Linq-Embed-Mistral Report

Junseong Kim, Seolhwa Lee, Jihoon Kwon, Sangmo Gu,
Yejin Kim, Minkyung Cho, Jy-yong Sohn, Chanyeol Choi

Linq

embedding@getlinq.com

Abstract

This report explores the enhancement of text retrieval performance using advanced data refinement techniques. We develop Linq-Embed-Mistral[1] by building on the E5-mistral and Mistral-7B-v0.1 models, focusing on sophisticated data crafting, data filtering, and negative mining methods, which are highly tailored to each task, applied to both existing benchmark dataset and highly tailored synthetic dataset generated via large language models (LLMs). Linq-Embed-Mistral excels in the MTEB benchmarks (as of May 29, 2024), achieving an average score of 68.2 across 56 datasets, and ranks 1st among all models for retrieval tasks on the MTEB leaderboard with a performance score of 60.2. This performance underscores its superior capability in enhancing search precision and reliability. Our contributions include advanced data refinement methods that significantly improve model performance on benchmark and synthetic datasets, techniques for homogeneous task ordering and mixed task fine-tuning to enhance model generalization and stability, and a streamlined evaluation process using 4-bit precision and a light retrieval evaluation set, which accelerates validation without sacrificing accuracy.

1 Introduction

In recent years, the convergence of large language models (LLMs) and information retrieval (IR) has garnered significant attention [1]. Especially, effective text retrieval is pivotal in integrating LLMs and IR systems as it greatly improves the system's capacity. Enhancing the text retrieval aspect is also crucial for frameworks like Retrieval-Augmented Generation (RAG), which incorporate current external information to overcome the static nature of LLMs, thus delivering dependable and dynamic answers to user inquiries [2]. This report explores extensive experiments focused on **improving text retrieval using advanced data refinement methods**, including sophisticated data crafting, data filtering, and negative mining techniques. These methods are applied to both (1) existing benchmark dataset, and (2) highly tailored synthetic dataset generated via LLMs. Recent studies highlight the efficacy of LLMs in generating synthetic data, primarily for enhancing human-labeled datasets or improving performance [3–7]. This motivates us to investigate a critical question:

- Can we rely on LLM-generated data to improve retrieval performances? If not, how can we enhance its quality for this specific task?

We employ advanced methods such as data crafting with extensive prompt engineering, data filtering, and negative mining guided by teacher models, which are highly tailored to each task, to improve the quality of the synthetic data generated by LLM. Our efforts aim to create high-quality triplet datasets (query, positive example, negative example), significantly improving text retrieval performances.

[1] https://huggingface.co/Linq-AI-Research/Linq-Embed-Mistral

1.1 Research Highlights

Similar to the SFR [8], our Linq-Embed-Mistral represents a notable progression in text-embedding models, leveraging the robust bases of E5-Mistral [5] and Mistral-7B-v0.1 [9].

The key experimental points are:

- Linq-Embed-Mistral performs well in the MTEB benchmarks, with an average score of 68.2 across 56 datasets. This places it 1st among publicly accessible models listed on the MTEB leaderboard and 3rd overall.
- The model shows a significant enhancement in the retrieval performance, ranking 1st among all models listed on the MTEB leaderboard with a performance score of 60.2.
  - Within the Mistral Model Series, a suite of models based on the foundational Mistral architecture, SFR enhances E5-Mistral by adding a specially curated dataset of MTEB tasks. In contrast, our approach focuses solely on **creating and integrating more sophisticated synthetic datasets**. This has increased our model's score from 56.9 for E5-Mistral and 59.0 for SFR to an 60.2.

Our contribution points are as follows:

1. Our proposed Data Refinement Methods, which include sophisticated data crafting, filtering, and negative mining, significantly enhance the model's ability to identify misleading documents. By improving the quality of the benchmark dataset and addressing issues in the synthetic dataset generated by GPT-4, these methods ensure more accurate and reliable results.
2. We propose Homogeneous Task Ordering and Mixed Task Fine-tuning, which enhance the model performance by promoting better generalization and training stability, especially when mixed task fine-tuning is limited to within 20 steps. Here, homogeneous task ordering provides precise insights into task ordering effects, whereas Mixed Task Fine-tuning mitigates the catastrophic forgetting.
3. We design Streamlined Evaluation, which uses 4-bit precision and a light retrieval evaluation set. This speeds up the process of validation, where our streamlined evaluation has negligible performance differences, compared with the full-scale evaluation. Our design allows a single GPU to evaluate one checkpoint in approximately 5 hours, with retrieval tasks specifically taking around 4 hours.

2 Backgrounds & Motivation

2.1 Learning Embeddings: Transition from Multi-Stage to Single-Stage Training, Integration of Synthetic Data

The research on learning good embeddings has undergone a significant evolution, shifting from multi-stage to single-stage training methodologies, and from reliance on human-labeled data to the integration of synthetic data. Traditionally, methods including Contriever [10], OpenAI Embeddings [11], E5 [12], BGE [13], and Gecko [7] have adopted a multi-stage training paradigm to mitigate the limitations of labeled data in terms of task diversity and language coverage. These approaches involve (1) initial pre-training phase on large-scale, weakly-supervised text pairs using contrastive loss, followed by (2) fine-tuning on smaller, high-quality datasets to enhance the performance.

Recent advancements, however, have seen a move towards single-stage training approaches in models like E5-Mistral [5] and GritLM [6]. E5-Mistral, in particular, has demonstrated that extensive auto-regressive pre-training enables large language models (LLMs) to acquire robust text representations with minimal fine-tuning needed to transform them into effective embedding models. This finding suggests that contrastive pre-training has a negligible impact on the model quality, indicating that long periods of such pre-training are no longer necessary. This shift highlights a significant trend in the research on embeddings, emphasizing the growing efficacy of synthetic data and streamlined training processes.

