DISKCO : Disentangling Knowledge from Cross-Encoder to Bi-Encoder

Ankith MS*
Amazon
India
ankiths@amazon.com

Arindam Bhattacharya*
Amazon
India
aribhat@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

In the field of Natural Language Processing (NLP), sentence pair classification is important in various real-world applications. Bi-encoders are commonly used to address these problems due to their low-latency requirements, and their ability to act as effective retrievers. However, bi-encoders often under-perform compared to cross-encoders by a significant margin. To address this gap, many Knowledge Distillation (KD) techniques have been proposed. Most existing KD methods focus solely on utilizing the prediction scores of cross-encoder models and overlook the fact that cross-encoders and bi-encoders have fundamentally different input structures. In this work, we introduce a novel knowledge distillation approach called DISKCO, which **DIS**entangles the **K**nowledge learned in **Cro**ss-encoder models especially from multi-head cross-attention models and transfers it to bi-encoder models. DISKCO leverages the information encoded in the cross-attention weights of the trained cross-encoder model, and provide it as contextual cues for the student bi-encoder model during training and inference. DISKCO combines the benefits of independent encoding for low-latency applications with the knowledge acquired from cross-encoders, resulting in improved performance. Empirically, we demonstrate the effectiveness of DISKCO on proprietary and on various publicly available datasets. Our experiments show that DISKCO outperforms traditional knowledge distillation methods by upto 2%.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**; **Neural networks**; • **Information systems** → *Clustering and classification*.

ACM Reference Format:
Ankith MS, Arindam Bhattacharya, Ankit Gandhi, Vijay Huddar, Atul Saroop, and Rahul Bhagat. 2024. DISKCO : Disentangling Knowledge from Cross-Encoder to Bi-Encoder. In *Companion Proceedings of the ACM Web Conference 2024 (WWW '24 Companion), May 13–17, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3589335.3648333

*Both authors contributed equally to this work.

WWW '24 Companion, May 13–17, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-0172-6/24/05.
https://doi.org/10.1145/3589335.3648333

1 INTRODUCTION

In recent years, Natural Language Processing (NLP) has witnessed significant advancements, revolutionizing various domains such as information retrieval, question answering, sentiment analysis, and more. Among the multitude of NLP tasks, sentence pair classification plays a crucial role in numerous real-world applications, including query-product relevance classification in e-commerce [2, 25], text-based or (Query, Document) information retrieval [22], and identifying duplicate questions in online platforms like Quora/Stack overflow. Traditionally, there have been two main approaches to sentence pair or query-document classification: cross-encoder (CE) models and bi-encoder (BE) models. Cross-encoder models consider both sentences jointly and learn a joint representation for the pair. In contrast, bi-encoder models independently embed each sentence and compute a similarity score between the embeddings of the two sentences using a simple distance metric. While cross-encoder models have demonstrated remarkable performance on various benchmarks, they suffer from high inference costs, making them impractical for real-time, large-scale applications. They are also unsuitable for nearest neighbor styled retrieval task [22]. On the other hand, bi-encoder models offer lower latency but often underperform compared to cross-encoder models, primarily due to the lack of joint context between the sentences. The performance gap between bi-encoders and cross-encoders has motivated researchers to explore knowledge distillation (KD) methods [12], aiming to transfer the knowledge learned by cross-encoders to bi-encoders. However, existing KD methods predominantly focus on utilizing prediction scores from the cross-encoder model, neglecting the fact that cross-encoders and bi-encoders have different input structures. Further, authors in [3] show that KD using imitation loss (imitating the prediction of CE model in BE model) may not yield optimal knowledge transfer due to the structural and learning paradigm differences between CE and BE models. More recently, ERNIE-Search [20] proposed a cascaded distillation of cross-encoder → late-interaction (e.g., ColBERT) → bi-encoder models, and revisits the relationships between them for gradually transferring the knowledge to bi-encoders.

The contributions of this work are as follows:

(1) We propose a novel knowledge distillation approach called DISKCO. Our approach transfers knowledge from cross-Encoder models to bi-encoder models, leveraging the strengths of both architectures. We first train the cross-encoder model, and then build the hash map of tokens to important tokens using cross-connections leveraging the attention weights. This fixed hash map is finally provided as additional context in bi-encoders during their training and inference. By disentangling and transferring this cross-attention weights, we enhance the performance of Bi-Encoder models while maintaining their advantages of independent encoding for low-latency applications.
(2) To the best of our knowledge, this work is the first of its kind to provide a human-interpretable explanation of the knowledge transfer mechanism from cross-encoder to bi-encoder models as the hash map built is completely decipherable.
(3) We demonstrate the effectiveness of our approach on proprietary and publicly available MSMARCO and GLUE datasets, showcasing improved performance compared to traditional knowledge distillation methods.

The rest of the paper is organized as follows. In Section 2, we provide background and related work on sentence pair classification and knowledge distillation. Section 3 presents the details of our DISKCO approach, including the disentanglement of knowledge and the transfer process. Experimental setup, results, and ablation studies are presented in Section 4. Finally, we conclude the paper in Section 5, summarizing our contributions and discussing potential future directions.

2 RELATED WORK

2.1 Cross Encoders, Bi-Encoders and Late Interaction

Cross encoders [26] are a type of transformer-based models that aim to capture the relationship between input pairs. These models take two inputs, typically a pair of sentences or sequences, and encode them together into a shared representation. By jointly considering both inputs, cross encoders can effectively model interactions and dependencies between the input elements, leading to improved performance on various downstream tasks.

Bi-encoders [16], on the other hand, use separate (possibly shared, as in case of Siamese networks) 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 due to the decoupled inputs to the network.

