BAB 2
LANDASAN TEORI

Penelitian ini didasarkan pada penelitian terdahulu dan relevan dengan pencarian berbasis konteks, yang dijabarkan sebagai berikut.

2.1 Penelitian Terhadulu

Tabel 2.1 merupakan tabel perbandingan terhadap penelitian sekarang dan terdahulu.

Tabel 2.1. Penelitian Terdahulu

No	Aspek/Fitur Utama	Y. Kim (2023)	Emir Öztürk (2024)	Jiahui Zhao (2024)	Penelitian ini
1	Sistem pencarian berbasis konteks	✓	×	✓	✓
2	Implementasi model SBERT	✓	×	×	✓
3	Penggunaan Qdrant untuk *database* dan *similarity search engine*	×	✓	×	✓
4	Implementasi pada dokumen *Regional Trade Agreement* (RTA)	×	×	✓	✓

Berdasarkan perbandingan pada Tabel 2.1, terlihat adanya *research gap* yang signifikan di mana belum terdapat penelitian yang secara komprehensif mengintegrasikan penggunaan model SBERT dengan *vector database* Qdrant khusus pada *domain* dokumen *Regional Trade Agreement* (RTA). Penelitian terdahulu oleh Kim (2023) dan Zhao (2024) memang telah menerapkan pencarian berbasis konteks, namun belum memanfaatkan arsitektur *siamese* dari SBERT dengan skalabilitas pencarian ANN yang ditawarkan oleh Qdrant. Di sisi lain, penelitian Öztürk (2024) hanya berfokus pada analisis performa teknis Qdrant tanpa adanya implementasi model *embedding* kalimat spesifik maupun penerapan pada domain hukum. Oleh karena itu, kebaruan dan pembeda utama penelitian ini terletak pada penggabungan model SBERT untuk menghasilkan *embedding* dengan Qdrant sebagai mesin pencari dan manajemen *vector*, guna mengatasi kendala pencarian semantik pada dokumen perjanjian dagang internasional yang kompleks.

2.2 Normalized Discounted Cumulative Gain (NDCG@K)

Normalized Discounted Cumulative Gain (NDCG) merupakan salah satu metrik evaluasi peringkat yang umum digunakan, khususnya dalam sistem pencarian informasi atau *web search*. NDCG memiliki dua keunggulan utama dibandingkan metrik evaluasi biner lainnya. Pertama, NDCG mendukung *graded relevance* (relevansi bertingkat), sehingga dapat mengakomodasi berbagai derajat relevansi dokumen (misalnya skala 0–3) dan tidak hanya membedakan dokumen sebagai relevan atau tidak relevan secara biner. Kedua, NDCG menerapkan fungsi diskon terhadap posisi peringkat. Hal ini mencerminkan perilaku pengguna yang cenderung lebih memperhatikan dokumen pada peringkat atas dibandingkan dokumen di peringkat bawah [11, 12].

Metrik NDCG@K dihitung dengan membagi nilai *Discounted Cumulative Gain* (DCG) dengan nilai *Ideal Discounted Cumulative Gain* (IDCG) pada batas posisi K tertentu, yang dirumuskan pada 2.1.

$$NDCG@K = \frac{DCG@K}{IDCG@K} \tag{2.1}$$

Adapun komponen-komponen utama dalam perhitungan NDCG adalah sebagai berikut.

1. *Cumulative Gain (CG)*
   Merupakan jumlah total dari seluruh nilai relevansi dokumen tanpa mempertimbangkan posisi peringkatnya dalam daftar hasil. Dimana, rumusnya dijabarkan pada 2.2.

$$CG@K = \sum_{i=1}^{K} rel_i \tag{2.2}$$

   di mana:

   a. $CG@K$: *Cumulative Gain* pada batas K dokumen teratas.
   b. K: Batas jumlah dokumen teratas yang dievaluasi.
   c. rel_i: Derajat relevansi dokumen pada posisi ke-i.
   d. i: Indeks posisi peringkat dokumen dalam daftar (1 hingga K).