Models including E5-Mistral/GritLM, and Gecko generate and utilize well-curated synthetic data for training. They define multiple task categories and use these to generate new query-positive-negative

Table 1: Categorized Issue Types for Each Task

Task	Issue Types
Short-Long (Retrieval)	- Word Length Control - Self-Explanation Issues & False Positives and False Negatives - Diversity and Content Moderation Issues - Refutation Issues - Few-Shot Generation Problems
Long-Short (Classification)	- GPT-4 overfitting problem - Output label diversity
Short-Short (Matching)	- Content Repetition in Hard Negatives - Lack of Entity Information - Insufficient Colloquial Language
Long-Long (Matching)	- Content Repetition in Hard Negatives - Overly Difficult Hard Negatives - Few-Shot Generation Problems
STS (Semantic Textual Similarity)	- Duplication and Diversity Issues - False Positives and False Negatives
bitext (Multilingual Translation)	- False negatives

triplets. The former group generates all components of the query-positive-negative triplet, while the latter, exemplified by Gecko, generates only the query and task description, subsequently performing LLM-based positive and negative mining to complete the triplet.

2.2 The Importance of Data Quality in Training Embedding Models

The significance of data quality in training embedding models has been extensively emphasized.

SFR [8] : This research demonstrates that eliminating false negatives, the number of hard negatives, and the impact of hard negative mining significantly affect model performance. It shows that the quality of hard negatives greatly influences the performance, indicating that high-quality hard negatives are crucial for achieving optimal results.

Gecko [7] : Gecko introduces an LLM-based positive and negative mining method, leveraging large language models (LLMs) to identify more relevant positive passages and suitable hard negatives for generated queries. For a given query and task description, candidate passages are extracted from a corpus pool using a pre-trained embedding model. These candidates are then ranked by an LLM, with the top-ranked passage selected as the positive and the 20th neighbor as the hard negative. Although queries are generated from given passages, this method can change the global positive in approximately 15% of cases. The performance varies significantly based on the quality of the selected positives and hard negatives, underscoring the importance of these elements.

GritLM [6] : In comparisons of training datasets, GritLM highlights that models trained with E5 (synthetic data) outperform those trained with MEDI and MEDI2 (which have better negatives than MEDI) by a wide margin. This superior performance is attributed to the high quality of hard negatives and the diversity of tasks generated by GPT-4 in the E5 dataset. An inspection of samples supports this conclusion, emphasizing the critical role of data quality in the training process.

2.3 Our Observation: Issues in Data Generation Using GPT-4-Turbo and E5-Mistral's Synthetic Data Strategy

Using GPT-4-Turbo (gpt-4-turbo-2024-04-09), we generated data by closely adhering to the synthetic data strategy employed by E5-Mistral [5]. This strategy includes six tasks: short-long (retrieval), long-short (classification), long-long (matching), short-short (matching), STS (semantic textual similarity), and bitext (translation). For each of these tasks, we categorized the issue types as outlined in Table 1 and identified several data quality issues, which are detailed in Table 2, 3, 4, 5, 6, 7.

Table 2: Description of issue types in the STS (Semantic Textual Similarity) task.

STS (Semantic Textual Similarity)

- Duplication
  - The tendency to generate identical sentences necessitates deduplication.
- Topic Diversity
  - A high number of similar queries are generated (e.g., quantum computing, data analysis, parks, dogs and cats, "The quick brown fox...").
- False Positives
  - Some generated positives are of lower quality than the generated hard negatives.
- False Negatives
  - In score-based generation for STS tasks, false negatives can occur when the positive and negative examples differ by only 0.5 points in their relatedness scores.
- Limited Negative Patterns
  - General statements: Some negative examples use abstract level of terminologies, e.g., hypernym or holonym. (e.g., “Multivariable calculus is pivotal in understanding dynamic economic models.” vs. “Calculus is used in some basic economic calculations.”)
  - Counterfactual statements: Statements that swap subjects and objects (e.g., "He likes ball" vs. "She likes ball").
  - Lack of Negation: GPT-generated data rarely includes negation patterns (e.g., "I like you" vs. "I don’t like you").

Table 3: Description of issue types in the Long-Short (Classification) task.

Long-Short (Classification)

- GPT-4 Overfitting Problem
  - GPT-4-turbo tends to generate specific words with high probability when certain keywords are included, despite using random seed and temperature settings of 1.0.
- Output Label Diversity
  - The typical label types and relationships used for evaluation might not be sufficiently represented in the generated data due to the variety of tasks, which can range from very detailed to very broad.

Table 4: Description of issue types in the Short-Long (Retrieval) task.

Short-Long (Retrieval)

- Inconsistent Word Length
  - GPT-4-turbo struggles with controlling the word length, particularly for non-English languages.
- Self-Explanation Issues
  - Rationale Inclusion: The rationale on the positiveness/negativeness of the GPT-generated passages is added to the generated positive/negative passages.
  - Query Inclusion: Queries are sometimes included verbatim in the passages.
- False Positives and False Negatives
  - Quality of Hard Negatives: Although the generated hard negatives are less relevant to the given query than the positives, some hard negatives generated by the prompts still provide general or partially answerable responses.
  - Quality of Given Positives: In some cases, the given positive passage is ambiguous, and a more relevant positive sample exists within the passage pool. This causes issues during training because the model may incorrectly treat the more relevant passage as a negative, while using the ambiguous passage as a positive.
  - False Negatives: Some GPT-generated hard negatives are actually false negatives, being semantically similar to the positives but incorrectly classified as negatives.
  - Higher Incidence in Non-English Languages: False positives and false negatives occur more frequently in non-English languages.
- Lack of Representation
  - Synthetic data lacks representation on topics such as same-sex marriage, feminism, and abortion.
- Safety Concerns
  - Socially sensitive topics such as climate change and gun ownership are affected by content moderation, restricting discussion or representation.
- Incorrect Refutations as False Positives
  - When generating refutations, the model sometimes produces supportive results instead, resulting in false positives.

Table 5: Description of issue types in the Short-Short (Matching) task.

Short-Short (Matching)

- Content Repetition in Hard Negatives
  - Some hard negatives repeat the content of the positives verbatim, making it challenging to classify them as true negatives.
- Lack of Entity Information
  - The generated data lacks sufficient information about entities such as personal names, place names, movie titles, and game titles.
- Insufficient Colloquial Language
  - There is a lack of colloquial language in the generated data, which is necessary for creating more realistic and relatable passages.

Table 6: Description of issue types in the Long-Long (Matching) task.

Long-Long (Matching)

- Content Repetition in Hard Negatives
  - Some hard negatives repeat the content of the positives verbatim, making it challenging to classify them as true negatives.
- Overly Challenging Hard Negatives
  - The generated hard negatives are often too difficult, leading to a lack of clearly distinguishable negatives.

Table 7: Description of issue types in the Bitext (Multilingual Translation) task.

