BISON : BM25-weighted Self-Attention Framework for Multi-Fields Document Search

Xuan Shan
Microsoft STCA
Beijing, China
xuanshan@microsoft.com

Chuanjie Liu
Microsoft STCA
Beijing, China
chuanli@microsoft.com

Yiqian Xia
Microsoft STCA
Beijing, China
yiqxia@microsoft.com

Qi Chen
Microsoft Research Asia
Beijing, China
chenqi@microsoft.com

Yusi Zhang
Microsoft STCA
Beijing, China
yuszhang@microsoft.com

Angen Luo
Microsoft STCA
Beijing, China
anluo@microsoft.com

Yuxiang Luo
Microsoft STCA
Beijing, China
yuxlu@microsoft.com

Abstract

Recent breakthrough in natural language processing has advanced the information retrieval from keyword match to semantic vector search. To map query and documents into semantic vectors, self-attention models are being widely used. However, typical self-attention models, like Transformer, lack prior knowledge to distinguish the importance of different tokens, which has been proved to play a critical role in information retrieval tasks. In addition to this, when applying WordPiece tokenization, a rare word may be split into several different tokens. How to translate word-level prior knowledge into WordPiece tokens becomes a new challenge for the semantic representation generation. Moreover, web documents usually have multiple fields. Due to the heterogeneity of different fields, simple combination is not a good choice. In this paper, We propose a novel **B**M25-we**I**ghted **S**elf-Attenti**ON** framework (BISON) for web document search. By leveraging BM25 as prior weights, BISON learns weighted attention scores jointly with query matrix Q and key matrix K. We also present an efficient whole word weight sharing solution to mitigate prior knowledge discrepancy between words and WordPiece tokens. Furthermore, BISON effectively combines multiple fields by placing different fields into different segments. We demonstrate BISON is more efficient to capture the topical and semantic representation both in query and document. Intrinsic evaluation and experiments conducted on public data sets reveal BISON to be a general framework for document ranking task. It outperforms BERT and other modern models while retaining the same model complexity with BERT[1].

1 Introduction

Nowadays, most search engines use two ranking phases, recall and precision, to retrieve ideal results from a massive amount of documents in order to obtain milliseconds query response time. The recall

[1] The source code is available at https://github.com/cadobe/bison

Preprint. Under review.

phase applies a coarse-grained search to quickly select a small set of candidates from billions of documents using low-cost metrics. Then some complex ranking algorithms are used to prune the results in the precision phase. Traditionally, the recall phase is built on top of an inverted index using keyword match with some query alterations. However, it is hard to cover all the alteration cases and well understand user's intention. With recent breakthrough in deep learning, web content can be more meaningfully represented as vectors. Vector search has been attracting more attention recently to remedy the disadvantages of traditional keyword-based approach. It leverages high efficient Approximate Nearest Neighbor (ANN) search algorithms to retrieve relevant results according to the vector distance. To achieve this, the most important part is to map query and documents into semantic vector representation.

Building a suitable model to learn query/document embedding representation for retrieval tasks is challenging, not only because the mismatching between query and document, but also multiple fields of document should be taken into consideration. Recently, Transformer based models like BERT(7) are being widely enabled to tackle these issues (21; 25). However, when leveraging the vanilla Transformer, token's attention score is contributed from all others' without distinction. By adding position and segment signals can slightly alleviate the homogeneity, but it is still not token-wise. In fact, in information retrieval community, it is well-known that some tokens are more important than others in contextually representing according to the prior knowledge, thus an emphasis on these topical tokens is critical. Lacking such information makes the deep model difficult to represent the topic. Some studies have proved that BERT is not so good in topic learning without considering the prior knowledge. To name a few, by involving knowledge graph information into masking language model tasks, ERNIE model(30) achieves new SOTA on several NLP tasks. Kim et al. (13) significantly improves speech-enhancement performance by integrating a Gaussian-weight into self-attention. All of these aforementioned challenges increase the complexity of encoding query and document into meaningful and precise semantic vector space. BM25(28) has advanced information retrieval in last decades. A word with high BM25 score shows its uniqueness in query or document. It has been widely adopted in traditional learning to rank tasks, unfortunately seldom studies investigate to integrate it into Transformer.

