BAB 2
LANDASAN TEORI

2.1 Penelitian Terdahulu

Tabel 2.1 adalah tabel perbandingan penelitian terdahulu berdasarkan beberapa aspek, yaitu implementasi *vector search*, model *embedding* Multilingual-E5-Large, *vector database* Qdrant, algoritma HNSW, serta penggunaan domain resep masakan Indonesia.

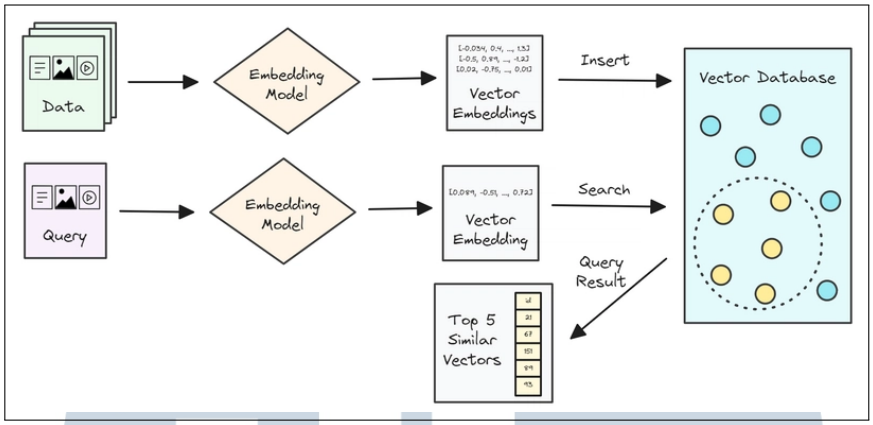
Tabel 2.1. Matriks Perbandingan Fitur Penelitian

No	Aspek/ Fitur Utama	Lin et al. (2023) [13]	Baur et al. (2025) [14]	Quisi-Peralta et al. (2024) [15]	Siregar et al. [7]	Liang et al. [9]	Penelitian ini
1	Implementasi sistem pencarian berbasis *vector search*	✓	✗	✓	✗	✓	✓
2	Implementasi model Multilingual-E5-Large	✗	✗	✗	✗	✓	✓
3	Penggunaan Qdrant sebagai *vector database*	✗	✓	✓	✗	✗	✓
4	Penggunaan algoritma HNSW untuk pencarian vektor	✓	✗	✗	✗	✗	✓
5	Implementasi pada resep masakan Indonesia	✗	✗	✗	✓	✗	✓

Berdasarkan Tabel 2.1, penelitian sebelumnya telah membahas penerapan metode yang berkaitan dengan sistem pencarian informasi. Lin et al. (2023) melakukan studi eksperimen untuk mengevaluasi implementasi *vector search*, sedangkan Baur et al. (2025) mengembangkan sistem *chatbot* yang menggunakan Qdrant sebagai *vector database* untuk melakukan *semantic search* dalam proses *retrieval* pada arsitektur *Retrieval-Augmented Generation* (RAG). Sementara itu, Quisi-Peralta et al. (2024) mengimplementasikan *vector search* pada sistem pencarian informasi untuk meningkatkan relevansi hasil pencarian. Perbandingan tersebut menunjukkan bahwa penelitian sebelumnya memiliki fokus yang berbeda, misalnya pada penggunaan Qdrant, algoritma HNSW, dan atau implementasi sistem pencarian berbasis *vector search*. Namun penelitian-penelitian tersebut, belum mengimplementasikan *vector search* dengan model *embedding* Multilingual-E5-Large, algoritma HNSW, serta Qdrant sebagai *vector database* pada domain resep masakan Indonesia. Oleh karena itu, penelitian ini bertujuan untuk mengembangkan sistem pencarian resep masakan Indonesia berbasis *vector search* dengan menggunakan model Multilingual-E5-Large, algoritma HNSW, dan Qdrant sebagai *vector database*.

2.2 Vector Search

Vector search adalah metode pencarian dengan menggunakan model *embedding* yang mengonversi data teks menjadi representasi numerik (vektor) berdimensi tinggi dalam ruang matematis [16]. Representasi tersebut memungkinkan objek dengan kemiripan makna memiliki jarak yang saling berdekatan dalam ruang vektor [17]. Dengan demikian, algoritma pencarian dapat memahami konteks dan hubungan yang lebih dalam di balik kata-kata yang ada di dalam data [18]. Berbeda dengan pendekatan tradisional seperti *keyword search* yang bergantung pada kecocokan kata secara literal [19]. *Vector search* memberikan hasil pencarian yang akurat dan relevan secara kontekstual [8]. Pendekatan ini bermanfaat ketika pengguna kesulitan dalam menentukan kata kunci yang sesuai untuk mendeskripsikan informasi yang dicari. Adapun tahapan dalam proses *vector search* adalah sebagai berikut [16].



