Introduction to Matryoshka Embedding Models

Upvote 73

ished February 23, 2024

ate on GitHub

tomaarsen
Tom Aarsen

Xenova
Joshua

osanseviero
Omar Sanseviero

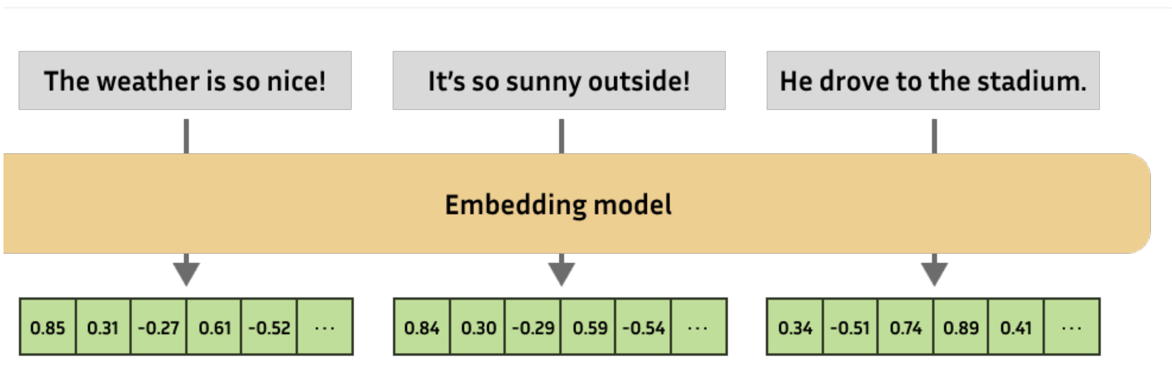
is blogpost, we will introduce you to the concept of Matryoshka Embeddings and ain why they are useful. We will discuss how these models are theoretically trained 1ow you can train them using Sentence Transformers.

tionally, we will provide practical guidance on how to use Matryoshka Embedding els and share a comparison between a Matryoshka embedding model and a regular edding model. Finally, we invite you to check out our interactive demo that showcases ower of these models.

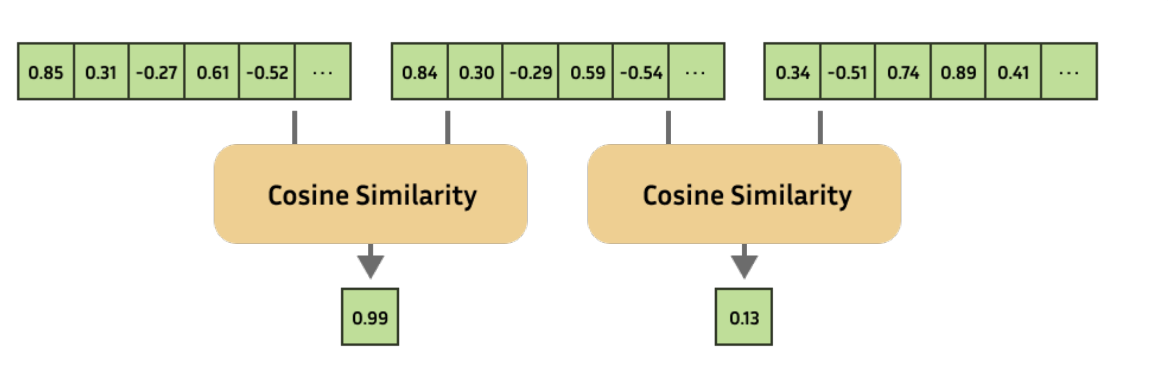
ble of Contents

derstanding Embeddings

:ddings are one of the most versatile tools in natural language processing, enabling :itioners to solve a large variety of tasks. In essence, an embedding is a numerical :sentation of a more complex object, like text, images, audio, etc.



:mbedding model will always produce embeddings of the same fixed size. You can compute the similarity of complex objects by computing the similarity of the :ctive embeddings!

