Analisis Teknik Embedding Model *NV-Embed* pada *Large Language Models* Berbasis *Retrieval Augmented Generation*

Tengku Muhammad Rafi Rahardiansyah[1], Rizal Setya Perdana[2], Tirana Noor Fatyanosa[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]tengkumrafir@ub.ac.id, [2]rizalespe@ub.ac.id, [3]fatyanosa@ub.ac.id

Abstrak

Large Language Models (LLMs) berbasis *Retrieval-Augmented Generation* (RAG) menghadirkan tantangan dalam menghasilkan embedding yang akurat untuk meningkatkan performa retrieval dan generasi teks. NV-Embed adalah model embedding baru yang dirancang untuk mengatasi keterbatasan model embedding sebelumnya dengan pendekatan latent attention dan pelatihan contrastive instruction-tuning. Penelitian ini dilakukan dengan melakukan implementasi NV-Embed menggunakan bantuan *PyTorch*. Dokumen PDF diolah melalui tahap *pre-processing*, *tokenization* dan *vectorization*. Dokumen PDF yang telah diolah dan disimpan didalam *vector database* akan digunakan sebagai referensi untuk memperkaya hasil *response* berdasarkan informasi yang tersedia pada LLM dan informasi dari dokumen PDF yang didapatkan melalui *RAG pipeline*. Teknik embedding NV-Embed dievaluasi menggunakan metrik *precision*, *recall*, *F1-score* untuk *retrieval*, serta *BLEU* dan *ROUGE* untuk generasi teks. Hasil pengujian menunjukkan bahwa NV-Embed unggul dalam tugas retrieval dengan tingkat *precision* sebesar 0.906, *recall* sebesar 0.994, dan *f1-score* sebesar 0.948. Pada tugas generasi teks, NV-Embed mencapai hasil *BLEU* sebesar 0.899 dan metrik *ROUGE* juga menunjukkan hasil yang sangat baik, dengan *ROUGE-1* sebesar 0.955, *ROUGE-2* sebesar 0.951, dan *ROUGE-L* sebesar 0.955. Analisis terhadap performa NV-Embed menunjukkan bahwa pendekatan latent attention meningkatkan kualitas embedding dalam menangkap hubungan semantik antar kata. Penelitian ini memberikan kontribusi penting terhadap pengembangan model embedding dalam LLMs berbasis RAG dan membuka peluang untuk penelitian lebih lanjut.

Kata kunci: *nv-embed, embedding, large language models, retrieval-augmented generation, NLP*

Abstract

Large Language Models (LLMs) based on Retrieval-Augmented Generation (RAG) pose challenges in generating accurate embeddings to enhance retrieval and text generation performance. NV-Embed is a novel embedding model designed to address the limitations of previous embedding models through a latent attention approach and contrastive instruction-tuning training.This research implemented NV-Embed using PyTorch. PDF documents were processed through pre-processing, tokenization, and vectorization stages. The processed and stored documents in the vector database were utilized as references to enrich responses by combining the information available in the LLM with additional information retrieved from the PDF documents through the RAG pipeline. The NV-Embed embedding technique was evaluated using precision, recall, and F1-score metrics for retrieval, as well as BLEU and ROUGE for text generation.The results demonstrated that NV-Embed excelled in retrieval tasks, achieving a precision of 0.906, recall of 0.994, and F1-score of 0.948. In text generation tasks, NV-Embed achieved a BLEU score of 0.899, and the ROUGE metrics also showed excellent results, with ROUGE-1 at 0.955, ROUGE-2 at 0.951, and ROUGE-L at 0.955.The analysis of NV-Embed's performance revealed that the latent attention approach significantly improved embedding quality in capturing semantic relationships between words. This research contributes significantly to the development of embedding models in RAG-based LLMs and opens opportunities for further exploration.

Keywords: *nv-embed, embedding, large language models, retrieval-augmented generation, NLP*

1. PENDAHULUAN

Artificial Intelligence (AI) telah mencapai kemajuan besar dalam *Natural Language*

Processing (NLP), seperti kemampuan mesin untuk memahami dan menghasilkan bahasa alami (Xu et al., 2021). *Large Language Models* (LLMs), seperti *Generative Pre-trained Transformer* (GPT-3), telah memperlihatkan kemampuan signifikan dalam menghasilkan teks (Kasneci et al., 2023). Namun, GPT-3 memiliki keterbatasan, terutama dalam penalaran pada tugas-tugas kompleks seperti code data (Zhao et al., 2023).

