IBM Slate.rtvr.125m_v2 model for Retrieval

Documentation copied from https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/fm-slate-125m-english-rtrvr-v2-model-card.html?context=wx&audience=wdp

Model Description

The slate.125m.english.rtrvr-v2 model is a standard sentence transformers model based on bi-encoders. The model produces an embedding for a given input e.g. query, passage, document etc. At a high level, our model is trained to maximize the cosine similarity between two input pieces of text e.g. text A (query text) and text B (passage text), which result in the sentence embeddings q and p. These sentence embeddings can then be compared using cosine similarity.

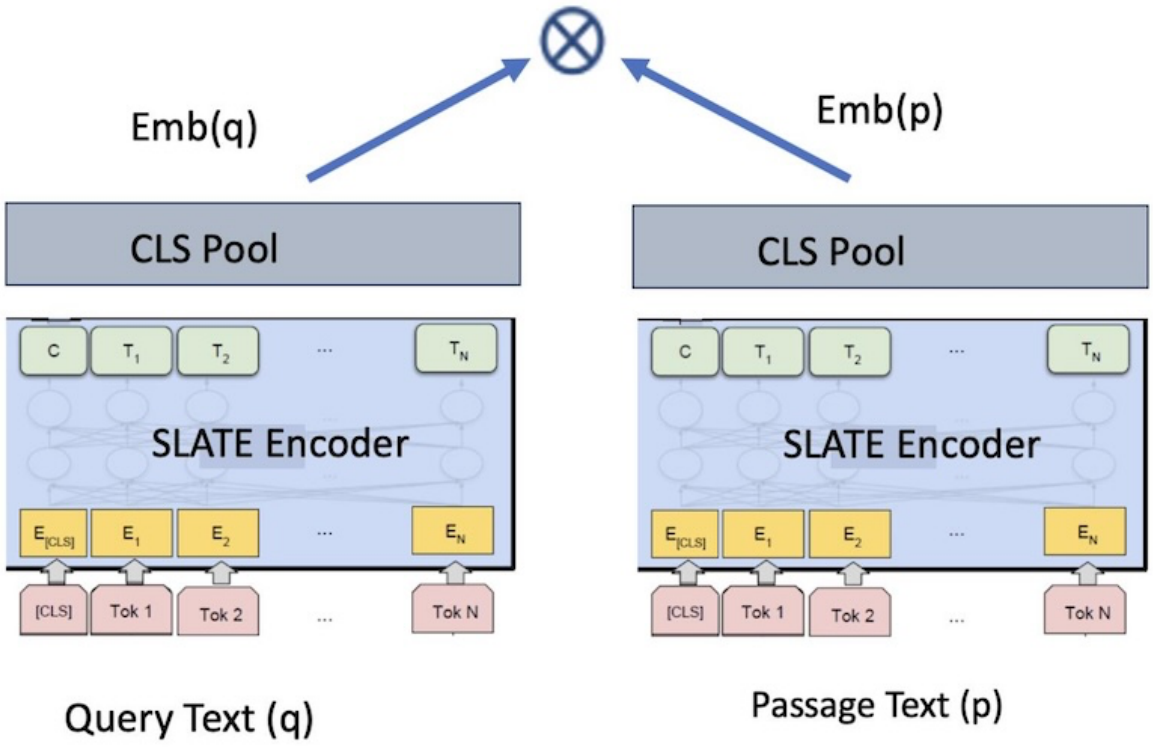

Figure 1. Bi-encoder Embeddings Model for Retrieval

Base Language Model

The underlying Language Model (LM) for our embeddings is “slate.125m.english”. It has the same architecture as a RoBERTa-base transformer model (12 layers) and has ~125 million parameters and an embedding dimension of 768. Specifically, slate.125m.english was fine tuned from “slate.125m.english” (formerly, WatBERT). Our final model is called slate.125m.english.rtrvr. The suffix denotes that the underlying model architecture was fine tuned for retrieval-based tasks.

Training Algorithm

Most embedding models that are either state-of-the-art or at the top of the MTEB leaderboard are typically trained in 3 stages:

1. Task Specific (retrieval-based) pre-training
2. Task specific fine-tuning on mined pairs
3. Fine-tuning on supervised pairs.

We follow the same approach and finally perform a model fusion by averaging the weights of different trained models.

slate.125m.english.rtrvr is produced by performing "model fusion" - averaging the weights of the following models, both trained in stages but having the following variations:

- Model 1 fine-tuned on all three stages mentioned above
- Model 2 distilled from a model fine-tuned on all stages above.