2. *Discounted Cumulative Gain (DCG)*

   Merupakan akumulasi nilai relevansi yang telah diberi bobot penalti (diskon) berdasarkan posisi peringkat dokumen. Dokumen yang relevan namun berada di peringkat bawah akan memberikan kontribusi skor yang lebih kecil. Yang dijabarkan pada rumus 2.3.

$$DCG@K = \sum_{i=1}^{K} \frac{rel_i}{\log_2(i+1)} \tag{2.3}$$

   di mana:

   a. $DCG@K$: *Discounted Cumulative Gain* pada batas K dokumen teratas.

   b. K: Batas jumlah dokumen teratas yang dievaluasi.

   c. rel_i: Derajat relevansi dokumen pada posisi ke-i.

   d. i: Indeks posisi peringkat dokumen dalam daftar (1 hingga K).

   e. $\log_2(i+1)$: Fungsi diskon logaritmik berdasarkan posisi peringkat.

3. *Ideal Discounted Cumulative Gain (IDCG)*

   Merupakan nilai DCG maksimum yang dapat diperoleh jika seluruh dokumen dalam daftar hasil diurutkan secara sempurna berdasarkan derajat relevansi tertingginya. IDCG berfungsi sebagai nilai normalisasi agar skor NDCG berada pada rentang 0 hingga 1, yang dijabarkan pada 2.4

$$IDCG@K = \sum_{i=1}^{K} \frac{rel_i^{ideal}}{\log_2(i+1)} \tag{2.4}$$

   di mana:

   a. $IDCG@K$: *Ideal DCG* nilai DCG pada susunan peringkat sempurna.

   b. rel_i^{ideal}: Derajat relevansi dokumen pada posisi ke-i setelah diurutkan menurun secara sempurna.

Penggunaan fungsi diskon $\log_2(i+1)$ bertujuan untuk menerapkan konsep *diminishing returns*. Melalui fungsi ini, bobot nilai relevansi akan menurun secara logaritmik seiring bertambahnya posisi i. Sebagai contoh, dokumen pada posisi 1 mendapatkan bobot diskon penuh sebesar 1.0, posisi 2 menurun menjadi 0.63, dan

pada posisi 10 bobotnya hanya tersisa 0.3 [11]. Keterangan variabel yang digunakan dalam rumus di atas adalah sebagai berikut.

a. $NDCG@K$: Nilai akhir efektivitas perangkatan dalam rentang 0–1.

b. K: Batas jumlah dokumen teratas yang dievaluasi (*cut-off*).

c. rel_i: Derajat relevansi dokumen pada posisi ke-i.

d. i: Indeks posisi peringkat dokumen dalam daftar (1 hingga K).

e. $\log_2(i+1)$: Fungsi diskon berdasarkan posisi peringkat.

f. $IDCG@K$: Nilai DCG ideal pada susunan peringkat sempurna.

2.3 Mean Reciprocal Rank (MRR)

Mean Reciprocal Rank (MRR) merupakan metrik evaluasi dalam *Information Retrieval* yang mengukur seberapa cepat sistem menempatkan dokumen relevan pertama pada posisi teratas dalam daftar hasil pencarian. Menurut Craswell, *Reciprocal Rank* menghitung nilai kebalikan (*reciprocal*) dari posisi di mana dokumen relevan pertama ditemukan. Nilai tersebut bernilai 1 jika dokumen relevan berada di posisi pertama, 0,5 jika di posisi kedua, 0,33 jika di posisi ketiga, dan seterusnya [13]. Ketika dirata-ratakan terhadap seluruh kueri, metrik ini disebut *Mean Reciprocal Rank*. Berbeda dengan NDCG yang mempertimbangkan seluruh urutan peringkat dan tingkat relevansi bertingkat (skor 0–3).