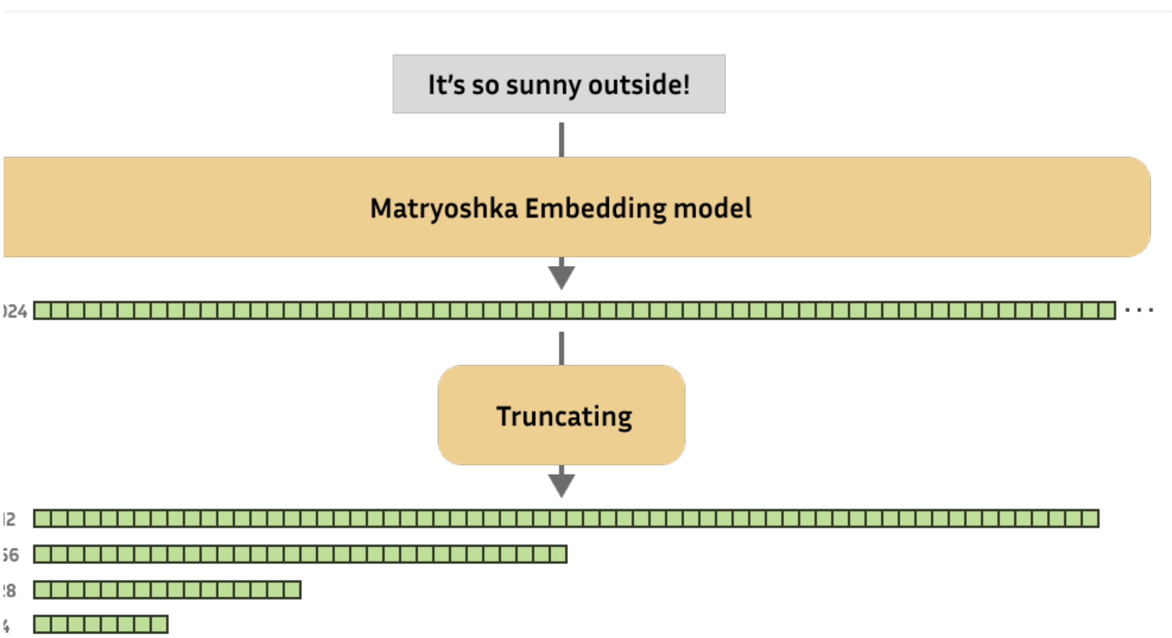


has an enormous amount of use cases, and serves as the backbone for nmendation systems, retrieval, one-shot or few-shot learning, outlier detection, arity search, paraphrase detection, clustering, classification, and much more!

Matryoshka Embeddings

search progressed, new state-of-the-art (text) embedding models started producing ddings with increasingly higher output dimensions, i.e., every input text is sented using more values. Although this improves performance, it comes at the cost iciency of downstream tasks such as search or classification.

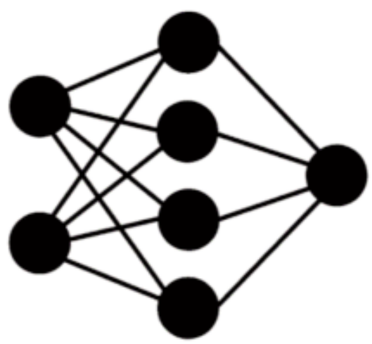
equently, Kusupati et al. (2022) were inspired to create embedding models whose ddings could reasonably be shrunk without suffering too much on performance.



e Matryoshka embedding models are trained such that these small truncated ɜddings would still be useful. In short, Matryoshka embedding models can produce ıl embeddings of various dimensions.

Matryoshka Dolls

hose unfamiliar, "Matryoshka dolls", also known as "Russian nesting dolls", are a set of len dolls of decreasing size that are placed inside one another. In a similar way, yoshka embedding models aim to store more important information in earlier nsions, and less important information in later dimensions. This characteristic of yoshka embedding models allows us to truncate the original (large) embedding uced by the model, while still retaining enough of the information to perform well on ıstream tasks.



1. Compute Matryoshka Embedding

ıy would you use 🪆 Matryoshka Embedding models?

variable-size embedding models can be quite valuable to practitioners, for example:

Shortlisting and reranking: Rather than performing your downstream task (e.g., nearest neighbor search) on the full embeddings, you can shrink the embeddings to a smaller size and very efficiently "shortlist" your embeddings. Afterwards, you can process the remaining embeddings using their full dimensionality.

