Modularized Transfomer-based Ranking Framework

Luyu Gao **Zhuyun Dai** **Jamie Callan**
Language Technologies Institute
Carnegie Mellon University
{luyug, zhuyund, callan}@cs.cmu.edu

Abstract

Recent innovations in Transformer-based ranking models have advanced the state-of-the-art in information retrieval. However, these Transformers are computationally expensive, and their opaque hidden states make it hard to understand the ranking process. In this work, we modularize the Transformer ranker into separate modules for text *representation* and *interaction*. We show how this design enables substantially faster ranking using offline pre-computed representations and light-weight online interactions. The modular design is also easier to interpret and sheds light on the ranking process in Transformer rankers.[1]

1 Introduction

Neural rankers based on Transformer architectures (Vaswani et al., 2017) fine-tuned from BERT (Devlin et al., 2019) achieve current state-of-the-art (SOTA) ranking effectiveness (Nogueira and Cho, 2019; Craswell et al., 2019). The power of the Transformer comes from self-attention, the process by which all possible pairs of input tokens interact to understand their connections and contextualize their representations. Self-attention provides detailed, token-level information for matching, which is critical to the effectiveness of Transformer-based rankers (Wu et al., 2019).

When used for ranking, a Transformer ranker takes in the concatenation of a query and document, applies a series of self-attention operations, and outputs from its last layer a relevance prediction (Nogueira and Cho, 2019). The entire ranker runs like a black box and hidden states have no explicit meanings. This represents a clear distinction from earlier neural ranking models that keep separate text representation and distance (interaction) functions. Transformer rankers are slow (Nogueira et al., 2019), and the black-box design makes it hard to interpret their behavior.

We hypothesize that a Transformer-based ranker simultaneously performs text representation and query-document interaction as it processes the concatenated pair. Guided by this hypothesis, we decouple representation and interaction with a *MO*dualarized *RE*ranking *S*ystem (MORES). MORES consists of three *Transformer* modules: the Document Representation Module, the Query Representation Module, and the Interaction Module. The two Representation Modules run independently of each other. The **Document Representation Module** uses self-attention to embed each document token conditioned on all document tokens. The **Query Representation Module** embeds each query token conditioned on all query tokens. The **Interaction Module** performs attention from *query* representations to *document* representations to generate match signals and aggregates them through self-attention over query tokens to make a relevance prediction.

By disentangling the Transformer into modules for representation and interaction, MORES can take advantage of the indexing process: while the interaction must be done online, document representations can be computed offline. We further propose two strategies to pre-compute document representations that can be used by the Interaction Module for ranking.

Our experiments on a large supervised ranking dataset demonstrate the effectiveness and efficiency of MORES. It is as effective as a state-of-the-art BERT ranker and can be up to 120× faster at ranking. A domain adaptation experiment shows that the modular design does not affect the model transfer capability, so MORES can be used under low-resource settings with simple adaptation techniques. By adapting individual modules, we discovered differences between represen-

[1]Open source code at https://github.com/luyug/MORES

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 4180–4190,
November 16–20, 2020.

tations and interaction in adaptation. The modular design also makes MORES more interpretable, as shown by our attention analysis, providing new understanding of black-box Transformer rankers.

2 Related Work

Neural ranking models for IR proposed in previous studies can be generally classified into two groups (Guo et al., 2016): *representation-based* models, and *interaction-based* models.

Representation-based models learn latent vectors (embeddings) of queries and documents and use a simple scoring function (e.g., cosine) to measure the relevance between them. Such methods date back to LSI (Deerwester et al., 1990) and classical siamese networks (Bromley et al., 1993). More recent research considered using modern deep learning techniques to learn the representations. Examples include DSSM (Huang et al., 2013), C-DSSM (Shen et al., 2014), etc. Representations-based models are efficient during evaluation because the document representations are independent of the query, and therefore can be pre-computed. However, compressing a document into a single low-dimensional vector loses specific term matching signals (Guo et al., 2016). As a result, previous representation-based ranking models mostly fail to outperform interaction-based ones.

Interaction-based models, on the other hand, use a neural network to model the word-level interactions between the query and the document. Examples include DRMM (Guo et al., 2016) and K-NRM (Xiong et al., 2017). Recently, Transformers (Vaswani et al., 2017), especially BERT (Devlin et al., 2019) based Transformers, have been widely used in information retrieval ranking tasks (Nogueira and Cho, 2019; Dai and Callan, 2019; Qiao et al., 2019). BERT-based rankers concatenate query and document into a single string and apply self-attention that spans over the query and the document in every layer. Rankers using pre-trained Transformers such as BERT has become the current state-of-the-art (Craswell et al., 2019). However, the performance gains come at the computational cost of inferring the many token-level interaction signals at the evaluation time, which scales quadratically to the input length. It is an open question whether we can *combine* the advantages of representation-based and interaction-based approaches. Little research has studied this direction prior to this work.

