SGPT: GPT Sentence Embeddings for Semantic Search

Niklas Muennighoff
Peking University
muennighoff@stu.pku.edu.cn

Abstract

Decoder transformers have continued increasing in scale reaching hundreds of billions of parameters. Due to their scale the same decoder sets state-of-the-art results on various language tasks via prompting or fine-tuning. Yet, these large foundation models remain unusable for the related fields of semantic search and sentence embeddings. This prevents possibly new state-of-the-art results and forces organizations to train and maintain separate models. To this end, we propose SGPT to use decoders for sentence embeddings and semantic search via prompting or fine-tuning. At 5.8 billion parameters SGPT improves on the previously best sentence embeddings by a margin of 7% and outperforms a concurrent method with 175 billion parameters as measured on the BEIR search benchmark. Code, models and result files are freely available at https://github.com/Muennighoff/sgpt.

1 Introduction

Semantic search consists of two parts: *Search* refers to finding the top k answers from a document corpus given a query. *Semantic* refers to understanding the documents and queries beyond keywords. Transformers [45] are the dominant semantic architecture [8, 44] competing with non-semantic models like BM25 [41]. Search applications like Google [26] or Bing [54] rely on transformers to provide semantically relevant results. However, they have been limited to BERT-like encoder-only transformers [26, 54, 10, 39, 12, 29].

Meanwhile, GPT-like decoder-only transformers [35] have been the focus of recent scaling efforts of up to 540 billion parameters [9]. Increasing language model parameters has been repeatedly shown to improve downstream zero-shot and fine-tuning performance on a variety of language tasks [5, 37, 9]. For example, increasing scale has allowed decoder-only transformers to outperform all encoder-only and catch-up with encoder-decoder transformers on the SuperGLUE benchmark [47, 9].

However, the related fields of semantic search and language embeddings have not been part of the proliferation of decoders. They are dominated by comparatively small encoders [44], as it remains unclear how to extract semantically meaningful embeddings from decoders and use them for semantic search. Methods to do so are desirable for two reasons:

Performance Taking advantage of the available scale of decoders has the potential to produce new state-of-the-art results in search. Available encoders are orders of magnitude smaller [10, 23, 38]. Google search, for example, processes an estimated 4 billion searches daily [19], thus better search models could have wide-reaching impacts.

Compute Savings Large-scale pre-trained decoders have been reused for different tasks via prompting or fine-tuning [5, 37, 9]. A well-performing method to extract embeddings from billion parameter decoders may prevent the need to train and maintain separate encoder and decoder models. Training just one large decoder and reusing it for search prevents additional cost to the environment [2].

Preprint. Under review.

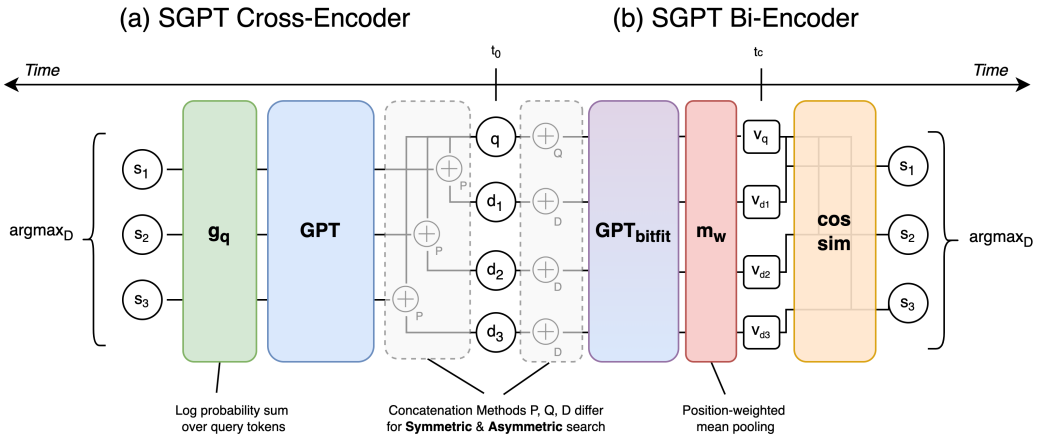

Figure 1: Given a query q, documents d_{1-3}, SGPT ranks the documents with scores s_{1-3}. (a) The Cross-Encoder concatenates queries and documents and encodes them together. Scores are extracted log probabilities. (b) The Bi-Encoder separately encodes queries and documents. Resulting document vectors v_{1-3} can be cached and retrieved at time t_c, when a new query comes in. Scores are cosine similarities.

In this work, we propose SGPT to apply decoder-only transformers to semantic search and extract meaningful sentence embeddings from them. We distinguish four settings: Cross-Encoder vs Bi-Encoder, Symmetric vs Asymmetric. See Figure 1 and §2.

