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
asaroop@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 ***CU**rriculum learning based real-time **P**red**I**ction using **D**istillation* 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 an 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 interaction 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 $Embedding_{Transformer}$. Then the predicted semantic similarity between the query and the product is measured using the cosine similarity as $\hat{y} = sim(q,p) = \cos(S(p), S(q))$. We train the teacher model using the binary cross entropy loss, computed as $loss_{ce}(\hat{y}, y) = y \cdot \log(\hat{y}) + (1-y) \cdot \log(1-\hat{y})$.

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

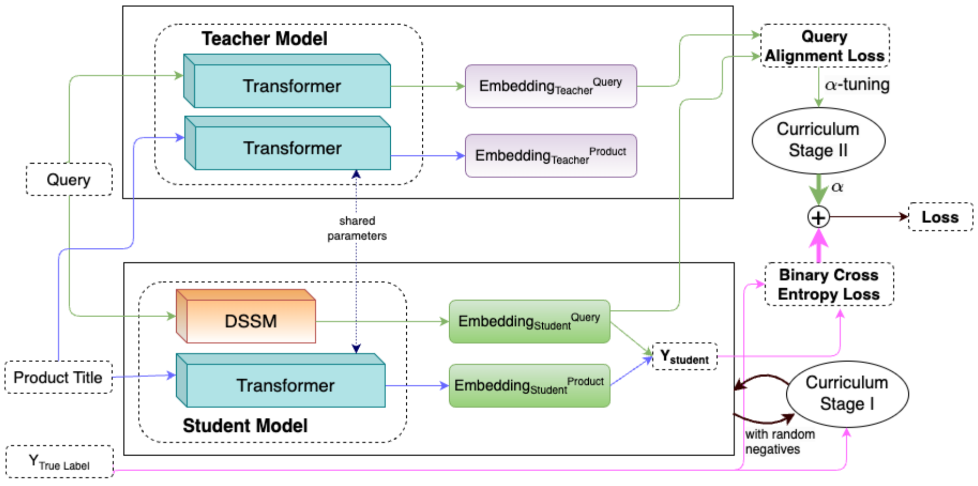

Figure 1: Our proposed architecture for learning asymmetric bi-encoder student model having DSSM architecture on the query arm and BERT architecture on the product arm with Alignment loss and Curriculum learning

to imitate the final output of the teacher. We call this loss the *imitation* loss, which is computed as $loss_{imitation}(\hat{y}_T, \hat{y}_S) = -\hat{y}_T \log(\hat{y}_S)$.

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 $Embedding_{DSSM}$ generated by the student model. Then the alignment loss for a query q is computed as

$$loss_{alignment}(S(q), D(q)) = 1 - \cos(S(q), D(q)).$$

In our experiments, we noted that if we introduce $loss_{ce}$ in the student model, it performs better. Furthermore the $loss_{imitation}$ 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 have 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, α is 0, and the model is effectively learning only from `true` labels. Gradually we scale up α, 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:

$$loss_{KD} = (1 - \alpha) \cdot loss_{ce} + \alpha \cdot loss_{alignment},$$

where α 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 use DSSM layer with a dense layer of size 128, same as that of the BERT arm. The student models are also trained for 10 epochs, and uses the same optimizers and schedulers as teacher. The weights of the product arm are frozen and only the query arm is trained.

Table 1: Area under ROC curve of various models. Cross entropy, imitation and alignment losses are represented as CE, IL and AL respectively. Stages of curriculum learning (CL) used are also indicated.

ID	Experiment	CL Stage	AUC (%)
T	Teacher	I	87.25
B	Student: CE (No KD)	I	83.15
S1	Student: IL	I	84.53
S2	Student: IL + AL	I	84.81
S3	Student: CE + AL	I	85.32
S4	Student: CE + IL	I	84.59
S5	Student: CE + IL + AL	I	84.80
H	CUPID	I & II	**86.17**

Metrics We use area under the ROC curve to compare the results of the baseline models with CUPID. This allows us the compare the performance of the model without forcing a choice of desired precision or recall, which may vary based on the requirements. In addition, AUROC also gives an indication of the probability of a negative being ranked higher than a positive, which is an important information when dealing with applications such as product search.

4.2.1 Results

We now compare the results of CUPID and the baselines on our dataset. Table 1 shows the area under the ROC curve for the models. The teacher model (**T**) achieves an AUC of 87.25%. For baseline (**B**), we show the performance of a model with DSSM for query arm and a BERT for product arm that is trained only using the training data, with no knowledge distillation. As expected, its performance falls short of the teacher. We then show the effects of various losses described in Section 3. **S1** uses only the imitation loss. **S2**, which adds alignment loss increases the performance by around 50 basis points. Replacing imitation loss with BCE in **S3** gives us approximately another 50 basis points boost. The **S4** configuration shows us that using BCE and imitation loss together doesn't provide significant boost over just imitation. Similar observation is made in **S5** which is presented for completeness. There we notice that addition BCE to **S2** complicates the model and performs no better.

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.

ID	Experiment	CL Stage	AUC (%)
T	Teacher	I	89.04
B	Student: CE (No KD)	I	72.41
S1	Student: IL	I	74.63
S2	Student: IL + AL	I	76.14
S3	Student: CE + AL	I	76.59
S4	Student: CE + IL	I	75.02
S5	Student: CE + IL + AL	I	74.80
H	CUPID	I & II	**79.17**

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.

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 $11.6ms$, which is almost $4\times$ that of the DSSM students, which is $3.2ms$. In real time, $11ms$ 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 improvement of CUPID over the baselines is more pronounced for this dataset.