Untuk mengatasi permasalahan ini, pada Juli 2021, OpenAI mengenalkan Codex, yaitu model GPT yang telah di fine-tune pada large corpus code dari GitHub. Hasilnya menunjukkan bahwa Codex mampu mengatasi permasalahan berat dalam pemrosesan code. Selain itu, pada Januari 2022 ditemukan pendekatan baru dengan melakukan training pada teks dan code embedding (Zhao et al., 2023).

Embedding adalah teknik yang mengonversi kata-kata menjadi representasi numerik untuk menangkap hubungan semantik antar kata lebih baik. Pendekatan ini berhasil meningkatkan kinerja dalam berbagai aplikasi NLP. Penelitian lain menunjukkan bahwa modifikasi embedding dan penggunaan memori eksternal mampu meningkatkan panjang *context window*, seperti pada *LongLLama*, yang memperpanjang *context window* dari 2K menjadi 8K (Zhao et al., 2023). Embedding menjadi salah satu pendekatan penting dalam meningkatkan performa LLM.

Namun, model LLM masih memiliki kelemahan, seperti menghasilkan informasi yang salah atau "halusinasi". *Hybrid Models* yang menggabungkan *parametric* dan *non-parametric memory* menjadi solusi untuk memperbaiki pengetahuan secara langsung (Lewis et al., 2020). Pendekatan ini dikenal sebagai *Retrieval Augmented Generation* (RAG), yang memanfaatkan retriever berbasis embedding untuk mengakses informasi eksternal terbaru tanpa mengubah parameter model (Lee et al., 2024).

Model embedding seperti BERT (*Bidirectional Encoder Representations from Transformers*) telah mendominasi beberapa tahun terakhir. BERT menggunakan pendekatan bidirectional untuk memahami konteks kata dalam teks, yang memungkinkan pembelajaran mendalam tanpa pelabelan yang rumit (Devlin et al., n.d.). SBERT (*Sentence-BERT*) kemudian diperkenalkan untuk meningkatkan kinerja BERT dalam mengukur kesamaan semantik antar kalimat dengan lebih efisien, terutama dalam tugas-tugas seperti pencarian semantik dan clustering teks (Reimers & Gurevych, 2019).

Transformer decoder-only LLMs juga mampu mengungguli model bidirectional dalam menghasilkan embedding yang lebih efisien (Lee et al., 2024). Namun, ini memerlukan fine-tuning dengan dataset besar yang tidak selalu tersedia secara publik. Model embedding baru, seperti NV-Embed, dirancang untuk meningkatkan performa model berbasis decoder-only dalam melakukan embedding.

Penelitian ini bertujuan untuk menganalisis embedding model NV-Embed dengan fokus mengevaluasi akurasi dan efektivitas NV-Embed. Diharapkan penelitian ini memberikan wawasan mendalam tentang keunggulan dan keterbatasan NV-Embed.

2. LANDASAN KEPUSTAKAAN

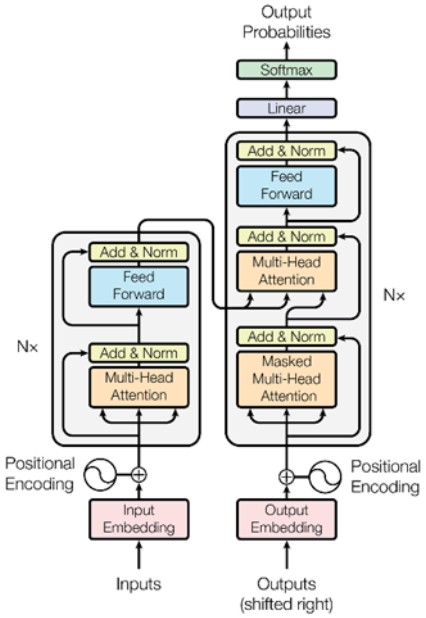
2.1. *Large Language Models* (LLMs)

Large Language Models (LLM) adalah model deep learning yang dirancang untuk memahami, menghasilkan, dan memanipulasi teks alami. LLM beroperasi dengan mempelajari representasi vektor dari kata, kalimat, atau dokumen, dan menggunakan parameter dalam jumlah besar untuk menyandikan hubungan semantik antar kata dalam teks (Brown et al., 2020; Devlin et al., n.d.).