There are several research directions aiming to reduce the computational cost of Transformer models. One line of research seeks to compress the big Transformer into smaller ones using model pruning (Voita et al., 2019) or knowledge distillation (Hinton et al., 2015; Sanh et al., 2019). Another line of research aims to develop new Transformer-like units that have lower complexity than the original Transformer. For example, (Child et al., 2019) introduces sparse factorizations of the attention matrix which efficiently compute subsets of the attention matrix. The focus of this work is an efficient framework to combine Transformers for ranking; all aforementioned techniques can be applied to individual Transformers within our framework, and are therefore orthogonal to this paper.

3 Proposed Method

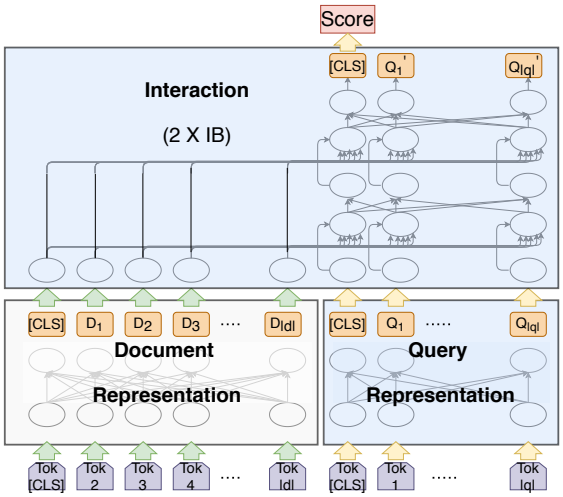
In this section, we introduce the Modularized Reranking System (MORES), how MORES can speed up retrieval, and how to effectively train and initialize MORES.

3.1 The MORES Framework

A typical Transformer ranker takes in the *concatenation* of a query qry and a document doc as input. At each layer, the Transformer generates a new contextualized embedding for each token based on its attention to all tokens in the concatenated text. This formulation poses two challenges. First, in terms of speed, the attention consumes time quadratic to the input length. As shown in Table 1, for a query of q tokens and a document of d tokens, the Transformer would require assessments of $(d+q)^2$ pairs of tokens. Second, as query and document attention is entangled from the first layer, it is challenging to interpret the model.

MORES aims to address both problems by disentangling the Transformer ranker into document representation, query representation, and interaction, each with a dedicated Transformer, as shown in Figure 1. The document representation is query-agnostic and can be computed off-line. The interaction uses query-to-document attention, which further reduces online complexity. This separation also assigns roles to each module, making the model more transparent and interpretable.

Figure 1: An illustration of the attention within a MORES model using two layers of Interaction Blocks (2× IB). Representation Modules only show 1 layer of attention due to space limits. In a real model, Document Representation Module and Query Representation Module are deeper than shown here.



The two **Representation Modules** use Transformer encoders (Vaswani et al., 2017) to embed documents and queries respectively and independently. In particular, for documents,

$$H_l^{doc} = \text{Encoder}_l^{doc}(H_{l-1}^{doc}) \quad (1)$$
$$H_1^{doc} = \text{Encoder}_1^{doc}(\text{lookup}(doc)) \quad (2)$$

and for queries,

$$H_l^{qry} = \text{Encoder}_l^{qry}(H_{l-1}^{q}) \quad (3)$$
$$H_1^{qry} = \text{Encoder}_1^{qry}(\text{lookup}(qry)) \quad (4)$$

where lookup represents word[2] and position embeddings, and Encoder represents a Transformer encoder layer. Query and document Representation Modules can use different numbers of layers. Let M and N denote the number of layers for document and query representations respectively. The hidden states from the last layers are used as the Representation Modules' output. Formally, for a document of length d, query of length q, and model dimension n, let matrix $D = H_M^{doc} \in \mathbb{R}^{d \times n}$ be the output of the Document Representation Module and $Q = H_N^{qry} \in \mathbb{R}^{q \times n}$ be the output of the Query Representation module.

The **Interaction Module** uses the Representation Modules' outputs, Q and D, to make a relevance judgement. The module consists of a stack of Interaction Blocks (IB), a novel attentive block that performs query-to-document cross-attention, followed by query self-attention[3], as shown in Figure 1. Here, we write cross-attention from X to Y as $\text{Attend}(X, Y)$, self-attention over X as $\text{Attend}(X, X)$ and layer norm as LN. Let,

$$Q_{\text{x}} = \text{LN}(\text{Attend}(Q, D) + Q) \quad (5)$$
$$Q_{\text{self}} = \text{LN}\left(\text{Attend}(Q_{\text{x}}, Q_{\text{x}}) + Q_{\text{x}}\right) \quad (6)$$

Equation 5 models interactions from query tokens to document token. Each query token in Q attends to document embeddings in D to produce relevance signals. Then, Equation 6 collects and exchanges signals among query tokens by having the query tokens attending to each other. The output of the first Interaction Block (IB) is then computed with a feed-forward network (FFN) on the query token embeddings with residual connections,

$$\text{IB}(Q, D) = \text{LN}\left(\text{FFN}\left(Q_{\text{self}}\right) + Q_{\text{self}}\right) \quad (7)$$

We employ multiple Interaction Blocks to iteratively repeat this process and refine the hidden query token representations, modeling multiple rounds of interactions, producing a series of hidden states, *while keeping document representation D unchanged*,

$$H_l^{IB} = \text{IB}_l(H_{l-1}^{IB}, D) \quad (8)$$
$$H_1^{IB} = \text{IB}_1(Q, D) \quad (9)$$

The Interaction Block (IB) is a core component of MORES. As shown in Table 1, its attention avoids the heavy full-attention over the concatenated query-document sequence, i.e. $(d + q)^2$ terms, saving online computation.

To induce relevance, we project the [CLS] token's embedding in the last (K^{th}) IB's output to a score,

$$score(qry, doc) = \mathbf{w}^T \text{CLS}(H_K^{\text{IB}}) \quad (10)$$

3.2 Pre-Compute and Reuse Representation

MORES's modular design allows us to precompute and reuse representations. The Query Representation Module runs once when receiving the new query; the representation is then repeatedly used to rank the candidate documents. More importantly, the document representations can be built offline. We detail two representation

[2]We use WordPiece tokens, following BERT.

[3]We use multi-head version of attention in the Interaction Blocks (IB).

Table 1: Time complexity of MORES and a typical Transformer ranker, e.g., a standard BERT ranker. We write q for query length, d for document length, n for Transformer's hidden layer dimension, and N_{doc} for number of candidate documents to be ranked for each query. For interaction, Reuse-S1 corresponds to document representation reuse strategy, and Reuse-S2 projected document representation reuse strategy.

	Total, 1 Query-Document Pair	Online, 1 Query-Document Pair	Online, N_{doc} Documents
Typical Transformer Ranker	$n(d+q)^2+n^2(d+q)$	$n(d+q)^2+n^2(d+q)$	$(n(d+q)^2+n^2(d+q))N_{doc}$
Document Representation	nd^2+n^2d	0	0
Query Representation	nq^2+n^2q	nq^2+n^2q	(nq^2+n^2q)
Interaction w/ Reuse-S1	$n(qd+q^2)+n^2(q+d)$	$n(qd+q^2)+n^2(q+d)$	$(n(qd+q^2)+n^2(q+d))N_{doc}$
Interaction w/ Reuse-S2	$n(qd+q^2)+n^2(q+d)$	$n(qd+q^2)+n^2q$	$(n(qd+q^2)+n^2q)N_{doc}$

reuse strategies with different time vs. space trade-offs: 1) a document representation reuse strategy that stores the Document Representation Module's output, and 2) a projected document representation reuse strategy that stores the Interaction Module's intermediate transformed document representations. These strategies have the same overall math, produce *the same* ranking results, and only differ in time/space efficiency.

Document Representation Reuse Strategy (Reuse-S1) runs the Document Representation Module offline, pre-computing document representations D for all documents in the collection. When receiving a new query, MORES looks up document representations D for candidate documents, runs the Query Representation Module to get a query's representation Q, and feeds both to the Interaction Module to score. This strategy reduces computation by not running the Document Representation Module at query time.

Projected Document Representation Reuse Strategy (Reuse-S2) further moves document-related computation performed in the Interaction Module offline. In an IB, the cross-attention operation first projects document representation D with key and value linear projections (Vaswani et al., 2017)

$$D_k = DW_k,\ D_v = DW_v \tag{11}$$

where W_k, W_v are the projection matrices. For each IB, Reuse-S2 pre-computes and stores D_{proj}[4],

$$D_{proj} = \{DW_k, DW_v\} \tag{12}$$

Using Reuse-S2, the Interaction Module no longer needs to compute the document projections at online evaluation time. Reuse-S2 takes more storage: for each IB, both key and value projections of D are stored, meaning that an Interaction Module with l IBs will store $2l$ projected versions of D. With this extra pre-computation, Reuse-S2 trades storage for further speed-up.

Table 1 analyzes the online time complexity of MORES and compares it to the time complexity of a standard BERT ranker. We note that MORES can move all document only computation offline. Reuse-S1 avoids the document self attention term d^2, which is often the most expensive part due to long document length. Reuse-S2 further removes from online computation the document transformation term n^2d, one that is linear in document length and quadratic in model dimension.

3.3 MORES Training and Initialization

MORES needs to learn three Transformers: two Representation Modules and one Interaction Module. The three Transformer modules are *coupled* during training and *decoupled* when used. To train MORES, we connect the three Transformers and enforce module coupling with end-to-end training using the pointwise loss function (Dai and Callan, 2019). When training is finished, we store the three Transformer modules separately and apply each module at the desired offline/online time.

We would like to use pre-trained LM weights to ease optimization and improve generalization. However, there is no existing pre-trained LM that involves cross-attention interaction that can be used to initialize the Interaction Module. To avoid expensive pre-training, we introduce BERT weight assisted initialization. We use one copy of BERT weights to initialize the Document Representation Module. We split another copy of BERT weights between Query Representation and Interaction Modules. For MORES with l IBs, the first $12-l$ layers of the BERT weights initialize the Query Representation Module, and the remaining

[4]We pre-compute for all attention heads in our multi-head implementation

l layers' weights initialize the Interaction Module. This initialization scheme ensures that Query Representation Module and the IBs use consecutive layers from BERT. As a result, upon initialization, the output of the Query Representation Module and the input of the first IB will live in the same space. In addition, for IBs, query to document attention initializes with the same BERT attention weights as query self-attention. In practice, we found initializing query to document attention weights important; random initialization leads to substantially worse performance. Details can be found in subsection 4.2.

4 Effectiveness and Efficiency in Supervised Ranking

The first experiment compares the effectiveness and efficiency of MORES to a state-of-the-art BERT ranker for supervised ranking.

4.1 Setup

We use the **MS MARCO passage ranking collection** (MS MARCO) (Nguyen et al., 2016) and evaluate on two query sets with distinct characteristics: *Dev Queries* have a single relevant document with a binary relevance label. Following Nguyen et al. (2016), we used MRR@10 to evaluate the ranking accuracy on this query set. *TREC2019 DL Queries* is the evaluation set used in the TREC 2019 Deep Learning Track. Its queries have multiple relevant documents with graded relevance. Following Craswell et al. (2019), we used MRR, NDCG@10, and MAP@1000 as evaluation metrics. All methods were evaluated in a *reranking* task to re-rank the top 1000 documents of the MS MARCO official BM25 retrieval results.

We test MORES effectiveness with a varied number of Interaction Blocks (IB) to study the effects of varying the complexity of query-document interaction. Models using 1 layer of IB ($1\times$ IB) up to 4 layers of IB ($4\times$ IB) are tested.

We compare MORES with the BERT ranker, a state-of-the-art ranker fine-tuned from BERT, which processes concatenated query-document pairs. Both rankers are trained with the MS MARCO training set consisting of single relevance queries. We train MORES on a 2M subset of Marco's training set. We use stochastic gradient descent to train the model with a batch size of 128. We use AdamW optimizer with a learning rate of 3e-5, a warm-up of 1000 steps and a linear learning rate scheduler for all MORES variants. Our baseline BERT model is trained with similar training setup to match performance reported by Nogueira and Cho (2019). Our BERT ranker re-implementation has better performance compared to that reported by Nogueira and Cho (2019). The BERT ranker and all MORES models are implemented with Pytorch (Paszke et al., 2019) based on the huggingface implementation of Transformers (Wolf et al., 2019).

We aim to test that MORES' accuracy is equivalent to the original BERT ranker (while achieving higher efficiency). To establish equivalence, statistical significance testing was performed with a non-inferiority test commonly used in the medical field to test that two treatments have similar effectiveness (Jayasinghe et al., 2015). In this test, rather than testing to reject the null hypothesis H_0: $\mu_{\text{BERT}} = \mu_{\text{MORES}}$, we test to reject H_0' : $\mu_{\text{BERT}} - \mu_{\text{MORES}} > \delta$ for some small margin δ. By rejecting H_0' we accept the alternative hypothesis, which is that any reduction of performance in MORES compared to the original BERT ranker is inconsequential. We set the margin δ to 2% and 5% of the mean of the BERT ranker.