Another method used to bridge the gap between CE and BE is late interaction, user by ColBERT [15]. ColBERT is a contextualized late interaction model built on the BERT architecture for passage ranking in large-scale information retrieval systems. It has an architecture similar to Bi-Encoder, but instead of using a simple distance metric like cosine similarity, it utilizes late interaction layers to combine the embeddings returned by each Bi-encoder arm, and compute the final similarity scores.

2.2 Knowledge Distillation

Knowledge Distillation (KD)[12] is a widely researched topic that enables the transfer of knowledge from a complex, pre-trained model to a smaller, more computationally efficient model. The teacher-student framework is commonly used in knowledge distillation. In this approach, the larger model (teacher) provides soft targets, i.e., the probabilities of correct labels, to guide the training of the smaller model (student). Subsequently, the student model tries to imitate the output distribution of the teacher model. This framework has been successfully applied to various NLP tasks. With the rise of BERT [8] and the corresponding growth in textual data, there has been growing interest in applying KD to BERT models in the field of NLP. The distillation of large cross encoder models for sentence pairs presents two options: i. distilling it to a smaller cross-encoder, and ii. distilling it to a Bi-encoder.

2.2.1 Cross Encoder Knowledge Distillation. Distill BERT [30] is a seminal work along with (Tinybert[14], Fast BERT[18], Task specific BERT [33], Patient Knowledge Distillation BERT [32]. These propose using KD to transfer knowledge from a large BERT model to a smaller model. They showed that the distillation process enables the smaller model to capture contextual relationships effectively. While these approaches make BERT models faster by up to 60%, they cannot be used in real-time because e-commerce applications such as semantic matching [13, 23] have limited computational resources (especially GPUs) and strict latency requirements.

2.2.2 Bi-encoder Knowledge Distillation. TwinBERT [19] and PROD [17] proposes the distillation of cross encoders to twin tower BERT structure, thus, permitting pre-compute of document embeddings and cache in memory saving additional computational time. TwinBERT computes the embeddings of each input separately, and then applies a crossing layer to produce the final score. The crossing layers were either a residual network, or a simple cosine similarity score. PROD proposes a framework for progressive teacher and data distillation where the model and data complexity is gradually decreased. Another approach, utilized by ERNIE-search [20], is to distill CE to BE is to do a progressive distillation from CE to a late-interaction model, and from late-interaction model to a BE.

Our approach is distinct and complementary to existing methods. While other approaches focus on imitating scores from Cross-Encoder (CE) models or performing interaction distillation in methods like Colbert or ERNIE Search, we take a different route. Instead of imitating scores, we extract cross-interaction information from the teacher model in the form of its attention weights. These attention weights are then transferred to the Bi-Encoder (BE) model in a human-interpretable format. By leveraging attention weights, we provide meaningful contextual information to the BE model, enhancing its understanding of token interactions and improving its performance in a more transparent and interpretable manner.

2.2.3 Query Expansion. Query expansion (QE) is a technique that enhances the precision and recall of search results by augmenting an initial user query with additional terms or concepts derived from the original query or external sources [5]. It has been widely explored in various information retrieval and natural language processing applications, such as document retrieval, web search, and information retrieval systems [1]. The fundamental idea behind query expansion is to bridge the semantic gap between a user's query and the underlying documents in a collection by augmenting the query with terms that enhances the effectiveness of the retriever. The method for expanding a query ranges from manual QE [9] to various levels of automated QE (AQE). The methods for AQE vary widely, from information retrieval based methods [4], probabilistic approaches [7], using word embeddings [29] and language model based methods [35]. While at first glance, DISKCO appears to be another approach for AQE, it is more focused on distilling the knowledge that a cross-encoder model learns due to cross-attentions between the document and query terms rather than some form of correlation, such as polysemy and synonymy, that the other methods rely on. Our approach is also different from other AQE using self-attention [11] in that we do not train any model to perform query expansion. We simply rely on cross-encoder models to provide the necessary information about the cross-interaction between the query and document tokens.

3 OUR APPROACH

In this section, we present the proposed knowledge distillation approach. Figure 1 and 2 shows the overall architectural diagram for the proposed method. DISKCO capitalizes on the wealth of information stored in the attention modules of the CE model, allowing us to extract valuable contextual insights from the multi-head-attention mechanism. By leveraging this rich information, we enhance the performance of BE models, infusing them with important contextual information from the CE model. DISKCO has three stages - (i) training of cross-encoder models, (ii) building hash map of tokens to important tokens from learned cross-attention weights, (iii) learning bi-encoder models by providing hash map as additional contextual cue in the input. We discuss each of these steps in detail below.

Let us define the labeled dataset as

$$D = (\mathbf{q}_1, \mathbf{p}_1, y_1), (\mathbf{q}_2, \mathbf{p}_2, y_2), ..., (\mathbf{q}_n, \mathbf{p}_n, y_n)$$

where $\mathbf{q}_i, \mathbf{p}_i$ and y_i represent the query, document, and label, respectively. The queries $\mathbf{q}_i = (q_i^1, q_i^2, ...q_i^{t_q})$ and documents $\mathbf{p}_i = (p_i^1, p_i^2, ...p_i^{t_p})$ are composed of t_q and t_p tokens, respectively.