Task-specific pre-training

This stage uses the RetroMAE framework, to make our underlying LM more retrieval oriented. We initialize our base LM with slate.125m.english and continue with RetroMAE pre-training, using the data in Table 1. Our hyper-parameters are: learning rate: 2e-5, number of steps: 190000, GPUs: 24 A100 40GB.

Note: this is our base LM for the following 2 stages.

Fine-tuning with large scale unsupervised data

This model is initialized with the RetroMAE pre-trained model and is trained as follows.

We use a bi-encoder framework for training an embedding model, as in Figure 1. The RetroMAE pre-trained LM is fine-tuned with <*query, passage*> text pairs using a *contrastive loss* objective. We mine large scale pairs from various domains, as indicated in Table 2. The model is trained with diverse pairs, including classification tasks such as NLI (Natural Language Inference) which consists of matching a premise to the corresponding hypothesis. Our hyper-parameters are: learning rate: 2e-4; number of steps: 35000; GPUs: 8 A100_80GB, effective batch size: 16k pairs

Fine-tuning with small scale supervised data with hard negatives

Finally, the model is fine-tuned with high-quality supervised training pairs, with supervision coming from hard negative mining, for the retrieval task. The intermediate model checkpoints are iteratively used to mine dataset specific hard negatives, which are then used for supervised finetuning. This process aims to make the model more

robust by letting it learn from its own mistakes and helps in stabilizing with much smaller data.

We fine-tune the model by using a subset of datasets (as found via performing validation experiments on a held-out dataset) mentioned in Training Data section which are as follows: AllNLI, Squad, Stackexchange, NQ, HotpotQA, Fever and 5M subset from each of Specter, S2orc, WikiAnswers. Moreover, we also synthetically generate triples to create good quality pairs of question-answers, factual verification, etc using Mixtral-8x7B-Instruct-v0.1. To provide better performance for IBM-specific use cases, we also include pairs created from IBM Software Support data and IBM Docs.

Training hyper-parameters are learning rate: 2e-5; max query length: 64; max passage length: 512; max steps: 5000; effective batch size: 512; GPUs: 1A100_80GB

Training Data

Table 1. Pre-Training Data

Dataset	**Passages**
Wikipedia	36396918
Books Corpus	3401308
Stack Exchange	15999837

Table 2. Unsupervised and Supervised Fine-Tuning Data

Dataset	**Pairs**
SPECTER citation triplets	684100
Stack Exchange Duplicate questions (titles)	304525
Stack Exchange Duplicate questions (bodies)	250519
Stack Exchange Duplicate questions (titles+bodies)	250460
Natural Questions (NQ)	100231
SQuAD2.0	87599
PAQ (Question, Answer) pairs	64371441
Stack Exchange (Title, Answer) pairs	4067139
Stack Exchange (Title, Body) pairs	23978013

Table 2. Unsupervised and Supervised Fine-Tuning Data

Dataset	**Pairs**
Stack Exchange (Title+Body, Answer) pairs	187195
S2ORC Citation pairs (Titles)	52603982
S2ORC (Title, Abstract)	41769185
S2ORC_citations_abstracts	52603982
WikiAnswers Duplicate question pairs	77427422
SearchQA	582261
HotpotQA	85000
Fever	109810
Arxiv	2358545
Wikipedia	20745403
PubMed	20000000
Miracl En Pairs	9016
DBPedia Title-Body Pairs	4635922
Synthetic: Query-Wikipedia Passage	1879093
Synthetic: Fact Verification	9888
IBM: IBM Docs (Title-Body)	474637
IBM: IBM Support (Title-Body)	1049949

Usage

make sure you've sentence transformers installed

pip install -U sentence-transformers

from sentence_transformers import SentenceTransformer, util

```
model = SentenceTransformer('path_to_slate_model')


input_queries = [
' Who made the song My achy breaky heart? ',
'summit define']


input_passages = [
" Achy Breaky Heart is a country song written by Don Von Tress. Originally titled Don't Tell My Heart and performed by The Marcy Brothers in 1991. ",
"Definition of summit for English Language Learners. : 1 the highest point of a mountain : the top of a mountain. : 2 the highest level. : 3 a meeting or series of meetings between the leaders of two or more governments."]


query_embeddings = model.encode(input_queries)


passage_embeddings = model.encode(input_passages)


print(util.cos_sim(query_embeddings, passage_embeddings)
```

The maximum sequence length of this model is 512 tokens.