Salah satu arsitektur kunci yang mendasari LLM adalah Transformer. Transformer pertama kali diperkenalkan oleh Vaswani et al. (2017).

2.2. Transformer

Transformer adalah arsitektur neural network yang pertama kali diperkenalkan oleh Vaswani et al. (2017) dalam makalah berjudul "*Attention is All You Need*." Transformer memiliki arsitektur sebagai ditampilkan dalam Gambar 1.



Gambar 1. Arsitektur Model Transformer

Sumber: Vaswani et al., 2017

Transformer terdiri dari dua komponen utama, yaitu *encoder* dan *decoder*. *Encoder* menerima sequence input dan memprosesnya menjadi representasi yang dapat dipahami oleh model, sementara *decoder* menghasilkan *sequence output* berdasarkan representasi yang dihasilkan oleh *encoder*. Arsitektur ini secara khusus dirancang untuk mengandalkan *attention mechanism* dalam setiap lapisan, yang memungkinkan model untuk fokus pada bagian-bagian yang relevan dari input sekuens (Vaswani et al., 2017).

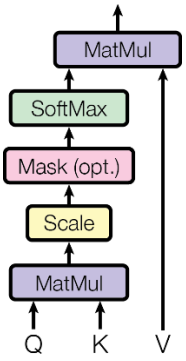
Encoder dan *decoder* memiliki kesamaan arsitektur, yang membedakkan decoder memiliki tambahan lapisan *masked self-attention* untuk memastikan bahwa setiap langkah tidak bergantung pada langkah di masa depan, namun bergantung pada informasi dari langkah-langkah sebelumnya.

2.3. *Self Attention*

Self-attention merupakan jenis attention yang digunakan dalam transformer, memungkinkan model untuk menghitung bobot antar elemen dalam sekuens input itu sendiri, bukan antar input dan output (Vaswani et al., 2017). Perhitungan untuk menghitung bobot attention sebagai ditampilkan dalam persamaan 1.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^{\top}}{\sqrt{d_k}}\right)V \qquad (1)$$

Proses *self-attention* dalam Transformer dimulai dengan mengonversi setiap elemen dalam sekuens input menjadi tiga vektor, yaitu *query* (Q), *key* (K), dan *value* (V). *Query* dan *key* digunakan untuk menghitung bobot attention, yang mencerminkan relevansi antara elemen input yang berbeda. Skor *attention* diperoleh dengan mengambil *dot product* antara *query* dan *key*, lalu hasilnya dinormalisasi menggunakan fungsi *softmax* untuk memastikan total bobot adalah 1.



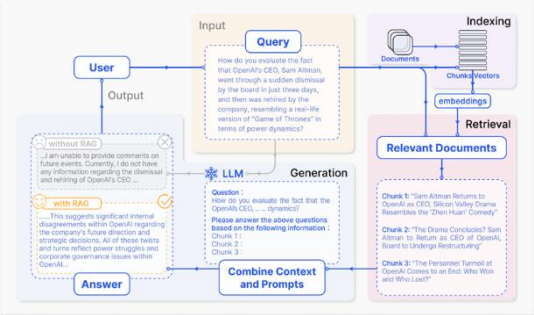
Gambar 2. *Scaled Dot-Product Attention*

Sumber: Vaswani et al., 2017

Proses ini sebagai ditampilkan dalam Gambar 2. menghasilkan representasi akhir dari *input sequence*, di mana elemen-elemen yang lebih relevan mendapatkan bobot yang lebih besar dalam pembentukan representasi tersebut (Vaswani et al., 2017).