3.1 Teacher Model: Cross Encoder

In our approach, we utilize a cross-encoder (CE) model to build the teacher model (refer to Figure 1). The CE model employs a transformer [34] encoder with cross-attention, enabling it to capture rich interactions between query and document tokens. Transformer encoder comprises of stacked identical blocks and the key component being multi-head attention. It plays a crucial role in generating powerful contextual embeddings, leading to it's success in various NLP tasks. To facilitate this, we concatenate the query $\mathbf{q}_i$ and document $\mathbf{p}_i$ using a special token, resulting in an input sequence $J = (\texttt{[CLS]}\ \ q_i^1 q_i^2 ... q_i^{t_q}\ \texttt{[SEP]}\ \ p_i^1, p_i^2, ...p_i^{t_p}\ \texttt{[SEP]})$. In the multi-head attention block, the input embeddings are transformed by passing them through a linear layer to generate query, key, and value matrices. These matrices capture different aspects of the input and allow the model to attend to relevant information during the encoding process. The attention score is computed by taking the dot product between the query and key vectors, followed by a softmax activation to obtain attention weights. Finally, the contextual representation for each token is derived by multiplying the attention weights with the corresponding value matrix. The use of multi-head allows the transformer to capture diverse and nuanced relationships between tokens. Each attention head focuses on different patterns or aspects within the input, enabling the model to attend to multiple sources of information simultaneously.

We use the embedding of the special token `CLS` from the last layer of the transformer as a pooled input representation for the classification task. This pooled representation is then passed through a classification layer, to obtain a cross-encoder prediction score.

3.2 Hash map building using cross-attention scores

After training cross encoder, we utilize the cross-attention scores at each layer of the model to build DISKCO hash $\mathcal{H}$. A DISKCO hash is a dictionary containing mapping from query tokens to product tokens that it cross-attends to across all the training examples with an aggregated attention score. This map $\mathcal{H}$ is augmented as additional information to bi-encoder model for effective knowledge transfer. Let a_{ij}^l denotes the attention score of i^{th} token for j^{th} token in the input sequence at l^{th} layer (averaged over all the attention heads of layer) of the cross-encoder model. Let $\mathcal{Q}$ denotes the set of distinct tokens on query side and $q_{t_1}, q_{t_2}, \ldots q_{t_{|\mathcal{Q}|}}$ be the corresponding tokens, and $\mathcal{P}$ denotes the set of distinct tokens on product side (we use the term product and document interchangeably in the paper). Then, the DISKCO hash is created as follows.

Get the cross-attention scores from attention map a_{ij}^l (see Figure 1) across all the training samples. We leverage all a_{ij}^l's such that i^{th} token is from the query side and j^{th} token is from the product side, i.e., $i \in \mathcal{Q}$ and $j \in \mathcal{P}$,

$$\begin{aligned} &\text{Initialize } \mathcal{H}(q_{t_i}, p_i) = 0, \forall q_{t_i} \in \mathcal{Q}, \forall p_i \in \mathcal{P} \\ &\mathcal{H}(q_{t_i}, p_i) += \sum_l a^l_{q_{t_i} p_i}, \forall (\mathbf{q}, \mathbf{p}) \in D, \forall q_{t_i} \in \mathbf{q}, \forall p_i \in \mathbf{p} \end{aligned} \tag{1}$$

This hash map is shown in Figure 1 where query token q_{t_i} is attending to tokens $p_i^1, p_i^2, \ldots, p_i^{|t_i|}$ from the product side across all training samples, and their aggregated attention weights are represented as $\mathcal{H}(q_{t_i}, p_i^j) = w_i^j$. For instance, for the token "shoe", the hash map list strongly affiliated tokens such as "backpack", "sneakers", "flip-flops", "running", "accessories". We describe the complete algorithm for populating hash map in Algorithm 1.

3.3 Student Model: Bi-encoder

Bi-encoder models employ separate query and document arms using the transformer encoder (as shown in Figure 2). CE model has high inference cost, making it impractical for real-time deployment in latency-sensitive scenarios. Bi-encoder's design, coupled with

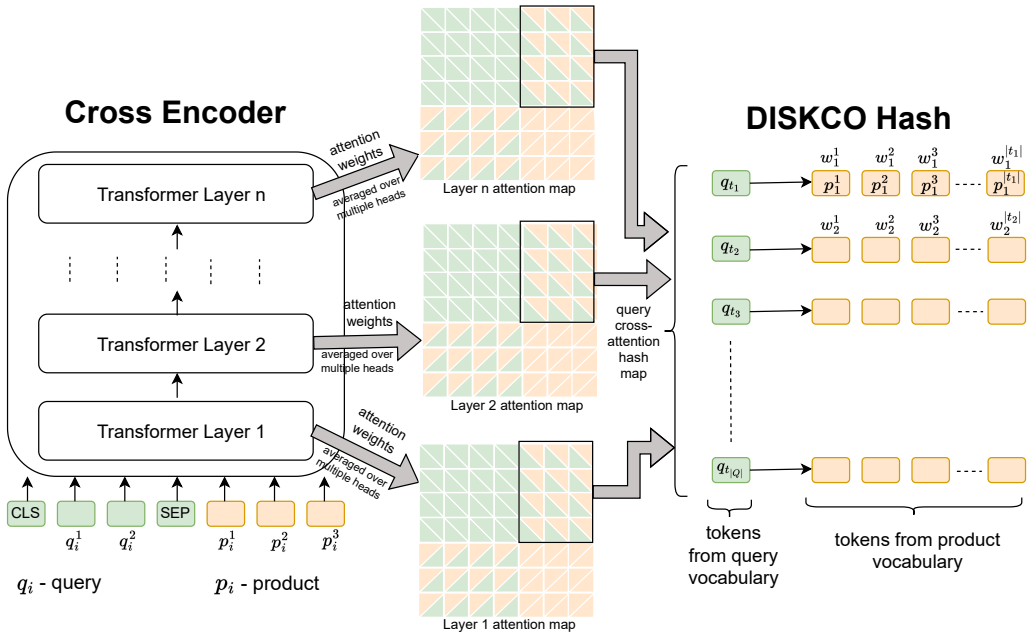

Figure 1: Architecture of DISKCO (part 1). Given the input query $\mathbf{q}_i$ and product $\mathbf{p}_i$ pairs, we train the cross-encoder model. For each training sample, we take the cross-attention weights (at every layer) from query side to product side, and build a DISKCO hash map $\mathcal{H}$. This hash map contains mapping from query tokens to product tokens they are cross-attending to across all training samples.