Evaluation

Baselines

For a fair comparison, we compare with the following baselines:

1. BM25 (a traditional model based on tf-idf)
2. ELSER (a commercial search algorithm provided by Elastic)
3. all-MiniLM-l6-v2: a popular open-source sentence transformers model. This model shares the same architecture as slate.30m.english.rtrvr, has been trained on more data without commercial-friendly licenses. Please see the huggingface model card for more details
4. E5-base: a recent open-source transformer model with very good performance on the BEIR benchmark. This is a base-sized model, which has the same

architecture as slate.125m.english.rtrvr. [Reference: Wang et.al., 2022: Text Embeddings by Weakly-Supervised Contrastive Pre-training]. Huggingface model card

5. E5-small: a smaller model within the open source E5 family. The embedding dimension of this model matches that of slate.30m.rtrvr (384) however it has 12 layers, and thus is larger and slightly slower. [Reference: Wang et.al., 2022: Text Embeddings by Weakly-Supervised Contrastive Pre-training]. Huggingface model card
6. BGE-base: a recent open-source transformer model with one of the best performances on the BEIR benchmark for the 768 embedding size. Huggingface model card
7. BGE-small: a recent open-source transformer model with one of the best performances on the BEIR benchmark for the 384 embedding size. Huggingface model card

We also compare the performance of these models with the older versions of the slate models, slate.125m.english.rtrvr-012024 and slate.30m.english.rtrvr-012024.

Our Evaluation benchmark: BEIR (MTEB's retrieval tab)

The BEIR benchmark contains of 15 open-source retrieval tasks evaluated under a zero-shot setting. BEIR focused on Diversity, including nine different retrieval tasks: Fact checking, citation prediction, duplicate question retrieval, argument retrieval, news retrieval, question answering, tweet retrieval, bio-medical IR, and entity retrieval. Further, it includes datasets from diverse text domains, datasets that cover broad topics (like Wikipedia) and specialized topics (like COVID-19 publications), different text types (news articles vs. Tweets), datasets of various sizes (3.6k - 15M documents), and datasets with different query lengths (average query length between 3 and 192 words) and document lengths (average document length between 11 and 635 words). BEIR uses the Normalized Cumulative Discount Gain (specifically, nDCG@10) metric for evaluation.

Long NQ

Long NQ is an IBM dataset designed for evaluating the full RAG pipeline, based on a subset of the NaturalQuestions dataset. The dev set has 300 answerable questions with a corpus of 178,891 passages from 2,345 Wikipedia documents. Long NQ also provides gold Wikipedia passages that are relevant for each question. During retrieval, the task is to obtain the relevant gold passage from the corpus for every question.

Results

Table 3. Performance comparison on the BEIR benchmark (MTEB retrieval tab)

Model	**BEIR-15 (NDCG@10)**
BM25	42.02
ELSER	49.01
all-miniLM-L6-v2	41.95
ES-small-v2	49.04
ES-base-v2	50.29
BGE-small	51.68
BGE-base	53.25
slate.30m.english.rtrvr 01.20.2024	46.91
slate.125m.english.rtrvr-01.20.2024	49.37
slate.30m.english.rtrvr 06.30.2024	49.06
slate.125m.english.rtrvr-06.30.2024	51.26

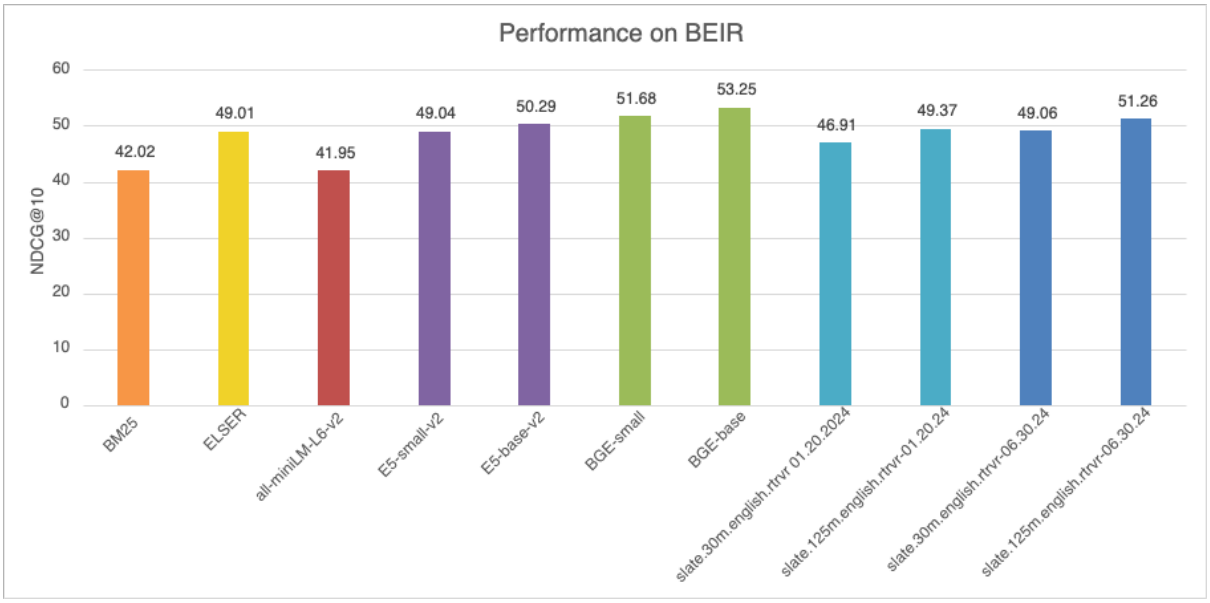

Figure 2. Performance comparison on the BEIR benchmark (MTEB retrieval tab)

Table 4. Performance comparison on the Long NQ dataset

Model	**LONGNQ (NDCG@10)**
all-miniLM-L6-v2	58.10
BGE-small	59.33
BGE-base	61.29
ES-small-v2	61.88
ES-base-v2	63.80
slate.30m.english.rtrvr 01.20.2024	59.94
slate.125m.english.rtrvr-01.20.2024	65.01
slate.30m.english.rtrvr 06.30.2024	62.07
slate.125m.english.rtrvr-06.30.2024	66.80

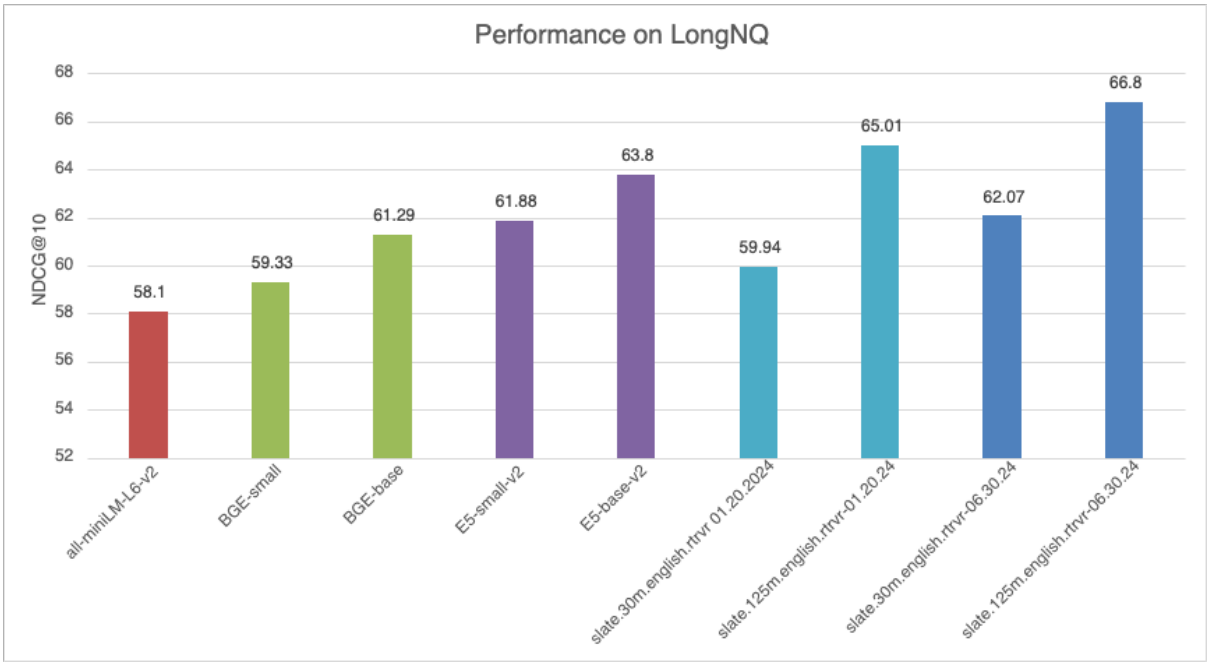

Figure 3. Performance comparison on the Long NQ dataset

Runtime Performance

The performance runtime is measured on a re-ranking task with 466 queries. For each query we re-rank the top-100 passages obtained by BM25 and we report the average time over all queries. The re-ranking was performed on a A100_40GB GPU.

Table 5. Run-time performance on re-ranking

Model	**Time/query**
all-miniLM-L6-v2	0.18 sec
E5-small	0.33 sec
E5-base	0.75 sec
BGE-small	0.34 sec
BGE-base	0.75 sec
slate.125m.english.rtrvr	0.71 sec
slate.30m.english.rtrvr	0.20 sec