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i} \tag{2.5}$$

di mana:

a. MRR: *Mean Reciprocal Rank* rata-rata *reciprocal rank* terhadap seluruh kueri, bernilai 0–1.

b. $|Q|$: Jumlah total kueri yang dievaluasi.

c. Q: Himpunan seluruh kueri.

d. rank_i: Posisi peringkat (*rank*) dokumen relevan pertama yang ditemukan untuk kueri ke-i.

e. i: Indeks kueri (1 hingga $|Q|$).

Craswell menjelaskan bahwa MRR mengasumsikan pengguna akan menelusuri daftar hasil secara berurutan hingga menemukan dokumen relevan pertama, semakin teratas posisi dokumen relevan pertama, semakin besar kontribusinya terhadap MRR [14]. Sifat MRR yang hanya membedakan dokumen sebagai relevan atau tidak relevan (biner) menjadikannya metrik yang adil untuk perbandingan antar sistem (*head-to-head benchmark*). Hal ini karena hasil evaluasi tidak dipengaruhi oleh perbedaan distribusi skor relevansi antar sistem yang diuji. Dalam domain pencarian dokumen hukum, Zheng et al. menggunakan MRR sebagai metrik utama untuk mengevaluasi sistem *legal retrieval* karena kemampuannya mengukur ketepatan posisi tanpa bergantung pada asumsi distribusi relevansi tertentu [15].

2.4 Regional Trade Agreement (RTA)

Menurut *World Trade Organization* (WTO) *Regional Trade Agreement* (RTA) merupakan perjanjian perdagangan timbal balik antara dua atau lebih mitra dagang, yang tidak harus berasal dari wilayah geografis yang sama. Dalam kerangka kerja perdagangan global, WTO menegaskan bahwa RTA tetap bersifat komplementer yang berarti bukan pengganti sistem perdagangan multilateral [16].

WTO mengatur notifikasi RTA melalui tiga landasan hukum, yaitu pasal XXIV *General Agreement on Tariffs and Trade* (GATT) 1994 difokuskan untuk perdagangan barang. *Enabling Clause* untuk perjanjian antar negara berkembang, serta pasal V *General Agreement on Trade in Services* (GATS) untuk perdagangan jasa [17]. WTO juga mengklasifikasikan RTA ke dalam empat jenis perjanjian, yaitu *Free Trade Agreement* (FTA), *Customs Union* (CU), *Economic Integration Agreement* (EIA), dan *Partial Scope Agreement* (PS) [17].

2.5 Vector Search

Untuk menangani kelemahan pencarian kata kunci tradisional adalah dengan menerapkan *vector search*. Dimana *vector search* bekerja dengan cara menafsirkan apa yang mungkin dimaksud oleh pengguna, yang berbeda dengan pencarian literal yang hanya akan memberikan hasil berdasarkan apa yang telah ditentukan oleh mesin pencarian [18]. Pendekatan ini tidak lagi mencocokan kata, namun memahami makna dan konteks pengguna.

Cara untuk memahami makna dan konteks pengguna tersebut adalah dengan memanfaatkan model *deep learning* seperti *transformer*. Model ini memungkinkan data tidak terstruktur seperti teks *trade agreement* untuk di *embed* ke dalam representasi vektor yang memiliki panjang vektor yang tetap *(fixed-length vector representations)*. Vektor-vektor inilah yang secara mendasar menangkap fitur semantik berdimensi tinggi dari data aslinya. Dan kesamaan semantik tersebut antara dua potong teks secara alami dapat direprensentasikan oleh jarak antar vektor [19].

2.6 Sentence-BERT (SBERT)

Sentence Bidirectional Encoder Representations from Transformers (SBERT) adalah evolusi dari model BERT yang mengadopsi struktur *siamese* dan *triplet network* untuk menghasilkan *sentence embedding* yang memiliki makna semantik mendalam. Modifikasi ini menghasilkan *sentence embedding* yang optimal untuk diukur tingkat kemiripannya menggunakan metrik *cosine similarity* pada tahap pencarian [8].