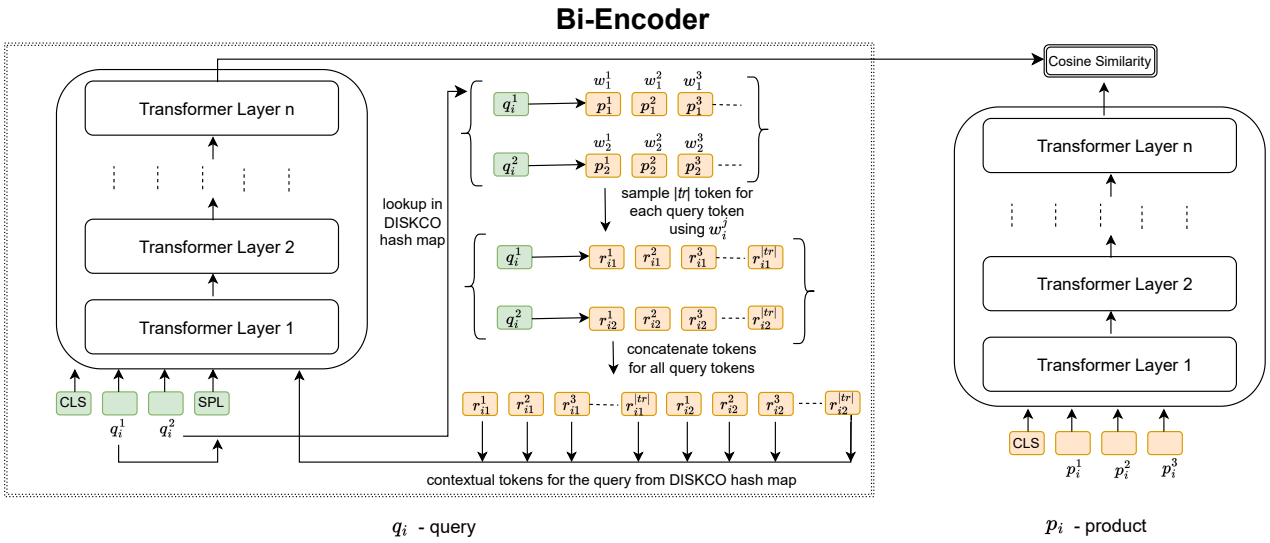

Figure 2: Architecture of DISKCO (part 2). We perform the lookup of query tokens in DISKCO hash map $\mathcal{H}$ and get all the contextual tokens. These contextual tokens are concatenated with the query tokens with a special token <SPL> in the input and the bi-encoder model is trained with this modified input.

dot/inner product operations for final classification score calculations, enables the BE model to perform in a computationally efficient manner. Moreover, the pre-computed and cached document representations in BE models significantly reduce inference time, making them suitable for real-time applications (refer to Table 1). However, bi-encoder performance falls short compared to the CE model due to the limitations of independent encoding. In this paper, we address this limitation by providing additional contextual information in the

Algorithm 1: Extracting Contextual Tokens from CE

Input: Dataset $D = \{(\mathbf{q}_i, \mathbf{p}_i, y_i)\}$, Trained Cross-Encoder model CE

Output: Contextual tokens look up table $\mathcal{H}$

Initialize an empty Contextual tokens look up table, and DISKCO hash map to 0

$\mathcal{H}(q_{t_i}, p_i) = 0, \forall q_{t_i} \in \mathcal{Q}, \forall p_i \in \mathcal{P}$

foreach $(\boldsymbol{q}_i, \boldsymbol{p}_i, y_i) \in D$ **do**

  foreach $l \in layers$ **do**

    /* Get the attention weights at layer l averaged over multiple heads by performing a forward pass through CE */

    $A_i^l = get\_attentions(CE, \mathbf{q}_i, \mathbf{p}_i, l)$

    /* take cross attention weights such that i is from the query side and j is from the document side */

    foreach $q_{t_i} \in tokens(\boldsymbol{q}_i)$ **do**

      foreach $p_j \in tokens(\boldsymbol{p}_i)$ **do**

        /* Update the lookup table */

        $\mathcal{H}(q_{t_i}, p_j) += a_{ij}^l$

form of DISKCO hash map $\mathcal{H}$. DISKCO unlocks a deeper understanding of how knowledge from the CE model can be effectively transferred to the BE model, providing meaningful context to the knowledge distillation process.

To substantiate our hypothesis, let's consider the relevance classification task in an e-commerce setting. Given a customer query like "puma shoes for men," the critical tokens could be "sneakers women backpack." The reasoning is that the model should comprehend that shoes and sneakers are related, shoes and backpacks are distinct product types, and men's and women's shoes differ. We hypothesize that such interactions are captured by the CE model's multi-head attention modules, which could be highly beneficial for the BE model. Besides providing a mechanism to transfer knowledge from the CE to the BE model, this method also offers a human-interpretable explanation of the knowledge transfer process.

We utilize a transformer model for the student model, which is independent of the configuration of the teacher model. Following the approach in [21], we have shared encoder for query and document to facilitate knowledge transfer between them. For every token q_i^j in the query $\mathbf{q}_i$, we perform lookup in hash map $\mathcal{H}$ and sample the tr associated tokens $r_{ij}^1 ... r_{ij}^{tr}$ based on their aggregated attention score w_i^j in the hash map. These pertinent tokens serve as contextual information for the query, assisting the student model in pinpointing significant interactions in the CE model and guiding its encoding process.