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

ID	Experiment	AUC (%)
T	BiEncoder Teacher	87.25
CT	CE Teacher	89.76
H	CUPID: BiEncoder Teacher	86.17
CH	CUPID: CE Teacher	81.53

Table 4: Impact of Curriculum Learning

ID	Experiment	CL Stage	AUC (%)
B-	Student: No KD	None	82.09
B	Student: No KD.	I	83.15
H-	CUPID	II	85.65
H	CUPID	I & II	**86.17**

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 α? How important is the Alpha scheduler? The parameter α 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 α 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 α, 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 α 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 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

Junjie Bai, Fang Lu, Ke Zhang, et al. 2019. Onnx: Open neural network exchange. `https://github.com/onnx/onnx`.

Arindam Bhattacharya, Ankit Gandhi, Vijay Huddar, MS Ankith, Aayush Moroney, Atul Saroop, and Rahul Bhagat. 2023. Beyond hard negatives in product search: Semantic matching using one-class classification (smocc). In *WSDM 2023*.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. 2020. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *Advances in neural information processing systems*, 33:4271–4282.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Hassan Sajjad, Preslav Nakov, Deming Chen, and Marianne Winslett. 2021. Compressing large-scale transformer-based models: A case study on bert. *Transactions of the Association for Computational Linguistics*, 9:1061–1080.

Mitchell A Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing bert: Studying the effects of weight pruning on transfer learning. *arXiv preprint arXiv:2002.08307*.

Jiafeng Guo, Yinqiong Cai, Yixing Fan, Fei Sun, Ruqing Zhang, and Xueqi Cheng. 2022. Semantic models for the first-stage retrieval: A comprehensive review. *ACM Trans. Inf. Syst.*, 40(4).

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

Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving efficient neural ranking models with cross-architecture knowledge distillation.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33:9782–9793.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013a. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, CIKM '13, page 2333–2338, New York, NY, USA. Association for Computing Machinery.

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

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding.

Omar Khattab and Matei Zaharia. 2020a. Colbert: Efficient and effective passage search via contextualized late interaction over bert.

Omar Khattab and Matei Zaharia. 2020b. Colbert: Efficient and effective passage search via contextualized late interaction over bert.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. *Advances in neural information processing systems*, 28.

Woosuk Kwon, Sehoon Kim, Michael W Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. 2022. A fast post-training pruning framework for transformers. *arXiv preprint arXiv:2204.09656*.

Chaozhuo Li, Bochen Pang, Yuming Liu, Hao Sun, Zheng Liu, Xing Xie, Tianqi Yang, Yanling Cui, Liangjie Zhang, and Qi Zhang. 2021. Adsgnn: Behavior-graph augmented relevance modeling in sponsored search.

Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2022. A survey of transformers. *AI Open*.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, and Qi Ju. 2020. Fastbert: a self-distilling bert with adaptive inference time. *arXiv preprint arXiv:2004.02178*.

Wenhao Lu, Jian Jiao, and Ruofei Zhang. 2020. Twinbert: Distilling knowledge to twin-structured bert models for efficient retrieval.

Yuxiang Lu, Yiding Liu, Jiaxiang Liu, Yunsheng Shi, Zhengjie Huang, Shikun Feng Yu Sun, Hao Tian, Hua Wu, Shuaiqiang Wang, Dawei Yin, and Haifeng Wang. 2022. Ernie-search: Bridging cross-encoder with dual-encoder via self on-the-fly distillation for dense passage retrieval.

Sourab Mangrulkar, Ankith M S, and Vivek Sembium. 2022a. Be3r: Bert-based early-exit using expert routing. In *KDD 2022*.

Sourab Mangrulkar, Ankith M S, and Vivek Sembium. 2022b. Multilingual semantic sourcing using product images for cross-lingual alignment. In *The Web Conference 2022*.

Yihuan Mao, Yujing Wang, Chufan Wu, Chen Zhang, Yang Wang, Yaming Yang, Quanlu Zhang, Yunhai Tong, and Jing Bai. 2020. Ladabert: Lightweight adaptation of bert through hybrid model compression. *arXiv preprint arXiv:2004.04124*.

Sachin Mehta, Marjan Ghazvininejad, Srinivasan Iyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2020. Delight: Deep and light-weight transformer. *arXiv preprint arXiv:2008.00623*.

Aditya Menon, Sadeep Jayasumana, Ankit Singh Rawat, Seungyeon Kim, Sashank Reddi, and Sanjiv Kumar. 2022. In defense of dual-encoders for neural ranking. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15376–15400. PMLR.

Michinari Momma, Chaosheng Dong, and Yetian Chen. 2022. Multi-objective ranking with directions of preferences. In *SIGIR 2022 Workshop on eCommerce*.

Priyanka Nigam, Yiwei Song, Vijai Mohan, Vihan Lakshman, Weitian, Ding, Ankit Shingavi, Choon Hui Teo, Hao Gu, and Bing Yin. 2019. Semantic product search.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Chandan K. Reddy, Lluís Màrquez, Fran Valero, Nikhil Rao, Hugo Zaragoza, Sambaran Bandyopadhyay, Arnab Biswas, Anlu Xing, and Karthik Subbian. 2022. Shopping queries dataset: A large-scale ESCI benchmark for improving product search. *arXiv*.

Nils Reimers and Iryna Gurevych. 2019a. Sentence-bert: Sentence embeddings using siamese bert-networks.

Nils Reimers and Iryna Gurevych. 2019b. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Nils Reimers and Iryna Gurevych. 2019c. Sentence-bert: Sentence embeddings using siamese bert-networks.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021. Colbertv2: Effective and efficient retrieval via lightweight late interaction.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019a. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019b. Patient knowledge distillation for bert model compression.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity.

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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. 2020. Lite transformer with long-short range attention. *arXiv preprint arXiv:2004.11886*.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. Deebert: Dynamic early exiting for accelerating bert inference.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit.