Blend and Match: Distilling Semantic Search Models with Different Inductive Biases and Model Architectures

Hamed Bonab
Amazon
hamedrab@amazon.com

Ashutosh Joshi
Amazon
jashutos@amazon.com

Ravi Bhatia
Amazon
rvbht@amazon.com

Ankit Gandhi
Amazon
ganankit@amazon.com

Vijay Huddar
Amazon
vhhuddar@amazon.com

Juhi Naik
Amazon
juhinaik@amazon.com

Mutasem Al-Darabsah
Amazon
mutasema@amazon.com

Choon Hui Teo
Amazon
choonhui@amazon.com

Jonathan May
Amazon
jnatmay@amazon.com

Tarun Agarwal
Amazon
tagar@amazon.com

Vaclav Petricek
Amazon
petricek@amazon.com

ABSTRACT

Commercial search engines use different semantic models to augment lexical matches. These models provide candidate items for a user's query from a target space of millions to billions of items. Models with different inductive biases provide relatively different predictions, making it desirable to launch multiple semantic models in production. However, latency and resource constraints make simultaneously deploying multiple models impractical. In this paper, we introduce a distillation approach, called *Blend and Match (BM)*, to unify two different semantic search models into a single model. We use a Bi-encoder semantic matching model as our primary model and propose a novel loss function to incorporate eXtreme Multi-label Classification (XMC) predictions as the secondary model. Our experiments conducted on two large-scale datasets, collected from a popular e-commerce store, show that our proposed approach significantly improves the recall of the primary Bi-encoder model by 11% to 17% with a minimal loss in precision. We show that traditional knowledge distillation approaches result in a sub-optimal performance for our problem setting, and our BM approach yields comparable rankings with strong Rank Fusion (RF) methods used only if one could deploy multiple models.

KEYWORDS

Semantic Search; Model Blending; Product Search; Ranking Distillation

ACM Reference Format:
Hamed Bonab, Ashutosh Joshi, Ravi Bhatia, Ankit Gandhi, Vijay Huddar, Juhi Naik, Mutasem Al-Darabsah, Choon Hui Teo, Jonathan May, Tarun Agarwal, and Vaclav Petricek. 2023. Blend and Match: Distilling Semantic Search Models with Different Inductive Biases and Model Architectures. In *Companion Proceedings of the ACM Web Conference 2023 (WWW '23 Companion), April 30-May 4, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3543873.3587629

WWW '23 Companion, April 30-May 4, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9419-2/23/04.
https://doi.org/10.1145/3543873.3587629

1 INTRODUCTION

Retail search engines are primarily keyword-based information retrieval (IR) systems comprising two main operations—matching and ranking [32]. When a customer issues a query, several matching systems work in parallel to filter millions of items into a candidate *matchset* that is relevant to the query. The recall-focused matchset can have thousands of items (products) which are then ranked based on the match quality of each item to the query [23]. Lexical algorithms like Okapi-BM25 [36, 37] score a query-product pair as a weighted sum of overlapping keywords. These approaches do not capture customer behavior signals (click, purchase, stream, etc.) and thus do not capture customer preferences [4, 26]. They are also brittle to morphological variation and spelling errors. Most commercial search engines thus deploy semantic matching in addition to lexical matching. This general schema is presented in Figure 1. In order to increase the coverage of relevant items in the matchset (i.e., recall), lexical and semantic models that generate a diverse matchset are deployed in parallel. However, the cost and efficiency constraints for real-world deployments limits the choices on the number of models that can be deployed.

Semantic matching learns representations of queries and products based on customer behavior and hence imputes products that customers prefer. Semantic matching can be implemented using Bi-encoders—two-tower models with one arm each for the query and product representation, with a final shared space to determine the similarity of the pair [22, 33, 43]. Semantic matching can also be implemented using eXtreme Multi-label Classification (XMC) systems. For fast inference, most XMC algorithms with textual inputs use sparse TF-IDF features and leverage different partitioning techniques on the label space to reduce complexity. In particular,

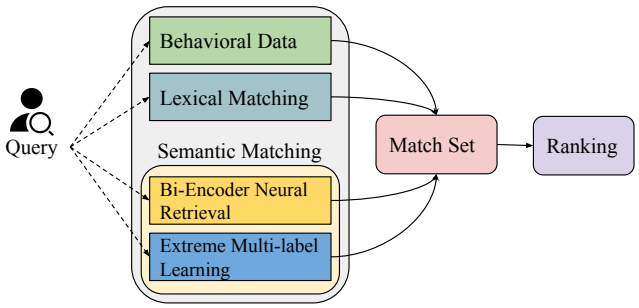


Figure 1: General Schema of Product Search.

Prediction of Enormous Correlated Output Spaces (PECOS) [45] partitions the label (product) space by clustering the labels into hierarchically granular clusters in an m-ary tree structure. Fast linear classifiers at each node select the branches to explore.

Since semantic models deal with hundreds of millions to billions of products, they are extremely large in their memory footprint. This limits the number of models that can be deployed in parallel. To our knowledge, large scale commercial systems deploy only one of these two model types. On the other hand, because the two approaches formulate the semantic matching problem differently, they have different inductive biases and thus predict a different set of (relevant) items.