To integrate this contextual information in BE, we formulate the contextualized query as

$$q_{BE} = (\texttt{[CLS]}\ \ q_i^1, q_i^2, ..., q_i^{t_q}\ \texttt{[SPL]} r_{i1}^1, r_{i1}^2 ..... r_{i1}^{|tr|}, \\ r_{i2}^1, r_{i2}^2 ..... r_{i2}^{|tr|}, ..., r_{it_q}^1, r_{it_q}^2 ..... r_{it_q}^{|tr|})$$

and for product, we refrain from adding any contextual information (this is discussed in our ablation studies 4.4) $p_{BE} = (\texttt{[CLS]}\ p_i^1, p_i^2 ... p_i^{t_p})$. The [CLS] token symbolizes the special classification token, while [SPL] marks the separation between the query and the relevant tokens. This construction guarantees that the student model acquires the necessary contextual prompts during the encoding process. Ultimately, we obtain the BE score using a dot product between query embedding and document embedding.

$$q_{embedding} = BE^{[CLS]}(q_{BE}) \tag{2}$$

$$p_{embedding} = BE^{[CLS]}(p_{BE}) \tag{3}$$

$$BE_{score} = cosine\_score(q_{embedding}, p_{embedding}) \tag{4}$$

Algorithm 2 describes the process for knowledge transfer from the trained teacher model, the Cross-Encoder, to the student model, the Bi-Encoder. DISKCO strategically infuses relevant contextual details during the encoding process, which provides the student model with a profound understanding of frequent token interactions across document, even without direct exposure to the document. In the next section, we show this augmentation significantly improves the model's performance across various datasets.

4 EXPERIMENTS

In this section, we present the experimental setup and methodology employed to investigate the effectiveness of DISKCO in enhancing the performance of a bi-encoder transformer model, and bridge the performance gap with respect to a cross encoder model. We describe the datasets used for training and evaluation, the baseline models which our model is compared against, as well as the architecture and hyperparameters of both the teacher and student models. Furthermore, we outline the training procedure and metrics utilized to assess the performance gains achieved through knowledge distillation.

We present the results of our method on MS-MARCO dataset. The dataset consists of real-world search engine queries and their corresponding documents, which are used as the context for answering the queries. The queries are sampled from anonymized Bing search logs, ensuring that they represent a wide range of topics and user intents. There are 880 thousand queries and 8.8 million passages. Each query has up to 10 passages and corresponding labels denoting if the passage was selected for the search query. We report the MRR@10, a standard metric for MS-MARCO to compare it with existing baselines.

To demonstrate the wide applicability of DISKCO, we also report the results on GLUE benchmark datasets, a collection of diverse natural language processing (NLP) tasks designed to assess the performance and generalization capabilities of various NLP models. GLUE consists of multiple datasets, each representing a different NLP task. We report the results on QNLI, QQP and STS. We also report DISKCO's performance on AI Crowd ESCI data [25].

We also present the results of DISKCO on multilingual dataset that have been sub-sampled from the Amazon e-commerce history

Algorithm 2: Training BE using DISKCO

Input: Dataset $D = \{(\mathbf{q}_i, \mathbf{p}_i, y_i)\}$, Contextual tokens look up table $\mathcal{H}$, Maximum number of contextual tokens tr for each query token
Output: Trained Bi-encoder Model BE θ_{BE}

Initialize Bi-encoder model using pre-trained model weights
foreach $(\boldsymbol{q}_i, \boldsymbol{p}_i, y_i) \in D$ **do**
  /* sample contextual tokens for each token in $\mathbf{q}_i$ from $\mathcal{H}$ based on aggregated attention scores */
  r_{ij}^k = sample_contextual_tokens($\mathbf{q}_i$, $\mathcal{H}$, tr)
  /* append contextual tokens to query */
  $\mathbf{q}_i^{context}$ = [CLS] $q_i^1, q_i^2, ..., q_i^{t_q}$ [SPL] $r_{i1}^1, r_{i1}^2 .....r_{i1}^{|tr|}$, $r_{i2}^1, r_{i2}^2 .....r_{i2}^{|tr|}$, ..., $r_{it_q}^1, r_{it_q}^2 ..... r_{it_q}^{|tr|}$
  /* Train BE using contextual inputs */
  Update $\theta_{BE} \leftarrow BCE(BE_{score}(\mathbf{q}_i^{context}, p_i), y_i)$

Table 1: Results of DISKCO and baselines on MSMARCO Dev set. The model types are (CE: cross encoder, BE: bi-encoder, LI: late interaction). Cache space is the storage required for caching the passages (p: #passages, d: embedding dimension, s: sequence length, t: #tokens). ⋆The inference times are approximate average time based on the architecture (see discussion in Section 4.1) for detailed explanation.

Model	Model Type	MRR@10	Inference Time (ms)	Cache Space
Cross Encoder	CE	44.00	2470⋆	O(1)
RocketQA [24]	BE	37.00	82	$O(p \cdot d)$
COIL [10]		35.50		
PAIR [27]		37.90		
RocketQAv2 [28]		38.80		
PROD [17]		39.30		
ColBERT [15]	LI	36.00	104	$O(p \cdot d \cdot s)$
ColBERTv2 [31]		39.70		
DISKCO	BE	39.50	89	$O(p \cdot d + t)$

log and human-audited for binary relevance labels, similar to the work of [25]. At this time, the data is not publicly available and is proprietary. We used a few hundred thousand records to train and another few hundred records to test the models. We also perform extensive ablation studies to show the effect of various parameters of DISKCO on the performance.

4.1 Results and Discussion

Table 1 shows the MRR@10 values for DISKCO and other baselines. All the models uses 12 layer pretrained transformers. DISKCO uses 12-layer XLMRoberta as the base model. We used a learning rate of 10^{-3}. Unless specified, as in ablation study, we use a maximum token length of 64 for the queries and 256 for the passages. For DISKCO, we sample 50 tokens per query token. For the baselines, we use the values reported in the respective papers. We present the results of a 12-layer XLMRoberta based cross encoder model with maximum token length of 320 to emphasize the gap between the Bi-encoder and cross-encoder models.

