JINA EMBEDDINGS: A Novel Set of High-Performance Sentence Embedding Models

Michael Günther and **Louis Milliken** and **Jonathan Geuter**
Georgios Mastrapas and **Bo Wang** and **Han Xiao**

Jina AI
Ohlauer Str. 43, 10999 Berlin, Germany
{michael.guenther,louis.milliken,jonathan.geuter,
georgios.mastrapas,bo.wang,han.xiao}@jina.ai

Abstract

JINA EMBEDDINGS constitutes a set of high-performance sentence embedding models adept at translating textual inputs into numerical representations, capturing the semantics of the text. These models excel in applications like dense retrieval and semantic textual similarity. This paper details the development of JINA EMBEDDINGS, starting with the creation of high-quality pairwise and triplet datasets. It underlines the crucial role of data cleaning in dataset preparation, offers in-depth insights into the model training process, and concludes with a comprehensive performance evaluation using the Massive Text Embedding Benchmark (MTEB). Furthermore, to increase the model's awareness of grammatical negation, we construct a novel training and evaluation dataset of negated and non-negated statements, which we make publicly available to the community.

1 Introduction

Sentence embedding models are an effective instrument for encoding the semantic nuances of words, phrases, and larger textual units into a continuous vector space. They encapsulate the complexities of contexts and lexical and grammatical interrelationships within a text, facilitating downstream tasks like information retrieval, semantic similarity evaluation, and text classification.

Despite the potential of these models, questions remain about the effectiveness of different data preprocessing strategies, the optimal loss function for training sentence embedding models, and the impact on performance of increasing the number of model parameters. This paper addresses these challenges.

We have develop a novel dataset specifically to train our sentence embedding models. Furthermore, we design a dataset specifically to sensitize our models to distinguish negations of statements from confirming statements. This paper also presents JINA EMBEDDINGS, a set of high-performance sentence embedding models trained on these datasets. The JINA EMBEDDINGS set is expected to comprise five distinct models, ranging in size from 35 million to 6 billion parameters. Three of those models are already trained and published. [1]

The JINA EMBEDDINGS models employ contrastive training on the T5 architecture [Raffel et al., 2020]. It's important to note that we opt to use the T5 model as our base due to its pre-training on a mixed set of downstream tasks. We argue that incorporating this approach can potentially enhance our ability to accurately gauge the effectiveness of our training strategy.

Our large-scale contrastive fine-tuning approach surpasses zero-shot T5 and delivers a performance level on par with other leading T5-based sentence embedding models such as Sentence-T5 [Ni et al., 2022a] and GTR [Ni et al., 2022b]. Consequently, this work demonstrates that high-quality sentence embeddings can be achieved with the judicious use of resources and innovative training methodologies.

2 Dataset Preparation

In order to develop models that excel across a wide range of tasks, we collate a comprehensive set of both public and custom datasets. These datasets target various retrieval objectives, such as e-commerce search, duplicate detection, web retrieval, article retrieval for question-answering, and text classification. Consolidating these datasets into a unified format facilitates concurrent model training for all tasks.

Definition of Format: Given the lack of non-relevance information in many of the datasets, we reformat each training item into pairs, designated as $(q, p) \in D_{pairs}$. Each pair includes a query

[1] jina-small-v1 , jina-base-v1 , jina-large-v1 are available at https://huggingface.co/jinaai, and are also ranked in the MTEB leaderboard on Hugging Face: https://huggingface.co/spaces/mteb/leaderboard.

Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023), pages 8–18
December 6, 2023 ©2023 Association for Computational Linguistics

string q and an associated target string p. To leverage explicit non-relevance judgments, we create an auxiliary set of triplets $(q, p, n) \in D_{triplets}$, which pair a query string q with a match p (positive) and a non-matching string n (negative).

Data Extraction: The methods used to extract pairs and triplets are specific to each source dataset. For example, given a question-answer dataset, we use questions as query strings and answers as target strings. Retrieval datasets often contain queries that can serve as query strings and relevant and non-relevant annotated documents which can operate as matching and non-matching strings.

Training Steps: Our training process is a two-step approach. Initially, we train on pairs and then fine-tune the model using the triplets, as detailed in Section 3.3.

2.1 Pairwise Data Preparation

The substantial size and inconsistent quality of many large datasets necessitates a rigorous filtering pipeline. We apply the following steps to filter training data:

De-Duplication: Duplicated entries within training data can negatively impact model performance [Hernandez et al., 2022], and potentially lead to overfitting. Consequently, we remove duplicate entries from our dataset. Considering the dataset's volume, we employ hash functions to identify and eliminate text pairs that map to duplicate hash values. We normalize whitespace and capitalization before checking for duplicates. Empty pairs and pairs with identical elements are also removed.