In the Bi-Encoder setting, we propose SGPT-BE using position-weighted mean pooling and contrastive fine-tuning of only bias tensors (BitFit [53]). We show that BitFit is competitive with full fine-tuning performance for both encoders (SBERT) [39] and decoders (SGPT) despite changing <0.1% of pre-trained parameters. When controlling for size, our decoders closely trail the performance of encoders. When scaling up, SGPT-BE-5.8B sets state-of-the-art results on BEIR and USEB for asymmetric and symmetric search.

In the Cross-Encoder setting, we propose SGPT-CE using log probability extraction of pre-trained GPT models. The method is applicable to symmetric or asymmetric search by changing the prompt. At scale, the model sets an unsupervised state-of-the-art on BEIR.

In summary, our contributions are three-fold:

- For SGPT-BE in §4, we develop a new pooling method and show the usefulness of bias-only fine-tuning for embeddings. At 5.8B parameters, it produces the best natural language embeddings available by a margin of 7% for the example of semantic search.
- For SGPT-CE in §3, we show how to use GPT for search via log probabilities without fine-tuning. At 6.1B parameters, it has the best unsupervised performance on BEIR by a margin of 8%.
- We provide free, more performant alternatives to commonly used endpoints, such as OpenAI's Search and Similarity Embeddings and the OpenAI Search endpoint available at `https://github.com/Muennighoff/sgpt`.

2 Related Work

In this section, we explain two dimensions fundamental to our work: Cross-Encoders vs Bi-Encoders and Symmetric vs Asymmetric Search. We highlight work in those areas relevant to ours.

Cross-Encoders encode query and document at the same time. BERT [10] is used as a Cross-Encoder by separating the query from the document with a $[SEP]$ token. They are then passed through the transformer network together. Each new query requires k forward passes given a corpus of k documents. There is no existing literature on using GPT models as Cross-Encoders, however,

we suspect the OpenAI Search API [32] uses a GPT-based Cross-Encoder. We include results based on querying their API, as well as a BERT-based state-of-the-art Cross-Encoder in our benchmarks.

Bi-Encoders encode query and document separately. SBERT [39] extends BERT to the Bi-Encoder setting via supervised fine-tuning and a pooling operation across the sequence output. The resulting document vectors can be cached. A new query requires only one forward pass through the transformer to produce the query vector. The query vector can then be scored against the cached document vectors with a similarity function. Embeddings from Bi-Encoders can be used for non-search tasks such as clustering or as input features of machine learning models [40]. While non-semantic models like keyword-based BM25 [41] remain extensively used, the field increasingly focuses on neural models using transformers [45]. Contriever [20] shows the utility of unsupervised contrastive training for pre-trained encoders. Their best model that we compare with also adds supervised fine-tuning as a third training stage. GTR [29] explores the effect of scaling up encoders on semantic search tasks also in a three-stage training setup. They find that despite keeping the embedding size fixed, more parameters increase the performance of encoders. Usage of GPT models as Bi-Encoders remains limited. Rather, there has been interest in using them as generators to produce search training data for encoders [42, 3]. Previous work has studied the differences in embeddings produced by various language models including BERT and GPT [11, 22, 6]. They have found GPT-2 embeddings to underperform on various word embedding benchmarks [11]. Concurrently to our work, the first trained GPT-based Bi-Encoder, cpt-text, was proposed [27]. They use a pre-trained decoder and employ two additional training stages, contrastive unsupervised pre-training and supervised fine-tuning. Their models are used for the OpenAI Similarity and Search Embeddings API [31]. Our Bi-Encoders differ from theirs in that we simplify the training process to only fine-tuning, use a novel pooling method and only train bias parameters. cpt-text is most similar to our Bi-Encoders, hence we include their results in our benchmarks.

Cross-Encoders tend to outperform Bi-Encoders [43], but are slower as vectors cannot be cached and reused. To balance the trade-offs, multi-stage architectures have been proposed [30, 34, 21]. In a two-stage re-ranking setup, the first model processes the entire corpus and the second model is only used on the top k documents returned by the first. In §3, we use Bi-Encoder (BM25 [41]) + Cross-Encoder re-ranking.

Asymmetric Search means queries and documents are not interchangeable. Finding answers given a question is an asymmetric search problem. Commonly, documents are much longer than queries [44]. We evaluate asymmetric search experiments on BEIR [44], a recently proposed benchmark consisting of 19 asymmetric search datasets.

Symmetric Search means queries and documents are interchangeable. Finding duplicate questions, where both queries and documents are questions, is a symmetric search problem. We evaluate symmetric search experiments on USEB [49], Quora from BEIR [44] and STS-B [7]. In Quora, queries are question titles and documents are question texts. They are often the same with average word lengths of 9.53 and 11.44, respectively [44]. Hence, we consider it more of a symmetric search task. We include Quora in both symmetric and asymmetric experiments.

