Just Another Way of Information Retrieval: Penggunaan *Hyperdimensional Computing* dalam Implementasi *Search Engine*

Yosef Nuraga Wicaksana
Fakultas Ilmu Komputer
Universitas Indonesia
Depok, Indonesia
yosef.nuraga@ui.ac.id

Vian Sebastian Bromokusumo
Departemen Ilmu Komputer dan Elektronika
Universitas Gadjah Mada
Sleman, Indonesia
viansebastianbromokusumo@mail.ugm.ac.id

Louis Widi Anandaputra
Departemen Ilmu Komputer dan Elektronika
Universitas Gadjah Mada
Sleman, Indonesia
louiswidianandaputra@mail.ugm.ac.id

***Abstrak*—*E-commerce* sebagai media transaksi berbagai produk secara *online* menjadi bagian penting kehidupan masyarakat modern. *Search engine* sebagai bagian penting *e-commerce* sedang banyak melalui pengembangan, mengimplementasikan teknologi berbasis *deep learning* yang memiliki kemampuan tinggi untuk merepresentasikan informasi dalam ruang vektor pada data teks yang terdapat pada *query* pengguna dan data produk. Penggunaan teknologi berbasis *deep learning* seperti BERT dengan basis Transformer digunakan untuk mengatasi lemahnya kemampuan metode konvensional seperti BM25 untuk menangkap informasi kontekstual. Akan tetapi, beban komputasi tinggi menjadi keterbatasan. Penelitian ini memperkenalkan FastVSS yang menggunakan pendekatan *hyperdimensional computing* dengan memanfaatkan operasi aljabar sederhana, membawa paradigma cara kerja otak manusia yang berbeda dengan *neural network*, guna memberikan efisiensi komputasi. Melalui pengetesan terhadap data *e-commerce*, FastVSS yang dapat digunakan untuk mencari produk, mampu memberikan indikasi performa yang lebih kuat. Akan tetapi, masih perlu dilakukan penelitian lebih lanjut untuk mengetahui kemampuan *hyperdimensional computing* pada berbagai kondisi.**

Kata Kunci*—*Vector Symbolic Search, Vektorisasi, Komputasi Terdistribusi

I. PENDAHULUAN

Penggunaan *e-commerce platform* sebagai sarana jual-beli telah menjadi bagian dari kehidupan modern masyarakat Indonesia. Kegiatan jual-beli yang dapat dilakukan melalui *platform e-commerce* meliputi transaksi tiket kereta dan pesawat, jual-beli barang, hingga transaksi properti dan tanah. Hal ini membuat terdapat banyaknya opsi yang dimiliki konsumen untuk memilih produk yang akan dibeli. Hal ini juga sesuai dalam penggunaan *search engine* sebagai sarana mencari produk yang sesuai dengan kebutuhan konsumen. *Search engine* sebagai salah satu penerapan *information retrieval* memberikan kesempatan yang besar bagi pengembangan-pengembangan yang melibatkan teknologi-teknologi maju seperti *machine learning* (ML) yang sedang banyak digunakan pada masa ini.

Pengembangan ML pada *search engine* berkaitan dengan data teks pada *input user* sebagai *query* dan data produk yang tersimpan dalam *database* seperti nama produk, deskripsi produk, dan fitur produk. Dalam implementasi *search engine*, *user* akan memasukkan *query* sebagai *input* dan akan mendapatkan *output* dari *search engine* berupa data yang relevan terhadap *query* yang diberikan oleh *user.* Akan tetapi, pemanfaatan data *querying* dan *produk* tidak bisa dilakukan secara langsung. Data yang terdapat dalam bentuk teks butuh melalui transformasi secara matematis agar dapat diterima oleh model *information retrieval* pada *search engine*, baik itu data *query* maupun data produk. Representasi informasi dari data teks yang digunakan pada *query* dan produk banyak terdapat pada representasi ruang vektor yang dapat diimplementasikan pada fungsi penghitungan jarak kemiripan untuk mendapatkan skor *relevance* seperti yang terdapat pada *cosine similarity* maupun paradigma *learning to rank* (LTR) seperti LambdaMART [1] yang menggunakan ML untuk mendapatkan skor *relevance* tersebut.

Representasi informasi dalam sebuah ruang vektor dapat menggunakan beberapa metode. Secara umum, ada 2 metode representasi teks menjadi data pada ruang vektor, *context dependent* dan *context independent.* Metode *context independent* cenderung lebih sederhana karena seringkali hanya menggunakan frekuensi terhadap kata-kata seperti *term frequency-inverse document frequency* (TF-IDF) yang memberi pembobotan pada suatu kata terhadap keseluruhan dokumen atau *Best Match* 25 (BM25) yang merupakan ekstensi terhadap TF-IDF dengan memperhitungkan panjang dokumen [2]. Akan tetapi, metode seperti BM25 memiliki kekurangan dalam menangkap informasi semantik pada dokumen yang panjang [3].