2.4. *Retrieval Augmented Generation* (RAG)

Retrieval-Augmented Generation (RAG) adalah metode yang memadukan dua pendekatan utama, *retrieval* (pencarian informasi) dan *generation* (generasi teks). RAG menggabungkan *parametric memory* (pengetahuan internal yang tersimpan dalam parameter model) dan *non-parametric memory* (pengetahuan eksternal yang tersedia melalui retrieval) untuk menghasilkan respons yang lebih relevan. RAG memanfaatkan model retriever untuk mengakses pengetahuan eksternal dan menggabungkannya dengan model generator yang menghasilkan teks berdasarkan pengetahuan yang ditemukan (Gao et al., n.d.).



Gambar 3. Proses RAG

Sumber: Gao et al., n.d.

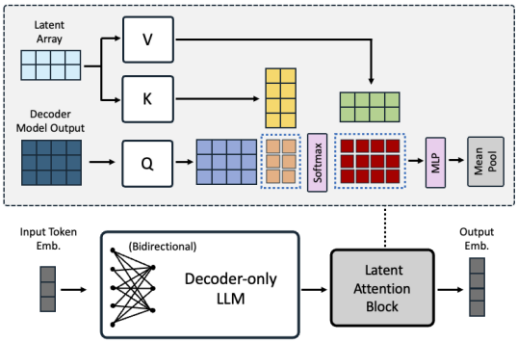
Proses RAG sebagai ditampilkan dalam Gambar 3. Pendekatan ini dirancang untuk mengatasi keterbatasan kecenderungan model menghasilkan konten yang tidak faktual atau tidak relevan, terutama ketika menangani pertanyaan di luar pengetahuan yang tersimpan dalam parameter model. Dengan melakukan retrieval atau pencarian dari database eksternal, RAG mampu memberikan jawaban yang lebih relevan terhadap pertanyaan yang diajukan, mengurangi risiko halusinasi atau "*hallucinations*" (Lewis et al., 2020).

2.5. Embedding

Embedding dalam LLM adalah representasi vektor berdimensi tinggi yang digunakan untuk memetakan kata, frasa, atau kalimat ke dalam bentuk numerik yang dapat diproses oleh model. Teknik embedding memungkinkan model bahasa memahami hubungan semantik antara kata-kata dengan memproyeksikannya ke dalam ruang vektor di mana kata-kata yang memiliki makna serupa terletak lebih dekat satu sama lain. Dalam *deep learning*, embedding menjadi penting karena menyediakan representasi informasi yang dapat digunakan oleh model untuk mempelajari pola dan hubungan dalam data teks (Mikolov et al., 2013).

2.6. NV-Embed

NV-Embed adalah model embedding terbaru yang dirancang untuk meningkatkan performa *decoder-only* pada LLMs dalam menghasilkan representasi numerik. Berbeda dengan model embedding berbasis *bidirectional* seperti BERT, NV-Embed menggunakan pendekatan baru yang memanfaatkan arsitektur latent attention layer serta pelatihan *contrastive instruction-tuning* secara bertahap. NV-Embed memperkenalkan latent attention layer, yang berfungsi untuk menghasilkan embedding yang lebih ekspresif. Desain arsitektur *Decoder-Only* dengan *latent attention layer* sebagai ditampilkan dalam Gambar 4.



Gambar 4. Desain Arsitektur *Decoder-Only* LLM dengan *Latent Attention Layer*

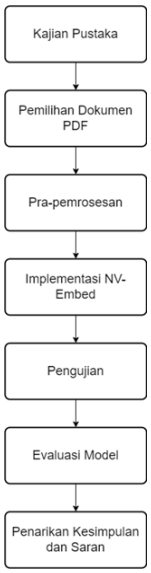
Sumber: Lee et al., 2024

Lapisan ini bertindak seperti bentuk *cross-attention*, di mana *output* dari *LLM decoder* bertindak sebagai *query* (Q) dan latent array yang dilatih digunakan untuk memperbaiki representasi kunci dan nilai (*key-value*). Teknik ini secara konsisten meningkatkan akurasi pada tugas pencarian (*retrieval*) dan tugas-tugas lainnya seperti klasifikasi dan kesamaan teks semantik (*semantic textual similarity*) dibandingkan dengan metode *pooling* rata-rata (*mean pooling*) yang biasa digunakan dalam model embedding *bidirectional* (Lee et al., 2024).