3 SGPT Cross-Encoder

3.1 Asymmetric Search

3.1.1 Method

Given a query q, and a document corpus D, we are interested in the most likely document d^*. Using Bayes' Theorem this can be expressed as:

$$d^* = \arg\max_{d \in D} P(d|q) = \arg\max_{d \in D} \frac{P(q|d)P(d)}{P(q)} = \arg\max_{d \in D} P(q|d)P(d) \tag{1}$$

Note that $P(q)$ is irrelevant as it is always the same when taking the $\arg\max$ over D. Due to variable document lengths and contents it is easier to compare $P(q|d)$ than $P(d|q)$. We hence compute the joint probability of the query tokens $q_{i,..,n}$ given the document tokens embedded in a prompt P as $p(q_i, ..., q_n|p_1, ..., p_{i-1})$ ignoring $P(d)$. As long as $P(d)$ does not vary excessively across the corpus D, this simplification should produce reasonable scores.

Training (→)	Unsupervised							Unsupervised + Supervised
Ranking (→)	Re-rank Top 0	Re-rank Top 10		Re-rank Top 100				
Model (→)	[41]	SGPT-CE		OpenAI Search		SGPT-CE		[44]
Dataset (↓)	**BM25**	**2.7B**	**6.1B**	**Ada**	**Davinci**	**2.7B**	**6.1B**	**BM25+CE♣**
MS MARCO	0.228	0.249‡	0.253‡	L	L	0.278‡	0.290‡	**0.413**‡
TREC-COVID	0.688	0.708	0.705	0.616	0.627	0.762	**0.791**	0.757
BioASQ	0.488	0.517	0.518	L	L	0.546	**0.547**	0.523
NFCorpus	0.306	0.319	0.323	0.336	**0.358**	0.333	0.347	0.350
NQ	0.326	0.358	0.366	L	L	0.384	0.401	**0.533**
HotpotQA	0.602	0.647	0.649	L	L	0.691	0.699	**0.707**
FiQA-2018	0.254	0.305	0.313	0.320		0.369	**0.401**	0.347
Signal-1M (RT)	0.330	**0.343**	0.342	0.313		0.320	0.323	0.338
TREC-NEWS	0.405	0.409	0.418	L	L	0.434	**0.466**	0.431
Robust04	0.425	0.438	0.444	L	L	0.449	**0.480**	0.475
ArguAna	**0.472**	0.379	0.376	0.166		0.293	0.286	0.311
Touché-2020	**0.347**	0.335	0.335	0.332		0.256	0.234	0.271
CQADupStack	0.326	0.364	0.367	0.328		0.405	**0.420**	0.370
Quora	0.808	0.810	0.810	0.786		0.792	0.794	0.825
DBPedia	0.320	0.341	0.340	L	L	0.367	0.370	**0.409**
SCIDOCS	0.165	0.176	0.177	0.161		0.186	**0.196**	0.166
FEVER	0.649	0.706	0.723	L	L	0.698	0.725	**0.819**
Climate-FEVER	0.186	0.179	0.189	L	L	0.138	0.161	**0.253**
SciFact	0.611	0.653	0.657	**0.727**		0.676	0.682	0.688
Average	0.428	0.444	0.447			0.450	0.462	**0.476**
Best on	2	1	0	1	1	0	**7**	5

Table 1: Re-ranking performances on BEIR [44]. OpenAI Search is to be distinguished from the OpenAI Embeddings endpoint. Please refer to Table 6 in the Bi-Encoder section for a benchmark with the OpenAI Embeddings endpoint. Results on the Search endpoint were produced in October 2021. Scores are **nDCG@10**. *L*: Dataset is too large for OpenAI's endpoint. ‡: Used for prompt-tuning (SGPT) or training (BM25+CE). ♣: Results from [44]. Other scores are from us. Average scores do not include MS MARCO.

Model Name	Ada (S)	Babbage (M)	Curie (L)	Davinci (XL)
Parameters	350M (300M)	1.3B (1.2B)	6.7B (6B)	175B (175B)

Table 2: OpenAI model parameter estimates. Based on comparing the embedding sizes from the OpenAI docs with the dimensions provided in [5]. In brackets are numbers for cpt-text models recently provided in [27]. They differ likely due to removing the language modeling head.

In practice, we use log probabilities [5, 36], computed via the log of the softmax of the model output. To have a constant query length $n+1-i$ and avoid abrupt text changes, documents are truncated from the left until the input fits the model's maximum sequence length. We apply these methods to re-rank top k documents returned by BM25 [41]. While re-ranking with BM25 bottlenecks performance, it speeds up experiments. It is not a necessary part of the architecture and therefore not depicted in Figure 1.

We experiment with publicly available pre-trained decoder transformers with 125M, 1.3B, 2.7B and 6.1B parameters [1, 48].

3.1.2 Results

We perform a search over 12 prompts using the MSMARCO [28] dataset as provided in BEIR [44]. The prompts and results are in Appendix §B.1. We select the prompt with the best score, P_G.

In Table 1, we benchmark the resulting SGPT-CE (SGPT-Cross-Encoder). We compare with OpenAI's Search endpoint, which is to be distinguished from their Embeddings endpoint. Please refer to Table 6 in the Bi-Encoder section for a benchmark with the OpenAI Embeddings endpoint. We provide

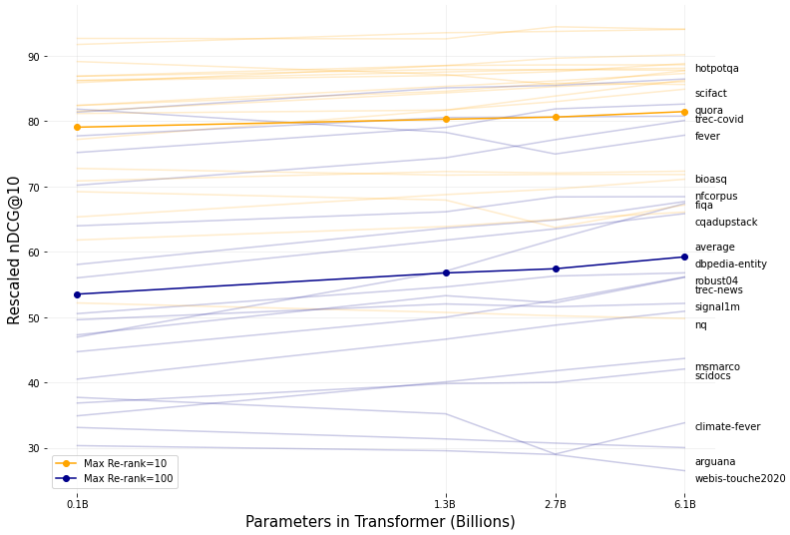

Figure 2: Scaling behavior across parameters and re-ranking for SGPT-CE on BEIR. Scores are rescaled nDCG@10 based on bounds defined in Appendix §A. Dataset labels are ordered by the *Max Re-rank=100* 6.1B performance. The higher on the y-axis, the more bottlenecked is the Cross-Encoder by BM25's performance. Average scores include MS MARCO.

parameter estimates for the OpenAI model names in Table 2. We also compare with the current state-of-the-art on BEIR [44], a BERT-based Cross-Encoder. BM25+CE consists of a pre-trained BERT model that is further fine-tuned on MS-MARCO [28] in a supervised fashion [44]. SGPT-CE consists solely of the pre-trained GPT model. However, SGPT-CE-6.1B has almost 15x more parameters than BM25+CE significantly increasing latency. In the *Re-rank Top 100* setting, the top 100 documents as returned by BM25 are re-ranked by the respective model. While SGPT-CE-6.1B wins on more datasets than the encoder-based state-of-the-art, its average score is worse. This can be alleviated by not using the same prompt P_G for all datasets. We show in §3.2 that SGPT-CE-6.1B can beat BM25+CE on Quora by changing the prompt.

In Figure 2, we investigate how performance scales with model size. As we are in a re-ranking setup, the Cross-Encoder performance is bounded by the documents returned by BM25. We provide the BM25 bounds and additional model results in Appendix §A. In a *Re-rank Top 10* setting, the model is significantly bottlenecked by BM25. SGPT-CE-6.1B reaches around 80% of the maximum possible performance. We hence observe high jumps in performance for datasets like HotpotQA [52] or TREC-COVID [46] as we move to top 100. In fact, the 0.791 nDCG@10 on TREC-COVID in Table 1 is not possible in a *Re-rank Top 10* setting as the bound is at 0.750. From the results, we infer that performance scales both as we re-rank more documents or increase model size.

3.2 Symmetric Search

We use the same methods outlined in §3.1.1, but adapt the prompt for symmetric search. We show this on the example of Quora in Table 3. In §2, we have explained why Quora is closer to symmetric search than asymmetric search. We search over several prompts on the smaller 125M parameter model and use the best one on the large model. By doing so, SGPT-CE-6.1B improves by 6% outperforming all Quora results in Table 1. We hypothesize that further customizing the prompt for each dataset could significantly improve performance. However, we highlight that searching prompts for all possible input types may not be feasible in practice and is not considered true few-shot learning [33]. Hence, the prompt we find for Quora may not generalize well to other symmetric search datasets, a key limitation of this method.

Id	Python	125M	6.1B
G	Documents are searched to find matches with the same content.\nThe document "{doc}" is a good search result for "{query}	0.764	0.794
quoraA	Questions are searched to find matches with the same content.\nThe question "{doc}" is a good search result for "{query}	0.766	
quoraB	Below are two similar questions asking the same thing.\nThe question "{doc}" is similar to "{query}	0.751	
quoraC	These two questions are the same: 1. {doc} 2.{query}	0.740	
quoraD	Question Body: {doc} Question Title:{query}	0.782	**0.830**
quoraE	Question Body: {shortdoc} Question Title: {shortquery}\n Question Body: {doc} Question Title: {query}	0.773	

Table 3: SGPT-CE symmetric search results on Quora. The sum of log probabilities from {query} is used as the re-rank score. Overflowing tokens are truncated from the left of {doc}. Top 100 documents are re-ranked. Scores are **nDCG@10**.