Bitext (Multilingual Translation)

- High Incidence of False Negatives
  - A significant number of false negatives are produced, regardless of score differences (between positives and negatives) as defined in the E5-Mistral prompt. To be specific, many negatives have the same meaning as the positives, with only slight difference in wording. This is because, in the multilingual settings, the generated negative is often just another translated version of the query. This issue is pronounced in multi-lingual settings, where hard negatives in translation tasks can inadvertently become positives.

3 Training Dataset

Recall that our focus is to improve the text retrieval using advanced data refinement methods, to both (1) existing benchmark dataset, and (2) highly tailored synthetic dataset generated via LLMs. In this section, we provide how our training data differs from the dataset used in conventional high-performing embedding models. To be specific, the training data used for models originated from E5-Mistral can be summarized as below:

- E5-Mistral : E5 data (human labeled) + synthetic data
- SFR : a specially curated dataset comprising MTEB (human labeled)
- GritLM : E5S data (human labeled) + synthetic data
- Ours : E5S data (human labeled) + synthetic data (well-curated)

As demonstrated in Table 8, the SFR model is trained on a specially curated dataset of MTEB tasks, including Retrieval, Clustering, Classification, STS, and Reranking tasks, while excluding development and testing sets. Notably, SFR uniquely treats the labels in clustering and classification tasks as documents, applying contrastive loss exclusively to their respective negatives and omitting in-batch negatives. This method utilizes the **labels in the evaluation tasks directly during training**, which SFR hypothesizes encourages models to regularize embeddings based on high-level concepts, resulting in better separation of data across different domains. Instead, we focused exclusively on the training dataset used by E5-Mistral. Specifically, we utilized the same dataset composition as in the E5S data from GritLM, augmenting it with S2ORC data to enhance the training dataset used by E5-Mistral. We further analyzed and refined the dataset with the task list used in the E5S data.

Recall that our objective is to **produce high-quality synthetic data using GPT and verify its effectiveness**. Our synthetic data does not incorporate labels but follows the conventional triplet form of query-positive-negative, comprised solely of documents. By doing so, we seek to reinforce

the human-labeled benchmark dataset with high-quality synthetic data, ensuring that the synthetic data never sees the labels in the evaluation tasks directly. This method adheres to E5-Mistral data structures while leveraging the advanced capabilities of GPT to enhance data quality.

All in all, our approach seeks to improve the quality and reliability of LLM-generated data by focusing on sophisticated synthetic data generation. This method stands in contrast to SFR's label-utilizing strategy, proposing that a well-crafted synthetic dataset can effectively support model training and generalization without direct reliance on evaluation task labels. The impact of adding the data used by SFR to our dataset setting is remained as a future work.

Table 8: Comparison of the benchmark dataset used for training various models. Considering the fact that MTEB benchmark is used to test each embedding model, it is less desired to train with datasets included in MTEB. However, one high-performing baseline (SFR model) is trained on a specially curated dataset comprising MTEB tasks, including Retrieval, Clustering, Classification, STS, and Reranking tasks, while excluding development and testing sets. Instead, the dataset composition of our benchmark dataset is identical to that of E5S data used in GritLM [6], which is the concatenation of the training dataset used by E5-Mistral and S2ORC data. Here, S2ORC may overlap with the Arxiv-related tasks in MTEB.

Task	Benchmark Dataset	Benchmark Dataset used for training E5-Mistral	Benchmark Dataset used in our model GritLM	Benchmark Dataset used for training SFR	MTEB English Evaluation Dataset
Retrieval	DuReader	O	O	X	X
	ELI5	O	O	X	X
	FEVER	O	O	O	O
	HotpotQA	O	O	O	O
	NLI	O	O	O	X
	MIRACL	O	O	X	X
	MSMARCO	O	O	O	O
	Mr.TyDi	O	O	X	X
	NQ	O	O	O	O
	S2ORC	X	O	X	X
	SQuAD	O	O	X	X
	T2Ranking	O	O	X	X
	TriviaQA	O	O	X	X
	Quora	O	O	O	O
	FiQA	X	X	O	O
	SciFact	X	X	O	O
	NFCorpus	X	X	O	O
	DBPedia	X	X	O	O
Clustering	arXiv	X	X	O	O
	bioRxiv	X	X	O	O
	medRxiv	X	X	O	O
Classification	AmazonReview	X	X	O	O
	Emotion	X	X	O	O
	MTOPIntent	X	X	O	O
	ToxicConversation	X	X	O	O
	TweetSentiment	X	X	O	O
STS	STS12	X	X	O	O
	STS22	X	X	O	O
	STSBenchmark	X	X	O	O
Reranking	SciDocs	X	X	O	O
	StackOverFlowDupQuestionRR	X	X	O	O

4 Proposed Data Refinement Methods

In this section, we provide details of our proposed data refinement methods used on (1) Benchmark Dataset and (2) Synthetic Data. Our methods include sophisticated data crafting, data filtering, and negative mining techniques.

