Energy Distance Investigation to Improve Computational Cost of LLMs

Gokul Natesan
Columbia University
gokul.natesan@columbia.edu

I. Introduction

In the field of natural language processing, retrieval augmented generation (RAG) stands as a pivotal application of generative artificial intelligence powered by large language models (LLMs). Its core aim is to provide relevant content for the generation phase and to mitigate the issue of hallucination commonly observed in large language models. Achieving this involves a two-step process: first, using a query to retrieve relevant passages, and second, generating an appropriate response based on both the query and the retrieved passages. The efficiency of the retrieval step is significant, as the quality of the generated response depends upon the information from the passages.

The prevailing methods for retrieval in RAG are predominantly based on cosine similarity measures, in which the query and passage are encoded into single vectors. While effective, this method overlooks the statistical distribution of the query and passage itself, potentially missing out on valuable information for retrieval. Alternative methods, such as ColBERT [7], utilize multiple vectors for encoding queries and passages, leading to potentially better results but at a higher computational cost. Therefore, we have identified a need to explore the potential of energy distance, a metric adept at measuring the distance between distributions as a novel strategy for enhancing information retrieval in RAG.

The primary objective of this investigation is to implement a novel approach for energy distance in information retrieval that bridges the gap between single-vector and multiple-vector-based methods. Specifically, we aim to compare the performance of energy distance with that of a single-vector-based method, with the goal of demonstrating improvements while maintaining a low computational cost. The scope of this study includes developing and evaluating the proposed energy distance approach within the context of information retrieval tasks, focusing on its effectiveness in retrieving relevant passages for subsequent generation steps.

II. Literature Review

A. Review of Relevant Literature

Recent advancements in information retrieval, exemplified by the ColBERT model introduced by Khattab and Zaharia [7], have shown improvements in retrieval performance. However, these gains often come at the expense of computational cost, rendering some state-of-the-art methods impractical for real-world applications. ColBERT, while highly effective in improving the interaction between queries and passages through contextualized representations from BERT, highlights the ongoing tradeoff between retrieval performance and computational efficiency.

The limitations of existing methods, particularly in terms of computational complexity, underscore the need for exploring alternative distance metrics that can offer competitive retrieval performance while mitigating computational overhead. Energy distance, introduced by Rizzo and Székely [6], has been commonly used in fields like computer vision but is relatively unexplored in information retrieval. One of the key advantages of energy distance lies in its ability to capture the statistical distribution of queries and passages. Unlike conventional approaches that focus solely on the distance between embeddings, energy distance considers the underlying distributional characteristics, potentially providing additional information crucial for retrieval accuracy. Preliminary studies have indicated that energy distance may offer comparable results to multi-vector-based methods while maintaining computational efficiency.

An important limitation of most retrieval methods lies in how they aggregate query tokens for similarity calculations. Many approaches use token-level similarity measures, to compute a single representation of the query before comparing it with the document representation. For example, ColBERT uses a maximum aggregation strategy over token-level similarities, selecting the most relevant token from the query-document interaction. Similarly, methods like DPR (Dense Passage Retrieval) employ an average aggregation of query token embeddings for comparison with document embeddings. However, these methods typically focus on either the query or document embeddings individually and do not consider the union of query and document tokens in their similarity computation.

While these approaches have been effective, they overlook the potential of using the complete set of query tokens and document tokens in conjunction. Few retrieval methods have explored a strategy where the union of query and document tokens is considered in the similarity computation. This approach captures all token-level interactions between the query and document, offering a more detailed and context-sensitive measure of similarity. Although computationally intensive, this method has the potential to uncover subtle relationships between query and document tokens that may be missed by traditional aggregation strategies.

Our work addresses another significant gap in the literature: the use of token embeddings in distance metrics within the Sentence-Transformers framework. Current implementations of Sentence-Transformers rely exclusively on sentence-level embeddings for distance computations, which aggregate token-level information into a single vector [12]. While this simplifies the computation, it loses the granularity of token-level interactions, which are particularly relevant in tasks such as retrieval-augmented generation (RAG). The novelty of our approach lies in integrating token embeddings directly into energy distance computations, providing a more nuanced similarity measure while retaining computational efficiency. To effectively utilize token embeddings, several changes were made to how they are stored and accessed during similarity score computations, ensuring compatibility with the energy distance framework. Furthermore, our approach uniquely integrates token embeddings for queries and sentence-level embeddings for documents, enhancing the adaptability of energy distance computations to retrieval tasks where fine-grained query representations are critical.