Pada tahap *training*, NV-Embed menggunakan pendekatan dua tahap. Tahap pertama, dilakukan pelatihan *contrastive* menggunakan *retrieval dataset* dengan memanfaatkan *in-batch negatives* dan *curated hard negative* yang dikurasi secara khusus sebagai contoh. Tahap kedua melibatkan *instruction-tuning* pada beragam *non-retrieval dataset* seperti klasifikasi dan pengelompokan, yang ternyata tidak hanya meningkatkan performa tugas *non-retrieval*, tetapi juga meningkatkan akurasi *retrieval* (Lee et al., 2024).

NV-Embed berhasil mencapai skor tertinggi pada *Massive Text Embedding Benchmark* (MTEB) dengan 69.32 pada 56 task, serta mencatat skor tertinggi pada 15 tugas retieval dari *BEIR benchmark*. Model ini menunjukkan kemampuan unggul dalam berbagai tugas embedding umum, termasuk *retrieval*, *reranking*, *klasifikasi*, pengelompokan, dan kesamaan tekstual *semantic* (Lee et al., 2024).

3. METODOLOGI



Gambar 5. Metodologi

3.1. Kajian Pusaka

Pada tahap ini, dilakukan kajian pustaka terkait model-model embedding yang sudah ada seperti BERT, Sentence-BERT dan GPT, serta NV-Embed sebagai model embedding terbaru. Literatur yang dibaca meliputi publikasi akademik, dokumentasi teknis, serta studi kasus terkait penerapan model-model.

3.2. Pemilihan Dokumen PDF

Kriteria dokumen pdf yang akan digunakan meliputi relevansi konten, kelengkapan informasi, bahasa yang digunakan, serta format dan kualitas file.

Sumber dokumen PDF berasal dari repositori akademik, jurnal ilmiah terindeks, serta database institusi akademik. Dokumen-dokumen ini diunduh dari platform terpercaya seperti *ResearchGate Google Scholar* dan *Arxiv*.

3.3. Pra-pemrosesan

Tahap pre-processing meliputi beberapa langkah penting, seperti memecah teks menjadi unit-unit dasar (token) yang akan digunakan oleh model embedding (*tokenization*) dan memetakan token yang telah diproses menjadi representasi numerik awal sebelum dimasukkan ke dalam model embedding.

3.4. Implementasi NV-Embed

Model NV-Embed diimplementasikan menggunakan bahasa pemrograman Python. Langkah ini melibatkan penggunaan framework deep learning *PyTorch*. Implementasi dilakukan dengan langkah-langkah berikut:

1. Menginstal dan mengonfigurasi environment sesuai dengan kebutuhan NV-Embed dan model lainnya.
2. Mengunakan langkah manualisasi sistematis pada bab analisis model.
3. Menggabungkan NV-Embed yang telah dibuat dengan *pipeline RAG.*
4. Melakukan eksperimen untuk memastikan bahwa model berfungsi dengan benar dan menghasilkan embedding yang diharapkan.

3.5. Pengujian

Pengujian dilakukan untuk mengevaluasi performa NV-Embed. Pengujian meliputi mengukur kemampuan model dalam menemukan dokumen atau bagian teks yang relevan dari corpus yang besar, menilai bagaimana embedding yang dihasilkan mempengaruhi kualitas teks yang dihasilkan oleh LLM dengan RAG. Pengujian akan dilakukan pada lingkungan penelitian sebagai ditampilkan dalam Tabel 1.

Tabel 1. Informasi Lingkungan Penelitian

No	Bagian	Spesifikasi
1	*Processor*	*Ryzen 9 4900HS (16 CPUs)*
2	*Memory*	*16GB Ram*
3	*VGA*	*NVIDIA GeForce RTX 2060 with Max-Q Design (5968 MB)*

3.6. Evaluasi

Evaluasi dilakukan dengan membandingkan hasil pengujian NV-Embed dengan model embedding yang sudah ada. Aspek evaluasi meliputi:

1. Akurasi *Retrieval*: Menggunakan metrik Precision, Recall, dan F1-Score untuk mengevaluasi hasil retrieval.
2. Kualitas Teks yang Dihasilkan: Menggunakan metrik BLEU dan ROUGE untuk menilai kualitas teks yang dihasilkan dalam tugas generation.