Language Filtering: Since we design our embedding models for English, we use the `fasttext-language-identification` model[2] based on the fasttext text classification method [Joulin et al., 2017] to remove non-English training items from the dataset.

Consistency Filtering: Consistency filtering means excluding training pairs with low semantic similarity. Previous studies suggest that eliminating low-similarity pairs using an auxiliary, albeit less precise, model boosts performance [Dai et al., 2023, Wang et al., 2022]. We employ the `all-MiniLM-L6-v2` model[3] for consistency filtering in this manner: We generate embeddings for 1M pairs $(q_i, p_i)_i$ randomly sampled from D_{pairs}. For every pair $(q, p) \in D_{pairs}$ in the dataset, we verify whether p is among the top two passages most similar to q based on the cosine similarity of their embeddings compared to all passages p_i, $i = 1, ..., 1\text{M}$.

The application of these preprocessing steps reduces the size of the dataset from over 1.5 billion mixed-quality pairs to 385 million high-quality pairs. This reduction permits us to train our model with significantly less data than typical embedding models without sacrificing embedding quality.[4]

2.2 Triplet Data Preparation

For the triplet dataset, we forego de-duplication and language filtering and we assume the quality of these datasets already meets our quality requirements. However, we validate the relevance of the "positive" item with respect to the "query" for each triplet in a manner similar to consistency filtering. Instead of contrasting the embedding cosine similarity $s(q, p)$ against a sample set, we compare it solely with the similarity $s(q, n)$ of the embeddings derived from the same triplet $(q, p, n) \in D_{triplets}$. This is accomplished using a cross-encoder model, which evaluates the pair directly without generating embedding representations. More specifically, we leverage the `ms-marco-MiniLM-L-6-v2` model[5] to verify whether the difference in retrieval scores determined by the model exceeds a threshold $r(q, p) - r(q, n) > \kappa$, with threshold $\kappa = 0.2$, and eliminate all other pairs. This methodology draws inspiration from the de-noising strategy proposed in [Qu et al., 2021].

2.3 Negation Data Preparation

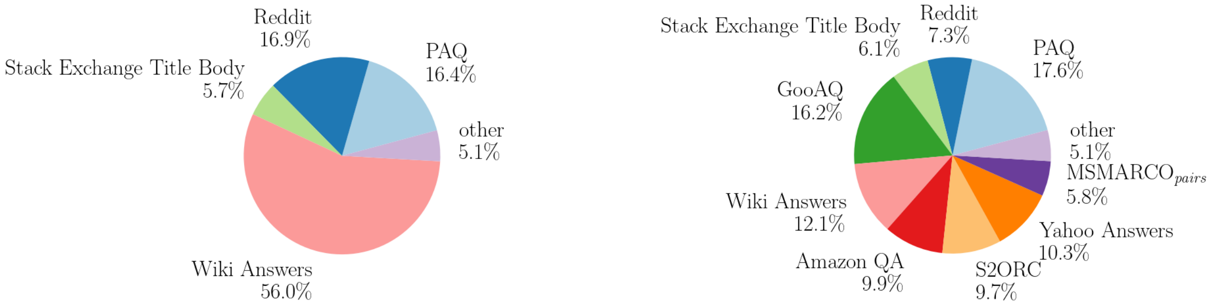
We observe that many embedding models struggle to accurately embed negations. For instance, when embedding the three sentences: "A couple walks hand in hand down a street.", "A couple is walking together.", and "A couple is not walking together.", the first two should be embedded close together, while the second and third, contradictory in

[2] `fasttext-language-identification` (https://huggingface.co/facebook/fasttext-language-identification)

[3] `all-MiniLM-L6-v2` model (https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2)

[4] For instance, models like `all-MiniLM-L6-v2` and `all-mpnet-base-v2` are trained on nearly 1.2 billion pairs, whereas other T5-based models such as `sentence-t5-base` or `sentence-t5-large` are trained on 2.2 billion pairs.

[5] `ms-marco-MiniLM-L-6-v2` (https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-6-v2)



(a) Original Distribution after Filtering

(b) Adjusted by Sampling Rates

Figure 1: The composition of 385 million pairwise data

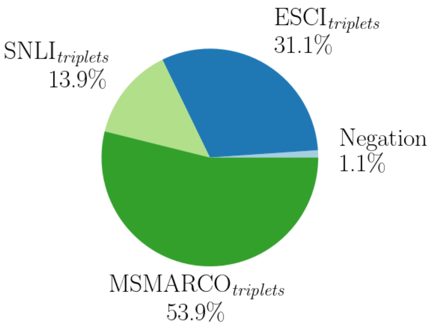


Figure 2: The composition of 927,000 triplets data

meaning, should be positioned further apart.[6] However, for instance, the `all-MiniLM-L6-v2` model assigns a cosine similarity of 0.7 to the first two sentences, while attributing a similarity of 0.86 to the second and third.[7]

We decide to address this problem by creating our own negation dataset[8]. This dataset, based on positive pairs from the SNLI dataset[9] and negatives created with GPT-3.5, comprises triplets (anchor, entailment, negative) akin to the example given above, where (anchor, entailment) form a positive pair and the "negative" contradicts both the "anchor" and "entailment", while remaining syntactically very similar to "entailment". This dataset forms a subset of our aforementioned triplet dataset, with training details provided in Section 3.3.

