Multi-Vector Attention Models for Deep Re-ranking

Giulio Zhou*
Carnegie Mellon University
giuliozhou@cmu.edu

Jacob Devlin
Google Research
jacobdevlin@google.com

Abstract

Large-scale document retrieval systems often utilize two styles of neural network models which live at two different ends of the joint computation vs. accuracy spectrum. The first style is *dual encoder* (or two-tower) models, where the query and document representations are computed completely independently and combined with a simple dot product operation. The second style is *cross-attention* models, where the query and document features are concatenated in the input layer and all computation is based on the joint query-document representation. Dual encoder models are typically used for retrieval and deep re-ranking, while cross-attention models are typically used for shallow re-ranking. In this paper, we present a lightweight architecture that explores this joint cost vs. accuracy trade-off based on *multi-vector attention* (MVA). We thoroughly evaluate our method on the MS-MARCO passage retrieval dataset and show how to efficiently trade off retrieval accuracy with joint computation and offline document storage cost. We show that a highly compressed document representation and inexpensive joint computation can be achieved through a combination of learned pooling tokens and aggressive downprojection. Our code and model checkpoints are available on GitHub.

1 Introduction

Classical information retrieval systems used weighted sparse keyword matching to retrieve relevant documents for incoming search queries. A common approach has been to re-rank these documents with a neural network model which takes the concatenation of the query text and document text as input, and emits a relevance score. We refer to this as a *cross-attention network*, depicted in Figure 1a. In modern NLP, these systems are typically pre-trained using a technique such as BERT (Devlin et al., 2019) and then fine-tuned on human-labeled relevance data (Han et al., 2020; Ding et al., 2020; Nogueira et al., 2019).

However, even if the re-ranking model is a state-of-the-art neural system, it is still limited by the documents that were produced by the retrieval system. Because of this, end-to-end neural approaches to retrieval have become popular to improve the relevance of the documents produced in the retrieval stage. These models typically take the form of a *dual encoder network* (also called a "two-tower network," "Siamese network," and "DSSM" (Huang et al., 2013), as depicted in Figure 1b), which emits a "query vector" and "document vector" conditioned on the query text and document text respectively. The relevance score is typically defined as the dot product (or cosine distance) between these vectors.

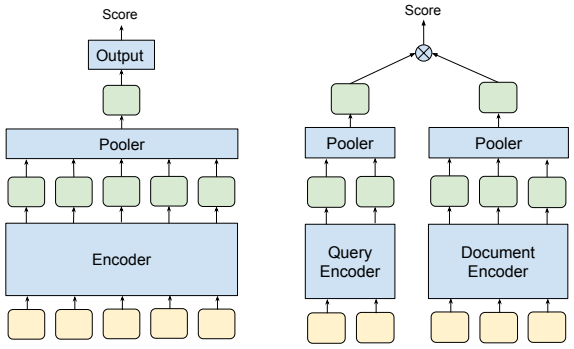
With a dual encoder model, each document vector in the corpus can be precomputed offline, and the query vector only needs to be computed once per incoming query. Since scoring each query-document pair is a simple dot product, a commodity CPU can score thousands of document in a few milliseconds. Moreover, retrieving the highest scoring documents for a query can be done in sub-linear time using approximate nearest neighbor algorithms such as hierarchical k-means clustering (Johnson et al., 2017; Guo et al., 2020).

In practice, modern IR systems are not restricted to this simple retrieve-then-rerank framework, and often use multi-stage re-ranking. For example, the system might first retrieve the top 1000 documents using a combination of a dual encoder and keyword retrieval, then re-rank with a very cheap cross-attention model, then finally re-score the top 50 with a more expensive cross-attention model.

There are two key costs to consider for these type of deep re-ranking models. The first is the number of bytes necessary to store the precomputed

*Work done while at Google.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 5452–5456
November 7–11, 2021.



(a) Cross-Attention model. (b) Dual Encoder model.