3.7. Penarikan Kesimpulan dan Saran

Bab ini menyimpulkan temuan utama dari penelitian yang telah dilakukan. Disajikan kesimpulan mengenai keunggulan dan keterbatasan arsitektur NV-Embed dalam menghasilkan representasi numerik.

Kemudian memberikan rekomendasi untuk penelitian di masa depan, termasuk optimasi dan langkah yang dapat ditempuh untuk mengoptimasi teknik embedding NV-Embed pada LLM berbasis RAG.

4. HASIL PENGUJIAN

4.1. Hasil Evaluasi

Evaluasi sistem NV-Embed dilakukan menggunakan berbagai metrik, seperti *precision*, *recall*, *f1-score*, *BLEU*, dan *ROUGE* untuk mengukur performa *retrieval* dan generasi respons.

Berdasarkan implementasi yang telah dilakukan didapatkan response yang telah berhasil dibuat sebagai berikut:

Tabel 2. Hasil *Generated Response*

Query	***Generated Response***
What is NV-Embed?	*NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models Chankyu Lee ∗1 Rajarshi Roy 1 Mengyao Xu 1 Jonathan Raiman 1 Mohammad Shoeybi 1 Bryan Catanzaro 1 1 NVIDIA Wei Ping ∗1 Abstract Decoder-only large language model (LLM)-based embedding models are begin- ning to outperform BERT or T5-based embedding models in general-purpose text embedding tasks, including dense vector-based retrieval. In this work, we introduce the NV-Embed model with a variety of architectural designs and training procedures to significantly enhance the performance of LLM as a versatile embedding model, while maintaining its simplicity and reproducibility. For model architecture, we propose a latent attention layer to obtain pooled embeddings, which consistently improves retrieval and downstream task accuracy compared to mean pooling or using the last <EOS> token embedding from LLMs. To enhance representation learning, we remove the causal attention mask of LLMs during contrastive training. For model training, we introduce a two-stage contrastive instruction-tuning method. It first applies contrastive training with instructions on retrieval datasets, utiliz- ing in-batch negatives and curated hard negative examples. At stage-2, it blends various non-retrieval datasets into instruction tuning, which not only enhances non-retrieval task accuracy but also improves retrieval performance. Combining these techniques, our NV-Embed model, using only publicly available data, has achieved a record-high score of 69.32, ranking No. 1 on the Massive Text Embedding Benchmark (MTEB) (as of May 24, 2024), with 56 tasks, encompassing retrieval, reranking, classification, clustering, and semantic textual similarity tasks. Notably, our model also attains the highest score of 59.36 on 15 retrieval tasks in the MTEB benchmark (also known as BEIR).*

Sebagai ditampilkan dalam Tabel 2, implementasi yang telah dilakukan berhasil mengembalikan *response* sesuai dengan informasi yang didapatkan dari dokumen PDF. Kemudian untuk mengukur hasil *response*, dengan menggunakan metrik *precision*, *recall*, *f1-score*, BLEU dan ROUGE, dilakukan pengujian dengan menggunakan *ground thruth* yang telah ditentukan:

Tabel 3. Hasil Evaluasi

Generated Response	*Ground Thruth*	Evaluasi
NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models Chankyu Lee ∗1 Rajarshi Roy 1 Mengyao Xu 1 Jonathan Raiman 1 Mohammad Shoeybi 1 Bryan Catanzaro 1 1 NVIDIA Wei Ping ∗1 Abstract Decoder-only large language model (LLM)-based embedding models are begin- ning to outperform BERT or T5-based embedding models in general-purpose text embedding tasks, including dense vector-based retrieval. In this work, we	*NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models. Decoder-only large language model (LLM)-based embedding models are begin- ning to outperform BERT or T5-based embedding models in general-purpose text embedding tasks, including dense vector-based retrieval. In this work, we introduce the NV-Embed model with a variety of architectural designs and training procedures to significantly enhance the performance of*	*Precision: 0.906 recall: 0.994 f1-score: 0.948 BLEU: 0.899 ROUGE-1: 0.955 ROUGE-2: 0.951 ROUGE-L: 0.955*