Table 2: Area under the ROC curve of DISKCO on multilingual dataset.

Model	Type	AUC
Cross-encoder	Teacher	92.10
Bi-encoder (2 layer)	Student-1	84.10
DISKCO (2 layer)		86.40
Bi-encoder (12 layer)	Student-2	88.60
DISKCO (12 layer)		89.90

From the table we see that DISKCO beats the purely bi-encoder models, such as RocketQA(v1/v2), COIL, PAIR and PROD. Additionally, we note that DISKCO fails to meet the baselines that use late interactions. To justify this, we consider the type of target applications that requires bi-encoder. They are usually large scale models that require fast inference. To perform this, we need to cache the document embeddings, and generate the query embeddings on the fly. Late interaction models require caching of all the tokens of the document in order to evaluate the late interaction layer. This increases the storage cost, and we found such solution to be infeasible, thus giving DISKCO an edge.

Table 1 also shows the approximate query time. Note that the time per query-passage pair for CE is $247ms$ but CE model needs to run once for each passage for a given query (10 for MSMARCO dateset). This is the main reason behind using bi-encoder models for real time retrieval. DISKCO has a slightly higher inference time than vanilla bi-encoder architecture, but is faster than late interaction models.

Finally, the table shows the caching space complexity of the models. CE models cannot cache anything and needs to run for each pair. The vanilla bi-encoder models caches the d-dimensional embeddings of the p passages in the dataset. DISKCO requires similar caching space, plus an additional $O(t)$ storage for the hash map. On the other hand, the late interaction models require storing the individual token embeddings for each passage of length s. This makes late-interaction models unsuitable for real time applications with large p and s.

4.2 Performance on GLUE tasks

We explored the applicability of DISKCO beyond ranking and relevancy tasks by investigating its potential in sentence similarity tasks, as well as relevancy classification of products. The results of our experiments are presented in Table 3, where we demonstrate the impact of DISKCO on similarity tasks across various GLUE benchmark datasets.

Consistent with the outcomes observed in our previous experiments, we observed a significant performance boost when utilizing DISKCO for sentence similarity tasks.

The improvements in performance were evident across all tested datasets, indicating the robustness and generalizability of DISKCO in wide range of NLP tasks.

Table 3: Performance of DISKCO on GLUE tasks. All numbers are accuracy, except for STS, which reports Spearman correlation, as consistent with GLUE benchmark.

Model	QNLI	AI Crowd	QQP	STS
BE	86.56	86.18	81.64	77.32
BE with KD	87.44	86.96	82.11	78.41
DISKCO without KD	88.84	87.43	83.43	80.16
DISKCO with KD	88.86	87.39	83.43	80.18

4.3 Real World Application: Irrelevant Result Detection for Multilingual Dataset

In Table 2, we present the results of our method on the real world dataset with data from eleven countries in six languages (including Arabic, Polish, Spanish, French, Portuguese and Turkish). We trained the teacher model with pre-trained `xlm-roberta` model. We also trained two student: Student 1 is a two layer transformer for the query(because of low latency requirement), and a twelve layer transformer for the product (no latency requirement on the product side, since the product embedding will be cached), Student 2 is a twelve layer transformer for the query, and a twelve layer transformer for the product. We notice a significant improvement with DISKCO over the Bi-encoder. This improvement is even more pronounced for smaller Bi-encoder(2 layer student) models. This shows that for smaller models, the context hash map compensate the drop in model's capability.

4.4 Ablation Studies

In this section, we study the impact of various factors such as the query length, number of tokens, and knowledge distillation loss on the performance of DISKCO.

4.4.1 ***What is the impact of varying the number of contextual tokens on model performance?*** In practical terms, when tokenizing the query, we impose a maximum sequence length, primarily to meet stringent latency requirements. If the extracted contextual tokens surpass this prescribed maximum, we are constrained to retain only those originating from the initial query tokens for model input. For instance, if the query is 'puma men shoes,' the contextual tokens may be limited to those derived from 'puma' and 'men' alone, omitting contextual tokens from 'shoes', all due to the imposition of the maximum sequence length to fulfill latency requirements. To overcome this constraint, we employ a random sampling approach, selecting contextual tokens from the entire set of query tokens. This ensures a more comprehensive representation of contextual information

Table 4: Examples of some contexts extracted from MSMARCO dataset.

Token	Context
iphone	aluminium display elite believe micro printer training
chocolate	popcorn western bar calorie artisan vacation lime
table	buffet poet card friends champagne camp restaurant
relief	billion golf emergency metabolism tropical mini cable
house	planning traditional homes party rules argent benefit

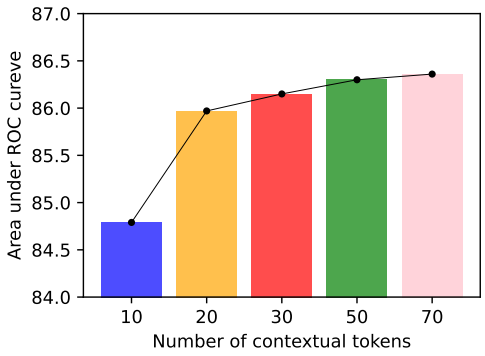

Figure 3: Performance of DISKCO with the number of contextual tokens on multilingual dataset (DISKCO 2 layer).

Our experiments, as illustrated in Figure 3, have demonstrated that including a limited number of DISKCO tokens provides significant value. This is evident from the substantial improvement observed between 10 and 20 tokens. However, it's important to note that adding tokens beyond 50 leads to diminishing returns, accompanied by a notable increase in computation cost.