III. Methodology

A. Data Collection and Preprocessing

In order to strengthen the experiment, we decided to use BEIR (Benchmarking Information Retrieval) for evaluating our energy distance and cosine similarity trained models [8]. A standard and unified benchmark for testing information retrieval models, BEIR is particularly useful for improving retrieval augmented generation (RAG) systems and evaluating dense retrieval techniques. The BEIR benchmarks includes 18 datasets covering a wide range of retrieval tasks and domains. These datasets are divided into 9 retrieval tasks, making BEIR a comprehensive framework for evaluating the generalization ability of information retrieval models.

For the initial evaluation, we decided to use the HotpotQA BEIR dataset to run our experiment. Since most BEIR datasets are designed only for zero-shot evaluation, it was ideal to choose a dataset which had predefined train, test, and validation splits keeping in mind that we wanted to fine-tune a model on our distance metrics (energy distance and cosine similarity). The HotpotQA dataset satisfies this objective while also providing further benefits listed below making it well-suited for information retrieval tasks.

1) **Multi-Hop Reasoning Requirement:** HotpotQA requires models to retrieve multiple pieces of evidence from different documents to answer a single query. This tests the retrieval system's ability to connect different information across documents.
2) **Open-Domain and Challenging Queries:** The queries are often complex, containing multiple entities or aspects, requiring models to understand relationships between concepts.
3) **Diverse and Noisy Corpus:** HotpotQA includes a mix of relevant and irrelevant documents, simulating real-world noise in retrieval tasks and challenging models to distinguish between them.
4) **Grounded in Real-World Scenarios:** The dataset reflects real-world knowledge-seeking behavior by simulating queries that require reasoning across multiple Wikipedia articles.
5) **Well-Annotated and High-Quality Ground Truth:** It provides relevant documents, distraction documents, and detailed question-answer pairs, ensuring high-quality evaluation of retrieval systems.
6) **Multi-Domain Generalization:** The multi-hop structure ensures models must generalize across different domains of knowledge, making it suitable for evaluating open-domain retrieval models.

B. Model Selection

Selecting an appropriate model for evaluating the retrieval benchmarks was a critical aspect of our experiment. The decision was guided by two key considerations:

1) Hardware resource constraints.
2) Ensuring the model had not been pre-trained on any specific distance metric to avoid bias.

All model training and evaluation were conducted on the Columbia University Terremoto high-performance computing cluster, which provided access to an NVIDIA V100 GPU with 16 GB of memory. Given the limitations of the available hardware, it was essential to choose a model that would not lead to excessively long runtimes or memory-related issues. Consequently, we opted to use a model with fewer than 100 million parameters.

Additionally, to ensure the fairness of our experiment, the selected model needed to be unbiased with respect to distance metrics. Certain Sentence-Transformer models are fine-tuned with specific distance metrics, such as cosine similarity, to optimize their embeddings for retrieval and semantic similarity tasks. However, our research aimed to evaluate whether energy distance could outperform cosine similarity as a distance metric for information retrieval. To facilitate a fair comparison, it was crucial to use a general-purpose pre-trained language model that had not been fine-tuned with any distance metric. Such models are typically trained on foundational objectives like masked language modeling, where the model predicts randomly masked words in a sentence, or next sentence prediction, where the model determines if two sentences follow each other. These pretraining tasks enable the model to learn a rich internal representation of the English language, which can then be leveraged for downstream tasks such as retrieval.

Based on these requirements, we selected the *distilbert-base-uncased* model for our investigation. This model, a distilled version of *bert-base-uncased*, contains approximately 67 million parameters, making it both computationally efficient and well-suited to the constraints of our hardware. Despite its reduced size, the distilled model retains the ability to learn the same underlying representations as its larger teacher model. Furthermore, *distilbert-base-uncased* had not been fine-tuned on any downstream tasks involving distance metrics, ensuring an unbiased starting point for our experiments. These attributes made it a practical and justified choice for our study.

C. Energy Distance Computation

To calculate the energy distance, Rizzo and Székely use the formula

$$D^2(F,G) = 2\mathbb{E}\|X-Y\| - \mathbb{E}\|X-X_0\| - \mathbb{E}\|Y-Y_0\| \geq 0$$

where

- $D^2(F,G)$ represents the energy distance between distributions F and G
- $\|X-Y\|$ denote Euclidean distances between distributions F and G
- $\|X-X_0\|$ and $\|Y-Y_0\|$ denote Euclidean distances within distributions F and G respectively

This calculation of energy distance was used for both inference and training in our experiment. Given the format of our queries and passages there were a few optimization strategies we took advantage of. Specifically, since the passages are encoded as a single vector we know that $\mathbb{E}\|Y-Y_0\| = 0$. Therefore, we can eliminate this computation from the energy distance due to the nature of our implementation.

Since we need to calculate the Euclidean distance within the distribution of query vectors $\|X-X_0\|$, and we have to compare the similarity scores of all possible query-document pairs during inference, we store the the result of $\|X-X_0\|$ to prevent repeated computations. This ensures that $\|X-X_0\|$ only needs to be calculated one time for each query during evaluation.

D. Training

In order to develop a model that would generate embeddings based on energy distance, we first fine-tuned a pretrained Sentence-Transformer model. After further review, it was decided to use a Multiple Negatives Ranking Loss function for training. Multiple Negatives Ranking Loss is well-suited for contrastive learning in retrieval tasks, where the objective is to train an embedding model to to maximize the similarity between a query and its corresponding relevant document while minimizing the similarity between the query and unrelated documents [9]. The Multiple Negatives Ranking Loss works as follows:

- For each anchor (e.g. query a_i), there is one positive pair (e.g. document p_i)
- All other positives in the batch (e.g. p_j for $j \neq i$ are treated as negative pairs for p_i)
- This encourages the model to maximize the similarity between a_i and p_i, while minimizing the similarity between a_i and all p_j (where $j \neq i$)

Essentially, the Multiple Negatives Ranking Loss is a specialized form of cross-entropy loss, where the softmax operation is applied across all candidate pairs within a batch. For each anchor a_i and its positive pair p_i, the loss is computed as:

$$\mathcal{L}_i = -\log \frac{\exp(\text{sim}(a_i, p_i))}{\sum_{j=1}^{n} \exp(\text{sim}(a_i, p_j))}$$

where:

- $\text{sim}(a_i, p_j)$ denotes the similarity score (e.g., cosine similarity or energy distance) between anchor a_i and candidate p_j,
- p_i is the positive sample corresponding to a_i, and
- the denominator, $\sum_{j=1}^{n} \exp(\text{sim}(a_i, p_j))$, sums over all candidates in the batch.

This formulation ensures that positive pairs are assigned high similarity scores while negative pairs are assigned low similarity scores, aligning the embeddings to the desired structure.

When adapting the MNRL to work with energy distance, additional considerations were necessary. Energy distance is a metric that quantifies the discrepancy between the distributions of two sets of vectors. A perfect match between distributions results in an energy distance of 0, while increasingly dissimilar distributions yield larger energy distance values with no upper bound. However, this behavior poses a challenge when used with the MNRL, as the original formulation of energy distance assigns higher values to dissimilar pairs. In contrast, the MNRL is designed to maximize similarity scores for positive pairs, and this mismatch can lead to suboptimal training outcomes.

In fact, during initial experiments, the loss values during training decreased smoothly, yet the evaluation performance of the model deteriorated. This discrepancy was attributed to the misalignment between the energy distance metric and the MNRL's optimization objective. To resolve this issue, we applied a simple yet effective transformation: the sign of the energy distance scores was flipped. By negating the energy distance, higher scores now corresponded to greater similarity, ensuring compatibility with the MNRL's optimization goal. This adjustment aligned the metric with the intended training dynamics, enabling the model to learn embeddings effectively for retrieval tasks.

To represent queries as multiple vectors, it was necessary to utilize the token embeddings of the queries. During training, both queries and documents were tokenized and passed as inputs into the Sentence-Transformer model. The output of the transformer's forward pass was stored in a dictionary containing the following components:

1) **sentence_embedding:** The final sentence embedding after applying a pooling strategy (mean pooling, max pooling, or CLS pooling). Shape of *[batch_size x embedding_dimension]*
2) **token embeddings:** The raw output of the transformer for each token in the input sequence. Shape of *[batch size x max_sequence_length x embedding_dimension]* where max_sequence_length is the maximum sequence length in the batch of queries
3) **attention mask:** A binary mask indicating which tokens are actual input tokens (1) and which are padding (0). Shape of *batch_size x max_sequence_length*

The output format provided by the Sentence-Transformer model allowed direct access to the token embeddings for queries, which were used in conjunction with sentence embeddings for documents to compute similarity scores. These similarity scores were then used to minimize the cross-entropy loss between the predicted outputs and the target labels. It is important to emphasize that these modifications were specifically applied when fine-tuning the model using energy distance as the similarity metric. In contrast, for models trained using cosine similarity, sentence embeddings were used for both queries and documents. As a result, no changes to the existing Sentence-Transformers framework were necessary for the cosine similarity approach.