Our model evaluation on the negation dataset, which includes a comparative analysis with other popular open-source models, is presented in Section 4.3.

2.4 Data Composition

Our dataset of text pairs, represented as $D_{pairs} = D_1 \sqcup \cdots \sqcup D_n$, is aggregated from 32 individual datasets. This amounts to a total of 1.6 billion pairs before filtering, which is subsequently reduced to a robust 385 million high-quality pairs after rigorous filtering.

In comparison, our dataset of triplets initially comprises a total of 1.13 million entries before filtering, streamlined to 927,000 triplets after filtering.

The composition of our datasets after filtering is illustrated in Figure 1a for the text pairs, and in Figure 2 for the triplets. Together, these form the final dataset for the training of the JINA EMBEDDINGS models.

3 Training

Training takes place in two distinct phases. The first phase centers on training the model using the voluminous quantity of text pairs, consolidating the semantics of an entire text phrase into a single representative embedding. The second phase uses the relatively small triplet dataset, comprising an anchor, an entailment, and a hard-negative, teaching it to differentiate between similar and dissimilar text phrases.

3.1 Training on Pairwise Data

Each model within the JINA EMBEDDINGS set is based on, and trained using, the zero-shot T5 models of corresponding size, as detailed in [Raffel et al., 2020]. The zero-shot T5 models are composed of encoder-decoder pairs. However, Ni et al.

[6] Although it could be argued that for certain tasks, like document retrieval, it might still be desirable for contradicting texts to be embedded closely. Regardless, in this example, the first two sentences should be assigned a higher similarity.

[7] Interestingly, our large model does *correctly* assign a cosine similarity of 0.77 to the positive pair, and only a similarity of 0.62 to the negative pair, after fine-tuning with our negation dataset.

[8] The negation dataset is available at https://huggingface.co/datasets/jinaai/negation-dataset

[9] https://huggingface.co/datasets/snli

[2022a] has demonstrated that it is more effective to calculate text embeddings using only the encoder component of the T5 models, as opposed to deploying both encoder and decoder. Consequently, the Jina Embeddings models use only the encoders of their respective T5 models.

During tokenization, Jina Embeddings models use SentencePiece [Kudo and Richardson, 2018] to segment input text and encode them into WordPiece tokens [Kudo, 2018]. Following the encoder model, a mean pooling layer is implemented to generate fixed-length representations from the token embeddings.

For the training process involving pairs, we employ InfoNCE [van den Oord et al., 2018], a contrastive loss function. This function calculates the loss for a pair $(q, p) \sim B$ within a batch $B \in D^k$ of text pairs, where the batch size is k, as follows:

$$\mathcal{L}^{pairs}_{\text{NCE}}(B) := \mathbb{E}_{(q,p)\sim B}\left[-\ln \frac{e^{s(q,p)/\tau}}{\sum_{i=1}^{k} e^{s(q,p_i)/\tau}}\right]$$

The loss is calculated by comparing the cosine similarity between a given question q and its target p, with the similarity to all other targets in the batch. We found that calculating the loss in both directions results in greater improvements during training. Accordingly, the loss is defined as follows:

$$\mathcal{L}^{pairs}(B) := \mathcal{L}^{pairs}_{\text{NCE}}(B) + \mathcal{L}^{pairs}_{\overline{\text{NCE}}}(B), \text{ where}$$

$$\mathcal{L}^{pairs}_{\overline{\text{NCE}}}(B) := \mathbb{E}_{(q,p)\sim B}\left[-\ln \frac{e^{s(p,q)/\tau}}{\sum_{i=1}^{k} e^{s(p,q_i)/\tau}}\right].$$

Intuitively, $\mathcal{L}^{pairs}_{\overline{\text{NCE}}}$ matches the target string to all query strings instead. The constant τ denotes a temperature parameter which we set to $\tau = 0.05$. This method of calculating the loss is based on a similar method in [Neelakantan et al., 2022].

3.2 Data Sampling in Pairwise Training

Rather than sequentially training on individual datasets, we opt for a parallel approach, training on all datasets concurrently. We postulate that this parallel training promotes enhanced model generalization across diverse tasks. Despite this, each training batch is exclusively composed of data from a single dataset. This ensures that loss calculations, performed across the entire batch, do not conflate data from different tasks.