本文要解决的问题

Inspired by this, in this paper we introduce BISON: a BM25-weighted Self-Attention framework to learn the distributed representations of query and document. It pre-computes inherent BM25 scores for query and document respectively, then taking this score as the guarding weight when performing self-attention. BISON leverages a 30,000 token vocabulary from WordPiece embedding (32), while BM25 is usually generated on natural word level, different words are mapped into different number of tokens. It is vital to pave a way to pass BM25 score from word level to token level. We propose a whole word weight sharing mechanism to bridge the discrepancy between words and tokens. For the multiple fields challenge in document side, we also demonstrate an innovative combined field representation to encode document to a unified vector space. Different with prior solutions, our combined field representation reduce document embedding to one vector, which is a dramatic storage and computation saving.

To the best of our knowledge, this is the first time that research work successfully integrates BM25 into self-attention based models as a guarding weight and embeds multi-fields document into one unified vector. BISON significantly improves the search relevance by intrinsic evaluation. We also measure BISON on public data set, the results show BISON is superior in quality without increasing model complexity.

2 Background and related work

The document ranking (also known as *ad-hoc retrieval*) task can be described as, given one query q, the system produces the best ranking of documents D from a mass of candidates. When it comes to deep learning era, Mitra and Craswell (17)give a detailed introduction about the researches made on information retrieval with deep neural networks. Deep neural models are usually equipped into search engines by a Siamese (symmetric) architecture(8; 29; 11) or an Interaction-focused manner(10; 16; 23). The major difference between these 2 architectures lies in when query interacts with document, the Siamese approach encodes query and document separately while Interactive way jointly learns query with document at the very beginning. For large scale document recall tasks, especially those that depend on vector search, the Siamese approach is preferred since a multitude of documents are supposed to be encoded without the help of query offline. To better facilitate document search tasks, our proposed framework BISON is built upon Siamese architecture.

2.1 Transformer models in document ranking

Pre-train language modeling has been proved to be effective on natural language processing tasks. One of such models, BERT(7), has been widely applied into retrieval-based tasks like document ranking(34) and question answering(33; 21). MS Marco(3) is a collection data set for multi-perspective web search tasks. So far[2], the top 10 winners in the leading board all leverage BERT as a basis. Typically, Nogueira et al.(22) built a multi-stage ranking architecture on BERT by formulating the ranking problem as pointwise and pairwise classification, respectively. Han et al. combined DeepCT retrieval model(6) with a TF-Ranking BERT ensemble(24). The DeepCT-Index produces term weights that can be stored in an ordinary inverted index for document ranking. Observed from another famous information retrieval data set ClueWeb09(4), the announced high results are also trained on Transformer based models. XLNet(35) claimed its state-of-the-art result, superior to RoBERTa(15), GPT(26) and BERT+DCMN(38).
Most of these studies consolidate on single field document. Although Zamani et al.(36) proposes a deep neural ranking model on multi-fields document ranking. Self-Attention based approaches have not been well studied yet for multi-fields document.