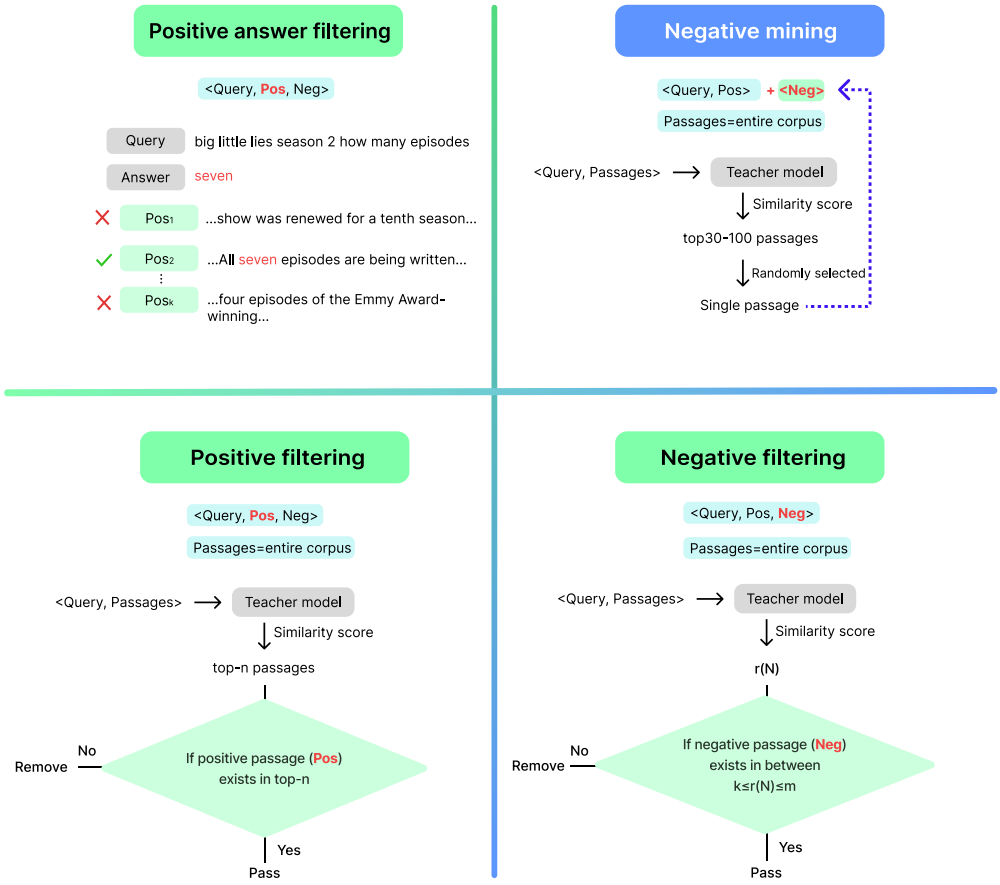

Figure 1: Overview of our proposed methods of refining the Benchmark Dataset.

4.1 Data Refinement on Benchmark Dataset

For each task, we applied various methodologies to identify the most effective combination, selectively implementing the best-performing strategies for each task. These methods include:

- **Data Source Selection**: When multiple data sources are available (e.g., HotpotQA from KILT or DPR), we tested each source and selected the most suitable one.
- **Positive Answer Filtering**: Utilizing only passages that contain the answer.
- **Positive Filtering**: Incorporating positives only if they are within the top-n rankings of the teacher model.
- **Negative Mining**: Using samples within the top 30-100 rankings of the teacher model as negatives.
- **Negative Filtering**: Implementing various strategies using teacher models, where negatives (N) are considered only if their rankings (r) fall within a specific top (n) to (m) range of the teacher model's rankings, i.e., $(k \leq r(N) \leq m)$.

The performance varies significantly depending on the combination of methods used for each task. For instance, in some tasks where answers are present, filtering positives based on the inclusion of

the answer may enhance the performance, while in other tasks, this approach may not be beneficial. It is crucial to tailor the refinement strategies to the specific requirements and characteristics of each task to achieve optimal performance. The specific processing methods applied for each task can be reviewed in Figure 1.

Table 9: Refinement methods for enhancing the quality of Synthetic Data across six tasks, demonstrating up to two independent methods per task.

	Short-Long (Retrieval)	**Long-Short (Classification)**	**Short-Short (Matching)**	**Long-Long (Matching)**	**STS (Semantic Textual Similarity)**	**bitext (Multilingual Translation)**
Refinement Method 1	Few-Shot Prompt Engineering (Answerable and Score-Based Generation) → Filtering with a Teacher Model → Filtering False Positives via LLMs for Refute Expressions	Zero-shot Prompt Engineering (Generalizing Task Descriptions) → De-Duplicating Positive/Negative Labels for Zero-shot Outputs	Few-Shot Prompt Engineering (Hard Negative, Entity, Colloquialism)	Few-Shot Prompt Engineering (Hard Negative)	Score-Based Filtering	Filtering with a Teacher Model
Refinement Method 2	Few-Shot Prompt Engineering (Topic Diversity) → Top-100 Negative Mining	Few-Shot Prompt Engineering (Types of Labels and Their Relationships)	-	Few-Shot Prompt Engineering (Controlling the difficulty level of hard negatives, Topic Diversity)	Few-Shot Prompt Engineering (Topic Diversity, Diverse negative patterns)	-

4.2 Data Refinement on Synthetic Data

As detailed in Table 9, the quality of synthetic data is scrutinized for each task, employing various types of few-shot prompt engineering, filtering, and negative mining to improve overall data quality. In Table 10, we provide detailed description of each issue and our proposed solution to handle the issue.

5 Training Details: Mostly Follow E5-Mistral and SFR

Our experimental setup integrates several strategies and techniques, drawing from SFR and E5-Mistral, aiming to optimize the model performance and the training efficiency.

Contrastive Loss We utilize the standard InfoNCE loss over in-batch negatives and hard negatives, formulated as follows:

$$\min \mathcal{L} = -\log \frac{\phi(q,d)}{\phi(q,d)+\sum_{n_i \in \mathcal{N}} \phi(q,n_i)},$$

where $\mathcal{N}$ denotes the set of all negative documents, and $\phi(q, d)$ is a function that computes the matching score between query q and document d. Following E5-Mistral, we adopt the temperature-scaled cosine similarity function:

$$\phi(q, d) = \exp\left(\frac{1}{\tau} \cos(h_q, h_d)\right),$$

where h_q and h_d are the embeddings of the query and the document, respectively, and the temperature is set to $\tau = 0.02$.

Fine-Tuning Procedure We fine-tuned the E5-Mistral-7b-instruct model[2] using a batch size of 2,080 and a learning rate of 1e-4, starting with a warm-up phase followed by linear decay. Despite the advantage of larger batch sizes noted by SFR, increasing from 2,048 to 8,192 showed no significant performance change. This fine-tuning process took approximately 30 hours on four A100 GPUs.

Maximum In-Device Batch Size We used A100 80GB GPUs to train with the maximum in-device batch size, akin to the SRF approach, considering the impact of the number of negative samples.

[2] `https://huggingface.co/intfloat/E5-Mistral-7b-instruct?ref=blog.salesforceairesearch.com`

Sequence Length While SRF used a maximum sequence length of 128 for queries and 256 for documents, we opt for a maximum sequence length of 4K, aiming to generalize better for (long query, long passage) tasks. For our experiments, we adhere to the settings of E5-Mistral, limiting evaluations to the first 512 tokens for efficiency, despite the model's capability to handle longer sequences.

Task-Homogeneous Batching According to SFR, constructing batches with samples from a single task enhances the difficulty of in-batch negatives and improves retrieval task performances. We followed such setting.

