Web-Scale Semantic Product Search with Large Language Models

Aashiq Muhamed[1(✉)], Sriram Srinivasan[1], Choon-Hui Teo[1], Qingjun Cui[1], Belinda Zeng[2], Trishul Chilimbi[2], and S. V. N. Vishwanathan[1]

[1] Amazon, Palo Alto, CA, USA
{muhaaash,srirs,choonhui,qingjunc,vishy}@amazon.com
[2] Amazon, Seattle, WA, USA
{zengb,trishulc}@amazon.com

Abstract. Dense embedding-based semantic matching is widely used in e-commerce product search to address the shortcomings of lexical matching such as sensitivity to spelling variants. The recent advances in BERT-like language model encoders, have however, not found their way to realtime search due to the strict inference latency requirement imposed on e-commerce websites. While bi-encoder BERT architectures enable fast approximate nearest neighbor search, training them effectively on query-product data remains a challenge due to training instabilities and the persistent generalization gap with cross-encoders. In this work, we propose a four-stage training procedure to leverage large BERT-like models for product search while preserving low inference latency. We introduce query-product interaction pre-finetuning to effectively pretrain BERT bi-encoders for matching and improve generalization. Through offline experiments on an e-commerce product dataset, we show that a distilled small BERT-based model (75M params) trained using our approach improves the search relevance metric by up to 23% over a baseline DSSM-based model with similar inference latency. The small model only suffers a 3% drop in relevance metric compared to the 20x larger teacher. We also show using online A/B tests at scale, that our approach improves over the production model in exact and substitute products retrieved.

Keywords: Matching · Retrieval · Search · Pretrained Language Models

1 Introduction

An e-commerce product search engine typically serves queries in two stages—matching and ranking, for efficiency and latency reasons. In the matching stage, a query is processed and matched against hundreds of millions of products to retrieve thousands of products that are relevant to the query. In the subsequent ranking stage, the retrieved products are scored against one or more objectives

A. Muhamed and S. Srinivasan—Equal contribution.

H. Kashima et al. (Eds.): PAKDD 2023, LNAI 13937, pp. 73–85, 2023.
https://doi.org/10.1007/978-3-031-33380-4_6

and then sorted to increase the likelihood of satisfying the customer query in the top positions. Matching is therefore a critical first step towards a delightful customer experience in terms of search latency and relevance, and the focus of this paper. Lexical matching using an inverted index [1] has been the industry standard approach for e-commerce retrieval applications. This type of matching retrieves products that have one or more query keywords appear in their textual attributes such as title and description. Lexical matching is favorable because of its simplicity, explainability, low latency, and ability to scale to catalogs with billions of products. Despite the advantages, lexical matching has several shortcomings such as sensitivity to spelling variants (e.g. "grey" vs "gray") or mistakes (e.g. "sheos" instead of "shoes"), proneness to *vocabulary mismatch* (e.g. hypernyms, synonyms), and lack of semantic understanding (e.g. "latex free examination gloves" does not match the intent of "latex examination gloves"). These issues are largely caused by the underlying term-based distributional representation for query and product that fails to capture the fine-grained relationship between terms. Researchers and practitioners typically resort to *query expansion* techniques to address these issues.

Dense embedding based semantic matching [2] has been shown to significantly alleviate the shortcomings of lexical matching due to its distributed representation that admits granular proximity between the terms of a query-product pair in low dimensional vector space [3]. To fulfill the low latency requirement, these semantic matching models are predominantly shallow and use a bi-encoder architecture. Bi-encoders have separate encoders for generating query and product embeddings and use cosine similarity to define the proximity of queries and products. Such an architecture allows product embeddings to be indexed offline for fast approximate nearest neighbor (ANN) search [4] with the query embedding generated in realtime. Recently, BERT-based models [5] have advanced the state-of-the-art in natural language processing but due to latency considerations, their use in online e-commerce information retrieval is largely limited to the bi-encoder architecture [6–8] which does not benefit from the early interaction between the query and product representations.