Gambar 2.1. *Workflow Vector Search*

Sumber: [20]

1. *Vector Representation*

   Data berupa teks, gambar, atau video dikonversi menjadi representasi numerik (vektor) menggunakan model *embedding*. Setiap vektor direpresentasikan sebagai *points* dalam ruang berdimensi tinggi.

2. *Query Conversion*

   Ketika pengguna mengirimkan kueri pencarian, sistem juga mengkonversi kueri pengguna menjadi vektor dengan model *embedding* yang sama dengan data.

3. *Similarity Measurement*

   Sistem menghitung kemiripan antara vektor data dan vektor kueri dengan metrik seperti *cosine similarity* yang mengevaluasi sudut antar vektor. Vektor yang lebih dekat menunjukkan kemiripan yang lebih tinggi.

4. *Ranking and Results*

   Sistem akan mengurutkan dari yang paling mirip hingga yang paling tidak mirip. Data dengan nilai kemiripan tertinggi ditampilkan di bagian atas hasil pencarian. Hasil tersebut dipilih karena hasil tersebut terkait secara kontekstual.

2.3 Model Multilingual-E5-Large

Model Multilingual-E5-Large adalah model *embedding* yang dikembangkan berdasarkan arsitektur XLM-RoBERTa [21]. Secara arsitektur, model ini menggunakan pendekatan *encoder-only transformer* [22]. Model ini memiliki

24 lapisan (*layers*) yang mendukung lebih dari 100 bahasa dengan kemampuan menghasilkan representasi vektor berdimensi 1024 untuk setiap teks yang di proses [23]. Selain itu, model ini memiliki batas maksimum panjang teks masukan sebesar 512 token dalam satu kali proses *encoding* [24].

Pada tahap awal, model ini mewajibkan penambahan *prefix* khusus untuk membedakan antara kueri dan data. Data diawali dengan *prefix* "passage: ", sedangkan kueri pengguna diawali *prefix* "query: " [25]. Setelah itu, teks diproses melalui tahap tokenisasi atau *tokenization* menggunakan *tokenizer* untuk dipecah menjadi unit-unit kecil yang disebut token. Token tersebut dapat berupa kata utuh, potongan kata (*subword*), maupun tanda baca. Pendekatan ini digunakan untuk mengatasi permasalahan *out-of-vocabulary* (OOV). Dengan memecah kata menjadi bagian yang lebih kecil, model tetap dapat merepresentasikan makna dari kata baru berdasarkan kombinasi token penyusunnya.

Setelah tahap tokenisasi, token yang dihasilkan kemudian dikonversi menjadi representasi vektor awal (*embedding*). Representasi vektor awal tersebut kemudian diproses melalui 24 lapisan *encoder* yang masing-masing terdiri dari mekanisme *self-attention* dan *feed-forward network*. Mekanisme *self-attention* memungkinkan model untuk menghitung bobot hubungan atau tingkat kepentingan setiap kata terhadap seluruh kata lainnya dalam sebuah kalimat tanpa bergantung pada jarak posisi kata, sehingga hubungan antar kata dapat dipahami secara kontekstual [26]. Kemudian, mekanisme *feed-forward network* berfungsi untuk memproses hasil dari mekanisme *self-attention* agar menghasilkan representasi vektor yang lebih informatif.

Setelah melewati 24 lapisan, hasil *embedding* dari setiap token diringkas menjadi satu vektor tunggal sebagai representasi akhir suatu teks melalui metode *mean pooling*. Metode ini bekerja dengan cara menghitung nilai rata-rata dari seluruh *embedding* token yang membentuk teks tersebut. Vektor representasi berdimensi 1024 yang dihasilkan ini kemudian digunakan dalam proses pengukuran kemiripan (*similarity measurement*) pada sistem *vector search*.