3 Proposed methods

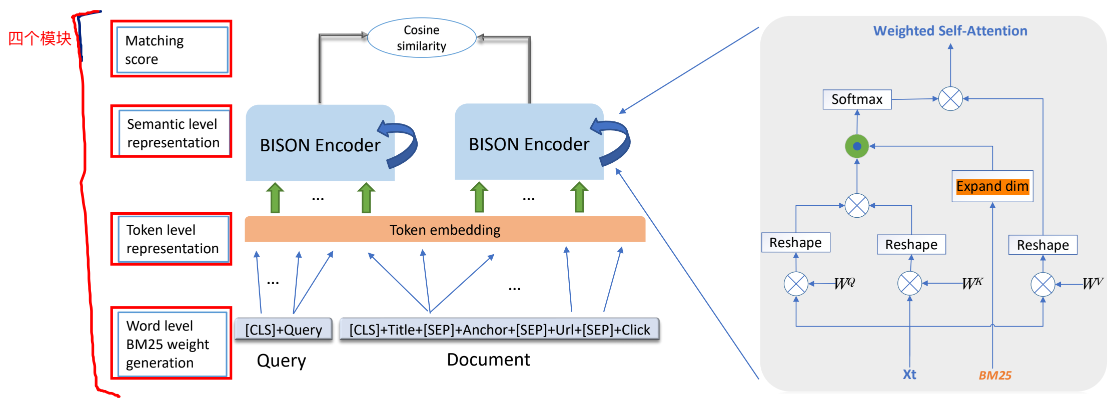

Figure 1: Illustration of BISON and the core unit Weighted Self-Attention. The blue arrows between word level weight generation and token level representation indicate the whole word weight sharing methodology. One word might map to single or multiple tokens.

3.1 Overview of BISON

The structure of BISON is shown in left part of Figure 1. BISON is comprised of four parts. For the word level BM25 weight generation, we simply prepend a [CLS] to query and use a combined fields representation to represent document(will be detailed discussed in Section 3.3 and Section 3.5). When computing BM25 as weight, as mentioned in Section 1 there is always an alignment challenge between WordPiece tokens and natural words. we propose a whole word weight sharing mechanism in Section 3.4 to map weight from whole words to WordPiece tokens. We also conduct experiments both on token and word level weight generation in Section 5.1 to prove the high efficiency of the whole word weight generation. In the token level representation layer, the same with BERT, we use the sum of token embedding, position embedding and segment embedding to form the token representation. Then BISON Encoder is responsible for encoding query and document into semantic space by Siamese structure which makes efficient online serving possible, we will describe it in Section 3.2. The semantic representation layer takes BISON Encoder by stacking 3 times. Lastly, we adopt *cosine similarity* to describe the matching score.

[2]As of 30th May, 2020.

3.2 BISON Encoder: Weighted Self-Attention

Let's define $\mathbf{X} \in \mathbb{R}^{d \times T}$ is a d-dimensional sequence embedding input of one query or document with length of T, x_i is the ith token in the sequence. Q,K and V are matrices initiated by $\mathbf{X}$ multiplying different weight matrices. The attention score matrix in such a sequence is denoted by $\mathbb{A} \in \mathbb{R}^{T \times T}$. For a token pair x_i, x_j, q_i and k_j are the column selection from Q and K according to i or j, its attention score A_{ij} is calculated in Scaled Dot-Product Attention as $A_{ij} = \frac{q_i \cdot k_j^T}{\sqrt{d}}$. In (31), they claim the attention unit is already a weighted sum of values, where the weight assigned to each value is learned from q_i and k_j. Whereas, in information retrieval area, it is well equipped with prior knowledge to represent the weight of one word. We enrich the attention calculation with these techniques. Assuming $w_i \in \mathbb{R}^T$ represents the importance weight of the ith token the sequence, w_i is a **non-trainable scalar**. In this paper we use BM25 to represent this importance. Its detailed generation of query and document will be introduced in Section 3.3. A new weighted attention score A_{ij}^w is computed as

$$A_{ij}^w = w_j \frac{q_i \cdot k_j^T}{\sqrt{d}}, A_{ji}^w = w_i \frac{q_j \cdot k_i^T}{\sqrt{d}} \tag{1}$$

q and k share the same shape only differ in random initialization. Symmetrically, the weighted attention score of A_{ji} can be represented in the right part of Eq. 1.

The right part of Figure 1 presents how Weighted Self-Attention works. With importing the weight information and packing all w_i into W, we define **Weighted Self-Attention** as

$$\boxed{\mathrm{WeightedSelfAttention}(Q, K, W, V) = \mathrm{softmax}(W \odot \frac{QK^T}{\sqrt{d}})V} \tag{2}$$

where W is one dimension vector and its multiplicand is a matrix . $\odot$ represents a Hadamard product by repeating W to perform element-wise multiplication. BISON Encoder picks this Weighted Self-Attention as its block unit. It is also built upon multi-head structure by concatenating several Weighted Self-Attention instances. With re-scaling by W^o, we can get a Complex Weighted Self-Attention (CWSA). A fully connected Feed-Forward network is then followed as the other sub-layer. In both sub-layers, layer normalization(2) and residual connection(9) are employed to facilitate the robustness of BISON Encoder.

$$\begin{aligned} &\mathrm{CWSA} = \mathrm{Concat}(\mathrm{WeightedSelfAttention_1}, ..., \mathrm{WeightedSelfAttention_n})W^o \\ &\mathrm{CWSA_{out}} = \mathrm{LayerNorm}(\mathrm{CWSA} + \mathrm{X}) \\ &\mathrm{BISONEncoder} = \mathrm{LayerNorm}(\mathrm{CWSA_{out}} + \mathrm{FeedForward}(\mathrm{CWSA_{out}})) \end{aligned} \tag{3}$$

3.3 BM25 weight generation

A key point of BISON is to find an appropriate way to represent word weight. As mentioned in Section 1, BM25 and its variants show the superiority in weight representation for document ranking tasks against other alternatives. We leverage BM25 to generate the weight scores for a query and BM25F(28) to compute the weight scores for a multi-fields document. BM25F is a modification of BM25 in which the document is considered to be composed from several fields with different degrees of importance in term of relevance saturation and length normalization. Both BM25 and BM25F depend on tf and idf, tf means TermFrequency, it describes the number of occurrences of the word in the field. While idf (InverseDocFrequency) is a measure of how much information the word provides, i.e., if it's common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word. For word i, $idf_i = \log \frac{N - df_i + 0.5}{df_i + 0.5}$, where N is a scalar [3] indicting how many documents we are serving in system and df is the number of documents where the word i appears.

Inherent Query BM25 The calculation of classic BM25 is based on tf in document. Since in BISON query and document are encoded separately, here we compute an inherent query BM25 by computing tf **within query** instead. An inherent BM25 term weight for query word i can be re-calculated as

$$\boxed{w_i^{BM25} = idf_i \frac{tf_i}{tf_i + k_1(1 - b + b\frac{l_q}{avl_q})}} \tag{4}$$

[3] Here we set it by 100,000,000

where tf_i is the term frequency of $word_i$ within query; l_q is the query length; avl_q is the query average length along the collection; k_1 is a free parameter usually chosen as 2 and 0<=b<=1 (commonly used is 0.75).

Inherent Document BM25F In BM25F, instead of using tf directly, empirically atf (AdjustedTermFrequency) is widely adopted. It is proposed by adding several field-wise factors. For a $word_j$ in document field c, its atf_j^c is defined in Eq. 5

$$atf_j^c = \frac{fw_c \cdot tf_j^c}{1.0 + fln_c \cdot (\frac{fl_c}{avl_c} - 1.0)} \tag{5}$$

where fw_c is the weight of field c; fln_c is the normalized field length for field c; tf_j^c is the term frequency of $word_j$ within field c; fl_c is the original field length of field c; avl_c is the average length for field c.

$$\boxed{w_j^{BM25F} = idf_j \frac{atf_j}{k_1 + atf_j}} \tag{6}$$

And its corresponding inherent BM25F score is computed in Eq. 6, where the calculation of idf_j is the same with idf_i.

3.4 Whole word weight sharing

Sub-word based approach has been proved efficient to alleviate out of vocabulary issue and limit vocabulary size. BERT uses WordPiece(32) to produce tokens from original raw text. One shortcoming of this methodology is that we cannot directly apply the word-level prior knowledge. Moreover, in some NLP tasks, token-level weight is not enough to distinguish the importance of different words. The latest BERT model has proved that upgrading the Mask Language Model task to whole word level[4] improves performance. In our task, the W used for attention score is also based on whole word weights. That is, we first collect and calculate weights in whole word level, then give the same word weight to tokens corresponding to one word. By this way, one WordPiece token may has different weight representation if it occupies in different words. We also conduct experiment in Section 5.1 to compare the effect of token-level weight generation and word-level. The results suggest the word-level manner is superior than token-level.

3.5 Combined fields representation

In ad-hoc retrieval tasks, there are always multiple sources of textual description (*fields*) corresponding to one document. Lots of studies (36; 27) reveal that different fields contain complementary information. Thus, to obtain a more comprehensive understanding of document, when encoding the document into semantic vector space, we need to take multiple fields into consideration.

The well-known fields for a document in web search are title, header, keyword, body, and the URL itself etc. These fields are primitive from the website and can be fetched from HTML tags. Another kind of fields, like anchor, leverage the description from the brother website. Via this way, we can infer with useful information from other documents. In addition to this, click signal is also with high quality and can be easily parsed from the search log. When a user clicked on the document *d* with a query *q*, we will add *q* to the clicked query field of *d*.

For performance consideration, we only pick *anchor*,*title*,*URL*,*clicked query* fields to do the document embedding. Body is not taken into consideration because body is pretty longer than others and it is hard to encode such long text into one unified space, which might diverge the representation. The special properties of these document fields make it difficult to unify them into one semantic space. One common approach (36) is to separately encode the multiple fields respectively and learn a joint loss across these fields.

We translate the segment definition of pre-next sentence in BERT to different fields in document by mapping multiple fields into different segments. Every segment has a max length constrain, we set it to 20 tokens for *anchor*, *URL* and *title* fields. For *clicked query* fields, since a popular document may exist a large magnitude of click instances, we only pick the top 5 clicked queries for one document with a max length of 68 tokens. For all these segment representation we pad it according to the need.

[4] https://github.com/google-research/bert

To obtain an unified document embedding, a [CLS] token is added at the beginning of the combined fields, and a [SEP] token is also inserted between each segment.

3.6 Optimization

We can achieve a sequence of semantic embeddings after BISON Encoder. Inspired by (7), using the embedding of [CLS] in the last layer as the matching features is already good enough. Nogueira et al.(21) also proves that in passage ranking task, adding more components upon Transformer does not help too much(25). Therefore, BISON uses the embedding of [CLS] as semantic representation for query and document, the matching score s is measured by cosine similarity on query and document vectors.

$$s = \cos(\mathrm{BISON}(\mathrm{query})_{\mathrm{cls}}^{\mathrm{last}}, \mathrm{BISON}(\mathrm{document})_{\mathrm{cls}}^{\mathrm{last}}) \tag{7}$$

We adopt a binary cross entropy loss to optimize the model, which determines whether a query-document is relevant or not. We also tried pair-wise loss and found it had no extra improvement. Prior works (25; 21) also confirm on this.

$$Loss = -y\log(\delta(\mathrm{w}\cdot\mathrm{s}+\mathrm{b})) - (1-\mathrm{y})\log(1-\delta(\mathrm{w}\cdot\mathrm{s}+\mathrm{b})) \tag{8}$$

where y is the label denoting if query-document is relevant, δ represents Sigmoid function. w and b are used to generate weighted cosine similarity to fit the Sigmoid function.

4 Experimentation

In this section, we evaluate the quality of BISON both on Bing's internal query set and public datasets. Taking efficiency and scalability into consideration, we build 3-layer BISON encoders both in query and document sides. Adam optimizer(14) is employed to train our model. The learning rate we used is 8e-5. We set the batch size to 300. Other hyber-parameters are the same with BERT. Our evaluation metrics are NDCG(Normalized Discounted Cumulative Gain)(12), NCG(Normalized Cumulative Gain) and MRR(Mean Reciprocal Rank). Detailed training settings and metrics calculation equations are in Appendix.

4.1 Data preparation

Similar with (18; 20; 11) we sample 30 million query-document pairs from Bing's search log. Each training instance combines a tuple of query q and its clicked multi-fields document d. since these pairs come from real user behavior, we treat these instances as positive labeling. For the negatives, we use a mixture random sampling approach.

NCE negative sampling Directly random picking a negative case is too easy for the model to learn, which weakens the model's generalization. Instead, we use the noise-contrastive estimation (NCE) to pick competitive negatives(19; 37). It always picks negatives within current training batch with the same size of positives.