introduce the NV-Embed model with a variety of architectural designs and training procedures to significantly enhance the performance of LLM as a versatile embedding model, while maintaining its simplicity and reproducibility. For model architecture, we propose a latent attention layer to obtain pooled embeddings, which consistently improves retrieval and downstream task accuracy compared to mean pooling or using the last <EOS> token embedding from LLMs. To enhance representation learning, we remove the causal attention mask of LLMs during contrastive training. For model training, we introduce a two-stage contrastive instruction-tuning method. It first applies contrastive training with instructions on retrieval datasets, utilizing in-batch negatives and curated hard negative examples. At stage-2, it blends various non-retrieval datasets into instruction tuning, which not only enhances non-retrieval task accuracy but also improves retrieval performance. Combining these techniques, our NV-Embed model, using only publicly available data, has achieved a record-high score of 69.32, ranking No. 1 on the Massive Text Embedding Benchmark (MTEB) (as of May 24, 2024), with 56 tasks, encompassing retrieval, reranking, classification, clustering, and semantic textual similarity tasks. Notably, our model also attains the highest score of 59.36 on 15 retrieval tasks in the MTEB benchmark (also known as BEIR).	*LLM as a versatile embedding model, while maintaining its simplicity and reproducibility. For model architecture, we propose a latent attention layer to obtain pooled embeddings, which consistently improves retrieval and downstream task accuracy compared to mean pooling or using the last <EOS> token embedding from LLMs. To enhance representation learning, we remove the causal attention mask of LLMs during contrastive training. For model training, we introduce a two-stage contrastive instruction-tuning method. It first applies contrastive training with instructions on retrieval datasets, utilizing in-batch negatives and curated hard negative examples. At stage-2, it blends various non-retrieval datasets into instruction tuning, which not only enhances non-retrieval task accuracy but also improves retrieval performance. Combining these techniques, our NV-Embed model, using only publicly available data, has achieved a record-high score of 69.32, ranking No. 1 on the Massive Text Embedding Benchmark (MTEB) (as of May 24, 2024), with 56 tasks, encompassing retrieval, reranking, classification, clustering, and semantic textual similarity tasks. Notably, our model also attains the highest score of 59.36 on 15 retrieval tasks in the MTEB benchmark (also known as BEIR).*

Sebagai ditampilkan dalam Tabel 3, hasil evaluasi menunjukkan bahwa sistem mencapai *precision* sebesar 0.906, *recall* sebesar 0.994, dan *f1-score* sebesar 0.948. Hasil menunjukan kemampuan model untuk mengambil dokumen relevan dengan tingkat kesalahan rendah serta menemukan hampir semua dokumen yang relevan.

Selain itu, hasil *BLEU* sebesar 0.899 menunjukkan bahwa respons yang dihasilkan memiliki kemiripan tinggi dengan teks referensi, baik dari segi struktur maupun konten. Metrik *ROUGE* juga menunjukkan hasil yang sangat baik, dengan *ROUGE-1* sebesar 0.955, *ROUGE-2* sebesar 0.951, dan *ROUGE-L* sebesar 0.955, yang mengindikasikan tingkat overlap yang tinggi antara respons yang dihasilkan dengan referensi, baik untuk unigrams, bigrams, maupun struktur teks keseluruhan.

4.2. Analisis Hasil Evaluasi

Berdasarkan hasil evaluasi, *pipeline* NV-Embed menunjukkan performa yang sangat baik dalam retrieval dan generasi respons. Kombinasi embedding berbasis NV-Embed dan *FAISS* (*Facebook Artificial Intelligence Semantic Search*) indexing memastikan dokumen relevan dapat diidentifikasi secara akurat, sedangkan reranking berbasis cosine similarity meningkatkan kualitas hasil retrieval dengan memprioritaskan dokumen yang benar-benar relevan.