4 SGPT Bi-Encoder

4.1 Symmetric Search

4.1.1 Method

Like in §3.1.1, we first experiment with decoder transformers that have only gone through unsupervised pre-training. In the Bi-Encoder setting, a pooling operation is commonly applied to the model's hidden states to reduce them to a vector whose size is irrespective of sequence length. SBERT [39] showed that a *MEAN* pooling mechanism outperforms *[CLS]* and *MAX* strategies for a BERT encoder. Due to the causal attention mask in an auto-regressive decoder transformer, tokens do not attend to future tokens like in an encoder transformer. Hence, only the last token has attended to all tokens in a sequence. To account for this information mismatch, we propose to give later tokens a higher weight using a position-weighted mean pooling method:

$$v = \sum_{i=1}^{S} w_i h_i \quad \text{where} \quad w_i = \frac{i}{\sum_{i=1}^{S} i} \tag{2}$$

where S is the sequence length, h_i the ith hidden state and v the query or document embedding. We compare weighted mean pooling with last token pooling, where the hidden state of the final token is the embedding, and regular mean pooling.

We follow recent work [16, 15, 20, 27] and perform supervised contrastive learning with in-batch negatives. Given matching query-doc pairs $\{q^{(i)}, d^{(i)}\}_{i=1}^{M}$, we optimize the cost function:

$$J_{\text{CL}}(\theta) = \frac{1}{M} \sum_{i=1}^{M} \log \frac{\exp(\tau \cdot \sigma(f_\theta(q^{(i)}), f_\theta(d^{(i)})))}{\sum_{j=1}^{M} \exp(\tau \cdot \sigma(f_\theta(q^{(i)}), f_\theta(d^{(j)})))} \tag{3}$$

where f_θ is the SGPT model outputting a fixed-size vector, σ cosine similarity and τ a temperature parameter set to 20 in our experiments. We use GradCache [14] to train with large batch sizes in a limited memory setting. We train on SNLI [4] and MNLI [51]. We limit the model sequence length to 75 tokens during both training and inference.

We fine-tune only bias parameters and freeze the rest of the model. This has been recently proposed as BitFit [53] for BERT encoders. It has been shown to be competitive with full fine-tuning in various scenarios [18, 50, 24]. Table 4 shows the number of parameters trained for BitFit models. Due to fewer gradient updates, BitFit significantly reduces GPU memory and time required per step. Further, adding a BitFit checkpoint to an instance with an existing full model will only require storing the different biases. An instance already serving a 22.5GB fp32 GPT-J-6B model requires an additional 22MB of storage to serve an SGPT-5.8B-bitfit model.

Model Name	SBERT-Base	SGPT-125M	SGPT-1.3B	SGPT-2.7B	SGPT-5.8B
Transformer (T.)	BERT	GPT-Neo	GPT-Neo	GPT-Neo	GPT-J
Total params	109M	125M	1.3B	2.7B	5.8B
Bias tensors per T. layer	8	5	5	5	3
Bias params	103K	74K	395K	658K	692K
Bias params %	0.094%	0.060%	0.030%	0.025%	0.012%

Table 4: SGPT parameter overview. Due to the removal of the final language modeling head SGPT-BE-5.8B has 206M parameters less than SGPT-CE-6.1B or GPT-J-6.1B. GPT-Neo models tie the language modeling head weights with the input embeddings, hence there is no parameter difference.