In this work, we propose a multi-stage training procedure to train a small BERT-based matching model for online inference that leverages a large pretrained BERT-based matching model. A large BERT encoder (750M parameters) is first pretrained with the masked language modeling (MLM) objective on the product catalog data (details in Sect. 2.1), we refer to the trained model as DS-BERT. Next, the DS-BERT model is pre-finetuned using our novel query-product interaction pre-finetuning (QPI) task (see Sect. 2.2), the trained model is referred to as QPI-BERT. We find that interaction pre-finetuning greatly improves training stability of bi-encoders downstream as well as significantly improves generalization. QPI-BERT is then cloned into a bi-encoder model architecture and finetuned with query-product purchase signal, we refer to this model as QPI-BERT-FT (see Sect. 2.3). Finally, a smaller QPI-BERT bi-encoder student model (75M parameters) is distilled from the QPI-BERT-FT teacher by matching the cosine similarity

score on the query-product pairs used for finetuning (see Sect. 2.4), we refer to this model as SMALL-QPI-BERT-DIS.

Through our offline experiments on a large e-commerce dataset, we show that the SMALL-QPI-BERT-DIS model (75M) suffers only a 3% drop in search relevance metric, compared to the QPI-BERT model with 20x its number of parameters (1.5B). This SMALL-QPI-BERT-DIS model improves search relevance by 23%, over a baseline DSSM-based matching model [2] with similar number of parameters and inference latency. Using an online A/B test we also show that the SMALL-QPI-BERT-DIS model outperforms the production model with 2% lift in both relevance and sales metrics.

Our work is closely related to the literature on semantic matching with deep learning. Some of the initial pre-BERT works include the Deep Semantic Similarity Model (DSSM) [2] , that constructs vector representations for queries and documents using a feedforward network and uses cosine similarity as the scoring function. DSSM-based models are widely used for real-time matching at web-scale [9,10]. This was later specialized for online product matching [3]. Post-BERT techniques leverage Pretrained Language Models (PLMs), such as BERT [5] to construct bi-encoders for matching tasks [8,11,12]. These techniques have broadly been applied to question-answering where the question and answer are from similar domains and interaction pre-finetuning is less essential. A recent work [13], proposes a multi-stage semantic matching training pipeline for web retrieval. However, unlike our approach, their focus is on deploying an ERNIE model (220M), while we study how large bi-encoders (1.5B) can be compressed to much smaller bi-encoders (70M) at web-scale using interaction pre-finetuning. In summary, the key contributions of this work are:

- We propose a multi-stage training procedure to effectively train a small BERT-based matching model for online inference from a much larger model (750 million to 1.5 billion parameters).
- We introduce a novel pre-finetuning task where a span masking and field permutation equivariant objective is used on joint query-product input text to help align the query and product representations. This task helps stabilize training and improve generalization of bi-encoders.
- We show using offline and online experiments at scale on an e-commerce website, that the proposed approach helps the small BERT SMALL-QPI-BERT-DIS model significantly outperform both a DSSM-based model (by 23%) in offline experiments and a production model in an online A/B test.

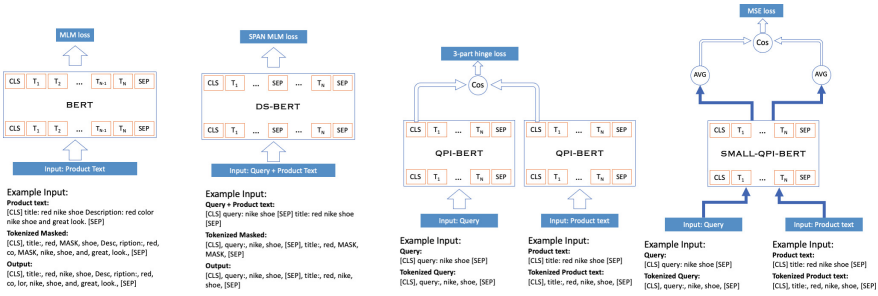
2 Methodology

In this section we describe our proposed four-stage training paradigm that consists of 1) domain-specific pretraining, 2) query-product interaction prefinetuning, 3) finetuning for retrieval, and 4) knowledge distillation to a smaller model. In the first three stages, we train a large BERT model for product matching and in the final stage we distill this knowledge to a smaller model that can be deployed efficiently in production (See Fig. 1).