Hasil *BLEU* dan *ROUGE* yang tinggi mencerminkan kemampuan sistem untuk menghasilkan respons yang sesuai dengan *query* dan teks referensi. Model LLM (*GPT-2 XL*) yang digunakan untuk menggabungkan query dengan dokumen relevan menunjukkan hasil yang cukup baik. Namun, terdapat beberapa keterbatasan dalam sistem, seperti batasan jumlah token pada LLM yang hanya berjumlah maksimal 1024 token, sehingga mengharuskan pemotongan teks panjang dan berpotensi kehilangan informasi penting. Selain itu, teknik summarization berbasis *TF-IDF* (*Term Frequency-Inverse Document*) bekerja baik pada dataset ini, tetapi mungkin kurang optimal untuk teks dengan struktur yang kompleks.

5. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Penelitian ini menunjukkan bahwa pipeline NV-Embed mampu memberikan performa unggul dalam retrieval informasi dan generasi respons berbasis teks. Dengan hanya memanfaatkan latent attention, NV-Embed yang telah dibuat berhasil menghasilkan embedding, sementara *FAISS indexing* mendukung proses retrieval yang cepat dan akurat. Evaluasi sistem menunjukkan hasil skor *precision* sebesar 0.906, *recall* sebesar 0.994 dan *f1-score* sebesar 0.948.

Kemudian skor *BLEU* sebesar 0.899 dan *ROUGE-1* sebesar 0.955 menunjukkan bahwa respons yang dihasilkan memiliki kemiripan tinggi dengan teks referensi, baik dari segi struktur maupun konten. Secara keseluruhan, sistem ini mampu menjawab *query* dengan respons yang relevan.

5.2. Saran

Untuk pengembangan lebih lanjut, disarankan untuk menggunakan model LLM yang lebih besar, seperti GPT-3 atau GPT-4 untuk mengatasi batasan jumlah token dan meningkatkan kualitas respons.

Kemudian teknik *abstractive summarization* dapat diimplementasikan untuk menghasilkan ringkasan yang lebih alami dan informatif.

Kemudian dapat membatasi informasi yang akan diambil dan diolah dari dokumen PDF. Optimasi infrastruktur juga perlu dilakukan dengan memanfaatkan perangkat keras yang lebih kuat untuk mempercepat proses pelatihan dan inferensi, terutama ketika menangani dataset besar atau model yang lebih kompleks.

Selain itu, sistem ini dapat dikembangkan lebih lanjut untuk domain khusus, seperti data medis, hukum, atau ilmiah, dengan menambahkan dataset spesifik melalui metode *fine-tune*. Dengan pengembangan ini, pipeline NV-Embed diharapkan dapat memberikan hasil yang lebih baik dan harapannya dapat memberikan kontribusi yang lebih besar dalam bidang NLP berbasis RAG.

6. DAFTAR PUSTAKA

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). *Language Models are Few-Shot Learners*. http://arxiv.org/abs/2005.14165

Devlin, J., Chang, M.-W., Lee, K., Google, K. T., & Language, A. I. (n.d.). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. https://github.com/tensorflow/tensor2tensor

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., & Wang, H. (n.d.). *Retrieval-Augmented Generation for Large Language Models: A Survey*. https://github.com/Tongji-KGLLM/

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., … Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. In *Learning and Individual Differences* (Vol. 103). Elsevier Ltd. https://doi.org/10.1016/j.lindif.2023.102274

Lee, C., Roy, R., Xu, M., Raiman, J., Shoeybi, M.,

Catanzaro, B., & Ping, W. (2024). *NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models*. http://arxiv.org/abs/2405.17428

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. http://arxiv.org/abs/2005.11401

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*. http://arxiv.org/abs/1301.3781

Reimers, N., & Gurevych, I. (2019). *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. http://arxiv.org/abs/1908.10084

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*. http://arxiv.org/abs/1706.03762

Xu, Y., Liu, X., Cao, X., Huang, C., Liu, E., Qian, S., Liu, X., Wu, Y., Dong, F., Qiu, C. W., Qiu, J., Hua, K., Su, W., Wu, J., Xu, H., Han, Y., Fu, C., Yin, Z., Liu, M., … Zhang, J. (2021). Artificial intelligence: A powerful paradigm for scientific research. In *Innovation* (Vol. 2, Issue 4). Cell Press. https://doi.org/10.1016/j.xinn.2021.100179

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., … Wen, J.-R. (2023). *A Survey of Large Language Models*. http://arxiv.org/abs/2303.18223