Number of Hard Negatives Although SFR found that using seven hard negatives per query yields the best results, our experiments showed diminishing returns with more hard negatives as data quality improved. Therefore, we opt to use only one hard negative, using the mE5-base model as the teacher for negative mining, unlike SFR's use of the BGE-base model[3].

Top 30-100 Negative Mining We found that negative sampling significantly affects the performance more than the in-device batch size. As reported by SFR, selecting the top 30-100 negative samples impacts the performance most significantly, underscoring the importance of precise negative sample selection.

LoRA Adapters We added LoRA adapters with alpha 32 and rank 16 to all linear layers. Despite no significant performance difference between ranks 16 and 8, we chose rank 16 for its greater stability and reduced fluctuation during training.

E5-Mistral Style One-Sided Instruction Prefix The E5-Mistral approach uses one-sided instructions for asymmetric datasets, where only queries receive instructions, allowing the document corpus to be encoded once, cached, and reused across tasks. As below, GritLM explains that even symmetric tasks are handled one-sidedly during training but evaluated in a two-sided format, ensuring the consistency and the reliability of similarity measures.

"During training, even symmetric tasks are handled in a one-sided manner, though they are evaluated in a two-sided format. This is feasible because the cosine similarity function used during training is transitive. Therefore, if a sentence with instructions (A) is similar to a sentence without instructions (B), and B is similar to another sentence with instructions (C), it can be inferred that A is also similar to C. This ensures consistency and reliability in similarity measures despite the one-sided instruction approach during training" [6].

In-Batch Negative Strategy in Clustering and Classification The embedding model for classification or clustering tasks can be misled by the in-batch negatives technique, as the "passage" within a batch might belong to the same class and therefore are not actual negatives. SFR uniquely treats labels as documents in clustering and classification tasks, applying contrastive loss exclusively to their respective negatives and avoiding in-batch negatives to prevent misleading the embedding model. Conversely, Gecko [7] addresses the issue of false negatives by assigning a unique ID to each triple (query, positive example, negative example), making in-batch negatives trivial for the model to distinguish. For long-short and long-long synthetic data tasks, our method applied the unique ID method, similar to Gecko, exclusively in few-shot scenarios where generative diversity decreases, and did not use it in zero-shot scenarios.

Other Training Techniques We applied various advanced training techniques, including gradient checkpointing, mixed precision training (fp16), and DeepSpeed ZeRO-3, to enhance training efficiency and performance.

[3] `https://huggingface.co/BAAI/bge-base-en`

Table 10: Summary of Issues and Proposed Solutions for Improving Synthetic Data Quality.