When training a model on the two distance metrics, our initial focus was on optimizing the learning rate and the number of epochs. The preliminary training experiments tuned these hyperparameters while keeping other parameters fixed, based on the default configuration provided by the Sentence-Transformers framework. Table I lists the hyperparameters that remained constant for both the energy distance and cosine similarity fine-tuned models.

TABLE I
FIXED HYPERPARAMETERS USED FOR MODEL TRAINING

Hyperparameter	**Value**
Train Batch Size	16
Max Sequence Length	128
Optimizer	Adam
Epsilon (ϵ)	1×10^{-8}
Weight Decay	0.01
Warmup Steps	0.1
Max Gradient Norm	1.0
Evaluation Steps	2500
Save Best Model	True
Use AMP	True

Across all experiments, the models were trained with a batch size of 16 and a maximum sequence length of 128 tokens. The Adam optimizer was used with a weight decay of 0.01, and the learning rate warmup was set to 10% of the total training steps. Gradient clipping was applied at 1.0 to prevent exploding gradients.

For the first set of training experiments, the learning rate and the number of epochs were systematically varied. Specifically, learning rates of 1×10^{-5}, 2×10^{-5}, and 3×10^{-5} were tested, while the number of epochs was set to 1, 3, and 5. A validation set, consisting of 20% of the HotpotQA development set queries (1089 queries and 2160 relevant documents), was used for evaluation. Only relevant documents were included in this validation set, with irrelevant documents excluded.

Model evaluation was conducted every 2500 steps, totaling 10625 steps for the entire training process. Additionally, evaluation runs were performed at the end of each epoch, resulting in approximately five evaluation runs per epoch. The best-performing model checkpoint was saved based on validation performance. The Sentence-Transformers *InformationRetrievalEvaluator* was modified to use the NDCG@10 metric instead of MAP@100 for validation, aligning the evaluation more closely with the objectives of this task.

Due to the initial implementation of the energy distance metric being unoptimized for runtime performance, validation evaluations were limited to a subset of the dataset. Despite these constraints, the results from the first training run indicated that the cosine similarity model slightly outperformed the energy distance model in terms of NDCG@10.

TABLE II
TRAINING RUN 1: ENERGY DISTANCE SIMILARITY MODEL VALIDATION RESULTS (NDCG@10)

Epoch / LR	**NDCG@10**
Epoch 1, LR 1e-5	0.8826
Epoch 3, LR 1e-5	0.8981
Epoch 5, LR 1e-5	0.9076
Epoch 1, LR 2e-5	0.8919
Epoch 3, LR 2e-5	0.9055
Epoch 5, LR 2e-5	0.9090
Epoch 1, LR 3e-5	0.8928
Epoch 3, LR 3e-5	0.9083
Epoch 5, LR 3e-5	**0.9122**

TABLE III
TRAINING RUN 1: COSINE SIMILARITY MODEL VALIDATION RESULTS (NDCG@10)

Epoch / LR	**NDCG@10**
Epoch 1, LR 1e-5	0.8960
Epoch 3, LR 1e-5	0.9091
Epoch 5, LR 1e-5	0.9131
Epoch 1, LR 2e-5	0.8967
Epoch 3, LR 2e-5	0.9129
Epoch 5, LR 2e-5	**0.9173**
Epoch 1, LR 3e-5	0.8985
Epoch 3, LR 3e-5	0.9138
Epoch 5, LR 3e-5	0.9170

Analysis of the validation results revealed a consistent improvement in performance as the number of epochs increased from 1 to 3 to 5, across both the energy distance and cosine similarity models. This trend suggested that extending training to 10 epochs might further enhance model performance. For subsequent runs, we adopted this approach, ensuring the model checkpoint with the highest validation score was saved.

While refining the training process, we also identified the importance of optimizing the scale hyperparameter. Commonly referred to as the inverse of the temperature, the scale hyperparameter in *MultipleNegativesRankingLoss* acts as a multiplier on similarity scores before applying the cross-entropy loss. Its role is critical in shaping the model's training dynamics. Larger scale values amplify the distinction between positive and negative examples, effectively increasing the classification margin. This results in more confidently skewed softmax probabilities, where positive examples are closer to 1 and negatives are closer to 0. Consequently, a larger scale encourages the model to produce highly distinct similarity scores, aiding in better classification of positive and negative pairs. However, overly large scale values may lead to overfitting, as the model could become excessively confident in its predictions. Conversely, smaller scale values reduce this distinction, potentially causing the model to treat positive and negative pairs more similarly. This can lead to slower convergence but may be beneficial for preventing overfitting in certain scenarios. The optimal scale depends on the dataset and the range of similarity scores. Adjusting the scale appropriately can significantly improve model performance, as supported by prior research [11].

In summary, the scale hyperparameter influences how the cross-entropy loss penalizes positive and negative pairs, thereby impacting the model's gradient updates and overall training behavior. To investigate its effects, we conducted further training experiments, testing scale values of 20, 50, 100, and 200. These experiments were performed while keeping the learning rates at 1×10^{-5}, 2×10^{-5}, and 3×10^{-5}. The results of these runs are presented below.

The results from Training Run 2 highlight the critical role of the scale hyperparameter in improving the validation performance of both the energy distance and cosine similarity models. For the energy distance model, the best validation NDCG@10 score of 0.9297 was achieved with a scale of 200 and a learning rate of 1×10^{-5}. This result suggests that higher scale values enhance the model's ability to distinguish between positive and negative pairs, leading to better retrieval performance. Similarly, the cosine similarity model attained its highest NDCG@10 score of 0.9329 at a scale of 200 and a learning rate of 2×10^{-5}. This observation reinforces the idea that tuning the scale parameter is crucial for optimizing retrieval quality across different similarity metrics.

While the cosine similarity model outperformed the energy distance model by approximately 0.003 on this validation set, the performance gap has narrowed significantly compared to previous experiments. This narrowing indicates that proper tuning of the scale hyperparameter can mitigate the performance differences between the two approaches. Overall, the findings underscore the importance of scale optimization in retrieval tasks, as it directly impacts the model's ability to assign appropriate similarity scores to document pairs. For future experiments, scale tuning will remain a key area of focus to further enhance model performance.

TABLE IV
TRAINING RUN 2: ENERGY DISTANCE MODEL VALIDATION RESULTS (NDCG@10)

Epoch / LR / Scale	NDCG@10
Epoch 10, LR 1e-5, Scale 20	0.9293
Epoch 10, LR 1e-5, Scale 50	0.9263
Epoch 10, LR 1e-5, Scale 100	0.9279
Epoch 10, LR 1e-5, Scale 200	**0.9297**
Epoch 10, LR 2e-5, Scale 20	0.9254
Epoch 10, LR 2e-5, Scale 50	0.9246
Epoch 10, LR 2e-5, Scale 100	0.9274
Epoch 10, LR 2e-5, Scale 200	0.9267
Epoch 10, LR 3e-5, Scale 20	0.9249
Epoch 10, LR 3e-5, Scale 50	0.9230
Epoch 10, LR 3e-5, Scale 100	0.9195
Epoch 10, LR 3e-5, Scale 200	0.9259

TABLE V
TRAINING RUN 2: COSINE SIMILARITY MODEL VALIDATION RESULTS (NDCG@10)

Epoch / LR / Scale	NDCG@10
Epoch 10, LR 1e-5, Scale 20 (Default)	0.9177
Epoch 10, LR 1e-5, Scale 50	0.9281
Epoch 10, LR 1e-5, Scale 100	0.9275
Epoch 10, LR 1e-5, Scale 200	0.9281
Epoch 10, LR 2e-5, Scale 20 (Default)	0.9219
Epoch 10, LR 2e-5, Scale 50	0.9316
Epoch 10, LR 2e-5, Scale 100	0.9286
Epoch 10, LR 2e-5, Scale 200	**0.9329**
Epoch 10, LR 3e-5, Scale 20 (Default)	0.9203
Epoch 10, LR 3e-5, Scale 50	0.9252
Epoch 10, LR 3e-5, Scale 100	0.9234
Epoch 10, LR 3e-5, Scale 200	0.9248

E. Evaluation

Once a Sentence-Transformer model was trained, it was evaluated on the Massive Text Embedding Benchmark (MTEB), a benchmark suite designed to evaluate the performance of large-scale text embedding models. MTEB provides a standardized framework for comparing different text embedding methods across a variety of tasks and datasets. The retrieval evaluation process in MTEB is designed to assess the quality of embeddings generated by models in tasks like information retrieval. The dataset (HotpotQA) consisted of queries and documents where each query was associated with one or more relevant documents. Both queries and documents were encoded into dense vector representations using the embedding model being evaluated. Similar to the training process, token embeddings were generated for each query while sentence embeddings were generated for each document. For each query, similarity scores were calculated between the query embedding and all document embeddings. Before model inference could be ran against the benchmarks, energy distance was added as a similarity function to the MTEB Retrieval Evaluator in order for computation between all possible pairs of queries and documents. After the distance metric computation, documents were ranked for each query based on their similarity scores.

Since computing and ranking similarity scores for every query-document pair can be computationally expensive, especially when the number of documents is very large, the MTEB Retrieval Evaluator uses a heap data structure to efficiently manage and rank the top results for each query. As similarity scores are computed, a min-heap of size k is maintained for each query to keep track of the k largest scores (top-k documents). Therefore, we needed to ensure that higher similarity corresponded to higher scores, because a min-heap removes the smallest values at the top. Understanding the use of the min-heap in the retrieval evaluation process was imperative because it explained why initial evaluation runs using energy distance gave us abnormally low performance scores. Our solution was the same as what was implemented in model training. We flipped the sign of the energy distance calculation so that higher similarity corresponded to higher scores.

The document rankings were evaluated using metrics like Normalized Discounted Cumulative Gain (NDCG@k), Recall@k, and Precision@k. These metrics indicate how well the model retrieves relevant documents. Specifically for our experiment, we chose to use NDCG@k as our performance metric. Widely regarded as the standard metric for retrieval evaluation, NDCG@k evaluates the quality of ranking by considering both the relevance of documents and their positions in the ranked list [10]. Additionally, benchmarks like BEIR and MTEB often report NDCG@K as the primary metric, with $k = 10$.

For the initial inference runs, we decided to use a subset of the HotpotQA test data when evaluating our trained models. Due to the existing format of token embeddings generated by the Sentence-Transformer, parallel computations could not be performed. Thus, our preliminary energy distance implementation would be later revised so that query-document pairs can be processed simultaneously. We can point to three evaluation runs to illustrate the improvement of our trained models and performance optimizations for fully-scaled inference.

1) **Run 1:** Evaluating a trained Sentence-Transformer model on a subset of the HotpotQA test data with the correct retrieval evaluation set up for energy distance.
2) **Run 2:** Evaluating a trained Sentence-Transformer model on a subset of the HotpotQA test data with optimization of the scale hyperparameter.
3) **Run 3:** Evaluating a trained Sentence-Transformer model on the entire subset of the HotpotQA test data with optimization of the scale hyperparameter.

F. Performance Optimizations

As one of the objectives of our study was to implement the use of energy distance for information retrieval in an efficient manner, we put a lot of effort into optimizing our energy distance calculation. Our first version of energy distance was built off the Sentence-Transformers existing framework for handling token embeddings. In this format, the Sentence-Transformer *encode* function returns a list of two-dimensional tensors, which in our case corresponds to the entire set of query embeddings. Since the shape of a query embedding is *[sequence_length* $\times$ *embedding_dimension]*, the dimensions of the query embeddings are individually different as queries do not all have the same number of tokens. This prevents the query embeddings from being placed on a single tensor, thereby not allowing for parallelized computations and efficient processing. Therefore, in our initial energy distance implementation, we iterated through queries and documents to calculate similarity scores. This method was inefficient, as we could only run evaluation on small subsets of the test and validation data.

To improve the computational efficiency of our energy distance calculation, the first step was to change the manner in which token embeddings were stored. Since the current structure of using a list of two-dimensional tensors forced our distance metric to be computed using iterative loops, we recognized that the multi-vector query embeddings should be placed in a tensor. Our strategy was to alter the Sentence-Transformers *encode* function so that when generating token embeddings for queries, padded embeddings would be added. Incorporating padding ensured that the sequence length for each query embedding would correspond to the maximum sequence length of the entire query set. Subsequently, query embeddings could be stored in a single tensor whose shape is *[num_queries* $\times$ *max_sequence_length* $\times$ *embedding_dimension].*

Now that padded embeddings were added to the queries, we needed to ensure that this did not impact the calculated energy distance values. Looking at our version of energy distance, the pairwise difference between each query vector and the single document vector is taken before the Euclidean distances are calculated and then averaged. For that reason, we decided to use an attention mask to zero out the pairwise differences that involve a padded query vector. As already noted, the attention mask is stored after the queries are sent through the forward pass of the transformer model. When producing token embeddings, the Sentence-Transformer traditionally utilizes the attention mask to remove the padding from the embeddings in order to return the exact embedding generated by the model. Thus, our *encode* function was modified to return separate tensors for the token embeddings (including padding) and the attention masks. Our next iteration of the energy distance calculation accepted the query embeddings, document embeddings, and attention masks as inputs while properly using the mask to zero out the effects of adding padded embeddings. To deal with having zero values under a square root operation in our energy distance calculation, we made sure to use the PyTorch *torch.norm* function. This prevented NaN values from occurring when computing gradients during training.