4.2 Ranking Effectiveness

Table 2 reports the accuracy of MORES and the baseline BERT-based ranker. The experiments show that MORES with $1\times$ IB can achieve 95% of BERT performance. MORES with $2\times$ IB can achieve performance comparable to the BERT ranker with a 2% margin. Three IBs does not improve accuracy and four hurts accuracy. We believe that this is due to increased optimization difficulties which outweighs improved model capacity. Recall that for MORES we have one set of artificial cross attention weights *per IB* not initialized with real pre-trained weights. Performance results are consistent across the two query sets, showing that MORES can identify strong relevant documents (Dev Queries), and can also generalize to ranking multiple, weaker relevant documents (TREC2019 DL Queries).

The results show that MORES can achieve ranking accuracy competitive with state-of-the-art ranking models, and suggest that the entangled and computationally expensive full-attention Transformer can be replaced by MORES's lightweight, modularized design. Document

Table 2: Effectiveness of MORES models and baseline rankers on the MS MARCO Passage Corpus. $*$ and † indicate non-inferiority (Section 4.1) with $p < 0.05$ to the BERT ranker using a 5% or 2% margin, respectively.

	MS MARCO Passage Ranking			
	Dev Queries	TREC2019 DL Queries		
Model	MRR	MRR	NDCG@10	MAP
BERT ranker	0.3527	0.9349	0.7032	0.4836
MORES 1× IB	0.3334*	0.8953*	0.6721*	0.4516*
MORES 2× IB	0.3456†	0.9283†	0.7026†	0.4777†
MORES 3× IB	0.3423†	0.9271†	0.6980†	0.4687*
MORES 4× IB	0.3307*	0.9322†	0.6565*	0.4559*

Table 3: Ranking Accuracy of MORES when using / not using attention weights copied from BERT to initialize Interaction Module. The models were tested on the MS MARCO dataset with the Dev Queries.

	Dev Queries	TREC2019 DL		
	MRR@10	MRR	NDCG@10	MAP
copy	0.3456	0.9283	0.7026	0.4777
random	0.2723	0.8430	0.6059	0.3702

and query representations can be computed independently without seeing each other. With the contextualized representation, 2 layers of lightweight interaction are sufficient to estimate relevance.

We also investigate IB initialization and compare MORES 2× IB initialized by our proposed initialization method (copy self attention weight of BERT as IB cross attention weight), with a random initialization method (cross attention weights randomly initialized). Table 3 shows that random initialization leads to a substantial drop in performance, likely due to difficulty in optimization.

4.3 Ranking Efficiency

Section 3.2 introduces two representation reuse strategies for MORES with different time vs. space trade-offs. This experiment measures MORES' real-time processing speeds with these two strategies and compares them with measurement for the BERT ranker. We test MORES 1× IB and MORES 2× IB. Additional IB layers incur more computation but do not improve effectiveness, and are hence not considered. We record average time for ranking one query with 1000 candidate documents on an 8-core CPU and a single GPU.[5] We measured ranking speed with documents of length 128 and 512 with a fixed query length of 16. Tables 4 (a) and (b) show the speed tests for the two reuse strategies, respectively. We also include per document data storage size [6].

Table 4: Average time in seconds to evaluate one query with 1,000 candidate documents, and the space used to store pre-computed representations for each document. Len: input document length.

(a) Document Representation Reuse (Reuse-S1)

Len	Model	CPU Time		GPU Time		Space (MB)
128	BERT ranker	161s	-	2.70s	-	0
	MORES 1×IB	4s	40x	0.04s	61x	0.4
	MORES 2×IB	8s	20x	0.12s	22 x	0.4
512	BERT ranker	698s	-	13.05s	-	0
	MORES 1×IB	11s	66x	0.14s	91x	1.5
	MORES 2×IB	20s	35x	0.32s	40x	1.5

(b) Projected Document Representation Reuse (Reuse-S2)

Len	Model	CPU Time		GPU Time		Space (MB)
128	BERT ranker	161s	-	2.70s	-	0
	MORES 1×IB	2s	85x	0.02s	118x	1.5
	MORES 2×IB	5s	36x	0.05s	48x	3.0
512	BERT ranker	698s	-	13.05s	-	0
	MORES 1×IB	3s	170x	0.08s	158x	6.0
	MORES 2×IB	6s	124x	0.10s	124x	12.0

We observe a substantial speedup in MORES compared to the BERT ranker, and the gain is consistent across CPUs and GPUs. The original BERT ranker took hundreds of seconds – several minutes – to generate results for one query on a CPU machine, which is impractical for real-time use. Using Reuse-S1, MORES with 1× IB was 40x faster than the BERT ranker on shorter documents ($d = 128$); the more accurate 2× IB model also achieved 20x speedup. The difference is more profound on longer documents. As the length of the document increases, a larger portion of compute in BERT ranker is devoted to performing self-attention over the document sequence. MORES pre-computes document representations

[5] Details are in Appendix A.1.