Figure 1: Standard information retrieval architectures.

document representation, since this must be stored in fast-access memory (typically RAM) for every document in the corpus. The second is the cost of the "joint computation," which is the part of the model that combines the query and document representation in order to generate a relevance score.

In this work, we explore the *Multi-Vector Attention* (MVA) architecture as an extension of dual encoder networks. The MVA network produces a query matrix (rather than vector) which attends to a document matrix to produce a *query-dependent* document representation. The scalar relevance score is then computed in a manner similar to a standard dual encoder model. A mathematical description is given in Section 2.1.

Related Work. Several retrieval approaches have been proposed that apply lightweight query-document scoring on last-layer Transformer features. These consist of *multi-vector dual encoders* (Luan et al., 2020; Khattab and Zaharia, 2020; Li et al., 2020) that emit multiple query and document vectors which interact via dot products, and *multi-layer attention architectures* (Gao et al., 2020; Chen et al., 2020; MacAvaney et al., 2020).

The work presented here can be thought of as an extension to ColBERT (Khattab and Zaharia, 2020), where we explore various aspects of the output layer in order to compress the document representation even further. We found the `max()` operation to be unstable when used in conjunction with the more aggressive pooling and downsampling, so we instead used a differentiable attention operation.

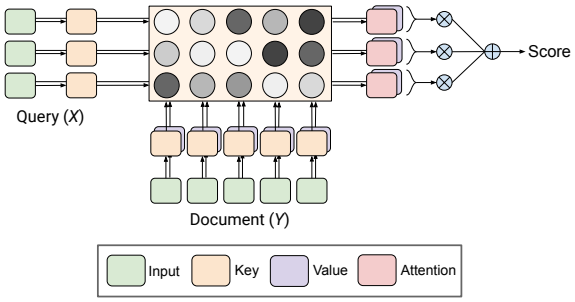


Figure 2: Multi-Vector Attention (MVA) Architecture.

2 Multi-Vector Attention Network

2.1 Model Architecture

Our architecture employs the standard form of "Transformer-style" dot product attention. Given input matrices $Q, K, V \in \mathbb{R}^{n \times h}$:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{h}})V$$

Single-headed attention is applied to query vectors $X \in \mathbb{R}^{q \times h}$ and document vectors $Y \in \mathbb{R}^{d \times h}$ using the learned projection matrices $W_{Q_K}, W_{Q_V}, W_{D_K}, W_{D_V} \in \mathbb{R}^{h \times h}$. These parameters are used to generate the intermediate query key, query value, document key, and document value matrices:

$$Q_K = XW_{Q_K} \quad Q_V = XW_{Q_V}$$

$$D_K = YW_{D_K} \quad D_V = YW_{D_V}$$

These matrices are passed into the operation $\text{Attention}(Q_K, D_K, D_V)$ to perform query token-dependent attention over document tokens for each individual query token X_i. The final relevance score between Q and D is given by:

$$\frac{1}{q}\sum_{i=1}^{q} {Q_V}_i^T \text{Attention}(Q_{K_i}, D_K, D_V)$$

which is the average dot product between query value vectors and their corresponding attention averaged document vector.

2.2 Pooling Architectures

First-K Tokens. In this pooling method, we truncate sequences to the first K tokens.

Multiple [CLS] Embeddings. We prepend query and document sequences with [CLS] embeddings $Q_{CLS} \in \mathbb{R}^h$ and $D_{CLS} \in \mathbb{R}^h$ that are retained following the encoder; this can be viewed

as a generalization of BERT's single-[CLS] embedding. In prior work, ColBERT (Khattab and Zaharia, 2020) applies this to the query and uses the untruncated sequence during scoring, referring to this as "query augmentation."

Temporal Pooling. We also explore a projection-based pooling approach that reduces sequences by a specified *pooling factor* ρ. We reshape the input sequence $X \in \mathbb{R}^{n \times h}$ into $\tilde{X} \in \mathbb{R}^{\frac{n}{\rho} \times \rho h}$, which concatenates every ρ consecutive elements into a single composite vector. Applying the Attention architecture projection layers to these composite vectors completes the pooling operation.