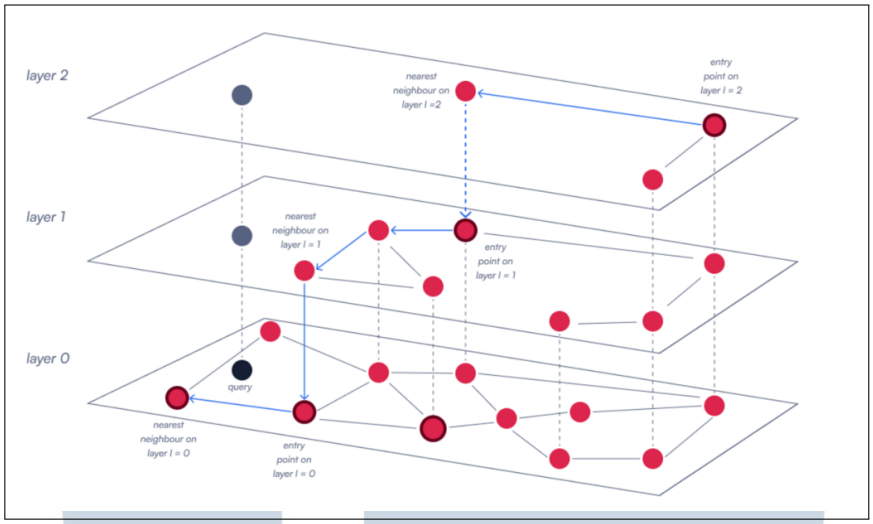
2.4 Algoritma HNSW

Hierarchical Navigable Small World adalah algoritma *approximate nearest neighbor search* yang dirancang untuk meningkatkan efisiensi pencarian vektor pada dataset berskala besar [27]. Algoritma ini membangun struktur graf bertingkat (*multilayer graph*), dengan setiap lapisan membentuk jaringan dunia

kecil (*navigable small-world network*) yang dapat dinavigasi, sehingga pencarian tetangga terdekat dapat dilakukan secara lebih cepat [28]. Setiap vektor direpresentasikan sebagai simpul (*node*) yang saling terhubung melalui tautan (*edge*) berdasarkan kedekatan jarak dalam ruang vektor [29]. Struktur graf tersebut dibagi ke dalam beberapa lapisan yang memungkinkan proses pencarian navigasi dari lapisan atas ke lapisan bawah untuk menemukan tetangga terdekat (*nearest neighbors*) secara cepat [30]. Lapisan teratas memiliki jumlah simpul (*node*) yang lebih sedikit serta sisi (*edge*) dengan koneksi jarak yang lebih panjang atau jauh untuk mempercepat pencarian, sedangkan lapisan yang lebih rendah memiliki jumlah simpul (*node*) yang lebih banyak serta sisi (*edge*) dengan koneksi jarak yang lebih pendek atau dekat untuk meningkatkan akurasi dan mendukung pencarian yang lebih detail [31].

Proses pencarian dimulai dari level atau lapisan paling atas pada graf sebagai titik awal navigasi. Pada tahap ini, algoritma mencari simpul (*node*) yang memiliki jarak paling dekat terhadap kueri. Simpul ini kemudian digunakan untuk melanjutkan pencarian ke level yang lebih rendah. Proses ini dilakukan secara bertahap pada setiap level hingga mencapai level terbawah. Setelah kandidat pada level terendah diperoleh, sistem menyusun dan mengurutkan hasil berdasarkan nilai kemiripan untuk menampilkan hasil pencarian yang paling relevan.

Gambar 2.2 adalah struktur jaringan (*multilayer*), yang terdiri dari beberapa lapisan (*layer*) yang saling terhubung, dengan setiap lapisan memiliki simpul (*node*) yang saling terkoneksi melalui *edge*. Visualisasi ini menggambarkan bagaimana simpul-simpul di setiap lapisan membentuk jaringan yang memungkinkan navigasi bertingkat pada algoritma HNSW.



Gambar 2.2. Struktur Jaringan Algoritma HNSW

Sumber: [32]

2.5 Vector Database Qdrant

Vector database adalah jenis basis data yang dirancang untuk menangani data berdimensi tinggi serta mendukung proses penyimpanan, pengindeksan, dan pencarian terhadap data yang direpresentasikan dalam bentuk vektor secara cepat [33]. Salah satu implementasinya adalah Qdrant, yang menyimpan setiap data dalam bentuk *point* yang terdiri dari vektor dan informasi tambahan dari dari data asli *payload* [34].