Permasalahan representasi informasi secara semantik yang dimiliki oleh metode representasi *context independent* dapat diatasi dengan penggunaan metode ekstraksi dan representasi *context dependent* seperti BERT [4] yang menggunakan *multi-head attention* dalam mengolah vektor matriks, diperkenalkan oleh arsitektur Transformer [5]. Namun, permasalahan yang dimiliki oleh model yang menggunakan arsitektur Transformer seperti BERT adalah kompleksitas komputasi dan kompleksitas ruang yang tinggi. Hal ini dapat dilihat dengan beberapa model yang mencoba mengurangi kompleksitas, seperti MAMBA [6]. Akan tetapi, model tersebut tetap memiliki kompleksitas yang tinggi karena sifat *state space machine* yang masih sulit untuk melalui proses *training.*

Perkembangan terkini pada model *information retrieval* masih banyak mengarah kepada pengembangan-pengembangan model yang menggunakan *deep learning*

yang bisa terlihat pada perkembangan *dense retrieval model* [7]. Penggunaan *bi-encoder* pada pasangan *query-document* menghasilkan metode yang dapat berfokus pada representasi ruang vektor *query* dan *document* secara terpisah. Akan tetapi, proses ini dapat memiliki beban komputasi yang berat ketika menggunakan model-model dengan arsitektur rumit seperti BERT. Sehingga, dibutuhkan metode yang dapat menangkap informasi dengan baik, namun memiliki kompleksitas yang rendah.

Hyperdimensional Computing (HDC) dapat digunakan sebagai alternatif terhadap metode proses representasi data teks terhadap ruang vektor. Perbedaan utama HDC terhadap metode berbasis *neural network* adalah nihilnya implementasi proses deterministik menggunakan *weights* dan *biases*, digantikan oleh proses stokastik pada operasi aljabar dengan ruang vektor menggunakan data yang digunakan dalam proses *training* [8]. Pengimplementasian *search engine* pada HDC memberikan proses inferensi yang lebih singkat dan penggunaan daya komputasi yang lebih rendah dibandingkan dengan *neural network* [9]. Hal tersebut dipengaruhi oleh operasi yang dilakukan oleh HDC hanya menggunakan operasi aljabar linear sederhana yakni multiplikasi, adisi, dan permutasi. Selain itu, komputasi HDC juga mudah untuk dilakukan paralelisasi ketika mengoperasikan vektor dengan sistem terdistribusi.

Maka, hasil dari penelitian ini bertujuan untuk memperkenalkan penggunaan HDC pada proses *information retrieval* dalam kasus *search engine* pada model FastVSS. Adapun secara khusus, jangkauan dari penelitian ini adalah pengembangan model yang dapat memenuhi kebutuhan pemrosesan data pada ruang vektor sekaligus memiliki performa yang efisien, serta melakukan implementasi pada *search engine* untuk data *e-commerce*.

II. Landasan Teori

Informasi yang akan digunakan dalam implementasi *information retrieval* ini akan melalui proses HDC. HDC atau yang juga dikenal sebagai *Vector Symbolic Architecture* merupakan sebuah arsitektur komputasi yang termotivasi dari kemampuan otak manusia dalam menyimpan dan menggabungkan konsep untuk mendapatkan konsep baru tanpa melupakan konsep pembentuknya [8]. Dalam merepresentasikan konsep, HDC menggunakan *hypervector* (HV) atau sebuah vektor yang terdapat pada ruang vektor dengan ribuan elemen. Pendekatan HDC memiliki kelebihan besar karena beberapa sifat unik dari HV. [8]. Pertama, sifat quasi-orthogonal dari HV, sehingga setiap vektor dapat merepresentasikan informasi secara unik. Kedua, ketahanan akan kesalahan perangkat keras seperti bit flip error yang menyebabkan perubahan data yang direpresentasikan. Hal ini mungkin dilakukan karena informasi dalam HDC direpresentasikan oleh ribuan elemen untuk tiap HV [8]. Selain itu, HDC juga efisien dari segi kecepatan komputasi karena memanfaatkan paralelisasi serta operasi aljabar linier sederhana seperti multiplikasi ($*$) untuk membentuk konsep pasangan antara variabel dengan nilai HV, adisi ($+$) untuk agregasi konsep dari beberapa HV, dan permutasi (Π) untuk merepresentasikan adanya urutan dalam proses agregasi yang dilakukan dengan melakukan *shift right* pada elemen HV.