Vektor *embedding* yang dihasilkan oleh SBERT merupakan representasi numerik yang memetakan data tekstual ke dalam ruang vektor dimensi tinggi. Kedekatan lokasi antar vektor (*locality*) dalam ruang tersebut secara langsung mengkodekan makna semantik dari inputnya, di mana kalimat dengan konteks serupa akan berada pada posisi yang berdekatan [20]. Kemampuan representasi ini telah banyak diimplementasikan untuk berbagai kebutuhan praktis, seperti *information retrieval*, sistem tanya jawab, analisis kesamaan tekstual semantik, hingga mesin rekomendasi [21].

Proses pembentukan representasi matematis ini diawali dengan tahap tokenisasi (*tokenization*), di mana teks mentah dipecah menjadi unit-unit kecil seperti kata atau *subwords*. Setiap *token* kemudian dipetakan menjadi nilai vektor awal melalui *lookup table* dalam matriks bobot arsitektur *neural network* [22]. Setelah melewati lapisan-lapisan *transformer*, SBERT melakukan tahap krusial yaitu operasi *pooling* (umumnya *mean pooling*). Tahap ini berfungsi untuk menggabungkan seluruh vektor *token* yang telah terkontekstualisasi menjadi satu vektor *embedding* berukuran tetap (*fixed-sized vector*) yang merepresentasikan makna kalimat secara utuh, yang nantinya siap untuk disimpan dan diproses lebih lanjut oleh *vector database* [8].

2.7 Approximate Nearest Neighbor Search (ANNS)

Approximate Nearest Neighbor Search (ANNS) merupakan sebuah paradigma pencarian yang dirancang untuk menemukan objek dalam basis data yang memiliki jarak (kemiripan) terdekat dengan kueri secara efisien. Paradigma ini dikembangkan karena dianggap sangat mahal secara komputasi untuk menemukan tetangga terdekat secara eksak di ruang *Euclidean* berdimensi tinggi, akibat fenomena *curse of dimensionality* [23]. *Curse of dimensionality* merujuk pada pertumbuhan eksponensial kompleksitas komputasi dan kebutuhan data seiring dengan bertambahnya dimensi ruang fitur [24].

Dengan kata lain, fenomena ini menyebabkan volume ruang meningkat sangat cepat sehingga data yang tersedia menjadi sangat jarang (*sparse*). Sebagai ilustrasi, sepuluh titik data mungkin cukup untuk mengisi ruang pada garis 1-dimensi, namun jumlah yang sama akan terasa sangat sedikit dalam kubus 3-dimensi, dan menjadi hampir tidak berarti dalam ruang berdimensi ratusan. Kondisi ini mengakibatkan hilangnya signifikansi perbedaan jarak, di mana jarak antara titik terjauh dan terdekat menjadi hampir seragam. Fenomena ini menimbulkan masalah seperti peningkatan komputasi, *overfitting*, hilangnya makna jarak, penurunan performa algoritma, serta tantangan visualisasi dan *clustering* [25].

Fenomena *curse of dimensionality* ini menuntut pendekatan aproksimasi untuk menjaga efisiensi pencarian. Oleh karena itu, algoritma ANNS dapat diklasifikasikan menjadi beberapa kategori utama yaitu: *tree-based methods, quantization-based methods, hashing-based methods*, dan *graph-based methods* [26]. Kategori-kategori ini menawarkan keseimbangan *(trade-off)* yang berbeda antara akurasi *(recall)*, kecepatan pencarian *(query time)*, waktu pembangunan indeks, dan penggunaan memori [27]. Berdasarkan jurnal-jurnal yang ada, secara umum *graph-based method* memiliki performa yang lebih baik dalam mencapai *trade-off* yang optimal antara tingkat akurasi dan efisiensi komputasi [26] [27] [28].