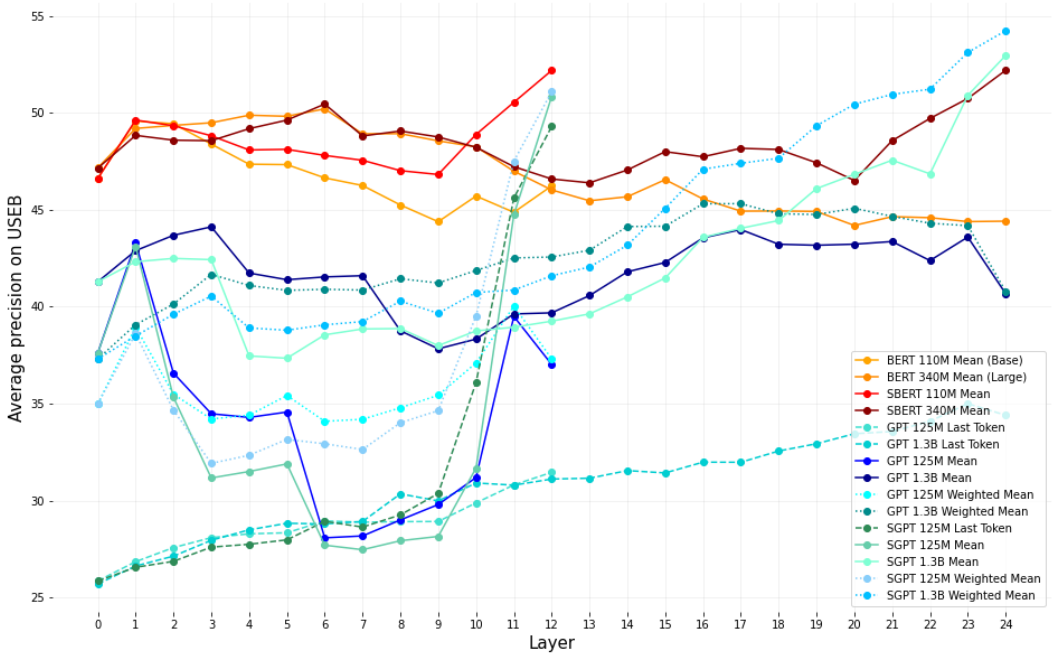

Figure 3: Performance on USEB [49] by taking the embeddings from certain layers. *S* models are fine-tuned on the same data with the same hyperparameters. Dashed, solid and dotted lines are last token, mean and weighted mean pooling, respectively. Shades of red are transformer encoders, while shades of blue are decoders. The 0th layer is the embeddings prior to the first transformer layer.

4.1.2 Results

Figure 3 shows average precisions on USEB [49] across different methods and layers. Similar to previous work [11], we find that in the unsupervised setting, decoder transformers (GPT) strongly underperform encoders (BERT). However, after fine-tuning on the same dataset with the same hyperparameters, decoders (SGPT) with 125M parameters closely trail the 110M parameter encoder (SBERT) for the 12th layer. Weighted mean pooling outperforms mean and last token pooling for SGPT 125M. When increasing SGPT size ten-fold, the last layer performance (24th layer) increases beyond that of SBERT models. The performance difference of weighted mean pooling compared to mean pooling further widens for SGPT 1.3B.

Table 5 provides performance on the individual USEB datasets, Quora and STS-B. STS-B scores should not be the focus of comparison due to the drawbacks highlighted in [49]. Despite training on less than 0.1% of parameters BitFit models are within +2 to -2% of fully fine-tuned ones. BitFit degrades performance more for decoders than encoders. This could be due to the missing bias parameters, see Table 4. [53] highlights the importance of the query bias vector for BERT, which is not present for SGPT models. *SGPT-5.8B-weightedmean-nli-bitfit* sets an out-of-domain state-of-the-art on USEB, but is outperformed by models trained in-domain in [49]. We observed performance gains by increasing the training batch size. *SGPT-5.8B-weightedmean-nli-bitfit* is trained with a

Dataset (→) / Method (↓)	AskU	CQA	TwitterP			SciDocs					Avg	Quora	STS-B
			TURL	PIT	Avg	Cite	CC	CR	CV	Avg			
OOD Unsupervised													
BM25 ♢	53.4	13.3	71.9	70.5	71.2	58.9	61.3	67.3	66.9	63.6	50.4	80.8†	
OOD Unsupervised + OOD Supervised (NLI)													
SBERT-base-nli-v2-bs64	52.8	11.6	75.5	71.5	73.5	68.0	70.6	71.1	73.5	70.8	52.2	78.9	83.9
SBERT-base-nli-v2-bf-bs64	53.8	11.7	**76.6**	72.9	74.8	67.5	70.6	70.8	73.0	70.5	52.7	78.8	81.8
SGPT-0.1B-weightedmean-nli-bs64	54.9	11.2	72.7	66.0	69.3	66.2	68.9	68.9	71.7	68.9	51.1	79.5	81.0
SGPT-0.1B-weightedmean-nli-bf-bs64	54.9	10.8	72.3	65.3	68.8	64.7	67.4	68.0	70.8	67.8	50.6	77.5	78.6
SGPT-0.1B-weightedmean-nli-bf-bs1024	55.7	11.1	72.8	66.5	69.6	65.1	67.8	68.6	70.5	68.0	51.1	79.0	79.5
SGPT-1.3B-weightedmean-nli-bf-bs1024	56.0	13.5	75.4	70.7	73.1	70.1	72.9	73.2	75.0	72.8	53.8	82.3	83.9
SGPT-2.7B-weightedmean-nli-bf-bs1024	57.5	14.0	75.8	71.0	73.4	72.3	75.4	74.7	76.5	74.7	54.9	82.6	84.7
SGPT-5.8B-weightedmean-nli-bf-bs1024	57.1	**16.0**	76.5	76.0	76.3	**75.0**	**78.2**	**77.1**	**78.3**	**77.2**	**56.6**	**84.7**	**85.7**
OOD Unsupervised + OOD Unsupervised + OOD Supervised (NLI, [27])													
Ada Similarity (Mar 2022)	55.9	13.6	**76.6**	76.2	76.4	66.7	71.0	71.3	73.7	70.7	54.1	82.2	
Curie Similarity (Mar 2022)	**57.3**	15.8	76.3	**78.9**	**77.6**	66.3	71.7	71.3	73.0	70.6	55.3	83.1	
Davinci Similarity (June 2022)	55.9		76.3	78.0	77.1								

Table 5: Results on USEB, Quora and STS-B. Metrics are **average precision** for USEB, **nDCG@10** for Quora and **Spearman** correlation for STS-B. **bf=BitFit**. **bs=Batch Size**. **OOD=Out-of-domain**, to contrast these numbers from in-domain numbers in [49]. However, fragments may be in-domain due to the large pre-training data of the transformer models. *SGPT-0.1B-weightedmean-nli* performs 2% worse than *SBERT-base-nli-v2* on USEB, but improves on Quora by 1%. Note that there is still a size difference of 14% between the two models. ♢: Results from [49] except when marked with †. CQADupstack and SciDocs differ from the same-name datasets in BEIR.

batch size of 1024. In Appendix §A, we provide results using a lower batch size and additional ablations. Results for Ada and Curie were obtained by querying the OpenAI Similarity Embeddings API in March 2022. They correspond to the cpt-text similarity models from [27] and we provide their parameters in Table 2.

4.2 Asymmetric Search

4.2.1 Method

If not otherwise specified, we follow the same setup as in §4.1.1. For asymmetric search, we train on MS-MARCO [28]. We limit the model sequence length to 300 tokens during both training and inference. We follow concurrent work [27] and add enclosing brackets to help the model distinguish between query and document. We embed the tokens of query q in two brackets as $[q_{0-n}]$. For documents, we use curly brackets: $\{d_{0-n}\}$. We add the token ids of the brackets to the already tokenized text to avoid the tokens intermingling. We refer to these special brackets as $specb$.

4.2.2 Results

Table 6 benchmarks **SGPT-BE-5.8B** (*SGPT-5.8B-weightedmean-msmarco-specb-bitfit*) on BEIR [44] with: (*a*) **BM25** [41], a non-semantic fast baseline (*b*) **SGPT-CE-6.1B** from §3 (*c*) **BM25+CE** [44], the current overall state-of-the-art on BEIR (*d*) **TAS-B** [17], the original Bi-Encoder state-of-the-art on BEIR (*e*) **Contriever** [20], a similar training scheme as [27] but using an encoder transformer (*f*) **GTR-XXL** [29], the current Bi-Encoder state-of-the-art on BEIR with 4.8 billion parameters using the BERT-like encoder transformer of T5 [38] (*g*) **cpt-text**, a GPT-like decoder transformer architecture concurrently proposed in [27]. Corresponding parameter estimates are in Table 2.

SGPT-5.8B achieves the best average nDCG@10 both on the BEIR subset selected in [27] and on the full BEIR benchmark. It outperforms the roughly same-sized cpt-text-L and the 30x larger cpt-text-XL by 8.1% and 4.2%, respectively. Yet, cpt-text models have gone through an additional unsupervised training stage [27] and are fully trained. SGPT-BE-5.8B fine-tunes just 700K parameters, 0.0004% of the parameters fine-tuned for cpt-text-XL [27]. See Table 2 for sizes. We suspect much of the difference to come from the cpt-text model's inferior last token pooling as shown in Figure 3. Further, we suspect that the benefits of the additional unsupervised contrastive pre-training stage diminish when followed by supervised contrastive fine-tuning. SGPT-BE-5.8B improves on the overall state-of-the-art, a Cross-Encoder, by 3%. It improves on the previously best sentence embeddings (Bi-Encoder) on BEIR, GTR-XXL, by 7%. However, these improvements come at a significant cost. GTR-XXL has 20% fewer parameters and its embeddings have 768 dimensions. SGPT-BE-5.8B produces embeddings with 4096 dimensions, hence requiring about 5x more storage. It took the model six days on one Nvidia A100 GPU to encode the entire BioASQ corpus with 15M