2.1 Domain-Specific Pretraining

In the first stage of training, we pretrain a large BERT model on a domain specific dataset for product matching. The language used to describe products (catalog fields) in the e-commerce domain significantly differs from the language used on the larger web. Product titles and descriptions use a subset of the entire vocabulary, are often structured to follow a specific pattern, and in general have a different distribution from the sources that publicly available language models are trained on. Therefore, using an off-the-shelf pretrained BERT-based model does not perform well when finetuned for the product matching task.

Instead of using an off-the-shelf pretrained BERT model, we construct a BPE vocabulary [14] from the catalog corpus comprising of billions of products in e-commerce domain. We then pretrain the model on text from the catalog, and use all of the catalog text fields such as title and description of products available by concatenating them along with their field names. Our pretraining objective is the standard masked-language-modeling (MLM) loss [5,15,16]. We refer to the model trained with this strategy as the DS-BERT model (See Fig. 1a).



(a) Stage 1: BERT model is pretrained to produce DS-BERT

(b) Stage 2: DS-BERT is pre-finetuned to produce QPI-BERT

(c) Stage 3: Bi-encoder QPI-BERT is trained with three-part hinge loss to produce QPI-BERT-FT

(d) Stage 4: QPI-BERT-FT is distilled to produce SMALL-QPI-BERT-DIS

Fig. 1. This figure shows the four stages involved in training an effective deployable model for semantic matching.

2.2 Query-Product Interaction Pre-finetuning

Bi-encoders are preferred over cross-encoder models with full interaction for matching due to their efficiency and feasibility at runtime. Bi-encoders are however notoriously difficult to train on query-product pairs due to training instabilities arising from gradient variance between the two inputs. Losing the capability to explicitly model the interaction between queries and products also results in worse generalization than the cross-encoder.

In the second stage of training we propose a novel self-supervised approach to incorporate query-product interaction in the large encoder which is critical to improving the performance on the product matching task. We use query-product

paired data to help the encoder learn the relationship between a query and a product using full cross-attention. To construct such a dataset, we first identify query-product pairs that share a relevant semantic relationship, for example, all products purchased for a given query can be considered relevant or query-product pairs can be manually labeled for relevance. In this paper, the dataset is constructed such that the query-product pairs are semantically relevant with a high probability $\alpha > 0.8$. The pre-finetuning dataset size (a few million examples) is much smaller than the pretraining dataset (a billion examples).

To perform pre-finetuning, we perform span MLM on the concatenated query and product text with a "[SEP]" token between them. At each iteration, we select spans from either the query text or product text (never both) to mask tokens. We sample span length (number of words) from a geometric distribution, till a predetermined percentage of tokens have been masked. The start of the span is uniformly sampled within the query or the product. During training we also observed that permuting the fields within the query and product, a form of field permutation equivariant training, also helped the model generalize better. We refer to the model trained with this strategy as QPI-BERT model (See Fig. 1b). Pre-finetuning with self-supervision on semantically relevant paired dataset boosts generalization for matching when a large noisy training set is available. This differs from previous works that use supervision on manually labeled data.

2.3 Finetuning for Matching

The third stage of training is to finetune the large teacher encoder QPI-BERT model in a bi-encoder setting for matching. We train a bi-encoder teacher as opposed to a cross-encoder teacher for retrieval as the extreme inefficiency in generating predictions for evaluation and slow training convergence rate makes it impractical to train cross-encoders for web-scale data and large models.

Let us denote the QPI-BERT model as M, query encoder as M_q, and product encoder as M_p, where the weights between query encoder and product encoder are shared. In our experiments sharing weights performed comparably to independently training them. For any query-product pair Q and P as inputs, we first generate the embedding Q_{emb} for query Q using M_q and embedding P_{emb} for product P using M_p using their "[CLS]" token representation. A cosine similarity score $s_{Q,P} = \cos(Q_{emb}, P_{emb})$ is used to compute relevance between them.

We train the bi-encoder using a three-part hinge loss. This loss requires the ground-truth data ($y_{Q,P}$) to be labeled with one of three possible values referred to as positive (1), hard negative (0) and random negative (-1). We use the purchased products for a given query as positive and any product uniformly sampled from the catalog as random negative. Identifying hard negatives is non-trivial [12,17], and in this work we choose a simple yet effective approach [3], where for a given query, all products that were shown to the user but did not

receive any interaction is a hard negative. The loss takes the following form:

$$\text{loss}_{Q,P}(y_{Q,P}, s_{Q,P}) = \begin{cases} \max(\delta_{pos} - s_{Q,P}, 0), & \text{if } y_{Q,P} = 1. \\ \max(\delta^{-}_{hn} - s_{Q,P}, 0) & \text{if } y_{Q,P} = 0. \\ \quad + \max(s_{Q,P} - \delta^{+}_{hn}, 0), & \\ \max(s_{Q,P} - \delta_{rn}, 0), & \text{if } y_{Q,P} = -1. \end{cases} \tag{1}$$

where δ_{pos} and δ^{-}_{hn} are the lower thresholds for the positive and hard negative data scores respectively and δ^{+}_{hn} and δ_{rn} are the upper thresholds for the hard negative and random negative data scores respectively. We refer to the model trained with this strategy as the QPI-BERT-FT model (see Fig. 1c).

2.4 Distillation and Realtime Inference

The final stage of our framework is to distill the knowledge of teacher QPI-BERT-FT to a smaller student bi-encoder BERT model (75M to 150M parameters) that meets the online latency constraint. We first pretrain and prefinetune the small model similar to QPI-BERT to generate SMALL-QPI-BERT model M. Then we clone the encoder to create a query encoder $\tilde{M}_Q$ and a product encoder $\tilde{M}_P$. Unlike the large model case, for the small model we observe that sharing parameters between encoders helps improve performance significantly. The query embedding $\tilde{Q}_{emb}$ and product embedding $\tilde{P}_{emb}$ for the student model are computed by averaging all token embeddings in the query Q and product P respectively. The relevance score for a query-product pair is compute using cosine similarity i.e., $\tilde{s}_{Q,P} = cos(\tilde{Q}_{emb}, \tilde{P}_{emb})$. The model is trained by minimizing the distance between the scores generated by QPI-BERT-FT teacher and the model using the mean squared error (MSE) loss function.

$$\text{loss}_{Q,P}(s_{Q,P}, \tilde{s}_{Q,P}) = (s_{Q,P} - \tilde{s}_{Q,P})^2 \tag{2}$$

In practice we observed that simple score matching using MSE outperformed other approaches such as using L2 loss on the embeddings directly, Margin-MSE [18] with random negatives, or contrastive losses like SimCLR [19] with random negatives. We refer to the model distilled with this strategy as the SMALL-QPI-BERT-FT model (see Fig. 1d). At runtime, for every query entered by the customer, we compute the query embedding and then retrieve top K products using ANN search [4]. To serve traffic in realtime, we cache the product embeddings and compute only the query embedding online. The retrieved products are served directly to customers or mixed with other results and re-ranked before displaying to the customer.

3 Empirical Evaluation

3.1 Experimental Setup

Data. We use the following multilingual datasets for different stages of training:

Domain-Specific Pretraining Data: We use ∼1 billion product titles and descriptions from 14 different languages. This data is also used to construct a sentencepiece [20] tokenizer with 256K vocab size.

Interaction Pre-finetuning Data: We use ∼15M query-product pairs from 12 languages and use weak supervision in the form of rules to label them as relevant or irrelevant. ∼80% of the pairs are relevant query-product pairs.

Finetuning for Matching Data: We use ∼330M query-product pairs subsampled from a live e-commerce service to train the model for matching. We maintain a positive to hard negative to negative ratio of 1:10:11. The pairs are collected from multiple countries with at least 4 languages. We use a validation dataset to compute recall that contains 28K queries and 1M products from the subsampled catalog. Human evaluation (Sect. 3.1) uses a held-out set of 100 queries.

Models. We experiment with several model variants, both small and large summarized in Table 1. All large models we train are based on DS-BERT, which is a multilingual BERT model with 38 layers, 1024 output dimensions and 4096 hidden dimensions. When the parameters for the query and product encoder are not shared, the model has twice the parameters of the encoder. The small models we train are multilingual BERT models with 2 layers, 256 output dimensions, and 1024 hidden dimensions. In addition, we use DSSM and XLMROBERTA as baselines. • XLMROBERTA: Publicly available XLMRoberta [21] model which is finetuned for matching as described in Sect. 2.3. • DSSM: Bi-encoder model with a shared embedding layer (output dimension of 256) followed by batch norm and averaged token embedding to represent the query and product [3]. To ensure effective use of vocabulary for DSSM, we create a different sentencepiece model with 300k tokens using the matching training data.

Metrics. *R@100:* This is the average purchase recall computed on the validation data for the top 100 products retrieved.

Relevance Metrics: To understand the true improvement in the quality of matches retrieved by the model, we use Toloka (toloka.yandex.com) to label the results produced by our models. For every query we retrieve 100 results and ask the annotators to label them as exact match, substitute, or other. We report the average percentage of exact (E@100). substitute (S@100), and other (O@100). We use E@100 + S@100 (E+S) to measure semantic improvement in the model.

Table 1. Bi-encoder model variants. Differences are number of parameters (Params), embedding dimensionality (ED), embedding type (ET), domain-specific pretraining (DS PT), QPI prefinetuning (QPI PFT), whether encoders share parameters (Shared), whether model is distilled from QPI-BERT-FT (Dis).

Models	Params	ED	ET	DS PT	QPI PFT	Shared	Dist
Large Models							
XLMROBERTA	1.1B	1024	CLS	N	N	N	N
DS-BERT	1.5B	1024	CLS	Y	N	N	N
QPI-BERT-FT	1.5B	1024	CLS	Y	Y	N	N
QPI-BERT-FT*	1.5B	1024	CLS	Y	Y[a]	N	N
QPI-BERT-FT-SH	750M	1024	AVG	Y	Y	Y	N
Smaller Models							
SMALL-QPI-BERT-FT	150M	256	CLS	Y	Y	N	N
SMALL-QPI-BERT-FT-AVG	150M	256	AVG	Y	Y	N	N
SMALL-QPI-BERT-FT-SH	75M	256	CLS	Y	Y	Y	N
SMALL-QPI-BERT-FT-SH-AVG	75M	256	AVG	Y	Y	Y	N
SMALL-QPI-BERT-DIS	75M	256	AVG	Y	Y	Y	Y
DSSM	75M	256	AVG	N	N	Y	N

[a] Classification objective instead of span masking objective on pre-finetuning data.

Training. We use Deepspeed (deepspeed.ai) and PyTorch for training models on AWS P3DN instances. We used LANS optimizer [22] with learning rate between $1e^{-4}$ and $1e^{-6}$ based on the model and for all models we use a batch size of 8192. During pre-finetuning, we use validation MLM accuracy to perform early stopping and for finetuning we use validation recall for stopping. When using the three-part hinge-loss in Eq. 1, $\delta_{pos} = 0.9$, $\delta^{+}_{hn} = 0.55$ and $\delta_{rn} = \delta^{-}_{hn} = 0.2$.

3.2 Offline and Online Results

Does our Training Strategy Help Improve Semantic Matching Performance Offline? For large models, we compare QPI-BERT-FT with XLMROBERTA, DS-BERT, and QPI-BERT-FT*, and for small models, we compare SMALL-QPI-BERT-FT SMALL-QPI-BERT-FT-SH with DSSM (Table 2). a) QPI-BERT outperforms other approaches both in R@100 and E+S. Among large models, the performance of DS-BERT is better than XLMROBERTA and QPI-BERT-FT* is better than DS-BERT. This clearly indicates progressive improvement with the different stages in our approach. b) We observe is that DSSM outperforms XLMROBERTA in all metrics indicating a vocabulary and domain mismatch between the catalog data and web data. Domain-specific pretraining is essential to performance when training the large models. c) We see that QPI-BERT-FT significantly outperforms QPI-BERT-FT* in all metrics, validating the importance of interaction pre-finetuning over mere supervision alone for matching. d) For small models,

we observe that the performance of SMALL-QPI-BERT-FT is very similar to DSSM, with SMALL-QPI-BERT-FT showing ∼45% relative lift in S@100 but, ∼8% relative drop in E@100, ∼4% relative lift in E+S, and ∼1% relative drop in R@100. When sharing parameters between the query and product encoder, and averaging embeddings, SMALL-QPI-BERT-FT-SH-AVG outperforms DSSM by ∼38% relative lift in S@100, ∼2% relative lift in E@100, ∼10% relative lift in E+S, and ∼2% relative lift in R@100. The results indicate that our strategy helps improve the performance overall and the improvements are higher for larger models (∼23% relative lift in E+S over DSSM). This reinforces our proposed approach: train a large model and distill the knowledge to a smaller model, instead of directly training a smaller model.

Can Distillation Preserve Large Model Performance? Given the large improvement in matching metrics for large models, we would ideally like to retain this improvement in smaller models using distillation. We compare SMALL-QPI-BERT-DIS with QPI-BERT-FT (Table 2) and observe a ∼3% relative drop in E+S and R@100. This shows that while there is small gap, it is possible to transfer most of the information from a 1.5B parameter large QPI-BERT-FT model to a 20x smaller SMALL-QPI-BERT-DIS model (75M parameter) using our approach.

Does Sharing Parameters in the Bi-encoder have an Impact on Retrieval Task Performance? To understand the effect of sharing parameters between query and product encoders in the bi-encoder setting, we compare QPI-BERT-FT-SH with QPI-BERT-FT among the large models and SMALL-QPI-BERT-FT-SHwith SMALL-QPI-BERT-FT, and SMALL-QPI-BERT-FT-SH-AVG, with SMALL-QPI-BERT-FT-AVGamong the small models (Table 2). We observe that sharing encoders has almost no impact on the performance of large models and the

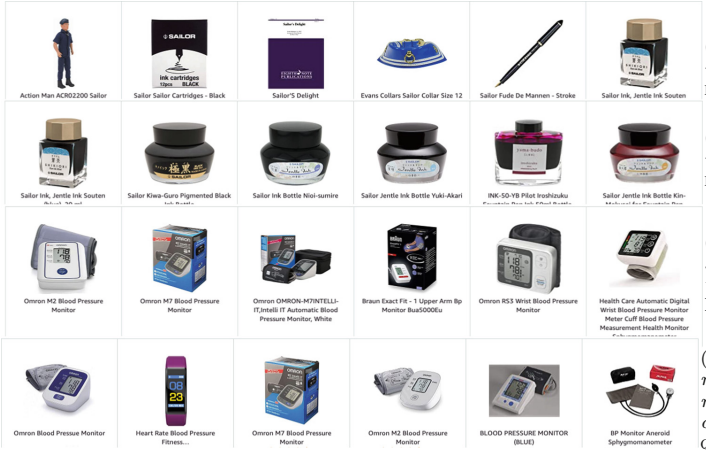
Table 2. Offline metrics of models on a multi-lingual e-commerce dataset

Models	R@100	E@100	S@100	O@100	E+S
Large Models					
XLMROBERTA	68.43	29.52	17.46	53.02	46.98
DS-BERT	73.98	43.17	17.55	39.28	60.72
QPI-BERT-FT	82.2	50.36	**20.5**	29.14	70.86
QPI-BERT-FT*	75.6	48.35	19.66	31.99	68.01
QPI-BERT-FT-SH	**83.35**	**51.08**	19.81	**29.11**	**70.89**
Smaller Models					
SMALL-QPI-BERT-FT	77.06	40.28	18.16	41.56	58.44
SMALL-QPI-BERT-FT-AVG	50.84	33.63	13	53.37	46.63
SMALL-QPI-BERT-FT-SH	79.3	43.98	18.52	37.5	62.5
SMALL-QPI-BERT-FT-SH-AVG	80.17	44.45	17.33	38.22	61.78
SMALL-QPI-BERT-DIS	**80**	**48.04**	**20.78**	**31.18**	**68.82**
DSSM	78.1	43.56	12.49	43.95	56.05

maximum relative drop in E+S and recall is ∼1% with QPI-BERT-FT-SH winning marginally. However, in the smaller models we observe that sharing parameters gives a large boost in performance with a relative lift of upto ∼32% in the E+S metric and ∼60% in R@100. When the model size is large enough, it is capable of learning independent encoders for both inputs. But, when the model is small, the model benefits from sharing parameters.

How Does our Approach Improve over a non-BERT-Based Model? To visualize the difference in matching quality between our BERT-based model and DSSM, we look at results for two queries, with DSSM retrieving more relevant products on one query and vice-versa on the other (Fig. 2). We observe that for query *"sailor ink"* QPI-BERT-FT performs better as all results are relevant products. For this query, DSSM behaves like a lexical matcher and fetches results for both *"sailor"* and *"ink"*. For query "omron sale bp monitor machine", DSSM retrieves all relevant matches. QPI-BERT-FT however, retrieves an irrelevant product (a fitness watch). While irrelevant, it still falls into the product type of "personal health" implying an error in semantic generalization. The significantly higher increase in S@100 compared to E@100 indicates that QPI-BERT-FT is a better semantic model as the representations must incorporate high-level concepts to match substitutes, that token-level exact matches cannot achieve.

What is the Latency Improvement of the Smaller BERT Model Compared to the Large Model? We have seen earlier that the large model can be effectively compressed to a 20x smaller model that incurs much lower inference latency. We compare the inference latencies of our models while generating query embeddings which is representative of realtime latency as the product embeddings are generated offline and indexed for ANN. We ignore the ANN latency as modern ANN search can be computed effectively in realtime (∼1 ms)



(a) Query: *"sailor ink"*; Method: DSSM.

(b) Query: *"sailor ink"*; Method: QPI-BERT-FT.

(c) Query: *"omron sale bp monitor machine"*; Method: DSSM.

(d) Query: *"omron sale bp monitor machine"*; Method: QPI-BERT-FT.

Fig. 2. Top 6 results obtained by DSSM and QPI-BERT-FT for queries *"sailor ink"* and *"omron sale bp monitor machine"*

[4]. Figure 3 shows the time it takes to compute query embedding (inference) for different query lengths (query length computed as number of tokens after tokenization) on an r5.4xlarge AWS instance. As expected, DSSM has the lowest inference time and QPI-BERT has the largest. Both SMALL-QPI-BERT and DSSM have embedding generation time of under 1ms upto 32 tokens making it feasible to serve realtime traffic. SMALL-QPI-BERT reduces the latency time by ∼60× compared to QPI-BERT with a relevance metric performance drop of only ∼3%.

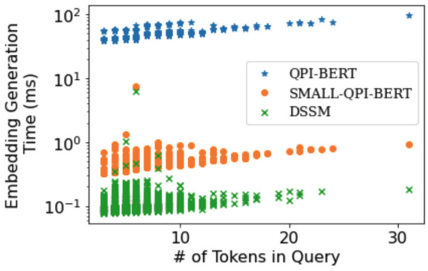


Fig. 3. Inference time for QPI-BERT, SMALL-QPI-BERT, and DSSM on r5.4xlarge

Table 3. A/B test results for SMALL-QPI-BERT-DIS rel. to production system.