Sebagai contoh pemanfaatan beberapa operasi tersebut, untuk membentuk sebuah *HV* yang merupakan representasi dari konsep "susu dari solo" maka akan diperlukan HV yang merepresentasikan variabel minuman ($HV_{minuman}$) dan variabel asal (HV_{asal}) serta *HV* yang merepresentasikan minuman susu (HV_{susu}) dan asal Solo (HV_{solo}). Jika diasumsikan telah memiliki $HV_{minuman}$ maupun HV_{asal} namun belum memiliki HV_{susu} dan HV_{solo}, maka kedua *HV* tersebut dapat disusun menggunakan beberapa *HV* yang merepresentasikan huruf melalui agregasi dengan operasi adisi serta memanfaatkan operasi permutasi untuk memperhatikan urutan huruf seperti yang ditunjukkan oleh persamaan (3.1) dan (3.2).

$$HV_{susu} = \Pi(\Pi HV_S + HV_U) + HV_S) + HV_u \tag{3.1}$$

$$HV_{solo} = \Pi(\Pi(\Pi HV_S + HV_O) + HV_L) + HV_O \tag{3.2}$$

Selanjutnya, perlu dibuat terlebih dahulu sebuah informasi yang menghubungkan variabel objek dengan objek observasi dan variabel warna dengan warna observasi. Informasi tersebut dapat dikatakan sebagai "minuman adalah susu" dan "asal adalah solo". Kedua informasi tersebut dapat diperoleh melalui operasi multiplikasi yang tertera pada persamaan (3.3) dan (3.4).

$$HV_{minuman\ \ adalah\ susu} = HV_{minuman} * HV_{susu} \tag{3.3}$$

$$HV_{asal\ \ adalah\ solo} = HV_{asal} * HV_{solo} \tag{3.4}$$

Terakhir, pembentukan konsep "susu dari solo" dilakukan dengan operasi adisi untuk mengagregasi informasi objek dengan informasi warna sesuai persamaan (3.5).

$$HV_{susu\ dari\ solo} = HV_{minuman\ \ adalah\ susu} + HV_{asal\ \ adalah\ solo} \tag{3.5}$$

Berbeda dengan arsitektur komputer modern yang menempatkan informasi pada satu tempat untuk melakukan komputasi melalui *random access memory* [10], arsitektur HDC menyimpan informasi pada dua tempat yakni *item memory* (IM) dan *associative memory* (AM), yang masing-masing telah terinisialisasi untuk digunakan pada proses yang berbeda. Komputasi HDC dimulai dengan proses *encoding* dengan memetakan sampel *S* menjadi sebuah HV_S dengan memanfaatkan *HV* yang telah diinisialisasi di dalam IM sebagai *embedding*. Proses inferensi kemudian dapat dilakukan dengan memanfaatkan *similarity function* seperti *cosine similarity* yang memiliki rentang -1 hingga 1 untuk membandingkan HV_S dengan berbagai HV_{CLASS} dalam AM. Hal ini dilakukan untuk untuk mendapatkan HV_{CLASS} yang memiliki similaritas tertinggi sebagai label dari *S*. Terakhir, proses *training* dalam HDC dapat dilakukan dengan melakukan pengoreksian yang memanfaatkan operasi adisi pada AM ketika terdapat kesalahan prediksi.

III. Metodologi

A. Perangkat dan Parameter Penelitian

Penelitian ini menggunakan beberapa perangkat lunak seperti pyTorch versi 2.2.0, Numpy dan Perangkat keras berupa CPU AMD Ryzen 7 5800H untuk mengevaluasi performa model. Selain itu, penulis juga menetapkan beberapa parameter *random state* pada perangkat lunak

yang digunakan yakni pyTorch dan Numpy untuk memastikan reprodusibilitas penelitian.

B. Dataset

Wayfair Annotation Dataset (WANDS) merupakan dataset yang dikembangkan (Chen Yan, et. al . 2022) [11] untuk mengevaluasi performa *search engine* dalam latar *e-commerce.* Dataset ini mengandung 42,994 kandidat produk, 480 *queries*, dan 233,448 label relevansi antara *query* dan kandidat produk.

WANDS terdiri dari tiga tabel; tabel Produk, tabel Kueri, dan tabel Label. Tabel Produk merupakan tabel data produk yang tersedia dalam dataset ini. Rincian dari tabel Produk dapat dilihat di tabel 1.

Atribut	Tipe	Deskripsi	*Missing Values*
product_id	Numerik	ID produk	0
product_name	Teks	Nama produk	0
product_class	Teks	Kelas produk	2852
category hierarchy	Teks	Kategori produk	1556
product_description	Teks	Deskripsi produk	6008
product_features	Teks	Fitur/atribut produk	0
rating_count	Numerik	Jumlah rating	9452
average_rating	Numerik	Rata-rata nilai rating	9452
review_count	Numerik	Jumlah review	9452

Tabel 1: rincian tabel Produk

Tabel Kueri merupakan tabel penghubung antara sebuah *query* dengan classnya. Tabel ini berguna untuk membantu pengelompokan berbagai macam *query* unik menjadi satu kelompok *query* . Contohnya, *query* '*kohler whitehaven farmhouse kitchen sink', 'sinks bed frame',* dan *'pedistole sink'* termasuk dalam *query class* (kelas kueri) '*Kitchen Sinks'*. Rincian dari tabel Kueri dapat dilihat di tabel 2.

Atribut	Tipe	Deskripsi	*Missing Values*
query_id	Numerik	ID kueri	0
query	Teks	Kueri	0
query_class	Teks	Kelas Kueri	6

Tabel 2: rincian tabel Kueri

Tabel Label merupakan tabel yang menyimpan data relevansi antara sebuah *query* dan sebuah produk. Contohnya, *query* dengan ID 0 memiliki relevansi '*Exact'* (tepat sasaran) dengan produk 25434, namun memiliki relevansi *'Irrelevant'* (tidak berhubungan) dengan produk 12088. Rincian dari Tabel Label dapat dilihat di tabel 3.

Atribut	Tipe	Deskripsi	*Missing Values*
id	Numerik	ID label	0
query_id	Numerik	ID kueri	0
product_id	Numerik	ID Produk	0
label	Teks (kategorikal)	Label	0

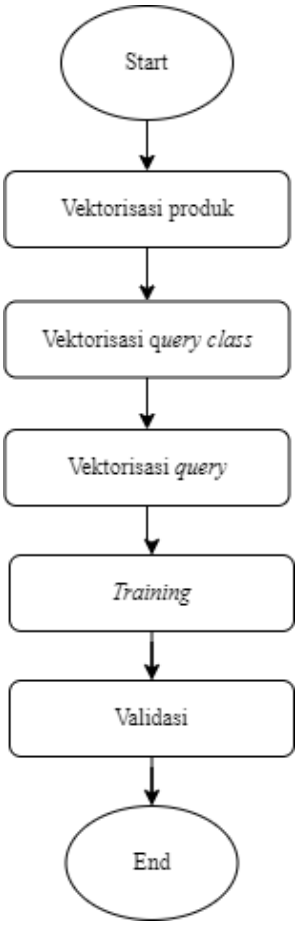
Tabel 3: rincian tabel Kueri

Dari rincian-rincian tabel diatas, terlihat bahwa dataset memiliki berbagai macam tipe data dan missing values. Pendekatan *preprocessing* yang digunakan adalah sebagai berikut:

1. *Missing values* pada data teks diimputasi menjadi "", yaitu sebuah *empty string.* Hal ini bertujuan untuk *placeholder* sementara, dikarenakan proses vektorisasi kedepannya akan menangani hal ini secara implisit.
2. *Missing values* pada data numerik diimputasi dengan nilai 0. Hal ini dilakukan dengan asumsi bahwa rating_count, average_rating, dan review_count yang kosong memiliki arti bahwa belum ada user yang melakukan review dan rating.
3. Atribut *product_features* dan *product_description* akan melalui tahap *cleaning* sederhana antara lain dibersihkan dari non-alfanumerik maupun kata penghubung, *stopwords*, serta tokenisasi, dan lemmatisasi.
4. Atribut Label pada tabel Label terdiri dari nilai unik *'Exact', 'Partial',* dan *'Irrelevant'*. Ketiga nilai ini diubah menjadi 2, 1, dan 0 secara berurutan, dengan *gain* atau nilai relevansi *query* sebesar 1, 0.5, dan 0. *Gain* sendiri akan digunakan dalam proses penghitungan DCG di tahap evaluasi.

C. Perancangan Model

Pada penelitian ini, penulis mengusulkan model *Fast Vector Symbolic Search* (FastVSS) sebagai *search engine* yang mampu memprediksi relevansi berdasarkan teks pencarian pengguna. Model FastVSS dirancang dengan pendekatan HDC yang efisien dari sisi daya komputasi dan latensi sehingga model akan memiliki skalabilitas yang baik sesuai dengan besar *knowledge base.* Secara umum, metode perancangan model direpresentasikan oleh Gambar 1 yang memiliki tahap vektorisasi produk hingga proses validasi sebagai evaluasi is performa model.