[6] We report un-compressed values. Compression can further reduce data storage.

Table 5: Domain adaptation on ClueWeb09-B. adapt-interaction and adapt-representation use MORES 2× IB. $*$ and † indicate non-inferiority (Section 4.1) with $p < 0.05$ to the BERT ranker using a 5% or 2% margin, respectively.

	Clueweb09-B					
	Title Queries			Description Queries		
	NDCG@20	MAP	Prec@20	NDCG@20	MAP	Prec@20
BERT ranker	0.3294	0.1882	0.3755	0.3597	0.2075	0.3881
MORES 1× IB	0.3059	0.1753	0.3407	0.3472	0.2009	0.3705
MORES 2× IB	0.3317†	0.1872†	0.3662†	0.3571†	0.2039†	0.3816†
MORES 3× IB	0.3299†	0.1841†	0.3679†	0.3476*	0.2008*	0.3763*
MORES 4× IB	0.3164*	0.1824*	0.3515	0.3472*	0.2012*	0.372*
adapt-interaction	0.3179*	0.1849†	0.3548	0.3385	0.1976*	0.3652
adapt-representation	0.3319†	0.1865†	0.3657*	0.3557†	0.2072†	0.3828†

Table 6: Domain adaptation on Robust04. adapt-interaction and adapt-representation use MORES 2× IB. $*$ and † indicate non-inferiority (Section 4.1) with $p < 0.05$ to the BERT ranker using a 5% or 2% margin, respectively.

	Robust04					
	Title Queries			Description Queries		
	NDCG@20	MAP	Prec@20	NDCG@20	MAP	Prec@20
BERT ranker	0.4632	0.2225	0.3958	0.5065	0.245	0.4147
MORES 1× IB	0.4394*	0.2097	0.3741*	0.4683	0.2263	0.3835
MORES 2× IB	0.4599†	0.2194†	0.3940†	0.4846*	0.2323*	0.4008*
MORES 3× IB	0.4551†	0.2135*	0.3934†	0.4854*	0.2334*	0.4006*
MORES 4× IB	0.4553†	0.2177†	0.3938†	0.4802	0.2309	0.3980*
adapt-interaction	0.4389	0.2117*	0.3723	0.4697	0.2249	0.3896
adapt-representation	0.4564†	0.2182†	0.3926†	0.4884*	0.2327*	0.4042*

and avoids document-side self attention, yielding up to 35x to 90x speedup on longer documents ($d = 512$).

Reuse-S2 – the projected document reuse strategy – further enlarges the gain in speed, leading to up to 170x speedup using 1× IB, and 120x speedup using 2× IB. Recall that Reuse-S2 precomputes the document projections that will be used in MORES' Interaction Module, which is of n^2d time complexity where n is the model hidden dimension (details can be found in the complexity analysis in Table 1). In practice, n is often large, e.g., our experiment used $n = 768$[7]. Reuse-S2 avoids the expensive n^2d term at evaluation time. Note that Reuse-S2 *does not affect accuracy*; it trades space to save more time.

5 Adaptation of MORES and Modules

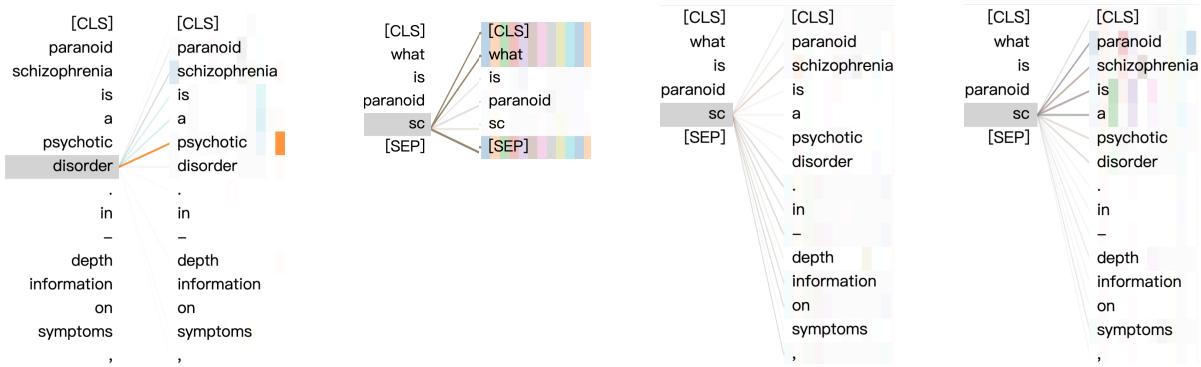
The second experiment uses a domain-adaptation setting to investigate whether the modular design of MORES affects adaptation and generalization ability, and how the individual Interaction and Representation Modules behave across domains.

[7]This follows model dimension in BERT

5.1 Setup

This experiment trains MORES using the MS MARCO dataset, and adapts the model to two datasets: ClueWeb09-B and Robust04. ClueWeb09-B is a standard document retrieval collection with 50M web pages crawled in 2009. Evaluation queries come from the TREC 2009-2012 Web Tracks. We used two variants of the queries: *Title Queries* is 200 short, keyword-style queries. *Description Queries* is 200 queries that are natural language statements or questions. Robust04 is a news corpus with 0.5M documents. Evaluation queries come from TREC 2004 Robust Track, including 250 *Title Queries* and 250 *Description Queries*. We evaluate ranking performance with NDCG@20, MAP, and Prec@20.

Domain adaptation is done by taking a model trained on MS MARCO and fine-tuning the model on relevant labels from the target dataset. Due to the small query sets in ClueWeb09-B and Robust04, we use 5-fold cross-validation for fine-tuning and testing. Data split, initial ranking, and document pre-processing follow Dai and Callan



(a) Document Representation (b) Query Representation (c) Interaction (1st IB) (d) Interaction (2nd IB)

Figure 2: Visualization of attention in MORES's Representation and Interaction Modules.

(2019). The domain adaptation fine-tuning procedures use a batch size of 32 and a learning rate of 5e-6 while having other training settings same as supervised ranking training.

5.2 Full Model Adaptation

The top 5 rows of Table 5 and Table 6 examine the effectiveness of adapting the full model of MORES. The adapted MORES models behave similarly as on MS MARCO: using two to three layers of Interaction Blocks (IB) achieves very close to BERT ranker performance on both datasets for both types of queries while using a single layer of IB is less effective. Importantly, our results show that the modular design of MORES does not hurt domain transfer, indicating that new domains and low resource domains can also use MORES through simple adaptation.

5.3 Individual Module Adaptation

With separate representation and interaction components in MORES, we are interested to see how each is affected by adaptation. We test two extra adaptation settings on MORES 2× IB: fine-tuning only Interaction Module on the target domain (adapt-interaction) or only Representation Modules (adapt-representation) on target domain. Results are shown in the bottom two rows of Table 5 and Table 6 for the two data sets.

We observe that only adapting the Interaction Module to the target domain is less effective compared to adapting the full model (MORES 2× IB), suggesting that changing the behaviour of interaction is not enough to accommodate language changes across domains. On the other hand, freezing the Interaction Module and only fine-tuning the Representation Modules (adapt-representation) produces performance on par with full model apdatation. This result shows that it is more necessary to have domain-specific representations, while interaction patterns are more general and not totally dependent on representations.

6 Analysis

The modular design of MORES allows Representation and Interaction to be inspected separately, providing better interpretability than a black-box Transformer ranker. Figure 2 examines the attention with MORES for a hard-to-understand query *"what is paranoid sc"* where "sc" is ambiguous, along with a relevant document *"Paranoid schizophrenia is a psychotic disorder. In-depth information on symptoms...."* [8]

In the Document Representation Module (Figure 2a), we can see that *"disorder"* uses *"psychotic"* and *"schizophrenia"* for contextualization, making itself more specific. In the Query Representation Module (Figure 2b), because the query is short and lacks context, *"sc"* incurs a broad but less meaningful attention. The query token *"sc"* is further contextualized in the Interaction Module (Figure 2c) using information from the document side – *"sc"* broadly attends to the document token in the first IB to disambiguate itself. With the extra context, *"sc"* is able to correctly attend to "schizophrenia" in the second IB to produce relevance signals (Figure 2d).

This example explains why MORES 1× IB performs worse than MORES with multiple IBs – ambiguous queries need to gather context from the document in the first IB before making relevance estimates in the second. More importantly, the example indicates that the query to document

[8] We only show the first 16 tokens due to space limitation.

attention has two distinct contributions: understand query tokens with the extra context from the document, and match query tokens to document tokens, with the former less noticed in the past. We believe MORES can be a useful tool for better interpreting and understanding SOTA black-box neural rankers.