4.4.2 ***How does the DISKCO model performance improve across various query lengths?*** Figure 4 shows the effect of the length of the query on the performance on DISKCO. We note that as we increase the length of the query, the gap between DISKCO and Bi-encoder reduces. For queries longer that length of 10, DISKCO under-performs Bi-encoders. This shows that the context added by DISKCO has greater effect when the query itself lacks the context. As we add more context to the query, the benefit of DISKCO reduces.

4.4.3 ***Why augment query, and not document?*** Typically, the tasks that benefit the most from distillation, such as retrieval and ranking, deal with asymmetric inputs. That is, the number of tokens in a *query* is far fewer than the number of tokens in the *document* or *title*. Adding tokens to the document does not provide significant additional information, because the mean pooling to generate the embedding of the product dilutes the effect of additional tokens.

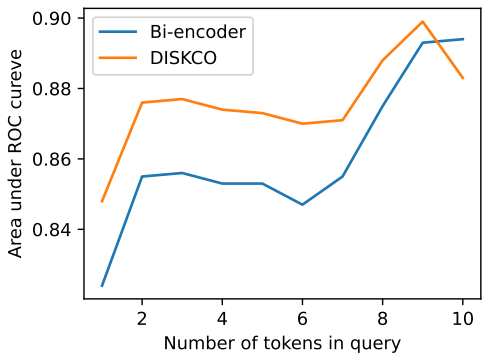

Figure 4: Performance of DISKCO (2 Layer) and Bi-encoder (2 Layer) with number of tokens in query on multilingual dataset.

Similar trend is noticeable i.e as the query length increase we observe diminishing returns for adding contextual tokens as seen in Figure 4.

4.4.4 ***How does the choice of cross-attention layer in DISKCO affect performance?*** The key idea behind DISKCO is to leverage the cross attention weights to identify the most import tokens in context of the query. For this, we looked at the cross attention weights of all layers. We found that the first layer cross attention weights provide the largest lift. For MS-MARCO dataset, using the first layer cross-attention weights, we get an MRR of 39.50. Using the sixth layer, it drops to 38.76. Using the twelfth layer cross-attention weights, the MRR degrades to 36.65. Averaging the attention weights across all the layers causes the MRR to drop to 37.47 compared to the first layer. Our hypothesis to explain this behavior is knowledge diffusion. Through knowledge diffusion, the cross encoder is able to leverage the information learned at earlier layers to enhance its representations at higher layers. While this helps the higher layers capture more abstract and context-aware features, as well as grammatical constructs [6], it *diffuses* the token specific attention weights, and cannot be leveraged by DISKCO.

4.4.5 ***Qualitative analysis of contextual tokens.*** Table 4 provides some randomly selected keywords and their context. We note that these contexts add various related information to the query, such as *display* for *iphone*, and *emergency* for *relief*. It also adds some less informative context, and empirically from our experiment we observe that the model learn to attend to the ones that add value during DISKCO training and ignore the noisy tokens.

4.4.6 ***Effect of filtering tokens based on labels.*** We extract the *important* tokens based on the cross-attention weights. This include adding tokens what were highly attended on to identify both relevant and irrelevant examples. From the distribution of scores of positive (relevant) pairs and negative (irrelevant) pairs, shown in Figure 5, we noticed that the model is more confused, that is, the scores are more widely spread for negative examples compared to positive examples. This led us to believe that adding the tokens extracted only from negative examples will give us the biggest boost. Note that the query is expected to have most of the terms that help in identifying relevant products, adding these additional contextual

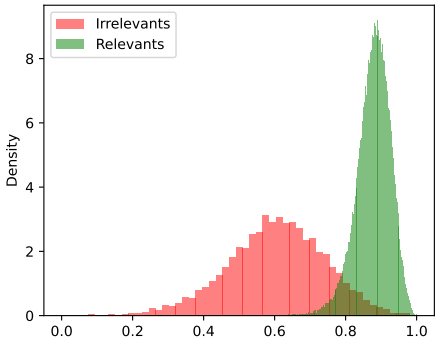

Figure 5: Distribution of the scores of relevant and irrelevant pairs on MSMARCO.

Table 5: Comparison of baseline with DISKCO when adding tokens from all pairs and only irrelevant pairs on MSMARCO Dev set.

Model	MRR@10
BE	37.18
DISKCO with negative tokens	38.39
DISKCO with all tokens	**39.50**

tokens would help weed out irrelevant products. However, as Table 5 shows, we found that adding tokens extracted from both positive and negative pairs increases the model performance more than when adding only the tokens from negative pairs.

4.4.7 ***Does the addition of standard knowledge distillation loss impact the performance of DISKCO?.*** Table 6 illustrates the impact of the standard knowledge distillation (KD) loss. When applying the conventional KD loss, the performance of the BE model sees a rise of 2.05%, whereas the proposed DISKCO method enhances performance by 4.02%. Interestingly, once DISKCO tokens are incorporated, the impact of the conventional KD loss on performance appears to be negligible. Our conjecture is that the KD loss's impact is likely diminished because the DISKCO tokens already bring ample information from the CE model into the BE model.

Table 6: Effect of standard Knowledge Distillation loss on Bi-encoder (BE) and DISKCO on MSMARCO Dev set.

Model	MRR@10
Cross Encoder	44.00
BE without KD loss	35.12
BE with KD	37.18
DISKCO without KD loss	39.15
DISKCO with KD	39.50

5 CONCLUSION

This paper introduces a new method called DISKCO, which aims to enhance bi-encoder models by leveraging cross encoder models. DISKCO utilizes the attention weights of the cross encoder to extract contextual tokens from documents. These contextual tokens are then incorporated into the keywords during the training of the bi-encoder model.