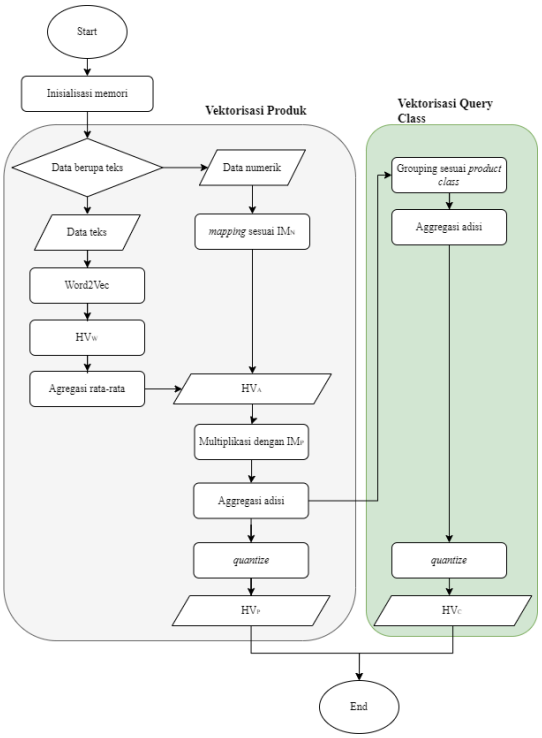
Gambar 1: alur perancangan model

a) Vektorisasi Produk dan Query Class

Agar dapat melakukan prediksi dengan pendekatan HDC, informasi yang tersimpan dalam *knowledge base* yakni produk dan query_class perlu ditransformasikan menjadi sebuah *HV* sehingga dapat disimpan dan dicari atau diambil dengan fungsi similaritas. Terdapat beberapa hal yang perlu dilakukan untuk melakukan proses vektorisasi. Pertama, menginisialisasi *item memory product* (IM_P) yang menyimpan delapan *HV* untuk merepresentasikan kedelapan atribut sampel produk dalam *dataset* WANDS, *item memory numeric* (IM_N) untuk memetakan data numerik menjadi HV_N, *item memory query* (IM_Q) yang menyimpan tiga *HV* untuk merepresentasikan variabel yang diperlukan *query* yakni teks *query*, *query_class*, dan produk, serta *associative memory* (AM) untuk menyimpan tiga HV_{CLASS} yang merepresentasikan label. Kedua, melatih Word2Vec untuk mentransformasikan kata menjadi HV_W yang memiliki dimensi sebesar 1000 dengan memanfaatkan atribut product_features dan product_description.

Proses vektorisasi produk kemudian dilakukan dengan mentransformasikan masing masing atribut produk menjadi HV_A. Pada atribut yang berupa data teks, transformasi dilakukan dengan memanfaatkan Word2Vec yang telah dilatih sebelumnya untuk mengubah kata-kata pada kalimat menjadi HV_W. Agar dapat merepresentasikan kalimat yang utuh, HV_W akan diagregasi dengan dirata-rata pada tiap elemennya. Berbeda dari atribut data teks, data numerik ditransformasikan dengan memetakan nilai menjadi indeks yang paling sesuai untuk mendapatkan HV_N dari IM_N. Kedelapan HV_A kemudian dimultiplikasi dengan IM_P untuk membentuk konsep pasangan antara variabel dan nilainya yang kemudian diagregasi dengan operasi adisi sehingga menghasilkan HV_P yang merepresentasikan sebuah sampel produk.

Selanjutnya, proses vektorisasi *query class* dilakukan dengan melakukan spesialisasi konsep produk sesuai *product class*, dimana spesialisasi adalah pengelompokan beberapa HV_P berdasarkan *product class* untuk diagregasi dengan operasi adisi pada masing-masing kelompok sehingga menghasilkan HV_C yang merepresentasikan sampel *query class*. Langkah terakhir pada tahap vektorisasi ini adalah kuantisasi pada tiap elemen dari seluruh HV_P maupun HV_C dengan fungsi *hyperbolic tangent* (Tanh), penulis mendefinisikan kuantisasi ini sebagai operasi *quantize*. Alur tahap vektorisasi ini sesuai pada ilustrasi Gambar 2.



Gambar 2: alur vektorisasi produk dan query class

b) Vektorisasi Query

Proses prediksi pada FastVSS akan memanfaatkan informasi dari teks *query, query class*, dan produk yang tersimpan dalam *knowledge base*. Langkah awal dari proses ini adalah transformasi teks *query* menjadi sebuah *HV*. Hal ini dilakukan dengan memanfaatkan Word2Vec untuk menghasilkan beberapa HV_W yang merepresentasikan kata-kata penyusun teks *query*. Beberapa HV_W tersebut kemudian di diagregasi

dengan dirata-rata pada tiap elemennya sehingga menghasilkan HV_Q.

Setelah didapatkannya HV_Q, perlu dibuat sebuah HV_S untuk merepresentasikan agregasi dari komponen HV_Q dengan HV_C dan HV_P yang disimpan dalam *database*. Konsep yang direpresentasikan HV_S dapat diinterpretasi sebagai "*Dicari* HV_Q *dalam kelompok* HV_C *menghasilkan* HV_P". Agar dapat membentuk konsep tersebut, diperlukan tiga komponen (C) yang perlu dihitung terlebih dahulu. Pertama, C_0 yang merepresentasikan konsep "*Dicari* HV_Q" dengan dihitung sesuai persamaan (4.1).

$$C_0 = HV_Q * \mathrm{IM}_{Q=0} \quad (4.1)$$

Kedua, C_1 yang merepresentasikan konsep "Kelompok HV_C" dengan dihitung sesuai persamaan (4.2).

$$C_1 = HV_C * \mathrm{IM}_{Q=1} \quad (4.2)$$

Terakhir, C_2 yang merepresentasikan konsep "Menghasilkan HV_P" dengan dihitung sesuai persamaan (4.3).

$$C_2 = HV_P * \mathrm{IM}_{Q=2} \quad (4.3)$$

Langkah selanjutnya, ketiga konsep tersebut diagregasi dengan operasi adisi untuk membentuk HV_S. Agar persebaran nilai pada tiap elemen terkontrol, maka tiap elemen akan dikuantisasi dengan operasi *quantize* yang telah didefinisikan sesuai dengan persamaan (4.4).

$$HV_S = quantize(C_0 + C_1 + C_2) \quad (4.4)$$

c) Training Model

Pada proses *training,* dilakukan pengadopsian metode dari model AdaptHD yang melakukan perhitungan pada similaritas menggunakan fungsi *cosine similarity* antara HV_S dengan setiap HV_{CLASS} yang tersimpan dalam *AM* untuk mendapatkan HV_{CLASS} dengan similaritas tertinggi ($HV_{CLASS\ prediksi}$) [12]. Apabila $HV_{CLASS\ prediksi}$ tidak sesuai dengan HV_{CLASS} yang merepresentasikan label ($HV_{CLASS\ label}$), maka akan dilakukan pengoreksian sebesar *learning rate* (*lr*) sesuai dengan persamaan (4.5) dan persamaan (4.6):

$$HV_{CLASS\ prediksi} = HV_{CLASS\ prediksi} - HV_s \times lr \quad (4.5)$$

$$HV_{CLASS\ label} = HV_{CLASS\ label} + HV_s \times lr \quad (4.6)$$

Walaupun pendekatan AdaptHD memungkinkan model meraih performa yang optimal, penambahan informasi secara berlebihan dapat menyebabkan *overfitting*. Untuk menghindari *overfitting*, penulis menggunakan syarat pembatasan yang akan menghentikan penambahan informasi saat kemiripan HV_S dengan $HV_{CLASS\ label}$ mencapai nilai yang tinggi seperti 0,95 [13].

Termotivasi dari syarat pembatasan untuk *overfitting*, penulis mengembangkan syarat baru untuk menghindari *underfitting*. Syarat ini didefinisikan dengan mempertimbangkan rentang hasil fungsi *cosine similarity* pada -1 hingga 0 yang merepresentasikan korelasi negatif atau nihilnya korelasi, maka sebuah prediksi yang benar namun memiliki similaritas yang terletak pada rentang tersebut mengindikasikan ketidakmampuan model dalam melakukan prediksi untuk sampel tersebut. Oleh karena itu akan dilakukan pengoreksian sesuai dengan persamaan (4.6).

D. Validasi

Pada penelitian ini, validasi model menggunakan metode *Leave-One-Group Out Cross Validation* (LOGOCV). LOGOCV merupakan metode validasi yang serupa dengan *Group K-Fold Cross Validation* dengan jumlah *fold* sebanyak jumlah grup pada *dataset* sehingga baik digunakan untuk mengukur performa model dengan tingkat bias yang rendah. Pada tahap validasi, perhitungan metrik *ranking* yang perlu pengurutan akan dilakukan dengan mengurutkan tiap grup berdasarkan label dan nilai similaritasnya.

E. Metrik

Terdapat beberapa metrik evaluasi yang digunakan pada penelitian ini. Secara umum, terdapat dua aspek yang akan dievaluasi. Aspek pertama merupakan aspek hasil *retrieval* dan *ranking* yang akan menggunakan *mean reciprocal rank* (MRR) dan *normalized discounted cumulative discounted gain* (NDCG), sedangkan aspek kedua merupakan aspek beban komputasi yang dibutuhkan untuk menjalankan satu proses *retrieval* dan *ranking* melalui latensi.

a) Mean Reciprocal Rank

MRR merupakan metode evaluasi hasil pencarian pada *search engine* yang melihat kualitas pengurutan hasil pencarian. MRR hanya menggunakan posisi *rank* suatu produk dan memberi informasi mengenai rata-rata posisi mengenai hasil pertama pada semua *query* [14] seperti yang dijelaskan persamaan (4.7).

$$MRR = \frac{1}{Q} \sum_{i=1}^{Q} \frac{1}{rank\ i} \quad (4.7)$$

b) Normalized Discounted Cumulative Gain

NDCG merupakan metrik lain yang digunakan untuk mengukur kualitas dari hasil pencarian. Komponen *cumulative gain* (CG) dari NDCG melibatkan perhitungan nilai relevansi, memberikan gambaran lebih spesifik mengenai kemampuan pencarian model [15]. Selain itu, Komponen perhitungan NDCG memiliki keterkaitan dengan komponen evaluasi seperti, *discounted cumulative gain* (DCG) dan *ideal discounted cumulative gain* (IDCG). Pada penelitian ini, nilai DCG didapat dari skor relevansi hasil prediksi model yang diurutkan

untuk menghasilkan *ranking* sementara. Lalu, nilai IDCG didapat dari perhitungan *ground truth* pada *ranking* data WANDS.

(1) Cumulative Gain

CG merupakan jumlah atas semua nilai gains pada suatu *query* dengan jumlah *ranks* sebanyak *n* yang diekspresikan oleh persamaan (4.8). *Gain* sendiri merupakan skor relevansi *query* dengan produk.

$$CG = \sum_{i=1}^{n} Gain_i \quad (4.8)$$

(2) Discounted Cumulative Gain

DCG mengadopsi konsep yang sama dengan CG dengan tambahan langkah diskon *Gain* berdasarkan peringkatnya. Langkah tersebut dapat dilihat pada persamaan (4.9) berikut.

$$DCG = \sum_{i=1}^{ranks} \frac{Gains}{log(i+1)} \quad (4.9)$$

(3) Ideal Discounted Cumulative Gain

IDCG sendiri dapat dikatakan serupa dengan DCG. Hanya saja, dihitung berdasarkan urutan skor relevansi ideal seperti yang terdapat pada persamaan (4.10).

$$IDCG = \sum_{i=1}^{ranks} \frac{sorted(Gains)}{log(i+1)} \quad (4.10)$$

(4) Normalized Discounted Cumulative Gain

NDCG, merupakan bentuk normal dari hasil pencarian yang dievaluasi dari DCG dan IDCG. Pada hakikatnya, NDCG dapat diinterpretasikan sebagai perbandingan relevansi aktual terhadap relevansi ideal. Persamaan (4.11) mendefinisikan NDCG sebagai hasil pembagian DCG terhadap IDCG.

$$NDCG = \frac{DCG}{IDCG} \quad (4.11)$$

c) Rata-Rata Latensi

Rata-rata latensi merupakan metrik yang digunakan untuk mengukur rata-rata waktu yang dibutuhkan oleh model untuk membuat sebuah prediksi dari tiap sampel *query* pada dataset. Dengan metrik ini proses penerimaan sampel *query* hingga proses prediksi diukur untuk mengevaluasi efisiensi latensi dari tiap model. Pada penelitian ini, penulis menggunakan satuan *milisecond* (ms).

IV. Hasil dan Pembahasan

Hasil dari *training* FastVSS yang dilakukan pada dataset WANDS dievaluasi menggunakan beberapa metrik, antara lain MRR, NDCG untuk 50 hasil pencarian, serta rata-rata latensi untuk satu kali pencarian. Alhasil, model FastVSS dapat digunakan sebagai metode *information retrieval* pada *search engine.* Nilai-nilai metrik *training* evaluasi FastVSS dapat dicermati di tabel 4.

Model	MRR	NDCG@50	Rata-Rata Latensi (ms)
FastVSS	0.9	0.820	0.11

Tabel 4: evaluasi FastVSS

FastVSS mendapatkan nilai MRR 0.9, NDCG@50 0.820, serta rata-rata latensi 0.11 ms. Nilai MRR dan NDCG yang mendekati 1 memberikan indikasi bahwa model memiliki performa yang cukup tinggi ketika memproses *query* untuk mengembalikan hasil pencarian.

Nilai MRR yang relatif tinggi mengindikasikan bahwa produk yang dikembalikan pada posisi pertama merupakan produk yang relevan. Lalu, pada nilai NDCG melebihi 0.8, skor relevansi yang diprediksi dapat dikatakan sesuai dengan *ground truth*. Terakhir, rata-rata latensi menunjukkan waktu yang singkat pada nilai 0.11 ms. Akan tetapi, perlu dilakukan percobaan lebih lanjut untuk mengetahui beban komputasi pada situasi lain.

Hasil tersebut menunjukkan indikasi bahwa FastVSS berhasil mengeluarkan produk-produk yang relatif tinggi relevansinya terhadap *query*, diikuti dengan pengurutan (*ranking*) dekat dengan ideal, serta berhasil melakukannya dengan cepat. Adapun perbandingan performa antara FastVSS dengan model LLM konvensional dapat dicermati di tabel 5.

Model	NDCG@50
Roberta Base	0.772
Roberta Large	0.773
SimCSE large	0.768
FastVSS (diusulkan)	**0.820**

Tabel 5: perbandingan FastVSS dengan LLMs

Dapat terlihat bahwa FastVSS memiliki indikasi kemampuan *retrieval* yang baik jika dibandingkan dengan model-model LLM yang kuat seperti Roberta Base yang memiliki 125 juta parameter, maupun Roberta Large yang memiliki 355 juta parameter [16]. Terdapat perbedaan nilai lebih dari 0.05 untuk setiap model, mengindikasikan potensi yang menjanjikan bagi pengembangan HDC untuk *information retrieval* pada *search engine*

V. Kesimpulan dan Saran

Melalui kebutuhan metode *search engine* yang cepat sekaligus memiliki kemampuan *modelling* yang baik, penelitian ini memperkenalkan FastVSS. Menggunakan basis representasi vektor dan dilanjutkan dengan operasi aljabar sederhana, proses pencarian dengan *query* sebagai masukkan dapat dilakukan. FastVSS memiliki nilai performa yang baik dibandingkan model lain seperti Roberta dan SimCSE. Untuk penelitian selanjutnya, penulis mengusung beberapa poin yang dapat dieksplorasi antara lain:

(1) Penggunaan hybrid HDC-Neural Network, serta metode vektorisasi awal yang memiliki performa lebih tinggi dibandingkan dengan Word2Vec guna meningkatkan kemampuan model memproses data,
(2) Pengujian FastVSS pada data dan metode train-test-split yang lebih bervariasi untuk melihat kemampuan pemodelan dan efisiensi model,
(3) Pengujian FastVSS pada *distributed computing*, baik GPU maupun *distributed systems*, terutama integrasi dengan sistem *online search engine*, serta
(4) Pengujian FastVSS pada implementasi *information retrieval* lainnya seperti RAG dan *question-answering*.

Ucapan Terima Kasih

Segala puji dan syukur peneliti panjatkan kepada Tuhan Yang Maha Esa atas bimbingan, berkat, dan kasih-Nya sehingga dapat menyelesaikan artikel ini. Kami juga berterima kasih kepada RISTEK Fasilkom UI yang telah memberikan kesempatan bagi kami untuk melakukan eksplorasi pada topik yang sulit. Terima kasih juga kami sampaikan kepada para dosen Universitas Gadjah Mada dan Universitas Indonesia yang telah memberikan masukkan terhadap pekerjaan kami. Terakhir kami berterima kasih kepada orang tua yang telah memberikan dukungan moral, material, dan spiritual kepada penulis.

Daftar Pustaka

[1] C. J. C. Burges, 'From RankNet to LambdaRank to LambdaMART: An Overview', Microsoft Research, 2010.
[2] G. Amati, 'BM25', in Encyclopedia of Database Systems, L. Liu and M. T. Özsu, Eds. Boston, MA: Springer US, 2009, pp. 257–260.
[3] Y. Lv and C. Zhai, 'When documents are very long, BM25 fails!', in Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, Beijing, China, 2011, pp. 1103–1104.
[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding', in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 4171–4186.
[5] A. Vaswani et al., 'Attention is All you Need', in Advances in Neural Information Processing Systems, 2017, vol. 30.
[6] A. Gu and T. Dao, 'Mamba: Linear-Time Sequence Modeling with Selective State Spaces', arXiv [cs.LG]. 2024.
[7] K. A. Hambarde and H. Proença, 'Information Retrieval: Recent Advances and Beyond', IEEE Access, vol. 11, pp. 76581–76604, 2023.
[8] P. Kanerva, 'Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors', Cognitive Computation, vol. 1, no. 2, pp. 139–159, Jun. 2009.
[9] D. Ma and X. Jiao, 'Hyperdimensional Computing vs. Neural Networks: Comparing Architecture and Learning Process', arXiv [cs.NE]. 2022.
[10] J. von Neumann, 'First draft of a report on the EDVAC', IEEE Annals of the History of Computing, vol. 15, no. 4, pp. 27–75, 1993.
[11] Y. Chen, S. Liu, Z. Liu, W. Sun, L. Baltrunas, and B. Schroeder, 'WANDS: Dataset for Product Search Relevance Assessment', in Advances in Information Retrieval, 2022, pp. 128–141.
[12] M. Imani, J. Morris, S. Bosch, H. Shu, G. D. Micheli, and T. Rosing, 'AdaptHD: Adaptive Efficient Training for Brain-Inspired Hyperdimensional Computing', in 2019 IEEE Biomedical Circuits and Systems Conference (BioCAS), 2019, pp. 1–4.
[13] A. Rahimi, S. Benatti, P. Kanerva, L. Benini, and J. M. Rabaey, 'Hyperdimensional biosignal processing: A case study for EMG-based hand gesture recognition', in 2016 IEEE International Conference on Rebooting Computing (ICRC), 2016, pp. 1–8.
[14] N. Craswell, 'Mean Reciprocal Rank', in Encyclopedia of Database Systems, L. Liu and M. T. Özsu, Eds. Boston, MA: Springer US, 2009, pp. 1703–1703.
[15] K. Järvelin and J. Kekäläinen, 'IR evaluation methods for retrieving highly relevant documents', in Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Athens, Greece, 2000, pp. 41–48.
[16] Z. Liu, W. Zhang, Y. Chen, W. Sun, T. Du, and B. Schroeder, 'Towards Generalizeable Semantic Product Search by Text Similarity Pre-training on Search Click Logs', in Proceedings of the Fifth Workshop on e-Commerce and NLP (ECNLP 5), 2022, pp. 224–233.