7 Conclusion

State-of-the-art neural rankers based on the Transformer architecture consider all token pairs in a concatenated query and document sequence. Though effective, they are slow and challenging to interpret. This paper proposes MORES, a modular Transformer ranking framework that decouples ranking into Document Representation, Query Representation, and Interaction. MORES is effective while being efficient and interpretable.

Experiments on a large supervised ranking task show that MORES is as effective as a state-of-the-art BERT ranker. With our proposed document representation pre-compute and re-use methods, MORES can achieve 120x speedup in online ranking while retaining accuracy. Domain adaptation experiments show that MORES' modular design does not hurt transfer ability, indicating that MORES can be adapted to low-resource domains with simple techniques.

Decoupling representation and interaction provides new understanding of Transformer rankers. Complex full query-document attention in state-of-the-art Transformer rankers can be factored into independent document and query representation, and shallow light-weight interaction. We further discovered two types of interaction: further query understanding based on the document, and the query to document tokens matching for relevance. Moreover, we found that the interaction in ranking is less domain-specific, while the representations need more domain adaptation. These findings provide opportunities for future work towards more efficient and interpretable neural IR.

Acknowledgments

This work was supported in part by National Science Foundation (NSF) grant IIS-1815528. Any opinions, findings, and conclusions in this paper are the authors' and do not necessarily reflect those of the sponsors. The authors would also like to thank Graham Neubig and Chenyan Xiong for helpful discussions and feedbacks.

References

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a Siamese time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2019. Overview of the trec 2019 deep learning track. In *TREC (to appear)*.

Zhuyun Dai and Jamie Callan. 2019. Deeper text understanding for ir with contextual neural language modeling. In *The 42nd International ACM SIGIR Conference on Research & Development in Information Retrieval*.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, pages 55–64.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531.

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

Gaya K Jayasinghe, William Webber, Mark Sanderson, Lasitha S Dharmasena, and J Shane Culpepper. 2015. Statistical comparisons of non-deterministic ir systems using two dimensional variance. *Information Processing & Management*, 51(5):677–694.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. *arXiv:1901.04085*.

Rodrigo Nogueira, Kyunghyun Cho, Yang Wei, Lin Jimmy, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv:1904.08375*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

Yifan Qiao, Chenyan Xiong, Zheng-Hao Liu, and Zhiyuan Liu. 2019. Understanding the behaviors of BERT in ranking. *CoRR*, abs/1904.07531.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International World Wide Web Conference*, pages 373–374.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*.

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.

Zhijing Wu, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2019. Investigating passage-level relevance and its role in document-level relevance judgment. In *Proceedings of the 42nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 55–64.

A Appendix

A.1 Implmentation Details

Training Details On MS MARCO passage ranking dataset, we trained MORES over a 2M subset of Marco's training set. We use stochastic gradient descent to train the model with a batch size of 128. We use AdamW optimizer with a learning rate of 3e-5, a warm-up of 1000 steps and a linear learning rate scheduler for all MORES variants. Our baseline BERT model is trained with similar training setup to match performance reported in (Nogueira and Cho, 2019). We have not done hyper-parameter search, and all training setup is inherited from GLUE example in the huggingface transformer code base (Wolf et al., 2019). Following (Dai and Callan, 2019), we run a domain adaptation experiment on ClueWeb09-B: we take trained model on MS MARCO, and continue training over ClueWeb09-B's training data in a 5-fold cross-validation setup. We use a batch size of 32 and a learning rate of 5e-6. We select from batch size of 16 and 32, learning rate of 5e-6, 1e-5 and 2e-5 by validation point-wise accuracy.

Speed Test Details GPU test was run on a single RTX 2080 TI, with CUDA 10.1. We use a separate CUDA stream to pre-fetch data to the GPU. CPU tests was run in a SLURM task environment with 8 Xeon Silver 4110 logical cores.

A.2 Parameter Details

All MORES models follow BERT's architecture for initialization, having 12 attention heads, 768 embedding dimension, 3072 feed forward network hidden dimension. MORES with one IB up to four IBs have parameters of 224M, 228M, 231M and 233M parameters respectively.

A.3 Datasets

We use MSMARCO, ClueWeb09-b and Robust04. The first is available at https://microsoft.github.io/msmarco/ and the latter two at http://boston.lti.cs.cmu.edu/appendices/SIGIR2019-Zhuyun-Dai. All input text are tokenized by BERT's WordPiece tokenizer without other pre-processing. We evaluate MS MARCO Dev query sets with its provided evaluation script and the rest with trec_eval (https://github.com/usnistgov/trec_eval).