2.3 Losses

We pre-train using standard BERT objectives, which include the Masked Language Modeling (MLM) task as well as the Next-Sentence Prediction (NSP) task. Our models are fine-tuned using a softmax loss with the <query, positive document, negative document> training triples provided by MSMARCO.

3 Experimental Setup

Datasets. We pre-train all of our models on the Colossal Clean Crawled Corpus (C4). For retrieval evaluation, we use the MS-MARCO (Bajaj et al., 2016) passage re-ranking dataset. We truncate query sequences to length 32 and documents to length 112. We observe that the average lengths of queries and documents in MS-MARCO are around 6 and 70 respectively.

Training setup. We use a 12-layer Transformer model with 12 attention heads and hidden size 768, equivalent in architecture to $\text{BERT}_{\text{Base}}$. We pre-train dual encoder and cross-attention models on C4 for 100,000 iterations on a v3-128 Cloud TPU, with batch size 8,192 and Adam with learning rate 3e-4. Our Dual Encoder is pre-trained directly on the MLM and NSP tasks rather than initialized as two identical BERT models (which is the conventional practice), and is therefore a stronger baseline than is typically found in the literature. We initialize MVA models from the Dual Encoder checkpoint and pre-train on the same MLM and NSP tasks for 20,000 iterations before fine-tuning to downstream tasks. We fine-tune on MS-MARCO using a batch size of 256 triples and Adam with learning rate 3e-5. We reserve a 10% split of the MS-MARCO training set as a validation set for tuning hyperparameters and report results on the MS-MARCO dev set.

Pooling Method	Document Timesteps	MSMARCO Dev MRR@10
All Tokens	~70	35.8
First-K	56	35.4
First-K	28	34.5
First-K	14	31.9
Multiple CLS	56	35.2
Multiple CLS	28	34.9
Multiple CLS	14	33.2
Temporal Pooling	56	35.4
Temporal Pooling	28	34.1
Temporal Pooling	14	32.8

Table 1: A comparison of pooling approaches applied to documents.

4 Ablation Experiments

Our results center around two important considerations that impact model deployment: (1) the amount of joint computation needed to score every query-document pair and (2) the cost to store the document representations offline. We first conduct ablation experiments to evaluate how independent architecture variations affect downstream accuracy.

Multiple [CLS] pooling for short document representations, and no pooling for query sequences. In Table 1, we compare three approaches for reducing the length of document sequences: truncation, multiple [CLS] embeddings, and temporal pooling. We find that [CLS] embeddings are clearly superior for producing short document representations, but that all approaches perform similarly for longer representations.

In Table 3, we examine the effect of the number of [CLS] tokens on the query side. In the first row, "All Tokens", the query representation corresponds to one [CLS] token plus each actual (non-padding) token in the query. Our results suggest that short queries do not benefit much from query pooling.

Projections lower costs with little quality drop. Down-projection is supported by changing the size of the attention head, which leads to a linear reduction in both joint computation and in offline document storage. In Table 2, we show via the accuracy trade-offs that hidden size projections outperform a comparable reduction in document length.

Multi-head Attention is unnecessary. In Table 4, we evaluate whether the single attention

Projection Size	MSMARCO Dev MRR@10
768	35.8
128	36.0
64	35.4
32	35.0
16	34.7

Table 2: An evaluation of projections (by using smaller attention head sizes) on the full query and document sequences.

Query Timesteps	MSMARCO Dev MRR@10
All Tokens (Avg. ~6)	35.8
1 CLS	32.6
2 CLS	33.5
8 CLS	34.9
33 CLS	35.9

Table 3: Multiple CLS pooling applied to the query sequence with full document sequences.

Document Timesteps	Num Heads	Projection Size	MSMARCO Dev MRR@10
All Tokens (Avg. ~70)	1	64	35.4
56	1	64	34.6
28	2	64	33.5
28	1	128	34.4
14	4	64	31.7
14	1	256	33.5

Table 4: A study of multi-headed attention applied to several document configurations with equivalent dimensionality.