In order to capture the most relevant products from the output of both Bi-encoders and PECOS, we propose a distillation mechanism called blending that uses the output of one model to teach the other model. Most distillation approaches focus on simplifying the student architecture and even multi-teacher distillation usually assumes similar inductive biases of the teachers and students [17, 20, 35]. We show, through offline as well as crowd-sourced evaluations, that blended model is able to substantially improve over the recall of the Bi-encoder model by 11% to 17% with a negligible loss in precision. Our experiments show that, for our problem setting, the existing knowledge distillation approaches provide suboptimal rankings. We also compare our blended model with strong Rank Fusion methods and show that our blended model provides comparable ranking performance, while only a single model needed for deployment. Rank Fusion methods on the other hand, would require deploying multiple models and hence is not a practical option for us.

In summary, the main contributions of this paper constitute:

- A blending architecture to combine two different formulations of semantic matching into a single model,
- Extensive experiments and analyses to show that our method is able to deliver the best of both worlds.

2 RELATED WORK

Our study is related to model compression and ranking distillation from the information retrieval domain. In this section, we briefly review the literature in these domains.

2.1 Model Compression.

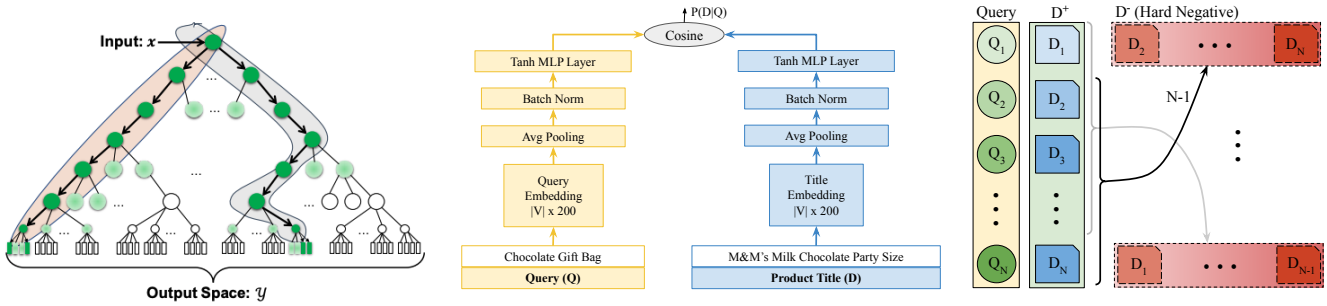
More accurate but complex models are difficult to deploy due to high inference time and resources. Model compression methods like pruning, quantization, and Knowledge Distillation (KD) mitigate this problem [5, 11, 18]. KD typically trains a (small) student model to replicate another model's outputs (a teacher, typically a larger and stronger model). Hinton et al. [19] show that for classification, obtaining softer probabilities of teacher models' logits results in better distillation. DistillBERT[39] employs a similar idea for a pre-trained BERT[15] model and provides 97% of BERT capabilities with increased speed and a smaller model size. You et al. [44] and Liu et al. [30] extend KD by incorporating multiple teachers in the framework. Most similar to our BM approach is Geras et al. [17], who introduce "model blending" as a variant of KD to blend a LSTM model predictions into a CNN. For blending, the teacher and student models are similar in complexity but different in inductive bias.

2.2 Ranking Distillation (RD).

A few studies extend findings from general model compression to ranking models, i.e., search and recommendation. Chen et al. [10] use listwise learning-to-rank in DarkRank distillation for image retrieval. Tang and Wang [41] reduce the neural recommendation systems' embedding size with the student model for RD with a pointwise loss function. Cohen et al. [13] aim to compress and approximate learning-to-rank ensemble methods using a simple feed-forward neural network, arguing on the universal function approximation capabilities of neural models. More recently, Kang et al. [25] propose a framework for transferring a teacher model's predictions and latent knowledge to a student using a listwise loss function. Vakili Tahami et al. [42] study the impact of KD on BERT-based retrieval models for chatbots. Lee and Kim [27] correct failures of the student model in recommenders by employing the discrepancy between the teacher and student model predictions. Choi et al. [12] study multi-teacher distillation for BERT-based cross-encoder and bi-encoder retrieval models to train more efficient bi-encoders. Reddi et al. [35] provides a unified framework for existing methods and propose RankDistil aiming to preserve the top-k rankings by matching the order of items of student and teacher, while penalizing items ranked low by the teacher. Hofstätter et al. [20] study distillation of different ranking architectures, similar to our work, by proposing a pairwise margin focused Mean Squared Error (MSE) loss function, i.e., *Margin-MSE*. Similar to their experiments, we report pointwise MSE and pairwise Margin-MSE as two RD-based indicative reference points for comparisons.

3 BACKGROUND

Here, we briefly describe the architecture of two semantic matching models we use, i.e. PECOS and Bi-encoder with Triplet Loss or BETL. The Bi-encoder and PECOS models that we use have been shown to improve search results in commercial systems [7, 33]. Both models are trained on sets of (query, product) pairs, where a pair is admitted to training data if a customer response beyond a search impression (e.g. a click or purchase) is observed for the product after issuing the query. We note that both models were designed to meet the memory and latency requirements and it is out of the scope of our



(a) PECOS Model Architecture (b) BETL Model Architecture (c) Hard Negative Mining

Figure 2: The Model Architecture of PECOS and BETL and the General Schema of Hard Negative Mining.

study to benchmark and compare resource consumption. We refer the reader to original papers [7, 33] for such comparisons.

