CUPID: Curriculum Learning Based Real-Time Prediction using Distillation

Arindam Bhattacharya  
Amazon, India  
aribhat@amazon.com

Ankith MS  
Amazon, India  
ankiths@amazon.com

Ankit Gandhi  
Amazon, India  
ganankit@amazon.com

Vijay Huddar  
Amazon, India  
vhhuddar@amazon.com

Atul Saroop  
Amazon, India  
asarooop@amazon.com

Rahul Bhagat  
Amazon, USA  
rbhagat@amazon.com

Abstract

Relevance in E-commerce Product Search is crucial for providing customers with accurate results that match their query intent. With recent advancements in NLP and Deep Learning, Transformers have become the default choice for relevance classification tasks. In such a setting, the relevance model uses query text and product title as input features, and estimates if the product is relevant for the customer query. While cross-attention in Transformers enables a more accurate relevance prediction in such a setting, its high evaluation latency makes it unsuitable for real-time predictions in which thousands of products must be evaluated against a user query within few milliseconds. To address this issue, we propose CUPID: a Curriculum learning based real-time Prediction using Distillation that utilizes knowledge distillation within a curriculum learning setting to learn a simpler architecture that can be evaluated within low latency budgets. In a bi-lingual relevance prediction task, our approach shows a 302 bps improvement on English and 676 bps improvement for low-resource Arabic, while maintaining the low evaluation latency on CPUs.

1 Introduction

Large-scale e-commerce search systems, such as those used by companies like Amazon, Walmart etc., typically employ a multi-step process to retrieve relevant products for a given query (Guo et al., 2022). The first step in this process is to generate a matchset that is approximately relevant to the query, followed by a series of steps that optimize for relevance, customer interest and other associated metrics (Momma et al., 2022). In such a setting, it is imperative to have features that accurately capture relevance between the customer’s query-intent and the candidate set of products in the matchset.

Recently, transformer-based models such as BERT have proven to be highly effective in estimating relevance between a query and a product by using cross-attention (Mangrulkar et al., 2022a; Nogueira and Cho, 2019; Wang et al., 2019), self-attention (dual-encoders) (Bhattacharya et al., 2023; Reimers and Gurevych, 2019a; Mangrulkar et al., 2022b), or late interaction (Khattab and Zaharia, 2020a; Santhanam et al., 2021; Lu et al., 2022) models. The use of cross-attention in transformers has been shown to be effective, as it allows the model to take into account both the query and the product when determining relevance (Menon et al., 2022; Hofstätter et al., 2020). However, these models come at a cost, as they require heavy computational resources and have a high latency even during evaluation. In large-scale search systems, it is important to perform real-time relevance predictions, as thousands of products need to be processed for each query, with strict latency requirements.

For this reason, dual-encoder models are more suitable, as they can provide real-time relevance predictions with low latency requirements. In such a setting, the task of estimating relevance is reduced to carrying out simple vector operations, typically a dot product of high-dimensional vectors, one representing the query and the others representing products. Under such a setting, the vectors representing products are pre-computed and cached, while those for the query are computed on-the-fly. Such on-the-fly computation of query vectors or embeddings in low latency settings restricts us from using a full stack of transformer layers, as is typical to models like BERT. To address the issue of latency in computation of query embeddings, we instead borrow from the architecture used in (Nigam et al., 2019) to be used for the query arm for our asymmetric dual-encoders, while continuing to use a full stack of transformers for the product arm (as shown in Figure 1). Their simple architecture is comprised of a word embedding layer and a mean-pool layer (based on (Huang et al., 2013a), referred loosely as...
DSSM (Deep Structured Semantic Model) henceforth), which is more suitable for real-time scenarios with low latency requirements. However, this model lacks the rich semantic representation of models built purely of transformers. To bridge this gap, we leverage knowledge distillation techniques, where we use the DSSM of the query arm as a student model to learn the rich semantic representation from the transformer model.

To address this issue, we propose CUPID: a Curriculum learning based real-time prediction using distillation that utilizes knowledge distillation within a curriculum learning setting to learn a simpler architecture that can be evaluated within low latency budgets. Our contributions to the literature can be summarized as follows:

• We show that our low-latency model benefits more through knowledge distillation from a structurally similar dual-encoder transformer model as a teacher, rather than from a cross-encoder transformer model. Even though the cross-encoder transformer model is more accurate, the student is able to learn better from a structurally similar teacher.

• We demonstrate that a learning regime, where a structurally similar student, optimizes for a cross-entropy loss for the first few epochs, followed by a curriculum-styled learning of the teacher embeddings using an alignment loss outperforms other alternative learning regimes.

2 Related Work

Cross Encoders and Bi-Encoders Cross encoders and bi-encoders are two distinct architectures used in sentence pair modeling. While both approaches aim to capture the relationship between two sentences, they differ in how they encode the sentences and produce their representations.

Cross encoders (Reimers and Gurevych, 2019b) jointly encode both sentences into a fixed-length representation. The shared encoder takes a pair of sentences as input and captures the interactions between them. This joint encoding helps capture both local and global interactions between the sentences, leading to improved representation learning.

Bi-encoders (Kiros et al., 2015), on the other hand, use separate encoders for each input sentence. Each sentence is encoded independently, producing two separate representations. These representations are then compared using a similarity function (e.g., dot product, cosine similarity) to determine the relationship between the sentences.

Both cross encoders and bi-encoders have their advantages and are suitable for different scenarios. Cross encoders excel at capturing the interaction between sentences, while bi-encoders are computationally efficient. The choice between the two architectures depends on the specific requirements. For real-time predictions, cross encoders are often infeasible, but bi-encoders excel due to the ability to utilize pre-computed indexes.

Low latency Transformers In recent years, with the state-of-the-art performance of transformers in NLP applications, there has been a demand to make transformers suitable for real-time e-commerce applications. And the reduction in computation time and latency is crucial for transformers to be viable for such use cases. There are three main themes (Lin et al., 2022) into which the advances in the design of low-latency transformers can be categorised. (I) architectural level, (II) component level, and (III) technique-based. At the architectural level (I), modifications are made at a higher level, such as the use of lightweight transformers like Funnel Transformer (Dai et al., 2020), Lite Transformer (Wu et al., 2020), and DeLighT (Mehta et al., 2020). Additionally, pruning techniques reported in (Kwon et al., 2022; Gordon et al., 2020; Mao et al., 2020; Hou et al., 2020) aim to reduce the size and computation by eliminating unimportant weights. Finally, Quantization-based approaches (Ganesh et al., 2021) and model compression (Bai et al., 2019) are used to compress weights and activations. At the component level (II), there is a focus on efficient self-attention, such as in the works of (Wang et al., 2020), and delayed interaction networks (Reimers and Gurevych, 2019c; Khattab and Zaharia, 2020b; Santhanam et al., 2021). Lastly, under the category of technique-based (III), research efforts have been made in areas such as early exit (Mangrulkar et al., 2022a; Zhou et al., 2020; Xin et al., 2020) and knowledge distillation (Hinton et al., 2015; Sanh et al., 2020).

Knowledge Distillation Knowledge Distillation (KD)(Hinton et al., 2015) is a widely researched topic that enables the transfer of knowledge from a complex, pre-trained model to a smaller, more computationally efficient model. With the rise of BERT (Devlin et al., 2018) and the corresponding growth in textual data, there has been growing interest in applying KD to BERT models in the field of NLP. Distill BERT (Sanh
et al., 2020) is one such seminal work along with (Tinybert(Jiao et al., 2020), Fast BERT(Liu et al., 2020), Task specific BERT (Tang et al., 2019), Patient Knowledge Distillation BERT (Sun et al., 2019a). These propose using KD to transfer knowledge from a large BERT model to a smaller model. While the Distill BERT makes BERT models faster by 60%, they cannot be used in real-time because e-commerce applications such as semantic matching (Huang et al., 2013b; Nigam et al., 2019) have limited computational resources (especially GPUs) and strict latency requirements. The latest TwinBERT (Lu et al., 2020) proposes the distillation of 12-layer BERT to 6-layer twin tower BERT structure, thus, permitting pre-compute of document embeddings and cache in memory saving additional computational time. However, the TwinBERT requires GPUs during inference to compute the query arm embeddings within the latency budgets.

3 Our Approach

In this section, we present our proposed approach CUPID that uses bi-encoder transformer models as teachers, and learn asymmetric student models having DSSM architecture in the query side and transformer architecture in the product side. Using cross-encoders as teachers is natural, however, in this work we show that using bi-encoder teacher yields better performance and allows better transferring of knowledge to bi-encoder student models (refer to Section 4.3). Bi-encoder teacher model enables better transfer of semantics to a simpler bi-encoder student model with better alignment of query embeddings. Later in the section, we present a very simple curriculum learning framework for training bi-encoder student model by progressively increasing the difficulty of task. We present the detailed architecture (also shown in Figure 1) below.

3.1 Teacher Model

The teacher model used is a Siamese BERT (SBERT) architecture. SBERT first computes fixed size contextual representation for an entity by mean pooling the BERT model’s output, followed by a dense layer with tanh activation to get entity embedding. We use the same BERT model for representing both query and product to enable the transfer of language semantics between them. Finally, the similarity \( \hat{y}_i \) between entities is determined by the cosine distance between the embeddings. Note that, although we use BERT, any transformer model can act as an alternative to the BERT model. In Section 4, we present results on some multilingual datasets, where we use the multilingual XLM RoBERTa transformers. It should also be noted that while cross-encoder models are widely used as teacher models in knowledge distillation tasks for NLP, for our application, we claim that a bi-encoder model is much better suited for the task of teaching an asymmetric bi-encoder student. Section 4.3 justify this claim.

Teacher Training Objective Let’s assume our training samples are represented by the tuple \((q_i, p_i, y_i)\), where \(q_i\) is a query entity and \(p_i\) is a product entity, and \(y_i\) is the ground truth label. Let \(S(\cdot)\) be the function returning the embedding Embedding\(\text{Transformer}\). Then the predicted semantic similarity between the query and the product is measured using the cosine similarity as \(\hat{y} = \text{sim}(q, p) = \cos(S(p), S(q))\). We train the teacher model using the binary cross entropy loss, computed as 

\[
\text{loss}_{ce}(\hat{y}, y) = y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y}).
\]

3.2 Student Model

In the teacher model, both arms use SBERT (or any transformer) models to generate query and product representations. While SBERT generates better representations of the entities, the latency of such large models are prohibitively high for real-time representation generation for queries. We therefore train a smaller model for query representation, and retain the SBERT model for product representation generation, which can be computed and indexed in advance, and do not require real-time latency. The high-level architecture of the query arm is similar to that of SBERT. The only difference is that we use embedding lookup and mean-pool layer instead of BERT encoder to generate the intermediate representation for query. We use the teacher BERT model to generate embedding for the product.

Student Model Training Here we discuss how we distill the knowledge from the query arm of the teacher SBERT models to the smaller student models.

The standard way to distill knowledge from teacher to student is to use a loss function over the predicted similarities between the query and the product of teacher model, \(\hat{y}_T\) and student model, \(\hat{y}_S\) (Sanh et al., 2020), that is, the student tries
to imitate the final output of the teacher. We call this loss the *imitation* loss, which is computed as
\[
\text{loss}_{\text{imitation}}(\hat{y}_T, \hat{y}_S) = -\hat{y}_T \log(\hat{y}_S).
\]

In this work, we use a loss which tries to align the representation generated by the student to that of the teacher similar to (Sun et al., 2019b; Li et al., 2021). We name this loss *alignment* loss. Let \(D(\cdot)\) be the function returning the embedding \(\text{Embedding}_{\text{DSSM}}\) generated by the student model. Then the alignment loss for a query \(q\) is computed as
\[
\text{loss}_{\text{alignment}}(S(q), D(q)) = 1 - \cos(S(q), D(q)).
\]

In our experiments, we noted that if we introduce \(\text{loss}_{ce}\) in the student model, it performs better. Furthermore the \(\text{loss}_{\text{imitation}}\) becomes superfluous, that is removing it doesn’t affect the model performance. Section 4 shows the effect of using all three losses, and shows that adding imitation loss does not add to the model performance and removing it does not impact any negative impact, but simplifies the training process by removing a component of the loss.

**Curriculum Learning for Student Models** Initially, the query arm of the student model is misaligned due to random initialization, which causes instability in learning. To address this, we perform two stage curriculum learning. In the first stage, we train the student model for few epochs using the positive training data and random negative data. The random negatives are generated by randomly shuffling the products forming pairs \((q_i, p_j)\) such that \(i \neq j\). This approach provides the model with easy examples compared to the negatives present in the dataset itself. This method initializes the DSSM weights to be more aligned to generate the expected query representation. In Section 4, we show that this kind of training stabilizes the training and improves the performance.

In the second stage of curriculum learning, we scale up the alignment loss gradually during the model training. So initially, \(\alpha = 0\), and the model is effectively learning only from true labels. Gradually we scale up \(\alpha\), making the models objective more complicated: reduce the cross entropy loss with true labels, and align the student’s query arm with that of the teacher to generate similar embeddings. The progressive increase in model’s task defines this stage of curriculum learning rather than the difficulty of examples.

With these improvements, we now arrive at the proposed knowledge distillation loss:
\[
\text{loss}_{KD} = (1 - \alpha) \cdot \text{loss}_{ce} + \alpha \cdot \text{loss}_{alignment},
\]
where \(\alpha\) is the scaling factor that varies from 0 to 1 and is incremented each epoch.

**4 Experiments and Results**
In this section, we compare the performance of CUPID with the state of the art methods of knowledge distillation. We also study the latency of the student models and compare them with the latency of teacher model to show why BERT based models are not suitable for real time predictions. We then present results for a real world application of CUPID on an internal dataset. Finally we show the effect each of the losses have on the performance of the models, and explain the choice of
a bi-encoder teacher model compared to a better performing cross-encoder model.

4.1 Datasets

We performed the experiments on the Shopping Queries Dataset (Reddy et al., 2022) that was made openly available as part of KDD Cup 2022 workshop. The training dataset has around 800 thousand samples and the test dataset has around 400 thousand. For each query, the training dataset provides on average a list of up to 13 potentially relevant results, together with relevance judgements (Exact, Substitute, Complement, Irrelevant) indicating the relevance of the product to the query. For our experiments, we consider the problem as a binary classification where a product is either relevant (exact, substitute or complement) or irrelevant. Both training and test dataset contains around 1 negative example per 10 positive examples. We also present the results of CUPID and the baselines on datasets from Arabic language locales that have been subsampled from a leading e-commerce company’s history log and human-audited for ESCI labels, similar to the work of (Reddy et al., 2022). At this time, the data is not publicly available and is proprietary. We used a few hundred thousand records to train and test the models.

4.2 Experiments

We implement the CUPID and the baseline models described in Section 3 using PyTorch (Paszke et al., 2019) library and Hugging Face (Wolf et al., 2019) transformers.

Teacher Models Both the biencoder model and the cross-encoder model uses bert-base-uncased as a pretrained model, with a dense layer of size 128 to generate both query and product representation. The teacher is trained for 10 epochs using Adam optimizer with exponentially decaying learning rate. We use a batch size of 256 to fit the model into GPU memory. Due to the imbalance inherent in the dataset, we use two standard approaches to stabilize the training. First, we accumulate the gradients across 10 batches before applying the Adam updates. Second, we use a weighted sampler while loading the training batch to ensure that we over-sample the negatives to retain a balanced class distribution.

Student Models The student models use the same BERT arm to generate the product representation as the teacher. For the query generation, the

<table>
<thead>
<tr>
<th>ID</th>
<th>Experiment</th>
<th>CL Stage</th>
<th>AUC (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>Teacher</td>
<td>I</td>
<td>87.25</td>
</tr>
<tr>
<td>B</td>
<td>Student: CE (No KD)</td>
<td>I</td>
<td>83.15</td>
</tr>
<tr>
<td>S1</td>
<td>Student: IL</td>
<td>I</td>
<td>84.53</td>
</tr>
<tr>
<td>S2</td>
<td>Student: IL + AL</td>
<td>I</td>
<td>84.81</td>
</tr>
<tr>
<td>S3</td>
<td>Student: CE + AL</td>
<td>I</td>
<td>85.32</td>
</tr>
<tr>
<td>S4</td>
<td>Student: CE + IL</td>
<td>I</td>
<td>84.59</td>
</tr>
<tr>
<td>S5</td>
<td>Student: CE + IL + AL</td>
<td>I</td>
<td>84.80</td>
</tr>
<tr>
<td>H</td>
<td>CUPID</td>
<td>I &amp; II</td>
<td>86.17</td>
</tr>
</tbody>
</table>
Table 2: Area under ROC curve of various models on Arabic data. Cross entropy, imitation and alignment losses are represented as CE, IL and AL respectively. Stages of curriculum learning (CL) used are also indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Experiment</th>
<th>CL Stage</th>
<th>AUC (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>Teacher</td>
<td>I</td>
<td>89.04</td>
</tr>
<tr>
<td>B</td>
<td>Student: CE (No KD)</td>
<td>I</td>
<td>72.41</td>
</tr>
<tr>
<td>S1</td>
<td>Student: IL</td>
<td>I</td>
<td>74.63</td>
</tr>
<tr>
<td>S2</td>
<td>Student: IL + AL</td>
<td>I</td>
<td>76.14</td>
</tr>
<tr>
<td>S3</td>
<td>Student: CE + AL</td>
<td>I</td>
<td>76.59</td>
</tr>
<tr>
<td>S4</td>
<td>Student: CE + IL</td>
<td>I</td>
<td>75.02</td>
</tr>
<tr>
<td>S5</td>
<td>Student: CE + IL + AL</td>
<td>I</td>
<td>74.80</td>
</tr>
<tr>
<td>H</td>
<td>CUPID</td>
<td>I &amp; II</td>
<td>79.17</td>
</tr>
</tbody>
</table>

Finally, CUPID with the weighted loss achieves the best results with over 300 basis points boost over the baseline with no knowledge distillation and more than 150 basis points above the standard KD method using imitation loss.

Table 3: Comparison with Cross-Encoder (CE) teacher

<table>
<thead>
<tr>
<th>ID</th>
<th>Experiment</th>
<th>AUC (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>BiEncoder Teacher</td>
<td>87.25</td>
</tr>
<tr>
<td>CT</td>
<td>CE Teacher</td>
<td>89.76</td>
</tr>
<tr>
<td>H</td>
<td>CUPID: BiEncoder Teacher</td>
<td>86.17</td>
</tr>
<tr>
<td>CH</td>
<td>CUPID: CE Teacher</td>
<td>81.53</td>
</tr>
</tbody>
</table>

Table 4: Impact of Curriculum Learning

<table>
<thead>
<tr>
<th>ID</th>
<th>Experiment</th>
<th>CL Stage</th>
<th>AUC (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-</td>
<td>Student: No KD</td>
<td>None</td>
<td>82.09</td>
</tr>
<tr>
<td>B</td>
<td>Student: No KD. I</td>
<td>I</td>
<td>83.15</td>
</tr>
<tr>
<td>H-</td>
<td>CUPID II</td>
<td>II</td>
<td>85.65</td>
</tr>
<tr>
<td>H</td>
<td>CUPID I &amp; II</td>
<td>I &amp; II</td>
<td>86.17</td>
</tr>
</tbody>
</table>

improvement of CUPID over the baselines is more pronounced for this dataset.