PS	Units	E@16	S@16	E+S@16	SR	Latency P99
+2.07%	+1.47%	−1.19%	+3.37%	+2.18%	−16.9%	+4 ms

How Well Does the Approach Perform Online? To measure the impact of our approach online, we experiment with SMALL-QPI-BERT-DIS in a multilingual large e-commerce service. The service augments matching results from several sources like lexical matchers, semantic matchers, upstream machine learning models, and advertised products. We replace only the production semantic matcher with our SMALL-QPI-BERT-DIS and perform an A/B test. We measure both customer engagement metrics and relevance quality metrics. For customer engagement metrics, we look at the change in number of units purchased and the amount of product sales (PS). For quality metric, we look at the change in user evaluated E@16, S@16, E+S@16 and sparse results (SR) which is the percentage of queries with less than 16 products retrieved. We observe (Table 3) that our approach significantly improves over the production semantic matcher and lead to a significant drop in SR. The reduction in E@16 and increase in S@16 suggests that our approach is learning latent semantic meaning to increase substitutes displayed to customers. We also observe that our model does not have a significant impact on latency (∼4 ms) and can be used at runtime.

4 Conclusion

In this work we develop a four-stage training paradigm to train an effective BERT model that can be deployed online to improve product matching. We introduce a new pre-finetuning task that incorporates the interaction between queries and products prior to training for retrieval which we show is critical to improving performance. Using a simple yet effective approach, we distill a large model to a smaller model and show through offline and online experiments that our approach can significantly improve customer experience. As future work, it would be interesting to incorporate other structured data from the e-commerce service to enhance representation learning, such as brand and product dimensions, as well as customer interaction data such as reviews.

Acknowledgement. We would like to thank Priyanka, Mutasem, Huajun, Jaspreet, Dhivya, Giovanni, Hemant, Anton, Tina, and RJ from Amazon Search.

References

1. Schütze, H., Manning, C.D., Raghavan, P.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
2. Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., Heck, L.P.: Learning deep structured semantic models for web search using clickthrough data. In: CIKM (2013)
3. Nigam, P., et al.: Semantic product search. In: SIGKDD (2019)
4. Malkov Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Trans. Pattern Anal. Mach. Intell. **42**(4), 824–836 (2018)
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NACCL (2019)
6. Reimers, N., Gurevych, I.: Sentence-BERT: sentence embeddings using Siamese BERT-networks. In: EMNLP-IJCNLP (2019)
7. Khattab, O., Zaharia, M.: Colbert: efficient and effective passage search via contextualized late interaction over BERT. In: SIGIR (2020)
8. Lu, W., Jiao, J., Zhang, R.: TwinBERT: distilling knowledge to twin-structured BERT models for efficient retrieval. ArXiv, vol. abs/2002.06275 (2020)
9. Huang, J.-T., et al.: Embedding-based retrieval in Facebook search. In: SIGKDD (2020)
10. Li, S., et al.: Embedding-based product retrieval in Taobao search. In: SIGKDD (2021)
11. Karpukhin, V., et al.: Dense passage retrieval for open-domain question answering. In: EMNLP (2020)
12. Xiong, L., et al.: Approximate nearest neighbor negative contrastive learning for dense text retrieval. In: ICLR (2021)
13. Liu, Y., et al.: Pre-trained language model for web-scale retrieval in baidu search. In: SIGKDD (2021)

14. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. In :Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Berlin, Germany, pp. 1715–1725, Association for Computational Linguistics, Aug. 2016
15. Liu, Y., et al.: RoBERTa: a robustly optimized BERT pretraining approach, ArXiv, vol. abs/1907.11692 (2019)
16. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: AlBERT: a lite BERT for self-supervised learning of language representations. In: ICLR (2020)
17. Ji, S., Vishwanathan, S.V.N., Satish, N., Anderson, M.J., Dubey, P.: BlackOut: speeding up recurrent neural network language models with very large vocabularies. In: ICLR (2016)
18. Hofstätter, S., Althammer, S., Schröder, M., Sertkan, M., Hanbury, A.: Improving efficient neural ranking models with cross-architecture knowledge distillation. ArXiv, vol. abs/2010.02666 (2020)
19. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: ICML (2020)
20. Kudo, T., Richardson, J.: SentencePiece: a simple and language independent subword tokenizer and detokenizer for neural text processing. In: EMNLP (2018)
21. Conneau, A.: Unsupervised cross-lingual representation learning at scale. In: ACL (2020)
22. Zheng, S., Lin, H., Zha, S., Li, M.: Accelerated large batch optimization of BERT pretraining in 54 minutes. ArXiv, vol. abs/2006.13484 (2020)