3.1 Prediction in Enormous and Correlated Output Spaces (PECOS)

PECOS is an XMC that deals with its extremely large label (product) space by successively partitioning it into smaller and smaller clusters [6, 9, 40, 45]. PECOS represents a product as an aggregation of its associated queries. It then clusters the product space into a hierarchical m-ary tree such that at each level, the products in a node are partitioned into m equal-sized child clusters (Figure 2a). Fast linear One-vs-Rest (OVR) classifiers are used at each level to determine the relevance of a cluster to a query. To meet latency constraints, PECOS uses sparse TF-IDF features of the input queries as detailed by Chang et al. [6]. For inference, PECOS uses beam search for traversing through the tree by choosing the top b child nodes at each level.

3.2 Bi-encoder with Triplet Loss (BETL)

The Bi-encoder architecture formulates the relevance problem as a metric learning task. We use a Bi-encoder model similar to Nigam et al. [33], but instead of the two-part hinge loss, we use a triplet loss function inspired by the work of Lu et al. [31]. To distinguish our model, we name it **Bi-encoder with Triplet Loss** or **BETL** (Fig. 2b). We use the following two-step procedure to train the model:

(1) **Warm-up.** We first train the model using pointwise Mean Squared Error (MSE) loss (Eq. 1). For every positive query-product pair, we randomly sample 3 negative products. N is the sum of N^+ positive examples and N^- randomly sampled negative examples. Let $\hat{y}_i$ be the model score for i-th query-product pair.

$$L_W = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{1}$$

(2) **Train.** Next, we train the model with a pairwise loss (Eq. 2). For every query Q_i and its corresponding product D_i^+, in the data, we sample hard negative products D_i^- on-the-fly while training (Figure 2c). y_i^+ and y_i^- are model predictions for D_i^+ and D_i^-, respectively. N^+ is the number of positive examples.

$$L_T = \frac{1}{N^+}\sum_{i=1}^{N^+}\log\left(1 + e^{(y_i^- - y_i^+)}\right) \tag{2}$$

Using the dense representation of both query and product title, we impute relevance as cosine similarity. For inference, we use the FAISS library [24] to build a KNN index and use approximate nearest neighbor methods to rank the closest k products for a query.

3.3 PECOS vs BETL

We summarize pros and cons of these two models for semantic matching as follows:

(1) PECOS training is 2-6x faster and requires cheaper hardware (only CPU) compared to BETL.
(2) Since PECOS uses clustering, model refreshes require frequent retraining from scratch, whereas BETL can incorporate new data with only index refreshes. Due to this PECOS also requires more frequent retraining.
(3) Since PECOS clusters labels (products) it sees in training, PECOS' prediction space is limited to only those products. BETL incorporates new products by forming a representation from text descriptions in the Bi-encoder's embedding space and thus is extensible to products not used during training.
(4) It is easier to distribute the KNN indexes across multiple servers, enabling arbitrarily fast inference speeds. PECOS on the other hand requires the entire tree to be present on a single server to facilitate beam search.
(5) Empirically, PECOS is better at capturing customer behavior (i.e. it has better recall), while BETL is better at making semantic connections between queries and products and hence can surface relevant products that are not reflected in customer behavior (i.e. it has better precision).

4 BLEND AND MATCH (BM)

Both BETL and PECOS models are powerful yet efficient models. Their problem formulations are totally different and combining their predictions can create a more robust semantic matching component. For example, in a preliminary analysis, we found that only about a third of the top-k predictions from PECOS and BETL are

in common. However, as described, latency and memory concerns make simultaneously deploying both models impractical. Hence, we propose to blend these into a single model.

The general schema of our BM training is provided in Fig. 3. We select BETL as the *Primary* and PECOS as the *Secondary* model. Our choice is mainly due to BETL's advantages of simpler architecture, maintenance, and deployment. We aim to teach PECOS predictions to BETL and achieve precision and recall measurements similar to the better individual model.

We enhance the primary model training such that it can incorporate the predictions of the secondary model along with the true labels. Our task is similar to the existing distillation methods with the key differences being that *(a)* we do not aim to reduce model size, and *(b)* the primary (cf. "student") model does not need to have the same formulation or architecture as the secondary (cf. "teacher"). In our case, the primary model is a fully trained (BETL) model, which we then teach to blend PECOS predictions using an enhanced loss function.

Most distillation works use MSE as the compression loss function [17, 19, 20, 39]. This is generally effective when both the teacher and student models have similar score distributions. In our case though, due to different inductive biases of PECOS and BETL, their score distributions diverge. Thus, when the primary (BETL) model tries to learn the loss distribution of the secondary (PECOS) model, it leads to a loss in performance akin to *catastrophic forgetting* [3]. Instead, we design a pairwise triplet loss function to blend PECOS predictions into our BETL training.

$$
\begin{aligned}
L_{BM} &= (1-\lambda)L_T + \lambda L_B \\
L_T &= \sum_{i=1}^{N^+} \log\left(1 + e^{(y_i^- - y_i^+)}\right) \\
L_B &= \sum_{i=1}^{N^+} \log\left(1 + e^{(y_i^+ - y_{pi}^+)^2}\right)
\end{aligned}
\tag{3}
$$

where y_i^+ and y_i^- are as defined in Eq. 2 and y_{pi}^+ is the PECOS prediction score for the original positive example. L_T is the original loss from Eq 2 that maximizes the distance between a positive and hard negative product, L_B is the blending loss that minimizes the distance between the primary and secondary models' predictions for a positive product, and $\lambda \in [0, 1]$ is the interpolation parameter between them. We refer to this blended BETL model trained using an interpolation of losses on PECOS predictions as ***BM-BETL***.

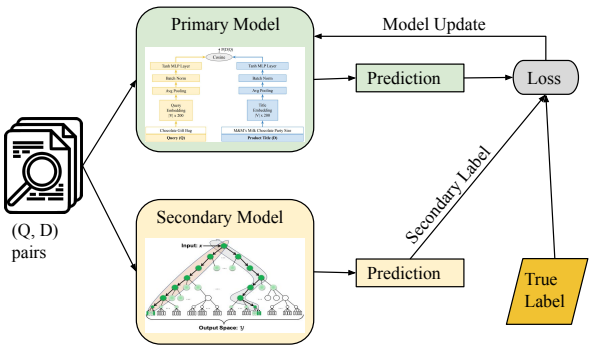


Figure 3: Blend and Match (BM) Training.

5 EXPERIMENTS

In this section we present experimental set-up and results, designed to compare the performance of BM-BETL against BETL and PECOS.

5.1 Experimental Setup

Dataset. We collect query-product pairs using behavioral data from de-identified search logs of an English speaking and a non-English speaking store of a popular retailer. Our collected data represents one year of search logs resulting in 43.6 million query-product pairs for the English speaking store and 78.8 million pairs for the non-English speaking store. We randomly sample 5% of each store's data for our evaluation. We ensure that none of the queries in the evaluation set occur in the training set.

Training PECOS and BETL. Both models use the same vocabulary of the most frequent 250k word unigrams, 50k word bigrams and 100k character trigrams. For BETL, we use FAISS [24] to search for the top-k nearest products for a query in the trained embedding space. We train PECOS with default parameters,[1] except we tune the maximum leaf size (English-store:100, non-English store:64) and beam size (English-store:50, non-English store:75). We train PECOS on a single machine with 96 vCPUs and 768GB of memory. We train BETL and BM-BETL on a single machine with 768GB memory and 8 NVIDIA V100 GPUs in a distributed fashion with a batch size of 256. We train and evaluate both PECOS and BETL models using the same training and evaluation set.

Training BM-BETL. During distillation it is customary to use all outputs from the teacher model to train the student model. However, we find that doing so during blending degrades the model performance. We also find that using only the relevant products (query-product pairs associated via behavior) to teach the BM-BETL model yields better performance than using all predictions. We hypothesize that online hard-negative sampling is much better at creating separation between relevant and irrelevant products and the negative predictions are not needed (see Section 7.4 for details). For relevant products not in the top-100, we use a fixed value of $25e^{-4}$ rather than the actual score which may be lower. Finally, we use $\lambda = 0.5$ (see Eq. 3), i.e., equal importance to both L_T and L_B (see Section 7.1).

Baselines. For our model comparison, we use two common loss functions used in traditional distillation approaches:

- **PointMSE.** A pointwise MSE loss function is a standard distillation loss function already used by many other studies for ranking models [12, 20, 42]. Our implementation is similar to the baseline used by Hofstätter et al. [20].

$$L_{PMSE} = \text{MSE}(y_i^+ - y_{pi}^+) + \text{MSE}(y_i^- - y_{pi}^-) \tag{4}$$

- **MarginMSE.** We implement the loss function proposed by Hofstätter et al. [20] for optimizing the margin between the scores of the relevant and the non-relevant sample items per query. The loss function utilize the margin between the positive and negative samples of the teacher model and

[1] https://github.com/amzn/pecos

Table 1: BM-BETL shows large gains in recall over BETL with a small loss in precision metrics. All highlighted improvements (in green) are significant at $\alpha = 0.05$ in a pairwise two-sample t-test.

	English-store		non-English store	
Model	P@16	R@100	P@16	R@100
PECOS	0.5316	**0.6244**	0.8187	**0.7657**
BETL-WU	0.5163	0.3634	0.6215	0.4779
BETL	**0.6200**	0.5415	**0.8484**	0.5998
PointMSE	0.0091	0.0470	0.3391	0.0919
MarginMSE	0.2584	0.5436	0.7941	0.5991
BM-BETL	0.6147	0.6006	0.8359	0.7012
△ BM-BETL Improvement				
vs. BETL	-0.85%	**10.91%**	-1.47%	**16.91%**
vs. PECOS	**15.63%**	-3.81%	2.10%	-8.42%
vs. MarginMSE	**137.89%**	**10.48%**	**5.26%**	**17.04%**

has been shown to perform better than other existing loss functions when the architecture of the models are different.

$$L_{MMSE} = \text{MSE}(y_i^+ - y_i^-, y_{pi}^+ - y_{pi}^-) \tag{5}$$

Evaluation Metrics. We measure the ranking performance with behavior-based recall metrics following other similar studies [8, 43], using historical purchase activity as the gold label. Using historical purchase activity data for performance evaluation might suffer from display bias—relevant products may not have been purchased. In order to estimate precision without display bias, we instead use crowd-sourced judgements to map each prediction as *Relevant* or *Irrelevant*; precision is the percentage of predictions that are Relevant.

$$P@k = \frac{1}{k}\sum_{i=1}^{k} isRelevant(\hat{y}_i) \tag{6}$$

where $isRelevant(\hat{y}_i) \in \{0, 1\}$ is the crowd-sourced decision on whether the predicted product $\hat{y}_i$ at rank i is relevant(1) or not(0).

$$R@k = \frac{1}{|Y|}\sum_{i=1}^{k} I(\hat{y}_i, y_i) \tag{7}$$

where $I(\hat{y}_i, y_i) \in \{0, 1\}$ is the indicator function indicating if the predicted product $\hat{y}_i$ matches the purchased ground truth product y_i and Y is the set of purchased products (ground truth) for a given query.

We are particularly interested in the precision of the 16 products appearing on the first page of results and on the 100 products retrieved overall, thus we report P(recision)@16 and R(ecall)@100 in subsequent sections[2]. Since manual evaluation is expensive and time-consuming, we randomly sample 200 queries and crowd-sourced the top-16 products of each model for each query.

[2]We note that other metrics such as mean Average Precision (mAP) and Recall@{1, 5, 16, 20, 100} follow the similar patterns. For simplicity we only report the most important measurements.

5.2 Experimental Results

Table 1 presents our main results. PECOS has a better recall with R@100 at 0.6244 and 0.7657 for the English and non-English stores respectively against 0.5415 and 0.5998 for BETL. On the other hand, BETL has better precision with P@16 at 0.6200 and 0.8484 against 0.5316 and 0.8187 for PECOS in the two stores. The aim of our model is to improve the recall of BETL (the production model) without affecting its precision. All improvements shown in green are significant at $\alpha = 0.05$ in a pairwise two-sample t-test. The blended model BM-BETL beats BETL in recall, achieving R@100 of 0.6006 (10.91% significant relative improvement) and 0.7012 (16.91% significant relative improvement) for the English and non-English stores, respectively. The loss in precision (P@16) from BETL is relatively small as compared to recall improvements: -0.85% and -1.47% respectively and not significant at $\alpha = 0.05$. For comparison, we also show that our model beats PECOS in precision significantly in the English-store, though recall is not significantly affected.

We also report the BETL warm-up checkpoint results (BETL-WU) and as it can be seen, the in-batch hard negative mining with triplet loss is effective in improving the model's performance. Comparing BM-BETL results with both distillation baselines suggests that for our problem setting existing distillation approaches are not helpful. PointMSE results show a drastic performance drop for both stores, indicating catastrophic forgetting. Interestingly MarginMSE results show slight changes in recall values while a significant drop in the precision of top-ranked relevant products.

6 BM VS. RANK FUSION (RF)

Rank Fusion (RF) [2, 16, 21, 28, 29, 34] is a family of methods used to aggregate the ranked results of different systems in order to provide robust results to the user. These methods require inference-time predictions from both PECOS and BETL in order to aggregate them. We cannot deploy both PECOS and BETL simultaneously, but we compare our blended model to few well-known RF methods to see how they perform against a theoretical ideal.

CombSum (sum of normalized scores from individual retrieval systems) and CombMNZ (multiplies CombSum score by the number of non-zero relevance scores) are widely used score-based approaches [16]. CombMIN and CombMAX return the min and max normalized score across systems respectively. We also test rank-based approaches — Reciprocal Rank Fusion (RRF) (sums of reciprocal rank for an item over each ranking [14]), Vector Space Model (VSM) (uses nearest neighbors [38]) and Rank-Biased Centroid (RBC) (centroid over a set of rankings from query variations for a topic [1]).

Table 2 presents P@16, computed using crowd-sourced judgements and R@100, computed using a random sample of queries of 50K queries, for the RF methods in the English store. We conduct pairwise two-sample t-tests of each method (including PECOS and BETL) with our BM-BETL predictions. At $\alpha = 0.05$, there is no significant difference between the P@16 of most of the RF methods compared to BM-BETL, indicating that our blended model performs as well as any RF method. Since RRF and RBC directly incorporate PECOS ranks, we see that they perform better than BM-BETL in terms of R@100. Similarly, CombMIN seems to benefit from normalized score distributions, though we cannot guarantee which of

Table 2: BM-BETL comparison with Rank Fusion. Significant difference at $\alpha = 0.05$ in a two-tailed paired t-test with BM-BETL is indicated by †.

	CombSUM	CombMNZ	CombMAX	CombMIN	RRF	VSM	RBC	BETL	PECOS	BM-BETL
P@16	**0.6200**	**0.6200**	**0.6200**	0.6059	0.5997	0.5681†	0.5988	**0.6200**	0.5316†	0.6147
R@100	0.5853†	0.5853†	0.5879†	0.6611†	**0.7398**†	0.5331†	0.7369†	0.5643†	0.6555†	0.6071

the Comb* methods would perform the best for arbitrary models. However, all RF methods need two (or more) models to be deployed simultaneously. BM-BETL achieves similar performance using a single model.

7 FURTHER ANALYSIS

In this section, for simplicity, we present our results only for the English store, noting that the non-English store follows similar trends.

7.1 λ Sensitivity Analysis

λ represents the importance given to the blending loss during the teaching phase (refer to Eq. 3). BETL-BM consists of blending a BETL model with an enhanced loss function.

Fig. 4 shows R@100 for different λ. With $\lambda = 0.0$ the function has not changed from BETL, thus the saturated model is insensitive to further training exposure, and no improvement is seen. With $\lambda = 1.0$ performance degrades drastically, indicating catastrophic forgetting due to the distribution shift in the relevance values. The best performance is found at $\lambda = 0.5$, where BETL is able to learn from PECOS without major disruption.

7.2 Smoothing PECOS Predictions

We study if PECOS score softening can provide better dark knowledge and help the blending performance [19]. Softmax softening is defined in Eq. 8. At temperature $T = 1.0$ there is no smoothing, while $T = \infty$ results in a uniform distribution with all labels equally likely. Normally, T is set to a value slightly over 1.0.

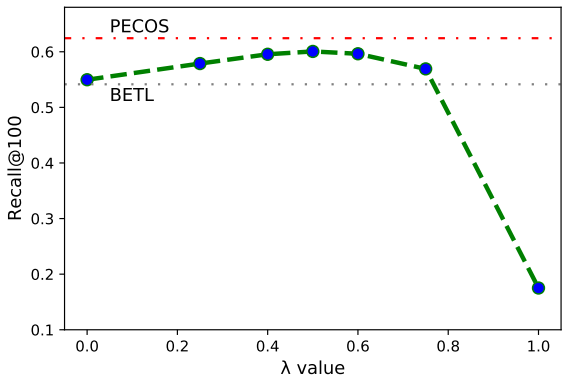


Figure 4: λ sensitivity analysis

Table 3: BM-BETL offline evaluation with smoothing out PECOS predictions during training with $\lambda = 0.5$

PECOS Smoothing	R@100
Original Scores	**0.6006**
Smoothing Scores with $T = 2.0$	0.5783
Smoothing Scores with $T = 5.0$	0.5764
Smoothing Scores with $T = 20.0$	0.5763
Reciprocal Ranking as the Score	0.5877

$$p_i = \frac{e^{(z_i/T)}}{\sum_j e^{(z_j/T)}} \tag{8}$$

Table 3 presents the R@100 for different values of T. In addition to temperature, we also smooth with PECOS reciprocal rankings ($\frac{1}{\text{rank}}$) as the score instead of the model's score. All of our results suggest that smoothing the secondary model's original score results in sub-optimal performance. We hypothesize that this is due to the differences between ranking and classification problems as well as blending vs. distillation. In a blending scenario similar to distillation, i.e., when both the models have similar form (say both are Bi-encoders with different architectures), smoothing the score distributions may provide better results. For an arbitrary secondary model though, we think it is preferable to not use smoothing.

7.3 Compare with PECOS as a "Ground Truth"

Our objective for BM-BETL is to improve the recall of BETL. Since PECOS has a higher recall, improving the recall of BETL will mean that the predictions of BM-BETL will be more similar to PECOS predictions than BETL predictions are to PECOS predictions. To see if BM-BETL predictions have moved closer to PECOS predictions, we consider PECOS ranking as truth data and evaluate BM-BETL and BETL using these relevance labels. For relevance labels, we considered top-100 PECOS rankings for the crowd-sourced evaluation set. To account for the distribution shift of the scores, we used a relevance score of $(101 - rank)$. As a result, product at rank 1 has a relevance 100, rank 2 $\rightarrow$ 99, and so on. This enabled us to compare BETL and BM-BETL against PECOS using traditional ranking evaluation metrics. Table 4 presents our results. As it can

Table 4: Compare with PECOS as "Ground Truth"

Model	NDCG	MAP	Prec@16	R@100
BETL	0.2296	0.0692	0.3185	0.1578
BM-BETL	**0.2573**	**0.0765**	**0.3630**	**0.1657**

be seen, with the blended model we obtain ranking more similar to PECOS according to all metrics when compared to BETL.

7.4 Usefulness of Irrelevants: an Ablation Study

We studied a variety of approaches to utilize our secondary model's predictions of products irrelevant to a query, along with relevant products. However, the blended model performance degraded whenever we used the irrelevant predictions. Here, we report a summary of most important directions one could examine. We restrict the irrelevant products explored to the top-k ($k = 100$) predictions of the PECOS model. For every query Q_i and the corresponding product relevant D_i^+, we extract PECOS ranking score as y_{pi}^+ and for irrelevant product, D_{pi}^-, PECOS prediction is noted as y_{pi}^-. We denote the Blended Models prediction as y_i^{BM+} and y_i^{BM-} for relevant and irrelevant products. We define the following general loss terms, in addition to L_T (Eq. 2) and L_B (Eq. 3).

- Blend loss term for relevant product sets:

$$L_{BR} = \sum_i \log\left(1 + e^{(y_i^{BM+} - y_{pi}^+)^2}\right) \tag{9}$$

- Blend loss term for irrelevant product sets:

$$L_{BI} = \sum_i \log\left(1 + e^{(y_i^{BM-} - y_{pi}^-)^2}\right) \tag{10}$$

A summary of alternative approaches we took for our BM framework is given below.

(1) **Replace.** We replace online model-based hard negatives with PECOS-predicted irrelevant products, To do so we randomly assign one irrelevant product to every D_i^+ during batch data preparation. The same loss, i.e. L_{BM} is used with the difference that in L_T we use model predictions for the irrelevant products.
(2) **Sample.** Similar to *Replace*, with the difference that for every D_i^+, we sample a lower ranked product from PECOS rankings weighted by their PECOS scores. The loss is similar to previous scenario.
(3) **Complement-R.** We keep the online model-based hard negatives and complement it with adding a PECOS based negative product, for every query-product pair. Choosing negative product is done similar to *Sample*.

$$L_{BMC} = \lambda \times [\alpha \times L_T + (1-\alpha) \times \sum_i \log\left(1 + e^{(y_i^{BM-} - y_i^+)}\right)] + (1-\lambda) \times [\beta \times L_{BR} + (1-\beta) \times L_{BI}]$$

(4) **Complement-S.** Similar to *Complement-R*, with the difference that choosing negative product is done similar to *Sample*. L_{BMC} is used as the loss function.
(5) **Semi-Relevant.** Investigating some of the PECOS top-ranked irrelevant products, we found that though they hadn't elicited customer behavior, manual evaluation still deemed these products as relevant. Based on this, we wanted to see whether considering such products as relevant (or relatively relevant) can help the blended model to better generalize. As a result, we take PECOS irrelevant products (above the threshold of 0.0025) as semi-relevant and defined a "confidence score" when we paired with the corresponding query. Considering 1.0 as the confidence for relevant query-product pairs, we examined each of these confidence values for our semi-relevant products: $\{0.5, 0.2, 0.0\}$. We also examined using reciprocal rank as a dynamic confidence score. This is similar to the RankDistil framework provided by Reddi et al. [35]. For the loss function, every batch of data gets a confidence vector c and multiplied to the original BM loss term—i.e., $L_{BMSR} = c \times L_{BM}$.

Table 5: PECOS False Positives Usefulness Analysis ($\lambda = 0.5$)

Exp. Scenario	R@100
Original Loss L_{BM}	**0.6006**
Replace	0.1365
Sample	0.0166
Complement-R, [$\alpha = 0.9$, $\beta = 0.9$]	0.5831
Complement-S, [$\alpha = 1.0$, $\beta = 0.9$]	0.5694
Semi-Relevant, [confidence= 0.0]	0.5289
Semi-Relevant, [confidence= 0.2]	0.5045
Semi-Relevant, [confidence= 0.5]	0.4606
Semi-Relevant, [confidence= $\frac{1}{rank}$]	0.5716

Table 5 presents our results. For simplicity, we do not present a sensitivity analysis on different combinations of hyper-parameters with our experiments. Instead we present the results of the best hyper-parameter combination for each loss. Choosing hard negatives via *Sample* drastically degrades performance and generally *Replace* is a better solution. This is also true for Complement experiments. As we see, weighting the loss terms with relevant products (large α and β) performs better than other approaches but not as well as only using relevant samples. For the case of Semi-Relevant hard negatives, we see that with higher confidence our results degraded. This again indicates that the score distribution shift between the two models plays a part in the performance of the blended model and it may be possible to tune hard-negative sampling and loss functions based on secondary model. Since the PECOS and BETL formulations are significantly different, we believe that our techniques can be used for an arbitrary secondary model.

8 CONCLUSION

This work presents an approach to blend two distinct semantic matching models into a single model. Our blending approach is able to improve performance of the primary model with the help of a secondary model. We show that our blended model, BM-BETL, improves over R@100 of BETL with a slight loss in precision. We similarly improve over P@16 of PECOS with a slight loss in recall. This study allows one to incorporate the strengths of multiple models into a single model that meets the memory and latency requirements of modern retail stores.

Our experiments suggested that (a) With totally different formulations of the semantic matching problem, existing knowledge distillation approaches are not helpful, (b) PECOS relevant predictions help bi-encoder training, (c) PECOS irrelevant predictions adversely impact blending performance when compared to online primary model-based hard negative mining, (d) Softening PECOS predictions result in sub-optimal performance, (e) Blend loss needs

to be similar to the models' original loss, and (f) Blending is able to achieve the overall robustness of strong rank fusion baselines.

Our proposed approach is flexible enough to apply to most distillation scenarios. In the case of product search, we plan to extend the proposed solution to a broader set of semantic matching (or even lexical matching) models. Since the secondary model is not intended to be deployed, any complex model can be used without the latency concerns. As a result, blending and model distillation combinations along with multi-teacher frameworks can be extended using our study. In addition, it would be interesting to blend the set of features every secondary model constructs into the primary model. For example, PECOS feature construction can be also added to our bi-encoder model.

9 ETHICS STATEMENT

We only use query and product interaction data for our work. Any identifiable user information is completely stripped before we can access the data. As our method is ultimately used to retrieve a set of products in an e-commerce store, incorrect predictions will not cause harm to the user besides an unsatisfactory experience.

REFERENCES

[1] Peter Bailey, Alistair Moffat, Falk Scholer, and Paul Thomas. 2017. Retrieval consistency in the presence of query variations. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 395–404.

[2] RODGER BENHAM. 2018. *Improving Recall In Text Retrieval Using Rank Fusion*. Ph.D. Dissertation. RMIT University Australia 5.

[3] Kuluhan Binici, Nam Trung Pham, Tulika Mitra, and Karianto Leman. 2021. Preventing Catastrophic Forgetting and Distribution Mismatch in Knowledge Distillation via Synthetic Data. *CoRR* abs/2108.05698 (2021). arXiv:2108.05698 https://arxiv.org/abs/2108.05698

[4] Leonid Boytsov and Eric Nyberg. 2020. Flexible retrieval with NMSLIB and FlexNeuART. *arXiv preprint arXiv:2010.14848* (2020).

[5] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 535–541.

[6] Wei-Cheng Chang, Daniel Jiang, Hsiang-Fu Yu, Choon Hui Teo, Jiong Zhang, Kai Zhong, Kedarnath Kolluri, Qie Hu, Nikhil Shandilya, Vyacheslav Ievgrafov, et al. 2021. Extreme multi-label learning for semantic matching in product search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2643–2651.

[7] Wei-Cheng Chang, Daniel Jiang, Hsiang-Fu Yu, Choon Hui Teo, Jiong Zhang, Kai Zhong, Kedarnath Kolluri, Qie Hu, Nikhil Shandilya, Vyacheslav Ievgrafov, Japinder Singh, and Inderjit S. Dhillon. 2021. *Extreme Multi-Label Learning for Semantic Matching in Product Search*. Association for Computing Machinery, New York, NY, USA, 2643–2651. https://doi.org/10.1145/3447548.3467092

[8] Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training Tasks for Embedding-based Large-scale Retrieval. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rkg-mA4FDr

[9] Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit S Dhillon. 2020. Taming pretrained transformers for extreme multi-label text classification. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3163–3171.