Trade-offs: Matryoshka models will allow you to scale your embedding solutions to your desired storage cost, processing speed, and performance.

w are 🪆 Matryoshka Embedding models trained?

eoretically

Matryoshka Representation Learning (MRL) approach can be adopted for almost all edding model training frameworks. Normally, a training step for an embedding model ves producing embeddings for your training batch (of texts, for example) and then some loss function to create a loss value that represents the quality of the produced eddings. The optimizer will adjust the model weights throughout training to reduce oss value.

Matryoshka Embedding models, a training step also involves producing embeddings our training batch, but then you use some loss function to determine not just the ty of your full-size embeddings, but also the quality of your embeddings at various rent dimensionalities. For example, output dimensionalities are 768, 512, 256, 128, 54. The loss values for each dimensionality are added together, resulting in a final loss e. The optimizer will then try and adjust the model weights to lower this loss value.

actice, this incentivizes the model to frontload the most important information at the of an embedding, such that it will be retained if the embedding is truncated.

Sentence Transformers

ence Tranformers is a commonly used framework to train embedding models, and it ntly implemented support for Matryoshka models. Training a Matryoshka embedding el using Sentence Transformers is quite elementary: rather than applying some loss tion on only the full-size embeddings, we also apply that same loss function on

:ated portions of the embeddings.

xample, if a model has an original embedding dimension of 768, it can now be trained ;8, 512, 256, 128 and 64. Each of these losses will be added together, optionally with : weight:

```
:om sentence_transformers import SentenceTransformer
:om sentence_transformers.losses import CoSENTLoss, MatryoshkaLoss

odel = SentenceTransformer("microsoft/mpnet-base")

ase_loss = CoSENTLoss(model=model)
oss = MatryoshkaLoss(
   model=model,
   loss=base_loss,
   matryoshka_dims=[768, 512, 256, 128, 64],
   matryoshka_weight=[1, 1, 1, 1, 1],


odel.fit(
   train_objectives=[(train_dataset, loss)],
   ...,
```

ing with `MatryoshkaLoss` does not incur a notable overhead in training time.

:ences:

[`MatryoshkaLoss`]

[`CoSENTLoss`]

[`SentenceTransformer`]

[`SentenceTransformer.fit`]

Matryoshka Embeddings - Training

he following complete scripts as examples of how to apply the `MatryoshkaLoss` in ice:

matryoshka_nli.py: This example uses the `MultipleNegativesRankingLoss` with `MatryoshkaLoss` to train a strong embedding model using Natural Language Inference (NLI) data. It is an adaptation of the NLI documentation.

matryoshka_nli_reduced_dim.py: This example uses the `MultipleNegativesRankingLoss` with `MatryoshkaLoss` to train a strong embedding model with a small maximum output dimension of 256. It trains using Natural Language Inference (NLI) data, and is an adaptation of the NLI documentation.

matryoshka_sts.py: This example uses the `CoSENTLoss` with `MatryoshkaLoss` to train an embedding model on the training set of the `STSBenchmark` dataset. It is an adaptation of the STS documentation.

w do I use 🪆 Matryoshka Embedding models?

eoretically

actice, getting embeddings from a Matryoshka embedding model works the same way th a normal embedding model. The only difference is that, after receiving the eddings, we can optionally truncate them to a smaller dimensionality. Do note that if mbeddings were normalized, then after truncating they will no longer be, so you may to re-normalize.

truncating, you can either directly apply them for your use cases, or store them such they can be used later. After all, smaller embeddings in your vector database should t in considerable speedups!

in mind that although processing smaller embeddings for downstream tasks ieval, clustering, etc.) will be faster, getting the smaller embeddings from the model is s fast as getting the larger ones.

entence Transformers

ntence Transformers, you can load a Matryoshka Embedding model just like any other
el, but you can specify the desired embedding size using the `truncate_dim` argument.
that, you can perform inference using the `SentenceTransformers.encode` function, and
mbeddings will be automatically truncated to the specified size.