Task	Issue Type	Description	Solution
Short-Long (Retrieval)	Word Length Control	**[Inconsistent Word Length]** GPT-4-turbo struggles with controlling the word length, particularly for non-English languages.	-
	Self-Explanation Issues / False Positives and False Negatives	**[Self-Explanation Issues]** • Rationale Inclusion: The rationale on the positiveness/negativeness of the GPT-generated passages is added to the generated positive/negative passages. • Query Inclusion: Queries are sometimes included verbatim in the passages. **[False Positives and False Negatives]** • Quality of Hard Negatives: Although the generated hard negatives are less relevant to the given query than the positives, some hard negatives generated by the prompts still provide general or partially answerable responses. • Quality of Given Positives: In some cases, the given positive passage is ambiguous, and a more relevant positive sample exists within the passage pool. This causes issues during training because the model may incorrectly treat the more relevant passage as a negative, while using the ambiguous passage as a positive. • False Negatives: Some GPT-generated hard negatives may not be pure negatives, worsening the training scheme. A significant number of negatives are actually false negatives, as they could be semantically similar to the corresponding positive documents but are incorrectly classified as negatives. • Higher Incidence in Non-English Languages: False positives and false negatives occur more frequently in non-English languages.	**[Instruction-Following for Task Description]** Align the task descriptions with instructions of the benchmark dataset. This alignment does not directly enhance final performance but facilitates quicker learning by providing clear guidelines for generating task descriptions in the style of the training data. **[Few-Shot Prompt Engineering; Answerable and Score-Based Generation]** This method involves defining a relatedness score ranging from 1 to 10 and determining the answerability of the generated positives and negatives as a boolean value. • A score of 1 ensures minimum relevance is maintained. • Highly tailored instructions are provided to guide the generation process. • Filtering is conducted based on an answerability value, which assesses whether the generated passage can adequately respond to the query. **[Filtering with a Teacher Model]** Implement filtering for: • Positives ranked below the top-100. • Negatives ranked above the top-50. • Instances where the positive rank is higher than the negative rank.
	Diversity and Content Moderation Issues	**[Lack of Representation]** Synthetic data lacks representation on topics like same-sex marriage, feminism, and abortion. **[Safety Concerns]** Socially sensitive topics such as climate change and gun ownership are affected by content moderation, restricting discussion or representation.	**[Few-Shot Prompt Engineering; Topic Diversity]** Mitigate the absence of representation and balance by: • Adding well-formed instructions and few-shot examples of refutations or arguments on underrepresented issues. • Incorporating entity diversity through prompt engineering with examples from Wikipedia and scientific topics.
	Refutation Issues	**[Incorrect Refutations as False Positives]** When tasked with generating refutations (e.g., refute, oppose, disagree, reject, deny, contradict), the model sometimes produces supportive results instead, resulting in false positives.	**[Filtering False Positives via LLMs for Refutation Expressions]** Classify and filter samples containing refutation expressions by: • Using Multiple LLMs to label the data as "support" or "refute." • Discard data labeled as "support" by either LLM.
	Few-Shot Generation Problems	When addressing other problems by generating data through few-shot examples and prompt engineering, several issues can arise: **[Quality Issues with Few-Shot Data]** Data generated from few-shot examples often include false negatives. Hard negatives generated in this manner can be more relevant or similar than the positive samples, causing quality issues.	**[Top-100 Negative Mining]** Use few-shot data for positives and employ top-100 negative mining for negatives to ensure quality and relevance.
Long-Short (Classification)	GPT-4 Overfitting Problem	**[GPT-4 Overfitting Problem]** GPT-4-turbo tends to generate specific words with high probability when certain keywords are included, despite using random seed and temperature settings of 1.0.	**[Zero-shot Prompt Engineering; Generalizing Task Descriptions]** Modify topic descriptions to more general expressions to mitigate overfitting.
	Output Label Diversity	**[Output Label Diversity]** The typical label types and relationships used for evaluation might not be sufficiently represented in the generated data due to the variety of tasks, which can range from very detailed to very broad.	**[Few-Shot Prompt Engineering; Types of Labels and Their Relationships]** Utilize few-shot style-following to ensure that label types and relationships—such as binary, multi-class, multi-label, ordinal, and hierarchical—used for classification tasks are well-represented. This improves the overall performance and reliability of classification models. Additionally, generate data using both zero-shot methods (to ensure diversity) and few-shot methods (to capture typical label types and relationships). **[De-Duplicating Positive/Negative Labels for Zero-shot Outputs]** De-duplicating zero-shot positive/negative labels is important to avoid redundancy.
Short-Short (Matching)	Content Repetition in Hard Negatives / Lack of Entity Information / Insufficient Colloquial Language	**[Content Repetition in Hard Negatives]** Some hard negatives repeat the content of the positives verbatim, making it challenging to classify them as true negatives. **[Lack of Entity Information]** The generated data lacks sufficient information about entities such as personal names, place names, movie titles, and game titles. **[Insufficient Colloquial Language]** There is a lack of colloquial language in the generated data, which is necessary for creating more realistic and relatable passages.	**[Few-Shot Prompt Engineering; Hard Negative, Entity, Colloquialism]** • Hard Negative Generation via GPT-4: Implement a method distinct from E5-Mistral to generate hard negatives using GPT-4, ensuring that the negatives do not repeat the content of the positives verbatim. This approach enhances the distinctiveness of hard negatives, improving the model's ability to accurately classify them. • Entity Inclusion: Utilize well-formed instruction to enrich the generated data with a diverse range of entities. By specifying the inclusion of various entities in the prompts, the generated passages will contain a broader spectrum of information, enhancing the diversity and comprehensiveness of the training data. • Colloquial Language: Utilize well-formed instruction to include colloquial expressions in the generated data. By crafting prompts that encourage the use of conversational language, the synthetic data will better mimic natural dialogue, making the training data more versatile and applicable to real-world scenarios.
Long-Long (Matching)	Content Repetition in Hard Negatives	**[Content Repetition in Hard Negatives]** Some hard negatives repeat the content of the positives verbatim, making it challenging to classify them as true negatives.	**[Few-Shot Prompt Engineering; Hard Negative]** • Hard Negative Generation via GPT-4: Implement a method distinct from E5-Mistral to generate hard negatives using GPT-4, ensuring that the negatives do not repeat the content of the positives verbatim. This approach enhances the distinctiveness of hard negatives, improving the model's ability to accurately classify them.
	Overly Difficult Hard Negatives	**[Overly Challenging Hard Negatives]** The generated hard negatives are often too difficult, leading to a lack of clearly distinguishable negatives, which could lead to several issues such as model confusion, training instability, and lack of clear distinction.	**[Few-Shot Prompt Engineering; Controlling the difficulty level of hard negatives]** • Prompt Engineering for Few-Shot Generation: Implement prompt engineering and use few-shot examples to create a balanced set of hard negatives that are challenging but not overly difficult. This ensures the availability of clearly distinguishable negatives, facilitating better performance.
	Few-Shot Generation Problems	When addressing other problems by generating data through few-shot examples and prompt engineering, several issues can arise: **[Quality Issues with Few-Shot Data]** Data generated from few-shot examples often include false negatives. Hard negatives generated in this manner can be more relevant or similar than the positive samples, causing quality issues.	**[Few-Shot Prompt Engineering; Topic Diversity]** • Generation of Diverse Topics: Generate totally different topic positives and negatives within the STEM domain to avoid the similarity issues seen in few-shot generated data.
STS (Semantic Textual Similarity)	Duplication and Diversity Issues	**[Duplication]** The tendency to generate identical sentences necessitates deduplication. **[Topic Diversity]** A high number of similar queries are generated (e.g., quantum computing, data analysis, parks, dogs and cats, "The quick brown fox...").	**[Few-Shot Prompt Engineering; Topic Diversity]** Use few-shot topics to enhance topic diversity in the generated data.
	False Positives and False Negatives	**[False Positives]** Some generated positives are of lower quality than the generated hard negatives. **[False Negatives]** In score-based generation for STS tasks, false negatives can occur when the positive and negative examples differ by only 0.5 points in their relatedness scores. **[Limited Negative Patterns]** • General statements: Some negative examples use abstract level of terminologies • Counterfactual statements: Statements that swap subjects and objects • Lack of Negation: GPT-generated data rarely includes negation patterns	**[Score-Based Filtering]** Ensure that positive and negative pairs have a score difference of at least 1.5 to avoid false negatives. **[Few-Shot Prompt Engineering; Diverse negative patterns for STS]** Implement strategies that include simple negations, significant changes in context, or more diverse counterfactual statements, such as altering demographic, temporal, spatial, or numeric information.
bitext (Multilingual Translation)	False Negatives	**[High Incidence of False Negatives]** A significant number of false negatives are produced, regardless of score differences (between positives and negatives) as defined in the E5-Mistral prompt. To be specific, many negatives have the same meaning as the positives, with only slight difference in wording. This is because, in the multilingual settings, the generated negative is often just another translated version of the query. This issue is pronounced in multi-lingual settings, where hard negatives in translation tasks can inadvertently become positives.	**[Filtering with a Teacher Model]** Reduce the frequency of samples where the positive and negative are highly similar by filtering out those with minimal differences in similarity ranks, retaining only 30% of such samples.

6 Homogeneous Task Ordering and Mixed Task Fine-tuning

In this section, we provide details of our proposed training methods, dubbed as Homogeneous Task Ordering and Mixed Task Fine-tuning , details of which are in the upcoming subsections. We first train a model using Homogeneous Task Ordering method, and then fine-tune this model using Mixed Task Fine-tuning method.

6.1 Homogeneous Task Ordering

Recall that existing embedding models (including SFR) rely on task-homogeneous batching, which constructs each batch with the samples from identical task. We follow this way of constructing the batches, but added another constraint on the training, **which specifies the orders of tasks handled in the consecutive batches**. To be specific, our batches of 1-epoch training is composed of n blocks, where we assigned samples in the batches and blocks in a way that the task order within each block is identical across n blocks. The primary objective of this proposed Homogeneous Task Ordering method is to clearly track how the performance changes as one block of batches (having specific task orders) are loaded. This provides us more precise insights on the effects of task ordering on the model performance.