[10] Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. 2018. Darkrank: Accelerating deep metric learning via cross sample similarities transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[11] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. 2017. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282* (2017).

[12] Jaekeol Choi, Euna Jung, Jangwon Suh, and Wonjong Rhee. 2021. Improving Bi-encoder Document Ranking Models with Two Rankers and Multi-teacher Distillation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2192–2196.

[13] Daniel Cohen, John Foley, Hamed Zamani, James Allan, and W Bruce Croft. 2018. Universal approximation functions for fast learning to rank: Replacing expensive regression forests with simple feed-forward networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1017–1020.

[14] Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. 758–759.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. 2019 Conf of the NAACL: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4171–4186. https://doi.org/10.18653/v1/N19-1423

[16] Edward A Fox and Joseph A Shaw. 1994. Combination of multiple searches. *NIST special publication SP* 243 (1994).

[17] Krzysztof J Geras, Abdel-rahman Mohamed, Rich Caruana, Gregor Urban, Shengjie Wang, Ozlem Aslan, Matthai Philipose, Matthew Richardson, and Charles Sutton. 2015. Blending lstms into cnns. *arXiv preprint arXiv:1511.06433* (2015).

[18] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision* 129, 6 (2021), 1789–1819.

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Distilling the knowledge in a neural network. *NIPS 2014, Deep Learning Workshop* (2014).

[20] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2021. Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation. arXiv:2010.02666 [cs.IR]

[21] D Frank Hsu and Isak Taksa. 2005. Comparing rank and score combination methods for data fusion in information retrieval. *Information retrieval* 8, 3 (2005), 449–480.

[22] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2333–2338.

[23] Thorsten Joachims, Hang Li, Tie-Yan Liu, and ChengXiang Zhai. 2007. Learning to Rank for Information Retrieval (LR4IR 2007). *SIGIR Forum* 41, 2 (Dec. 2007), 58–62. https://doi.org/10.1145/1328964.1328974

[24] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data* (2019).

[25] SeongKu Kang, Junyoung Hwang, Wonbin Kweon, and Hwanjo Yu. 2020. DE-RRD: A Knowledge Distillation Framework for Recommender System. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 605–614.

[26] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. *arXiv preprint arXiv:1906.00300* (2019).

[27] Youngjune Lee and Kee-Eung Kim. 2021. Dual Correction Strategy for Ranking Distillation in Top-N Recommender System. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3186–3190.

[28] David Lillis. 2020. On the Evaluation of Data Fusion for Information Retrieval. In *Forum for Information Retrieval Evaluation*. 54–57.

[29] David Lillis, Fergus Toolan, Rem Collier, and John Dunnion. 2006. Probfuse: a probabilistic approach to data fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. 139–146.

[30] Yuang Liu, Wei Zhang, and Jun Wang. 2020. Adaptive multi-teacher multi-level knowledge distillation. *Neurocomputing* 415 (2020), 106–113.

[31] Hanqing Lu, Youna Hu, Tong Zhao, Tony Wu, Yiwei Song, and Bing Yin. 2021. Graph-based Multilingual Product Retrieval in E-Commerce Search. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Papers*. 146–153.

[32] CD Manning, P Raghavan, and H Schütze. 2008. Xml retrieval. In *Introduction to Information Retrieval*. Cambridze University Press.

[33] Priyanka Nigam, Yiwei Song, Vijai Mohan, Vihan Lakshman, Weitian Ding, Ankit Shingavi, Choon Hui Teo, Hao Gu, and Bing Yin. 2019. Semantic product search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2876–2885.

[34] Rabia Nuray and Fazli Can. 2006. Automatic ranking of information retrieval systems using data fusion. *Information processing & management* 42, 3 (2006), 595–614.

[35] Sashank Reddi, Rama Kumar Pasumarthi, Aditya Menon, Ankit Singh Rawat, Felix Yu, Seungyeon Kim, Andreas Veit, and Sanjiv Kumar. 2021. Rankdistil: Knowledge distillation for ranking. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2368–2376.

[36] Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

[37] Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94*. Springer, 232–241.

[38] Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Commun. ACM* 18, 11 (1975), 613–620.

[39] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *NeurIPS 2019, Workshop on Energy Efficient Machine Learning and Cognitive Computing* (2019).

[40] Yanyao Shen, Hsiang-fu Yu, Sujay Sanghavi, and Inderjit Dhillon. 2020. Extreme Multi-label Classification from Aggregated Labels. In *International Conference on Machine Learning*. PMLR, 8752–8762.

[41] Jiaxi Tang and Ke Wang. 2018. Ranking distillation: Learning compact ranking models with high performance for recommender system. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2289–2298.

[42] Amir Vakili Tahami, Kamyar Ghajar, and Azadeh Shakery. 2020. Distilling knowledge for fast retrieval-based chat-bots. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in information retrieval*. 2081–2084.

[43] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *International Conference on Learning Representations*. https://openreview.net/forum?id=zeFrfgyZln

[44] Shan You, Chang Xu, Chao Xu, and Dacheng Tao. 2017. Learning from multiple teacher networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1285–1294.

[45] Hsiang-Fu Yu, Kai Zhong, and Inderjit S Dhillon. 2020. PECOS: Prediction for enormous and correlated output spaces. *arXiv preprint arXiv:2010.05878* (2020).