Our dataloader operates by initially selecting a dataset, followed by sampling the requisite number of data points from it to constitute a batch for the worker (refer to Section 4). Prior to training, the pairs within the datasets are thoroughly shuffled.

Sampling a dataset D_i follows a probability distribution ρ across all datasets D_i. The probability of sampling D_i is $\rho\left(D_i\right) = \frac{|D_i| s_i}{\sum_{j=1}^{n} |D_j| s_j}$ and is contingent upon the dataset's size $|D_i|$ and a scaling factor s_i.

Given the disparity in dataset sizes, it is critical to frequently sample from larger datasets to prevent overfitting on the smaller ones. Furthermore, we manipulate the sampling rates of datasets using scaling factors to prioritize training on high-quality datasets and achieve balance among text domains. In scenarios where datasets with higher sampling rates deplete their items before the completion of a training epoch, the dataset is reset, enabling the model to cycle through its items anew. This ensures that high-sampling-rate datasets contribute multiple times within a single training epoch.

Figure 1b displays the proportion of each dataset used based on their sampling rates. Following the creation of this adjusted distribution, the frequency of sampling from larger datasets significantly diminishes, resulting in only 180 million pairs actually being used during training.

3.3 Training on Triplet Data

Following the completion of pairwise training, the model progresses to the next phase which involves training on the triplet datasets. This phase uses a different loss function, leveraging negatives for improved model performance.

We experimented with various triplet loss functions and found that the best results are achieved through a combination of multiple commonly used triplet loss functions. Specifically, we use the extended version of the InfoNCE loss $\mathcal{L}^{triplets}_{NCE+}$, given by (2), which employs additional negatives [Reimers, 2023], the reverse InfoNCE loss $\mathcal{L}^{triplets}_{\overline{\text{NCE}}}$ from the initial training phase as given by (3), and the triplet margin loss function $\mathcal{L}^{triplets}_{3}$ as presented in (4) [Chechik et al., 2010].

The triplet function $\mathcal{L}^{triplets}_{3}$ determines the cosine similarity difference between the query and target $s(q, n)$, and the query and negative match $s(q, n)$. Furthermore, it establishes a minimal margin $\varepsilon = 0.05$ between these two values. If the negative is more similar to the query or the margin is violated, $\mathcal{L}_3^{triplets}$ returns a positive value. Otherwise, it yields 0, which is achieved through the application of the ReLU activation function. For the temperature parameter, we opted for a value of $\tau = 0.05$.

4 Evaluation

We conduct a comprehensive evaluation to compare our models against other state-of-the-art models (Section 4.1), investigate the impact of our filtering pipeline (Section 4.2), and evaluate the models' sensitivity to negation of statements (Section 4.3). Section 6 mentions details about the training.

To provide comprehensive results on the performance of models on various downstream tasks applicable to embeddings, we rely on the MTEB benchmark frameworks introduced by Muennighoff et al. [2023]. This also compromises all the retrieval tasks included in the BEIR [Thakur et al., 2021] benchmark. We also publish the code for executing it on our models on the Hugging Face pages of our model[10]. For evaluating models on the negation dataset, we use our own separate evaluation tool[11].

4.1 Performance Against State-of-the-Art Models

To gauge the performance of the JINA EMBEDDINGS set in relation to other similarly sized open-source and close-sourced models, we select representative models from five distinct size categories, as depicted in Table 1. Additionally, we include sentence-t5 and gtr-t5 xl and xxl models, which are based on T5 models with 3 billion and 11 billion parameters, respectively. This inclusion allows investigating the performance variation with models of such massive scales.

Table 6 presents the scores for MTEB's sentence similarity tasks, wherein the models within the JINA EMBEDDINGS set outshine their similarly sized counterparts across numerous tasks. Notably, the `jina-large-v1` model consistently delivers comparable, if not superior, results to models in the billion-parameter scale. `jina-base-v1` and `jina-small-v1` also exhibit competitive performances with models of analogous sizes, exceeding their peers on the BIOSSES[12] task. This highlights the benefits of training with highly diverse data sources.

Model	Parameters	Embedding Dimensions
sentence-t5-xxl	4.9b	768
gtr-t5-xxl	4.9b	768
gtr-t5-xl	1.2b	768
sentence-t5-xl	1.2b	768
`jina-large-v1`	330m	1024
gtr-t5-large	330m	768
sentence-t5-large	330m	768
all-mpnet-base-v2	110m	768
`jina-base-v1`	110m	768
gtr-t5-base	110m	768
sentence-t5-base	110m	768
`jina-small-v1`	35m	512
all-MiniLM-L6-v2	23m	384

Table 1: Model sizes and output dimensions

`jina-base-v1` consistently demonstrates performances similar to or better than gtr-t5-base, which was trained specifically for retrieval tasks [Ni et al., 2022b]. However, it seldom matches the scores of sentence-t5-base, which was trained on sentence similarity tasks [Ni et al., 2022a].