Training (→)	Unsupervised		U. + U.	Unsupervised + Supervised			Unsupervised + Unsupervised + Supervised			
Model (→)	[41]	SGPT-CE	[27]	[44]	[17]	SGPT-BE	[20]	[29]	OpenAI Embeddings [27]	
Dataset (↓)	**BM25**	**SGPT-6.1B**	**cpt-text-L♥**	**BM25+CE♣**	**TAS-B♣**	**SGPT-5.8B**	**Contriever♠**	**GTR-XXL♦**	**cpt-text-L♥**	**cpt-text-XL♥**
MS MARCO	0.228	0.290		0.413‡	0.408‡	0.399‡		**0.442**‡		
TREC-COVID	0.688	0.791	0.427	0.757	0.481	**0.873**	0.596	0.501	0.562	0.649
BioASQ	0.488	**0.547**		0.523	0.383	0.413		0.324		
NFCorpus	0.306	0.347	0.369	0.350	0.319	0.362	0.328	0.342	0.380	**0.407**
NQ	0.326	0.401		0.533	0.463	0.524	0.498	**0.568**		
HotpotQA	0.602	0.699	0.543	**0.707**	0.584	0.593	0.638	0.599	0.648	0.688
FiQA-2018	0.254	0.401	0.397	0.347	0.300	0.372	0.329	0.467	0.452	**0.512**
Signal-1M (RT)	0.330	0.323		**0.338**	0.289	0.267		0.273		
TREC-NEWS	0.405	0.466		0.431	0.377	**0.481**		0.346		
Robust04	0.425	0.480		0.475	0.427	**0.514**		0.506		
ArguAna	0.472	0.286	0.392	0.311	0.429	0.514	0.446	**0.540**	0.469	0.435
Touché-2020	**0.347**	0.234	0.228	0.271	0.162	0.254	0.230	0.256	0.309	0.291
CQADupStack	0.326	**0.420**		0.370	0.314	0.381	0.345	0.399		
Quora	0.808	0.794	0.687	0.825	0.835	0.846	0.865	**0.892**	0.677	0.638
DBPedia	0.320	0.370	0.312	0.409	0.384	0.399	0.413	0.408	0.412	**0.432**
SCIDOCS	0.165	0.196		0.166	0.149	**0.197**	0.165	0.161	0.177†	
FEVER	0.649	0.725	0.638	**0.819**	0.700	0.783	0.758	0.740	0.756	0.775
Climate-FEVER	0.186	0.161	0.161	0.253	0.228	**0.305**	0.237	0.267	0.194	0.223
SciFact	0.611	0.682	0.712	0.688	0.643	0.747	0.677	0.662	0.744	**0.754**
Sub-Average	0.477	0.499	0.442	0.520	0.460	**0.550**	0.502	0.516	0.509	0.528
Average	0.428	0.462		0.476	0.395	**0.490**		0.458		
Best on	1	2	0	3	0	**5**	0	3	0	4

Table 6: Comparison of BEIR state-of-the-art models. Keep model size, latency and training time in mind when inspecting this table. Further, this table compares 2 Cross-Encoders and 8 Bi-Encoders, whose respective trade-offs should be considered. Scores are **nDCG@10**. ‡: In-domain performance. ♣: Results from [44]. ♠: Results from [20]. ♦: Results from [29]. ♥: Results from [27] except when marked with †. Other scores are from us. Average scores do not include MS MARCO.

documents and an average 200 words each [44]. Its comparatively low performance on BioASQ may be improved by increasing the sequence length limit beyond 300, however, requiring additional compute. For SGPT-CE-6.1B, the sequence length limit was 2048 for the combined prompt on all datasets. The high performance on TREC-COVID for SGPT models could be due to the different pre-training datasets. The SGPT pre-training dataset, *The Pile* [13], contains data until mid-2020. This may give the models an information advantage on Covid-19. Lastly, we highlight that on Quora SGPT-BE-5.8B-msmarco is outperformed by SGPT-BE-5.8B-nli from Table 5. Given our classification of Quora as a symmetric search task in §2, this supports our overall distinction between asymmetric and symmetric search. We advise users of our models to classify their tasks as symmetric or asymmetric and use the appropriate model. For non-classifiable embedding tasks, both may work, but we recommend experimenting with embeddings from the symmetric models in §4.1 first.

5 Conclusion and Future Work

This work presented SGPT. Building on SBERT, we proposed modifications to GPT models to use them as Cross- or Bi-Encoders for semantic search.

SGPT-BE uses position-weighted mean pooling and fine-tuning of only bias tensors. At scale, it produces new state-of-the-art sentence embeddings. The model can be used for semantic search or other embedding tasks. We recommend using SGPT-BE-5.8B when compute and storage are of high availability and maximum performance is desired.

SGPT-CE extracts log probabilities of pre-trained GPT models to produce unsupervised state-of-the-art search results. The setup presented can only be used for semantic search. Storage can be limited, but compute should be of high availability for SGPT-CE-6.1B. The prompt and max re-rank parameter can be adjusted depending on performance and latency requirements.

Future research could fine-tune a GPT Cross-Encoder on MSMARCO similar to the BM25+CE model. We suspect that this should outperform the presented non-fine-tuned SGPT-CE model as well as SGPT-BE if enough documents are re-ranked. Further, the combination of SGPT with GPT for generative search results could be interesting. Possibly, SGPT embeddings could be injected into GPT models to generate answers. Lastly, a detailed study of the disadvantages of the missing biases in large GPT models could be helpful to consider their inclusion in the training of future large language models.

offensive content. OpenAI models are licensed by OpenAI API to customers via a non-exclusive, non-sublicensable, non-transferable, non-assignable, revocable license.[2]

B.3 Computational Cost

We use the OpenAI API to evaluate Search and Embeddings endpoints. We used tokens equivalent to around 5,000 USD. For SGPT experiments, we use one node with 8x NVIDIA A100 Tensor Core GPU with 40GB memory. For SGPT-CE the evaluation on the entire BEIR suite took around two weeks for the 5.8B model. For SGPT-BE symmetric search training took 21 hours, while asymmetric training took 60 hours for the 5.8B model. Our cluster was provided by Oracle.

[2] https://openai.com/api/policies/terms/