We presented results on the MSMARCO, GLUE and a multilingual internal datasets. We demonstrated superior performance compared to all the purely bi-encoder baselines. Over a bi-encoder with no knowledge distillation, DISKCO achieves 4% higher AUC. Over standard knowledge distillation, we show that DISKCO achieves a gain of 2%. Additionally, we performed extensive ablation studies to show the impact of the contextual token on the performance of bi-encoder models.

REFERENCES

[1] Hiteshwar Kumar Azad and Akshay Deepak. 2019. Query expansion techniques for information retrieval: a survey. *Information Processing & Management* 56, 5 (2019), 1698–1735.

[2] Arindam Bhattacharya, Ankit Gandhi, et al. 2023. Beyond hard negatives in product search: Semantic matching using one-class classification (SMOCC). In *WSDM 2023*.

[3] Arindam Bhattacharya, Ankith Ms, et al. 2023. CUPID: Curriculum Learning Based Real-Time Prediction using Distillation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*. Association for Computational Linguistics.

[4] Claudio Carpineto, Renato De Mori, Giovanni Romano, and Brigitte Bigi. 2001. An information-theoretic approach to automatic query expansion. *ACM Transactions on Information Systems (TOIS)* 19, 1 (2001).

[5] Claudio Carpineto and Giovanni Romano. 2012. A survey of automatic query expansion in information retrieval. *ACM Computing Surveys (CSUR)* 44, 1 (2012).

[6] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What Does BERT Look at? An Analysis of BERT's Attention. Association for Computational Linguistics, Florence, Italy.

[7] Hang Cui, Ji-Rong Wen, et al. 2002. Probabilistic Query Expansion Using Query Logs. In *Proceedings of the 11th International Conference on World Wide Web*. Association for Computing Machinery.

[8] Jacob Devlin, Ming-Wei Chang, et al. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. Association for Computational Linguistics.

[9] Efthimis N Efthimiadis. 1996. Query Expansion. *Annual review of information science and technology (ARIST)* 31 (1996), 121–87.

[10] Luyu Gao, Zhuyun Dai, et al. 2021. COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

[11] Albert Gordo, Filip Radenovic, and Tamara Berg. 2020. Attention-based query expansion learning. In *European Conference on Computer Vision*. Springer, 172–188.

[12] Geoffrey Hinton, Oriol Vinyals, et al. 2015. Distilling the Knowledge in a Neural Network. arXiv:1503.02531 [stat.ML]

[13] Po-Sen Huang, Xiaodong He, et al. 2013. Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*. Association for Computing Machinery.

[14] Xiaoqi Jiao, Yichun Yin, et al. 2020. TinyBERT: Distilling BERT for Natural Language Understanding. arXiv:1909.10351 [cs.CL]

[15] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery.

[16] Ryan Kiros, Yukun Zhu, et al. 2015. Skip-Thought Vectors. In *Advances in Neural Information Processing Systems*, Vol. 28. Curran Associates, Inc.

[17] Zheng Lin, Yeyun Gong, et al. 2023. PROD: Progressive Distillation for Dense Retrieval. *Proceedings of the ACM Web Conference 2023* (2023).

[18] Weijie Liu, Peng Zhou, et al. 2020. Fastbert: a self-distilling bert with adaptive inference time. *arXiv preprint arXiv:2004.02178* (2020).

[19] Wenhao Lu, Jian Jiao, et al. 2020. TwinBERT: Distilling Knowledge to Twin-Structured BERT Models for Efficient Retrieval. arXiv:2002.06275 [cs.IR]

[20] Yuxiang Lu, Yiding Liu, et al. 2022. ERNIE-Search: Bridging Cross-Encoder with Dual-Encoder via Self On-the-fly Distillation for Dense Passage Retrieval. *ArXiv* abs/2205.09153 (2022).

[21] Sourab Mangrulkar, Ankith M S, et al. 2022. Multilingual semantic sourcing using product images for cross-lingual alignment. In *The Web Conference 2022*.

[22] Priyanka Nigam, Yiwei Song, et al. 2019. Semantic Product Search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Association for Computing Machinery.

[23] Priyanka Nigam, Yiwei Song, et al. 2019. Semantic Product Search. arXiv:1907.00937 [cs.IR]

[24] Yingqi Qu, Yuchen Ding, et al. 2020. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. *ArXiv* abs/2010.08191 (2020).

[25] Chandan K. Reddy et al. 2022. Shopping Queries Dataset: A Large-Scale ESCI Benchmark for Improving Product Search. *arXiv preprint arXiv:2206.06588* (2022). arXiv:2206.06588

[26] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).

[27] Ruiyang Ren, Shangwen Lv, et al. 2021. PAIR: Leveraging Passage-Centric Similarity Relation for Improving Dense Passage Retrieval. *ArXiv* abs/2108.06027 (2021).

[28] Ruiyang Ren, Yingqi Qu, et al. 2021. RocketQAv2: A Joint Training Method for Dense Passage Retrieval and Passage Re-ranking. *ArXiv* abs/2110.07367 (2021).

[29] Dwaipayan Roy, Debjyoti Paul, Mandar Mitra, and Utpal Garain. 2016. Using word embeddings for automatic query expansion. *arXiv preprint arXiv:1606.07608* (2016).

[30] Victor Sanh, Lysandre Debut, et al. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108 [cs.CL]

[31] Keshav Santhanam, O. Khattab, et al. 2021. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. In *North American Chapter of the Association for Computational Linguistics*.

[32] Siqi Sun, Yu Cheng, et al. 2019. Patient Knowledge Distillation for BERT Model Compression.

[33] Raphael Tang, Yao Lu, et al. 2019. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136* (2019).

[34] Ashish Vaswani, Noam Shazeer, et al. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*.

[35] Zhi Zheng, Kai Hui, et al. 2020. BERT-QE: Contextualized Query Expansion for Document Re-ranking. arXiv:2009.07258 [cs.IR]