The evaluation of model performances on retrieval tasks, presented in Table 8, reflects a similar relationship among gtr-t5, sentence-t5, and JINA EMBEDDINGS. Here, gtr-t5 models, which have been specially trained on retrieval tasks, consistently score the highest for their respective sizes. JINA EMBEDDINGS models follow closely behind, whereas sentence-t5 models trail significantly. The JINA EMBEDDINGS set's capability to maintain competitive scores across these tasks underscores the advantage of multi-task training.

As illustrated in Table 7, `jina-large-v1` also achieves exceedingly high scores on reranking tasks, often outperforming larger models. Similarly, `jina-base-v1` surpasses gtr-t5-large and sentence-t5-large on several reranking tasks, which could once again be attributed to the specific training tasks of sentence-t5 and gtr-t5.

[10] https://huggingface.co/jinaai/jina-embedding-b-en-v1/blob/main/mteb_evaluation.py

[11] https://huggingface.co/jinaai/jina-embedding-b-en-v1/blob/main/negation_evaluation.py

[12] https://tabilab.cmpe.boun.edu.tr/BIOSSES/DataSet.html

$$\mathcal{L}^{triplets}(B) := \mathcal{L}^{triplets}_{NCE+}(B) + \mathcal{L}^{triplets}_{\text{NCE}}(B) + \mathcal{L}^{triplets}_{3}(B), \quad \text{where} \tag{1}$$

$$\mathcal{L}^{triplets}_{NCE+}(B) := \mathbb{E}_{(q,p,n)\sim B}\left[-\ln \frac{\exp(s(q,p)/\tau)}{\sum_{i=1}^{k} \exp(s(q,p_i)/\tau) + \exp(s(q,n_i)/\tau)} \right], \tag{2}$$

$$\mathcal{L}^{triplets}_{\text{NCE}}(B) := \mathbb{E}_{(q,p,n)\sim B}\left[-\ln \frac{\exp(s(p,q)/\tau)}{\sum_{i=1}^{k} \exp(s(p,q_i)/\tau)} \right], \tag{3}$$

$$\mathcal{L}^{triplets}_{3}(B) := \mathbb{E}_{(q,p,n)\sim B}\left[\text{ReLU}\Big(s(q,n) - s(q,p) + \varepsilon \Big) \right]. \tag{4}$$

Model	RR	RT	STS
sentence-t5-xxl	56.42	42.24	**82.63**
gtr-t5-xxl	**56.66**	**48.48**	78.38
gtr-t5-xl	55.96	47.96	77.80
sentence-t5-xl	54.71	38.47	81.66
jina-large-v1	**56.42**	44.81	80.96
gtr-t5-large	55.36	**47.42**	78.19
sentence-t5-large	54.00	36.71	**81.83**
all-mpnet-base-v2	**59.36**	43.81	80.28
jina-base-v1	55.84	44.03	79.93
gtr-t5-base	54.23	**44.67**	77.07
sentence-t5-base	53.09	33.63	**81.14**
jina-small-v1	53.07	38.91	78.06
all-MiniLM-L6-v2	**58.04**	**41.95**	**78.90**

Table 2: Average Scores for Reranking (RR), Retrieval (RT) and sentence similarity tasks (STS)

4.2 Impact of Filtering Steps

We evaluate the effectiveness of our dataset preprocessing pipeline by performing an ablation study. In this study, we fine-tune our smallest model on the Reddit dataset, where various preprocessing steps are individually applied. The corresponding results are presented in Table 3.

The ablation study's results underscore the value of both language and consistency filtering as crucial preprocessing steps. Their combined application results in the highest performance across the majority of benchmarks.

Specifically for the Reddit dataset, we observe a significant performance boost with the application of consistency filtering, while language filtering only marginally enhances the performance. We can account for this disparity by noting that the language filter removes only 17.4% of the Reddit data, while consistency filtering screens out 84.3[13]. Reddit samples are primarily in English, but many are positive pairs with very low similarity, making consistency filtering more effective than language filtering.

The effectiveness of these preprocessing steps, however, does exhibit variability across different datasets.

4.3 Effectiveness of Negation Data

To determine the effectiveness of our models on negation data, we evaluate them against the test split of our negation dataset, comparing the results with other open source models. We measure performance with respect to two metrics: one measures the percentage of samples where the model positions the *anchor* and *entailment* closer than the *anchor* and *negative* (which is an easy task, as the *anchor* and *negative* are syntactically dissimilar), the other measures the percentage of samples where the model positions the *anchor* and *entailment* closer than the *entailment* and *negative* (which is a hard task, as the *entailment* and *negative* are syntactically more similar than the *anchor* and *entailment*). The former is denoted by *EasyNegation*, the latter by *HardNegation*. The outcomes of these evaluations are displayed in Table 4. We assess our models both before and after fine-tuning on the triplet data, denoted as **<model>**$_{\textbf{pairwise}}$ and **<model>**$_{\textbf{all}}$, respectively.

From the results, we observe that across all model sizes, fine-tuning on triplet data (which includes our negation training dataset) dramatically enhances performance, particularly on the HardNegation task. Our models are on par with other state-of-the-art open-source models in terms of per-

[13] It is pertinent to note that the subsets filtered out overlap, thus the combined application of language and consistency filtering filters out *only* 86.8% of the data.

	Retrieval		
Data Preparation	Quora	SciFact	Trec-Cov
No Extra Filter	0.734	0.218	0.242
Language	0.741	0.218	0.250
Consistency	0.805	**0.381**	0.297
Language + Consistency	**0.806**	0.379	**0.306**

	Sentence Similarity						
Data Preparation	STS12	STS13	STS14	STS15	STS16	STS17	STS22
No Extra Filter	0.558	0.668	0.573	0.694	0.706	0.764	0.606
Language	0.561	0.668	0.579	0.697	0.704	0.765	0.609
Consistency	0.652	**0.728**	0.652	0.760	0.755	0.808	**0.610**
Language + Consistency	**0.653**	0.727	**0.656**	**0.764**	**0.757**	**0.810**	0.609

Table 3: Evaluation of Data-Preparation Effectiveness on the Reddit Dataset. Retrieval evaluated on nDCG@10, Sentence Similarity on Spearman.

formance, while achieving this with only a fraction of the training data required by their counterparts.

5 Related Work

The field of embedding models has seen significant advanced over the years, with the development of various models featuring diverse architectures and training pipelines. For instance, Sentence-BERT [Reimers and Gurevych, 2019] uses BERT to generate sentence embeddings. Similarly, Sentence-T5 [Ni et al., 2022a], based on the encoder architecture of T5, demonstrates superior performance over Sentence-BERT on numerous benchmarks. The study underscores the effectiveness of encoders for sentence embeddings, contrasting with another approach that explores the use of decoders [Muennighoff, 2022].

Knowledge distillation [Hinton et al., 2015] offers an alternative approach to model training. In this setup, a larger, pre-trained model acts as a mentor, instructing a smaller model during training. This methodology can be seamlessly integrated with a contrastive loss function, presenting an avenue for future investigation.

Embedding models can also be characterized based on their functionality. For instance, while some models are designed to solely embed queries, others are trained to embed queries along with specific instructions, generating task-dependent embeddings [Su et al., 2023]. An example of this using a T5-based model is the large dual encoder [Ni et al., 2022b], which is fine-tuned for retrieval tasks and computes a retrieval score directly.

Recent studies [Neelakantan et al., 2022, Wang et al., 2022] emphasize the benefits of contrastive pre-training coupled with fine-tuning on hard negatives. Both approaches have achieve state-of-the-art results on multiple benchmarks, with [Wang et al., 2022] also employing consistency filtering as part of their preprocessing pipeline.

6 Training Details

For training, we employ A100 GPUs and leverage the DeepSpeed stage 2 distributed training strategy [Rajbhandari et al., 2020] for effective multi-device management. For training our models we use the AdamW optimizer, coupled with a learning rate scheduler that adjusts the learning rate during the initial stages of training. The hyperparameters used across all three models throughout the training process are listed in Table 5.

7 Conclusion

This paper introduces the JINA EMBEDDINGS set of embedding models, demonstrating that competitive performance on various tasks can be achieved while substantially reducing the amount of training data, when compared to other models with comparable backbones. Through an extensive evaluation on the MTEB benchmark, we show that employing judicious data filtering techniques can lead to enhanced performance in comparison to training with a larger, yet lower-quality dataset. These findings significantly shift the paradigm, indicating that training large language models for embedding tasks can be conducted with less data than previously as-

	EasyNegation	HardNegation	Parameters	Training samples
jina-small-v1$_{\text{pairwise}}$	88.4%	8.4%	35m	385m
jina-base-v1$_{\text{pairwise}}$	93.0%	13.8%	110m	385m
jina-large-v1$_{\text{pairwise}}$	94.6%	16.6%	330m	385m
jina-small-v1$_{\text{all}}$	96.6%	35.2%	35m	386m
jina-base-v1$_{\text{all}}$	97.8%	54.6%	110m	386m
jina-large-v1$_{\text{all}}$	**98.2%**	65.4%	330m	386m
all-MiniLM-L6-v2	94.8%	29.4%	23m	1170m
all-mpnet-base-v2	97.4%	**67.6%**	110m	1170m
sentence-t5-base	96.0%	55%	110m	2275m
sentence-t5-large	**98.2%**	64.0%	330m	2275m

Table 4: Evaluating a Range of Models on the Negation Dataset: A Benchmark Analysis of JINA EMBEDDINGS Trained on Both Pairwise-Only and Combined Pairwise and Triplet Data. The negation dataset is available at https://huggingface.co/datasets/jinaai/negation-dataset

Hyperparameters	Value
# of devices	8
Sequence length	512
Model precision	32 bit
Learning rate	0.00005
# of steps for learning rate warm-up	500
Batch size for jina-small-v1	4096
Batch size for jina-base-v1	2048
Batch size for jina-large-v1	1024

Table 5: Hyperparameters

sumed, leading to potential savings in training time and resources.