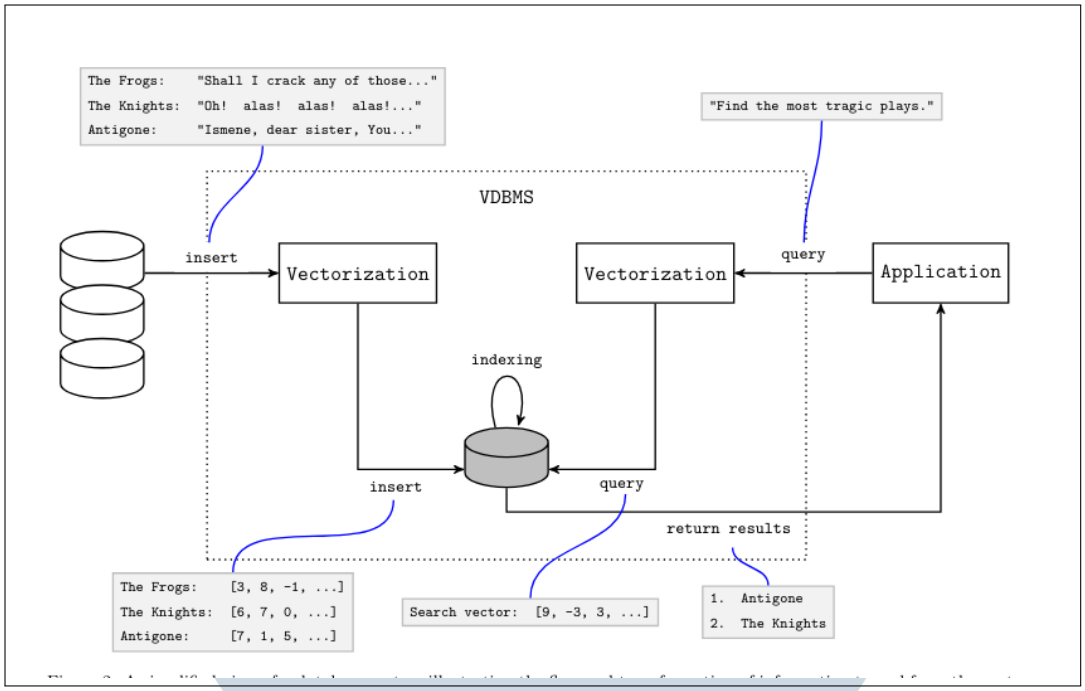
Metode *graph-based* bekerja dengan cara membangun *proximity graph*, di mana setiap vektor data direpresentasikan sebagai *node*, dan hubungan kedekatan antar data direpresentasikan sebagai *edge*. Algoritma pencarian pada struktur ini bekerja secara iteratif dengan menjelajahi lingkungan (*neighborhood*) dari *node* yang sedang dikunjungi untuk menemukan titik yang lebih dekat dengan kueri. Pendekatan ini memungkinkan pemrosesan kueri yang sangat cepat karena algoritma dapat langsung melompat menuju area yang relevan di dalam ruang fitur [26].

Salah satu contoh algoritma yang mengimplementasikan metode *graph-based* adalah *Hierarchical Navigable Small World* (HNSW), yang banyak di implementasikan dalam *vector database* karena efisiensinya dan skalabilitasnya. Karakteristik tersebut menjadikan HNSW sangat andal untuk berbagai aplikasi dunia nyata, seperti sistem rekomendasi dan pencarian teks, karena didukung oleh kemampuan eksplorasi lingkungan graf yang cepat (*fast neighborhood exploration*). [10]. Proses pencarian dimulai dari lapisan tertinggi untuk menemukan area umum yang paling dekat dengan kueri, kemudian berpindah turun ke lapisan yang lebih rendah secara bertahap hingga mencapai lapisan dasar yang mencakup seluruh dataset untuk menemukan tetangga terdekat dengan kueri [26].

2.8 Vector *Database* Management System (VDBMS)

Vector Database Management System (VDBMS) adalah sistem manajemen basis data khusus yang dirancang untuk mengelola data vektor berdimensi tinggi secara efisien. Sebagaimana kategori sistem manajemen basis data lainnya seperti relasional, dokumen, maupun graf, VDBMS didefinisikan sebagai sebuah perangkat lunak fungsional yang memiliki kapabilitas pengelolaan data secara utuh, melampaui batasan fungsionalitas sebuah pustaka perangkat lunak (*software library*) [10].

VDBMS umumnya mendukung pencarian kemiripan (*similarity search*) melalui metode indeksasi yang memungkinkan pencarian vektor serupa secara cepat dan akurat. Proses pencarian ini dilakukan untuk menemukan vektor-vektor yang paling menyerupai vektor kueri berdasarkan metrik jarak spesifik, seperti *Euclidean distance* atau *cosine similarity*. Kemampuan ini sangat bernilai dalam berbagai aplikasi di mana penemuan vektor serupa menjadi aspek yang krusial, seperti *retrieval system* pada gambar maupun teks [10].



Gambar 2.1. Cara vector *database* bekerja
Sumber [10]

Gambar 2.1 merupakan ilustrasi mengenai mekanisme kerja *vector database*, di mana alur dimulai dari sumber data tradisional yang ditransformasikan melalui proses *vector embedding*. Selain menyimpan vektor itu sendiri, sebuah VDBMS umumnya mengelola komponen data tambahan yang mencakup identitas unik (*identifier*), metadata, serta data asli (*payload*) [10]. Sebagai contoh dalam kasus *trade agreement*, sistem menyimpan hasil *embedding* yang menangkap pola semantik terkait isi perjanjian tersebut. Saat pengguna memberikan kueri dalam bahasa alami, kueri tersebut akan diubah menjadi *search vector* melalui proses vektorisasi yang sama. Nilai kedekatan antar vektor kueri dengan vektor-vektor yang telah terindeks di dalam *database* inilah yang memungkinkan ditemukannya dokumen perjanjian yang paling relevan secara semantik. Kemampuan integrasi antara vektor, metadata, dan kueri ini merupakan pilar utama dalam *Modern Information Retrieval Systems* yang memungkinkan mesin pencari memahami maksud pengguna secara lebih mendalam dan akurat [10].

2.9 Qdrant

Qdrant adalah *vector database* yang dirancang khusus untuk menyimpan dan melakukan pencarian pada vektor berdimensi tinggi, termasuk *embedding* yang

dihasilkan dari teks, gambar, maupun data tidak terstruktur lainnya [29]. Berbeda dengan sistem basis data relasional konvensional, Qdrant dioptimalkan untuk operasi pencarian berbasis kemiripan vektor (*vector similarity search*) [10]. Untuk mengimplementasikan pencarian tersebut, Qdrant menggunakan *Hierarchical Navigable Small World* (HNSW) sebagai algoritma indeksasi utamanya [29].

Hierarchical Navigable Small World (HNSW) merupakan salah satu algoritma indeksasi berbasis graf yang paling efektif untuk pencarian *approximate K-nearest neighbor search* (K-ANNS) pada data berdimensi tinggi. Diusulkan oleh Malkov dan Yashunin (2020), HNSW mengembangkan konsep *Navigable Small World* (NSW) dengan menambahkan struktur *hierarchical multilayer* yang memungkinkan pengskalaan kompleksitas pencarian mendekati $O(\log N)$ sambil mempertahankan *recall* yang tinggi [30].