try to use a model that I trained using `matryoshka_nli.py` with `microsoft/mpnet-base`:

```
:om sentence_transformers import SentenceTransformer
:om sentence_transformers.util import cos_sim

ıtryoshka_dim = 64
›del = SentenceTransformer("tomaarsen/mpnet-base-nli-matryoshka", truncate_dim=m

ıbeddings = model.encode(
   [
       "The weather is so nice!",
       "It's so sunny outside!",
       "He drove to the stadium.",
   ]

:int(embeddings.shape)
 => (3, 64)


 Similarity of the first sentence to the other two:
imilarities = cos_sim(embeddings[0], embeddings[1:])
:int(similarities)
 => tensor([[0.8910, 0.1337]])
```

Link to the model: tomaarsen/mpnet-base-nli-matryoshka

free to experiment with using different values for `matryoshka_dim` and observe how
affects the similarities. You can do so either by running this code locally, on the cloud
as with Google Colab, or by checking out the demo.

:ences:

SentenceTransformer

SentenceTransformer.encode

util.cos_sim

Matryoshka Embeddings - Inference

ick here to see how to use the Nomic v1.5 Matryoshka Model

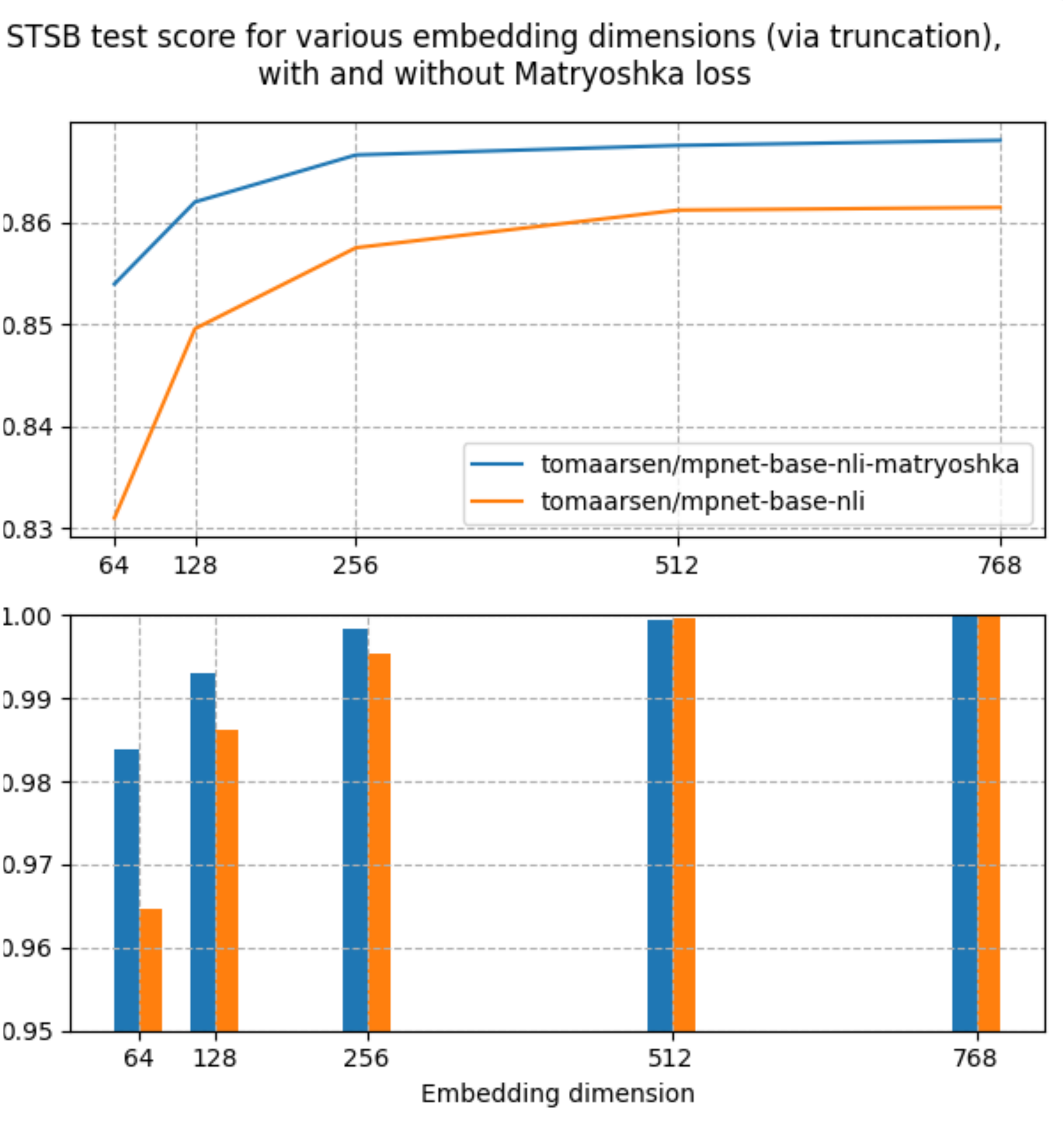
sults

that Matryoshka models have been introduced, let's look at the actual performance we may be able to expect from a Matryoshka embedding model versus a regular edding model. For this experiment, I have trained two models:

tomaarsen/mpnet-base-nli-matryoshka: Trained by running `matryoshka_nli.py` with `microsoft/mpnet-base`.

tomaarsen/mpnet-base-nli: Trained by running a modified version of `matryoshka_nli.py` where the training loss is only `MultipleNegativesRankingLoss` rather than `MatryoshkaLoss` on top of `MultipleNegativesRankingLoss`. I also use `microsoft/mpnet-base` as the base model.

of these models were trained on the AllNLI dataset, which is a concatenation of the and MultiNLI datasets. I have evaluated these models on the STSBenchmark test set ; multiple different embedding dimensions. The results are plotted in the following e:



e top figure, you can see that the Matryoshka model reaches a higher Spearman arity than the standard model at all dimensionalities, indicative that the Matryoshka el is superior in this task.