To further optimize computations, we modified the energy distance implementation to leverage tensor broadcasting. With queries, documents, and attention masks stored in tensors, broadcasting allowed us to efficiently compute pairwise differences across all token embeddings. Tensor broadcasting enables parallelized computations by automatically expanding smaller tensors to match the dimensions of larger ones without creating additional memory overhead. During training, we ensured that the expanded tensors were placed on the computation graph so that gradients could flow through them in the backward pass. This adjustment allowed for significant speedup in similarity score computations.

However, during evaluation, computing energy distance for all queries and documents at once posed GPU memory challenges. For example, storing multi-vector query embeddings for 5,447 queries with a maximum sequence length of 512 and embedding dimension of 768 required over 12 GB of GPU memory. This was infeasible given the additional memory demands for storing document embeddings and performing pairwise computations. To address this, we adopted a strategy of iterating through query batches while keeping the corpus embeddings on the CPU and loading them onto the GPU only when needed for evaluation. The *encode* function was modified to return a list of three-dimensional tensors for query batches and corpus chunks, allowing us to iterate through the query batches efficiently and reduce GPU memory usage.

Even with query batching, tensor broadcasting during pairwise difference computations caused memory bottlenecks. For example, using a batch size of 32 queries and a corpus chunk of 50,000 documents resulted in a tensor of shape *[32 $\times$ 50000 $\times$ max_sequence_length $\times$ embedding_dimension]*, which was too large to fit into GPU memory. To resolve this, we employed an efficient computation using the PyTorch *torch.einsum()* function. This function allowed us to perform parallelized computations for pairwise differences while significantly reducing the memory footprint. By using *torch.einsum()*, we avoided the need to expand tensors explicitly, enabling efficient computation of energy distance values without exceeding GPU memory limits.

To summarize, the following performance optimizations were made to the energy distance implementation and *encode* function:

- Changed the storage of token embeddings by padding query embeddings and storing them in a single tensor of shape *[num_queries $\times$ max_sequence_length $\times$ embedding_dimension]*.
- Added attention masks to zero out pairwise differences involving padded embeddings.
- Modified the *encode* function to return token embeddings with padding and attention masks as separate tensors.
- Utilized tensor broadcasting during training to enable parallelized computations while placing expanded tensors on the computation graph.
- Iterated through query batches during evaluation, keeping corpus embeddings on the CPU and loading them onto the GPU as needed.
- Replaced tensor broadcasting for pairwise differences with *torch.einsum()* to reduce memory usage while retaining computational efficiency.

G. Experimental Results

For the first two inference runs, the HotpotQA test subset was selected to ensure consistency and fairness in evaluation. The subset included all the queries (7,405 queries) and their corresponding relevant documents. To ensure a total of 50,000 documents, irrelevant documents were selected deterministically. Specifically, all irrelevant document IDs were sorted in ascending order, and the first N irrelevant documents were chosen to fill the remaining slots, where $N = 50,000 -$ *len(relevant documents)*. This deterministic selection ensures reproducibility across evaluation runs.

The table below presents the performance of the best models for both the energy distance and cosine similarity approaches from the first training run, where detailed validation results are provided in Tables 2 and 3. The evaluation metrics include NDCG@1, NDCG@5, and NDCG@10.

TABLE VI
INITIAL INFERENCE RESULTS ON HOTPOTQA TEST SUBSET

Model	**NDCG@1**	**NDCG@5**	**NDCG@10**
Energy Distance Model (Epoch 5, LR 3e-5)	0.82593	0.74766	0.77100
Cosine Similarity Model (Epoch 5, LR 2e-5)	0.83714	0.76573	0.78821

From the results, it can be observed that the cosine similarity model outperforms the energy distance model across all metrics (NDCG@1, NDCG@5, and NDCG@10). However, the energy distance model demonstrates competitive performance.

To further improve model performance, we extended the training to 10 epochs and optimized the scale hyperparameter during training. Table VII presents the inference results of the best-performing models for both the energy distance (ED) and cosine similarity metrics.

Tuning the scale hyperparameter allows the model to better balance the relative importance of embeddings during similarity calculation, leading to improved retrieval quality. Interestingly, the ED model showed better performance in certain metrics compared to the previous evaluation, with a smaller gap between ED and cosine similarity for NDCG@10. However, ED evaluation remains significantly slower due to its reliance on token embeddings, which involve more complex similarity calculations compared to the single-vector approach of cosine similarity.