Proses konstruksi HNSW dilakukan secara iteratif dengan memasukkan setiap titik ke dalam graf indeks untuk membentuk graf navigable dengan derajat terbatas yang ditentukan oleh parameter M. Untuk setiap elemen v yang dimasukkan, indeks *layer* maksimum l dipilih secara stokastik menggunakan distribusi probabilitas eksponensial yang dinormalisasi oleh konstanta $m_L = 1/\ln(M)$. Penentuan level ini memastikan bahwa panjang jalur karakteristik meningkat seiring indeks level. Pada dasarnya, *layer* teratas berisi *longest ranges link* yang akan dilalui pertama kali oleh algoritma pencarian, sedangkan *layer* terbawah berisi *shortest ranges link* yang dilalui terakhir. Yang kemudian berlangsung dalam dua fase.

a. Pada fase pertama, *greedy search* dilakukan secara iteratif dari *layer* teratas dimulai dari *entry point* yang telah ditentukan hingga ke *layer* ke-$(l+1)$.

b. Pada fase kedua, *greedy search* melanjutkan dari *layer* l hingga *layer* 0, di mana pada setiap level dipilih sebanyak ef *node* sebagai kandidat *edge*, dan maksimum M di antaranya dipilih sebagai tetangga v menggunakan algoritma *pruning* berbasis aproksimasi *Relative Neighborhood Graph* (RNG). Strategi *pruning* RNG ini menghapus *edge* redundan berdasarkan panjang *edge* (jarak antar *node*), sehingga mencapai *trade-off* akurasi-efisiensi yang lebih baik [31]. Pada *layer* 0, batas derajat ditingkatkan menjadi $2 \times M$ [32].

Dalam proses pencarian pada struktur HNSW, kemiripan antar vektor ditentukan menggunakan metrik kesamaan (*similarity metrics*). Salah satu metrik yang digunakan dalam penelitian ini adalah *Cosine Similarity*, yang mengukur kemiripan antara dua vektor berdasarkan sudut yang terbentuk di antara keduanya,

bukan berdasarkan jarak absolutnya. Secara matematis, *Cosine Similarity* antara dua vektor A dan B yang didefinisikan pada rumus 2.6.

$$\text{Cosine Similarity}(A,B) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.6}$$

Di mana variabel-variabel tersebut didefinisikan sebagai berikut.

a. Cosine Similarity(A,B): Skor kemiripan antara vektor A dan B, rentang -1 hingga 1.

b. A,B: Vektor representasi fitur hasil *embedding* teks.

c. A_i, B_i: Elemen ke-i pada masing-masing vektor.

d. n: Jumlah dimensi vektor.

e. i: Indeks elemen (1 hingga n).

f. $\|A\|, \|B\|$: Magnitudo (panjang Euclidean) masing-masing vektor.

Sebelum dilakukan perhitungan kemiripan, setiap vektor perlu diukur panjangnya menggunakan *L2 Norm* (jarak Euclidean). Perhitungan panjang *L2 Norm* untuk sebuah vektor $\mathbf{v}$ ditunjukkan pada Persamaan (2.7).

$$\|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^{n} v_i^2} \tag{2.7}$$

Proses normalisasi untuk setiap elemen vektor kemudian dilakukan agar setiap vektor memiliki panjang standar, yang dapat dirumuskan melalui Persamaan (2.8).

$$v_{norm} = \frac{v_i}{\|\mathbf{v}\|_2} \tag{2.8}$$

Di mana variabel-variabel tersebut didefinisikan sebagai berikut.

a. v_{norm}: Nilai elemen ke-i yang telah dinormalisasi (komponen dari *unit vector*).

b. v_i: Elemen asli ke-i dari vektor sebelum dilakukan proses normalisasi.

c. $\|\mathbf{v}\|_2$: Nilai *L2 Norm* atau panjang Euclidean dari sebuah vektor $\mathbf{v}$.

Melalui proses *L2 Norm* ini, setiap vektor dipastikan menjadi vektor satuan (*unit vector*) dengan panjang tepat satu ($\|A\| = \|B\| = 1$), sehingga perhitungan *Cosine Similarity* dapat disederhanakan menjadi *Dot Product* ($A \cdot B$) tanpa mengurangi akurasi pengukuran [33]. Dalam konteks penelitian ini, metrik tersebut digunakan oleh Qdrant untuk menentukan kemiripan semantik antara vektor kueri dengan vektor dokumen RTA yang tersimpan dalam indeks HNSW.