Model Name	Approx Joint Compute (mega-FLOPS)	Storage Cost (Floats)	MSMARCO Dev MRR@10
$\text{BERT}_{\text{Base}}$	11,000,000	0	36.0
ColBERT	286	8,960	34.9
CA (Ours)	11,000,000	0	38.0
DE (Ours)	0.8	768	29.6
MVA[1]	130	17,920	36.0
MVA[2]	25	3,072	34.7
MVA[3]	8	1,024	32.7

[1] 70 document timesteps, projection size 128.
[2] 24 document timesteps, projection size 64.
[3] 16 document timesteps, projection size 32.

Table 5: Re-ranking results for various model configurations. Cross-Attention (CA), Dual Encoder (DE), Multi-Vector Attention (MVA). $\text{BERT}_{\text{Base}}$ and ColBERT figures are from (Khattab and Zaharia, 2020).

First Pass Model	Num Re-ranked	Approx Joint Compute (mega-FLOPS)	MSMARCO Dev MRR@10
N/A	1000	11,000,000	38.0
DE	1	0.8	29.6
	10	110,000	35.3
	50	550,000	37.5
MVA (D=24, P=64)	1	25	34.7
	10	110,000	36.6
	50	550,000	37.9

Table 6: Multi-pass re-ranking using a cross-attention model. D = timesteps in document representation, P = projection size.

head used by MVA can be replaced by multiple smaller attention heads with the same combined dimensionality. We do so by examining a range of document sequence lengths where the length is varied using [CLS] embeddings, but where $(\text{length} \times \text{projection\_size})$ is held constant. We find that multiple projection heads worsen the results across all sequence lengths, and especially so for shorter document sequences.

Joint pre-training improves downstream performance. We find it helpful to first pre-train the MVA parameters for a small number of additional steps on the MLM and NSP tasks before fine-tuning on the downstream dataset. Pre-training a MVA model from scratch, however, is not necessary since the improvement in downstream accuracy is marginal.

5 Results

Optimal operating points. We present several optimal operating points that maximize MRR with respect to different amounts of allowed joint computation and storage cost. The configuration that achieves the highest MRR for reasonable cost uses the full query-document sequence and a projection size of 128. We also present two cheaper architecture variants that use aggressive projections and [CLS] pooling to lower computation, for a large improvement over dual encoders for small additional cost. Our approach attains comparable results to ColBERT (Khattab and Zaharia, 2020) without having to extensively pad the query with [CLS] tokens.

Improved Re-ranking with MVA. We simulate a three-stage re-ranking pipeline where the MSMARCO top-1000 candidates (originally retrieved by BM25) are first re-ranked using MVA (as well as the Dual Encoder) and then the top-K candidates further re-ranked with the cross-attention model. We show in Table 6 that a comparatively cheap MVA model substantially outperforms a Dual Encoder. Moreover, the amortized cost of cross-attention re-ranking enables a better operating point on the joint computation-accuracy curve.

6 Conclusion

We presented a Multi-Vector Attention (MVA) architecture for deep re-ranking that extends previous work on multi-vector dual encoders and attention architectures.

References

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Jiecao Chen, Liu Yang, Karthik Raman, Michael Bendersky, Jung-Jung Yeh, Yun Zhou, Marc Najork, Danyang Cai, and Ehsan Emadzadeh. 2020. Dipair: Fast and accurate distillation for trillion-scale text matching and pair modeling.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yingqi Qu Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2020. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2020. Modularized transfomer-based ranking framework. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4180–4190, Online. Association for Computational Linguistics.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*.

Shuguang Han, Xuanhui Wang, Mike Bendersky, and Marc Najork. 2020. Learning-to-rank with bert in tf-ranking.

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

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

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*, pages 39–48.

Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2020. Parade: Passage representation aggregation for document reranking. *arXiv preprint arXiv:2008.09093*.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2020. Sparse, dense, and attentional representations for text retrieval. *arXiv preprint arXiv:2005.00181*.

Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Efficient document re-ranking for transformers by precomputing term representations. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 49–58.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with bert.