However, we acknowledge the limitations of the current methodologies and the performance of the JINA EMBEDDINGS set. During the training on pairs, the sampling rate selection was based on a heuristic approach. Given the vast size of the search space for these sampling rates, we leaned on our intuition and dataset familiarity to prioritize higher-value datasets over their lower-value counterparts. This subjective approach, however, points to the need for more objective methods for future advancements.

Additionally, the JINA EMBEDDINGS set fell short on some tasks. For instance, calculating sentence similarity on our negation dataset (as described in Section 4.3) didn't meet our expectations (see Table 4) nor achieves competitive scores for classification and clustering tasks on the MTEB benchmark. These performance shortcomings suggest a possible deficit in the representation of these types of tasks in our training data, necessitating further investigation.

Looking ahead, we aim to refine our training processes to deliver models with improved performance and greater sequence length. Our future endeavors also include generating bilingual training data and training an embedding model capable of understanding and translating between two languages, thereby expanding the utility and versatility of the JINA EMBEDDINGS set.

References

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1): 5485–5551, 2020.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, 2022a.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, et al. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9844–9855, 2022b.

Danny Hernandez, Tom Brown, Tom Conerly, Nova DasSarma, Dawn Drain, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Tom Henighan, Tristan Hume, et al. Scaling laws and interpretability of learning from repeated data. *arXiv preprint arXiv:2205.10487*, 2022.

Armand Joulin, Édouard Grave, Piotr Bojanowski, and Tomáš Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, 2017.

Zhuyun Dai, Vincent Y Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith Hall, and Ming-Wei Chang. Promptagator: Few-shot dense retrieval from 8 examples. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=gmL46YMpu2J.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5835–5847, 2021.

Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, 2018.

Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, 2018.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David Schnurr, Felipe Petroski Such, Kenny Hsu, Madeleine Thompson, Tabarak Khan, Toki Sherbakov andcalculating Joanne Jang, Peter Welinder, and Lilian Weng. Text and code embeddings by contrastive pre-training. *CoRR*, abs/2201.10005, 2022. URL https://arxiv.org/abs/2201.10005.

Nils Reimers. http://plastimatch.org/doxygen/classHausdorff__distance.html##details Last Access: 14th July 2023, 2023.

Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(3), 2010.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark, 2023.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models, 2021.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.

Niklas Muennighoff. Sgpt: Gpt sentence embeddings for semantic search, 2022.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. One embedder, any task: Instruction-finetuned text embeddings, 2023.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.

Appendix

Model	BIOSSES	SICK-R	STS12	STS13	STS14	STS15	STS16	STS17	STS22	STS Benchm.
sentence-t5-xxl	80.43	**80.47**	78.85	**88.94**	**84.86**	**89.32**	84.67	**89.46**	65.33	**84.01**
sentence-t5-xl	73.12	79.98	**79.02**	88.80	84.33	88.89	**85.31**	88.91	64.32	83.93
gtr-t5-xxl	**81.91**	74.29	70.12	82.72	78.24	86.26	81.61	85.18	65.76	77.73
gtr-t5-xl	78.94	73.63	69.11	81.82	77.07	86.01	82.23	84.90	**66.61**	77.65
sentence-t5-large	78.93	**80.34**	**79.11**	**87.33**	**83.17**	**88.28**	**84.36**	88.99	62.39	**85.36**
gtr-t5-large	**84.86**	73.39	70.33	82.19	77.16	86.31	81.85	83.93	64.30	77.60
jina-large-v1	84.43	79.20	74.53	83.16	78.09	86.91	83.65	**90.16**	**64.89**	84.60
sentence-t5-base	75.89	80.18	**78.05**	**85.85**	**82.19**	**87.46**	**84.03**	**89.57**	62.66	**85.52**
gtr-t5-base	79.00	71.45	68.59	79.09	74.64	84.85	81.57	85.80	66.17	79.58
all-mpnet-base-v2	80.43	**80.59**	72.63	83.48	78.00	85.66	80.03	90.60	**67.95**	83.42
jina-base-v1	**83.58**	79.14	75.06	80.86	76.13	85.55	81.21	88.98	66.22	82.57
all-MiniLM-L6-v2	81.64	**77.58**	72.37	**80.60**	**75.59**	**85.39**	78.99	**87.59**	**67.21**	**82.03**
jina-small-v1	**82.96**	76.33	**74.28**	78.55	73.84	83.71	**80.03**	87.49	64.25	79.20
text-emb-ada-002*	86.35	80.60	69.80	83.27	76.09	86.12	85.96	90.25	68.12	83.17