TABLE VII
INFERENCE RESULTS WITH OPTIMIZED SCALE HYPERPARAMETER

Model	**NDCG@1**	**NDCG@5**	**NDCG@10**	**Eval. Time (s)**
Energy Distance Model (Epoch 10, LR 1e-5, Scale 200)	0.85591	0.78146	0.80377	46913.08
Cosine Similarity Model (Epoch 10, LR 2e-5, Scale 200)	0.86833	0.78505	0.80683	69.80
Energy Distance Model (Epoch 10, LR 2e-5, Scale 100)	0.87306	0.79061	0.81002	41587.92
Cosine Similarity Model (Epoch 10, LR 2e-5, Scale 100)	0.86212	0.78589	0.80751	70.99

From Table VII, we observe that:

- The ED model (Epoch 10, LR 2e-5, Scale 100) achieved the highest scores for NDCG@1, NDCG@5, and NDCG@10, outperforming the cosine similarity model on the same subset.
- Despite improvements, the evaluation time for the ED model remains significantly higher than that of the cosine similarity model. This is due to the ED model's use of token embeddings and more computationally intensive similarity calculations.
- Optimizing the scale hyperparameter helped improve retrieval performance, as seen in the higher NDCG scores compared to the previous evaluation.

It is crucial to evaluate models on the entire test set to fully understand their performance in a realistic setting. After making a few optimizations to our MTEB evaluator, we were able to run inference on the entire test set. The results of this comprehensive evaluation are shown in Table VIII.

TABLE VIII
INFERENCE RESULTS ON FULL HOTPOTQA TEST SET

Model	**NDCG@1**	**NDCG@5**	**NDCG@10**	**Eval. Time (s)**
Energy Distance Model (Epoch 10, LR 1e-5, Scale 200)	0.52546	0.46390	0.49259	5899.00
Cosine Similarity Model (Epoch 10, LR 2e-5, Scale 200)	0.54679	0.47753	0.50690	5183.98

From Table VIII, we observe the following:

- The performance of the energy distance model and the cosine similarity model is very close, with a small gap across all NDCG metrics.
- The runtime efficiency of the energy distance implementation is promising, with only a *13.8% slowdown* compared to cosine similarity.
- The experiment was conducted using a subset of the validation set, which included only 20% of the total queries and their corresponding relevant documents. While this approach provided faster evaluation during training, it may have limited the model's ability to generalize optimally. Using the full validation set could offer a more comprehensive assessment and potentially lead to better model selection, as it would capture a broader range of examples and scenarios.

CONCLUSION

The primary objective of this research was to explore innovative approaches for improving the retrieval step in Retrieval-Augmented Generation (RAG). Specifically, we investigated whether the use of multivector token embeddings for queries, combined with single-vector sentence embeddings for documents, could enable a novel similarity function based on energy distance. This approach aims to measure the distance between distributions rather than relying solely on point-based similarity metrics like cosine similarity, representing the key novelty of our work.

The experimental results demonstrated that the performance of the energy distance (ED) model was remarkably close to that of the cosine similarity model across multiple metrics, including NDCG@1, NDCG@5, and NDCG@10. While cosine similarity maintained a slight advantage in retrieval quality, the inference run times for the energy distance implementation were impressive given its reliance on token embeddings. This inevitably increases computational complexity compared to the single-vector operations of cosine similarity, yet the slowdown was kept within a reasonable range, demonstrating the potential scalability of energy distance for real-world retrieval tasks.

Despite the promising results, there remains substantial room for future work to comprehensively evaluate and compare the two distance metrics. In particular, the current experiments used only a subset of the validation set during training, which may have limited the model's ability to generalize effectively. Future research could incorporate the full validation set for a more robust evaluation. Additionally, training and evaluating models across multiple datasets and retrieval tasks could provide deeper insights into the contexts where energy distance might outperform cosine similarity. Certain datasets or tasks may better leverage the multivector representation and distributional comparison offered by energy distance, suggesting exciting directions for further exploration.

Acknowledgment

We would like to extend thanks for mentorship from members of IBM and Columbia - Yulong Li, John Kender, Parijat Dube, and Bishwaranjan Bhattacharjee.

References

[1] https://github.com/gkermit/energy-distance
[2] https://github.com/gnatesan/Sentence-Transformers-energydistance
[3] https://github.com/gnatesan/mteb-evaluator
[4] https://huggingface.co
[5] https://www.sbert.net
[6] https://pages.stat.wisc.edu/ wahba/stat860public/pdf4/Energy/EnergyDistance10.1002-wics.1375.pdf
[7] https://arxiv.org/abs/2004.12832
[8] https://huggingface.co/papers/2104.08663
[9] https://arxiv.org/pdf/1705.00652
[10] https://arxiv.org/pdf/1304.6480
[11] https://openreview.net/forum?id=LBA2Jj5Gqn
[12] https://sbert.net/docs/sentence_transformer/usage/semantic_textual_similarity.html