Untuk mengukur kedekatan antar vektor, digunakan metrik *cosine similarity* yang menghitung kosinus sudut antara dua vektor [35]. Jika dua vektor memiliki arah yang sama, artinya data tersebut mirip. Semakin kecil sudut yang terbentuk antara kedua vektor, maka tingkat kemiripannya semakin tinggi, sedangkan semakin besar sudut yang terbentuk antara kedua vektor, maka tingkat kemiripannya semakin rendah [35]. Perhitungan *Cosine similarity* ditunjukkan pada persamaan 2.1 berikut.

$$\text{Cosine Similarity}(A,B) = \frac{A \cdot B}{\|A\| \, \|B\|} \tag{2.1}$$

Keterangan:

a. A dan B adalah vektor representasi dari data atau kueri

b. $A \cdot B$ adalah hasil *dot product* antara vektor A dan B

c. $\|A\|$ dan $\|B\|$ adalah nilai magnitudo dari masing-masing vektor

Nilai cosine similarity berada pada rentang sebagai berikut.

a. Nilai 1 menunjukkan vektor identik atau tingkat kemiripan tinggi

b. Nilai 0 menunjukkan vektor orthogonal atau tegak lurus (tidak *related*) atau tidak adanya kemiripan

c. Nilai -1 menunjukkan vektor memiliki arah yang berlawanan

2.6 *Normalized Discounted Cumulative Gain* (NDCG)

Normalized Discounted Cumulative Gain (NDCG) adalah metrik evaluasi yang digunakan untuk mengukur kualitas hasil peringkat pada sistem pencarian dengan mempertimbangkan tingkat relevansi setiap item serta posisinya dalam daftar hasil [36]. Metrik ini menilai seberapa baik urutan hasil pencarian sistem dengan urutan relevansi ideal. Terdapat beberapa komponen yang digunakan dalam perhitungan NDCG, yaitu sebagai berikut.

1. CG (*Cumulative Gain*)
   CG adalah jumlah total nilai relevansi dari seluruh item yang terdapat pada hasil pencarian hingga peringkat ke-K tanpa mempertimbangkan posisi item dalam daftar peringkat. Perhitungan nilai CG hingga peringkat ke-K dapat dinyatakan pada Persamaan 2.2.

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

2. DCG (*Discounted Cumulative Gain*)
   DCG adalah nilai kumulatif relevansi item pada hasil pencarian hingga peringkat ke-K dengan mempertimbangkan posisi setiap item dalam daftar peringkat. DCG memberikan faktor diskon menggunakan fungsi logaritma, sehingga item yang berada pada posisi lebih rendah memberikan kontribusi yang lebih kecil dibandingkan item yang berada pada posisi yang lebih tinggi. Semakin tinggi relevansi dan semakin atas posisinya, maka kontribusi nilai DCG akan semakin besar. Perhitungan nilai DCG hingga peringkat ke-K ditunjukkan pada Persamaan 2.3.

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

3. IDCG (*Ideal Discounted Cumulative Gain*)
   IDCG adalah nilai DCG ideal yang diperoleh apabila data diurutkan berdasarkan relevansi tertinggi. IDCG digunakan sebagai pembanding untuk menilai seberapa dekat peringkat aktual dengan peringkat ideal. Nilai IDCG adalah nilai maksimum DCG yang dapat diperoleh apabila seluruh item diurutkan berdasarkan tingkat relevansi tertinggi. Perhitungan nilai IDCG hingga peringkat ke-K ditunjukkan pada Persamaan 2.4.

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

4. NDCG (*Normalized Discounted Cumulative Gain*)
   NDCG adalah nilai rasio yang diperoleh dengan membandingkan nilai DCG dan IDCG. Nilai NDCG berada pada rentang 0 hingga 1. Semakin mendekati nilai 1, maka kualitas peringkat hasil pencarian semakin baik dan semakin mendekati urutan ideal [37]. Nilai NDCG dihitung dengan membandingkan nilai DCG terhadap IDCG yang ditunjukkan pada Persamaan 2.5.

$$\text{NDCG@K} = \frac{DCG@K}{IDCG@K} \tag{2.5}$$

Keterangan:

a. K adalah jumlah peringkat yang dievaluasi.
b. i adalah posisi atau peringkat item pada hasil pencarian.
c. rel_i adalah nilai relevansi item pada peringkat ke-i.
d. rel_i^* adalah nilai relevansi item pada peringkat ke-i dalam urutan ideal.