Table 6: Spearman Correlation for Sentence Similarity Tasks

Model	AskUbuntu-DupQuestions	MindSmall-Reranking	SciDocsRR	StackOverflow-DupQuestions
sentence-t5-xxl	**66.16**	30.60	76.09	52.85
sentence-t5-xl	62.86	29.77	75.16	51.05
gtr-t5-xxl	63.23	**31.93**	**77.96**	**53.50**
gtr-t5-xl	63.08	31.50	76.49	52.79
sentence-t5-large	61.51	30.27	74.88	49.34
gtr-t5-large	61.64	**31.84**	76.39	**51.58**
jina-large-v1	**62.83**	31.48	**80.97**	50.38
sentence-t5-base	59.73	30.20	73.96	48.46
gtr-t5-base	60.86	31.33	73.71	51.01
all-mpnet-base-v2	**65.85**	30.97	**88.65**	**51.98**
jina-base-v1	62.40	**31.56**	79.31	50.11
all-MiniLM-L6-v2	**63.48**	**30.80**	**87.12**	**50.76**
jina-small-v1	60.25	30.68	74.16	47.18
text-emb-ada-002*	62.05	31.45	81.22	50.54

Table 7: Mean Average Precision (mAP@10) for Reranking Tasks

* text-emb-ada-002 appears in a separate category since no model size is known and the embedding size is much higher compared to other models.

Model	FEVER	HotpotQA	MSMARCO	NQ	Quora Retrieval	SciFact	TREC COVID	Argu Ana	Climate FEVER	DBPedia	FiQA 2018	NFCorpus	SCIDOCS	Touche 2020
sentence-t5-xxl	51.20	42.14	27.67	52.87	85.96	55.38	59.48	39.85	14.63	39.19	46.68	**35.08**	**17.17**	21.65
sentence-t5-xl	36.12	37.17	25.17	46.29	85.85	50.91	54.77	39.40	10.61	33.65	44.71	33.18	15.97	22.51
gtr-t5-xxl	**74.08**	**59.67**	**44.05**	**57.24**	**89.09**	**66.77**	51.90	**53.77**	**27.21**	**41.28**	**46.78**	34.18	15.88	**26.76**
gtr-t5-xl	72.18	58.91	43.52	56.16	88.91	64.2	**60.09**	52.81	27.01	39.74	44.19	33.34	15.71	25.26
sentence-t5-large	36.21	33.95	23.96	42.02	85.73	49.91	46.11	39.27	11.36	31.55	**43.55**	31.10	15.38	21.63
gtr-t5-large	**72.66**	**57.85**	**42.73**	**55.09**	**88.47**	**63.42**	56.68	**52.09**	**26.90**	**39.55**	42.79	**32.63**	15.51	**28.29**
jina-large-v1	71.90	54.95	40.34	51.40	88.09	59.76	**57.25**	46.48	21.26	34.13	37.27	32.24	**18.45**	20.73
sentence-t5-base	26.17	33.20	20.70	36.32	85.49	45.76	40.70	44.85	10.37	27.77	34.83	28.65	14.15	20.30
gtr-t5-base	68.93	**54.93**	**41.16**	**50.47**	**87.98**	59.74	56.05	**50.83**	**24.88**	**35.24**	35.15	30.22	14.00	**25.89**
all-mpnet-base-v2	50.86	39.29	39.75	50.45	87.46	**65.57**	51.33	46.52	21.97	32.09	**49.96**	**33.29**	**23.76**	19.93
jina-base-v1	**73.29**	52.78	37.77	47.87	87.63	59.40	**60.57**	49.01	21.48	32.44	34.06	30.38	17.63	18.59
all-MiniLM-L6-v2	51.93	46.51	**36.54**	**43.87**	**87.56**	**64.51**	47.25	**50.17**	**20.27**	**32.33**	**36.87**	**31.59**	**21.64**	**16.90**
jina-small-v1	**69.12**	**47.48**	31.80	38.89	85.69	52.40	**52.30**	43.57	17.25	28.28	25.19	25.96	15.29	16.67
text-emb-ada-002*	74.99	60.90	40.91	51.58	87.60	72.75	68.47	57.44	21.64	39.39	44.41	36.97	18.36	21.61

Table 8: Normalized Discounted Cumulative Gain (nDCG@10) for retrieval tasks