ıermore, the performance of the Matryoshka model falls off much less quickly than tandard model. This is shown clearly in the second figure, which shows the ırmance at the embedding dimension relative to the maximum performance. **Even at › of the embedding size, the Matryoshka model preserves 98.37% of the ɔrmance**, much higher than the 96.46% by the standard model.

e findings are indicative that truncating embeddings by a Matryoshka model could: 1) ficantly speed up downstream tasks such as retrieval and 2) significantly save on ge space, all without a notable hit in performance.

mo

is demo, you can dynamically shrink the output dimensions of the `nomic-ai/nomic-` `l-text-v1.5` Matryoshka embedding model and observe how it affects the retrieval ırmance. All of the embeddings are computed in the browser using 🤗 sformers.js.

Adaptive Retrieval w/ Matryoshka Embeddings

Powered by Nomic Embed v1.5 and 🤗 Transformers.js

Query

What is a panda?

Text

Once upon a time, in a land far, far away...
A panda is a large black-and-white bear native to China.
This is an example sentence.
Hello world.
Ailuropoda melanoleuca is a bear species endemic to China.
The typical life span of a panda is 20 years in the wild.
A panda's diet consists almost entirely of bamboo.
I love pandas so much!
Bamboo is a fast-growing, woody grass.
My favorite movie is Kung Fu Panda.
I love the color blue.

Compute Embeddings

ferences

Kusupati, A., Bhatt, G., Rege, A., Wallingford, M., Sinha, A., Ramanujan, V., ... & Farhadi, A. (2022). Matryoshka representation learning. Advances in Neural Information Processing Systems, 35, 30233-30249. https://arxiv.org/abs/2205.13147

Matryoshka Embeddings — Sentence-Transformers documentation. (n.d.). https://sbert.net/examples/training/matryoshka/README.html

UKPLab. (n.d.). GitHub. https://github.com/UKPLab/sentence-transformers

Unboxing Nomic Embed v1.5: Resizable Production Embeddings with Matryoshka Representation Learning. (n.d.). https://blog.nomic.ai/posts/nomic-embed-matryoshka

---

More Articles from our Blog

Train 400x faster Static Embedding Models with Sentence Transformers

By tomaarsen January… • △ 126

Visual Document Retrieval Goes Multilingual

By marco January 9, 2… • △ 64

---

unity

Preview

discussing this article