6.2 Mixed Task Fine-Tuning

Although task-homogeneous batching is prevalent method used in existing works, it also has some limitations. This batching strategy limits the learning process to one task per gradient update, leading to potential catastrophic forgetting. Catastrophic forgetting occurs when a neural network loses previously acquired knowledge about one task due to exclusive focus on another, resulting in performance fluctuations observed at specific intervals. Additionally, the randomization of task sequence can delay the training of certain tasks, further aggravating the problem.

Motivated by these issues, we use Mixed Task Fine-tuning, defined as below:

- **Mixed Task Fine-Tuning**: While maintaining a homogeneous batch within the device, this method involves mixing different tasks to compose each batch. This strategy aims to mitigate the risks associated with learning in a homogeneous environment by diversifying the learning inputs.

Our training framework involves an initial phase of homogeneous task fine-tuning for one complete epoch, followed by mixed task fine-tuning conducted over a few steps. This structured approach is designed to optimize learning across diverse tasks, thereby reducing performance fluctuations and the impact of catastrophic forgetting on untrained tasks at the current step.

In our experiments, mixed task fine-tuning within 20 steps demonstrated the best performance. Extending the training beyond this point resulted in a decline in performance.

7 Streamlined Evaluation

To facilitate rapid experimentation, we employed a combination of strategies that streamline the evaluation process, as demonstrated in Figure 2. These strategies include (a) the use of a light retrieval evaluation set and (b) evaluations conducted using 4-bit precision. Collectively, these approaches enable quick and effective experimentation. Using a single GPU, one checkpoint can be evaluated in approximately 5 hours, with retrieval tasks specifically taking about 4 hours.

7.1 Light Evaluation Set

For Retrieval Task We observed that evaluating the performance of a model on Retrieval tasks takes much longer than the evaluation on other tasks in MTEB benchmarks. Thus, in order to check the performance of a model rapidly, we designed a light retrieval evaluation set, which has negligible performance differences compared to the performances observed for the full evaluation set. This approach allows efficient assessment of the model's performance on retrieval tasks without significantly compromising accuracy, thus optimizing the overall evaluation process.

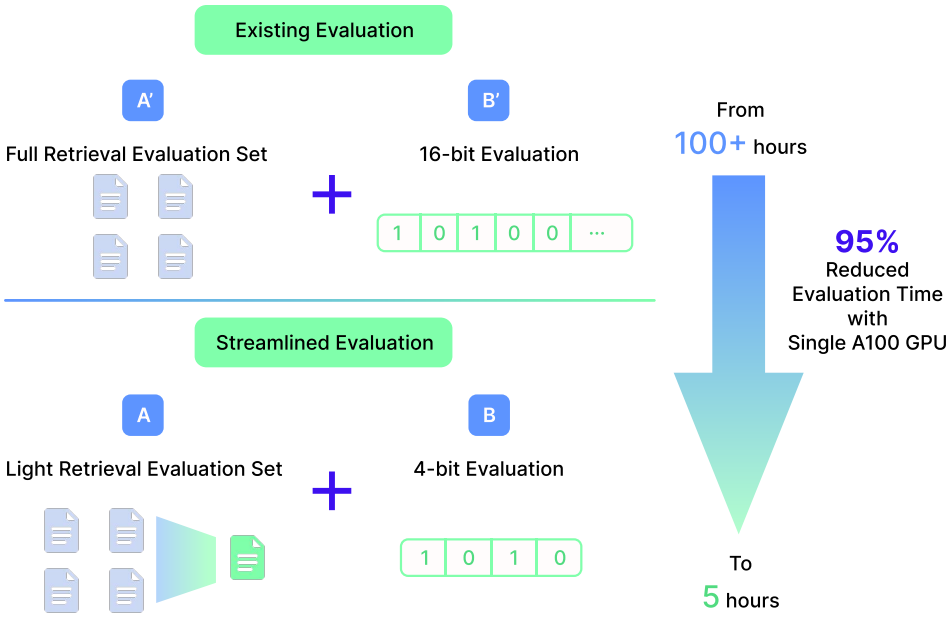

Figure 2: Strategies for streamlining the evaluation process.

Table 11: Comparison with publicly accessible models.

Model	BEIR Retrieval (15 datasets)	BEIR Retrieval Rank (as of May 29, 2024)	MTEB Average (56 datasets)	MTEB Average Rank (as of May 29, 2024)
Linq-Embed-Mistral	**60.2**	#1	68.2	#3
SFR-Embedding-Mistral	59.0	#3	67.6	#4
gte-Qwen1.5-7B-instruct	56.2	#5	67.3	#5
GritLM-7B	57.4	#6	66.8	#7
e5-mistral-7b-instruct	56.9	#7	66.6	#8

- **Corpus Sampling**: Among the entire corpus, the top 50 most relevant corpora for each query are extracted via the previously trained teacher model. Subsequently, sets will be created to eliminate overlaps, thereby reducing the number of corpora utilized for evaluation.
- **Query Sampling**: We sample 20% of queries with the balanced labels to further streamline the process.

For Other Tasks (Classification, Clustering, Pair Classification, Re-ranking, STS, Summarization) We omitted evaluation on large-scale sub-tasks.

7.2 4-bit Evaluation

Using 4-bit precision for evaluations allow a single GPU to process more samples compared to the case of using 16-bit precision. This results in a substantial increase in the processing speed, with observed improvements of up to approximately 40% in our GPU setting. This significant speedup comes without the cost of accuracy.

Table 12: Comparison with commercial models.

Model	BEIR Retrieval (15 datasets)	BEIR Retrieval Rank (as of May 29, 2024)	MTEB Average (56 datasets)	MTEB Average Rank (as of May 29, 2024)
Linq-Embed-Mistral (Linq)	**60.2**	#1	68.2	#3
voyage-large-2-instruct (Voyage)	58.3	#4	68.3	#2
google-gecko.text-embedding-preview-0409 (Google)	55.7	#14	66.3	#9
text-embedding-3-large (OpenAI)	55.4	#17	64.6	#18
Cohere-embed-english-v3.0 (Cohere)	55.0	#19	64.5	#20

8 Full Evaluation on MTEB

The Massive Text Embedding Benchmark (MTEB) stands as the most comprehensive benchmark for evaluating embedding models, incorporating 56 datasets across seven task types: classification, clustering, pair classification, re-ranking, retrieval, STS, and summarization.