4.3 Ablation Studies

In this section we look at the impact of some of the choices made by CUPID, studying their impact and comparing with the alternative approaches.

Why is curriculum-based learning important to the CUPID Model? Table 4 shows the effect of curriculum learning for the baseline and CUPID. B- and H- are the versions of baseline and CUPID without curriculum learning respectively. We notice that in each case, curriculum learning gives us a significant improvement in performance.

What is the role of the $\alpha$? How important is the Alpha scheduler? The parameter $\alpha$ is used to adjust the importance of the alignment loss in the CUPID model. In our experiments on non-English datasets, we found that a constant $\alpha$ could achieve an AUC of 85.32%. We hypothesized that the DSSM model, due to its basic architecture compared to the transformer model, would have difficulty learning the projection matrix and classification task jointly. Hence, by gradually increasing $\alpha$, we improved the model performance to 86.17%. We also note that the improvement is not due of the trade-off of achieving better results by letting the model have higher alignment loss, but $\alpha$ scaling actually helps reduce the alignment loss faster than when the alignment loss is not scaled, which justifies the hypothesis.

Why Bi-Encoder teacher and not cross-encoder? Cross encoder models are known to perform better in similarity matching tasks in NLP. So a natural question arises: why not train a cross-encoder

Latency Latency is a major concern for real time predictions. Here we compare the query arm latency of the teacher and student models to justify the need of a DSSM based students at the cost of some performance. We convert the models to ONNX before performing the inference. BERT has a latency of 11.6 ms, which is almost 4× that of the DSSM students, which is 3.2 ms. In real time, 11 ms is higher than the standard expected latency for real time applications. While the product representations can be pre-computed, this latency explains the need for a student model to generate the query representation. All the experiments to compute the latency was carried out for CPU on Amazon EC2 machines using p3.8xlarge instances with 2.3 GHz (base) and 2.7 GHz (turbo) Intel Xeon E5-2686 v4 processors.

4.2.2 Real World Application: Irrelevant Result Detection in Arabic Locales

Here we present the results of our method on the real world Arabic data. For this experiment, we trained the teacher model with pre-trained xlm-roberta model. This transformer model is trained on multiple languages and thus is more suitable for Arabic language data. The remaining parameters and the architecture of student models remain same. Table 2 shows the results of various methods on the Arabic data. Because of smaller size of dataset, larger vocabulary, and larger transformer model, the performance difference between the teacher and the students is larger. Also, the
model and use that as a teacher. Here we show that the cross encoder model performs better than the bi-encoder student. But due to the requirement of separate generation and indexing of query and product for real time predictions, we require a bi-encoder student. And due to the difference in semantics, the transfer of knowledge between a cross encoder teacher and a biencoder student doesn’t yield better results than with a biencoder teacher, as seen in Table 3. We notice that cross encoder teacher (CT) model performs better that biencoder teacher (T) but 250 basis points but the student trained with the bi-encoder teacher (H), which has access to alignment loss, greatly outperforms the student with cross encoder teacher (CH), which has access to only imitation loss which has a different semantics.

5 Conclusion

In this paper, we propose CUPID, a novel approach for knowledge distillation for real-time prediction of relevancy using curriculum learning. It uses a new loss function and two-stage curriculum learning framework to increase the influence of teacher gradually, resulting in a loss function that outperforms imitation learning based KD methods by up to 300 basis points.

References


Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.


Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Jiangnan Xia, Liwei Peng, and Luo Si. 2019. Structbert: Incorporating language structures into pre-training for deep language understanding.