The key experimental points are:

- Linq-Embed-Mistral performs well in the MTEB benchmarks, with an average score of 68.2 across 56 datasets. This places it 1st among publicly accessible models listed on the MTEB leaderboard and 3rd overall.
- The model shows a significant enhancement in the retrieval performance, ranking 1st among all models listed on the MTEB leaderboard with a performance score of 60.2.
  - Within the Mistral Model Series, a suite of models based on the foundational Mistral architecture, SFR enhances E5-Mistral by adding a specially curated dataset of MTEB tasks. In contrast, our approach focuses solely on **creating and integrating more sophisticated synthetic datasets**. This has increased our model's score from 56.9 for E5-Mistral and 59.0 for SFR to an 60.2.

Table 13: Comparison with the models that tops the MTEB leaderboard (as of May 29, 2024), highlighting the first and second place items in each task using bold and underlined formatting.

MTEB Rank (by May 29, 2024)	Model	Average (56 datasets)	Retrieval (15 datasets)	Classification (12 datasets)	Clustering (11 datasets)	Pair Classification (3 datasets)	Reranking (4 datasets)	STS (10 datasets)	Summarization (1 dataset)
3	Linq-Embed-Mistral	68.2	**60.2**	80.2	51.4	88.4	60.3	85.0	31.0
1	NV-Embed-v1	**69.3**	**59.4**	**87.4**	52.8	86.9	**60.5**	82.8	31.2
2	voyage-large-2-instruct	**68.3**	58.3	**81.5**	**53.4**	**89.2**	60.1	84.6	30.8
4	SFR-Embedding-Mistral	67.6	59.0	78.3	51.7	**88.5**	**60.6**	85.1	31.2
5	gte-Qwen1.5-7B-instruct	67.3	56.2	79.6	**55.8**	87.4	60.1	82.4	**31.5**
6	voyage-lite-02-instruct	67.1	56.6	79.3	52.4	86.9	58.2	**85.8**	31.0
7	GritLM-7B	66.8	57.4	79.5	50.6	87.2	60.5	83.4	30.4
8	e5-mistral-7b-instruct	66.6	56.9	78.5	50.3	88.3	60.2	84.6	31.4
9	google-gecko.text-embedding-preview-0409	66.3	55.7	81.2	47.5	87.6	58.9	**85.1**	**32.6**
18	text-embedding-3-large	64.6	55.4	75.5	49.0	85.7	59.2	81.7	29.9
20	Cohere-embed-enlgish-v3.0	64.5	55.0	76.5	47.4	85.8	58.0	82.6	30.2

Author Contributions

Contributions to the development of Linq-Embed-Mistral were made by

Junseong Kim (Project Leader; GPT Data Strategy, Experiment Design, Technical Guidance and Advice) **Seolhwa Lee** (AI Researcher; Dataset Filtering Mining, Modeling Experiments, GPT Data Generation) **Jihoon Kwon** (AI Intern; Benchmark Datasets for Training, Dataset Filtering Mining, Data Training Pipelines, Modeling Experiments, MTEB Evaluation) **Sangmo Gu** (AI Intern; GPT Data Strategy, GPT Data Filtering, GPT Data Generation) **Yejin Kim** (AI Intern; Baseline Models) **Minkyung Cho** (AI Intern; Baseline GPT Data) **Jy-yong Sohn** (Advisor; Technical Guidance and Advice) **Chanyeol Choi** (Advisor; Technical Guidance and Advice).

References

[1] Y. Zhu, H. Yuan, S. Wang, J. Liu, W. Liu, C. Deng, Z. Dou, and J.-R. Wen, "Large language models for information retrieval: A survey," *arXiv preprint arXiv:2308.07107*, 2023.

[2] Y. Huang and J. Huang, "A survey on retrieval-augmented text generation for large language models," *arXiv preprint arXiv:2404.10981*, 2024.

[3] Z. Dai, V. Y. Zhao, J. Ma, Y. Luan, J. Ni, J. Lu, A. Bakalov, K. Guu, K. B. Hall, and M.-W. Chang, "Promptagator: Few-shot dense retrieval from 8 examples," *arXiv preprint arXiv:2209.11755*, 2022.

[4] V. Jeronymo, L. Bonifacio, H. Abonizio, M. Fadaee, R. Lotufo, J. Zavrel, and R. Nogueira, "Inpars-v2: Large language models as efficient dataset generators for information retrieval," *arXiv preprint arXiv:2301.01820*, 2023.

[5] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei, "Improving text embeddings with large language models," *arXiv preprint arXiv:2401.00368*, 2023.

[6] N. Muennighoff, H. Su, L. Wang, N. Yang, F. Wei, T. Yu, A. Singh, and D. Kiela, "Generative representational instruction tuning," *arXiv preprint arXiv:2402.09906*, 2024.

[7] J. Lee, Z. Dai, X. Ren, B. Chen, D. Cer, J. R. Cole, K. Hui, M. Boratko, R. Kapadia, W. Ding, *et al.*, "Gecko: Versatile text embeddings distilled from large language models," *arXiv preprint arXiv:2403.20327*, 2024.

[8] M. Rui, L. Ye, J. Shafiq, Rayhan, X. Caiming, Z. Yingbo, and Y. Semih, "Sfr-embedding-mistral:enhance text retrieval with transfer learning." Salesforce AI Research Blog, 2024.

[9] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, *et al.*, "Mistral 7b," *arXiv preprint arXiv:2310.06825*, 2023.

[10] G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, and E. Grave, "Unsupervised dense information retrieval with contrastive learning," *arXiv preprint arXiv:2112.09118*, 2021.

[11] A. Neelakantan, T. Xu, R. Puri, A. Radford, J. M. Han, J. Tworek, Q. Yuan, N. Tezak, J. W. Kim, C. Hallacy, *et al.*, "Text and code embeddings by contrastive pre-training," *arXiv preprint arXiv:2201.10005*, 2022.

[12] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, and F. Wei, "Text embeddings by weakly-supervised contrastive pre-training," *arXiv preprint arXiv:2212.03533*, 2022.

[13] S. Xiao, Z. Liu, P. Zhang, and N. Muennighof, "C-pack: Packaged resources to advance general chinese embedding," *arXiv preprint arXiv:2309.07597*, 2023.