Hard negative integration The negatives from NCE sampling are all clicked documents, which only helps the model to learn entire non-related query-document pairs. To facilitate model with the capability to distinguish partial-related query-document pairs, we incorporate more difficult negatives by sampling 50 thousand queries from the search log and then sending these queries to the production system to retrieve 10 million partial-related query-document pairs as the hard negatives for these queries. These cases are added as companions of NCE negatives.

4.2 Evaluation

Baselines As outlined in Section 2, our baselines contain classic information retrieval matching methods, typical deep learning models for sentence encoding and advanced pre-train language models. Specifically, given their deserved reputations in information retrieval history, we choose TFIDF and BM25 as representatives of the classic methods. Many primitive deep learning studies explored how to encode sentences into embeddings. Among them, the Universal Sentence Encoder(5) and C-DSSM(29) are widely recognised to be more efficient and accurate. So we include these two models into our baselines. When stepping into language model pre-training boom, as illustrated in

Section 2.1, BERT-based model has dominated the document ranking tasks. So we adopt BERT and XL-Net(35) as the remaining baselines. To make a fair comparison, All of these competitor models are trained following best practises suggested in previous works. For BERT and XL-Net, we also apply a 3-layer setting in both query and document, and they are both fine tuned from public pre-train models. All these baseline models are evaluated strictly the same with BISON by averaging the results collected from 5 times training.

Table 1: Evaluation results. For XL-Net and BERT, we fine-tune them by initializing parameters with first 3 layer of released model from Google.

Model	**Intrinsic Query Set**				**MS Marco**	
	NCG@20	NDCG@1	NDCG@10	NDCG@20	MRR@10	MRR@20
TF-IDF	0.4561	0.1828	0.3062	0.3308	0.1835	0.1917
BM25	0.4889	0.2061	0.3472	0.3687	0.2068	0.2141
USE	0.2335	0.0860	0.1171	0.1333	0.0627	0.0648
C-DSSM	0.4254	0.1900	0.3113	0.3272	0.1461	0.1506
XL-Net	0.5316	0.2528	0.4017	0.4210	0.2597	0.2659
BERT	0.6550	0.3346	0.5154	0.5351	0.2624	0.2677
BISON	**0.6827**	**0.3361**	**0.5243**	**0.5473**	**0.2706**	**0.2762**

Intrinsic evaluations The intrinsic evaluations are performed in a common used manner(1) to evaluate the quality of semantic embedding representation. We pick 1.4k representative queries along with the corresponding 7 million query-document pairs from Bing's search log as the test set. For the deep learning models, the documents are ranked by the cosine similarity score. Each query-document is human labelled with five standard categories: Perfect,Excellent,Good,Fair and Bad. On one hand, we use NDCG computed at positions one, ten and twenty for precision measurement, on the other hand, we leverage NCG at twenty for recall evaluation. NCG cares more about document recall quality without considering the ranking positions. All performance numbers are averaged over queries for each run. As shown in left part of Table 1, USE has the worst performances since it only performs better on homogeneous data and query-document are obviously heterogeneous. BISON significantly outperforms all baselines across all metrics even with BERT and XL-Net. Typically, with increasing recall count from 1 to 10, 20, the NDCG gain gradually grows.

Evaluation on MS Marco The document full ranking task in MS Marco is similar with our scenario as the document contains multi fields with title, url and body. We use the same setting with internal evaluation set (Only the title and url fields are used to form a multi-fields document). In this task, training data contains 6 million query-document pairs with a simple binary positive/negative labeling while evaluation set includes 5k queries and 3 million documents. For each query, top 1000 documents with the highest similarity scores are returned. Following official guidance, MRR@10 and 20 are used as the performance metrics. MRR first calculates per query's reciprocal of the rank at which the first relevant document is retrieved, then averages them across queries. We can see from the right part of Table 1 that BISON also shows the best results. This demonstrates that BISON not only addresses the real encoding problem for industrial search engine with additional high quality field (clicked query), but also extends to be a general solver for ordinary document ranking task in information retrieval community.

5 Analysis

5.1 Ablation study

As aforementioned, three key components empower the embedding quality of BISON. To further investigate the individual contribution of each part, we carefully design the ablation study. Specifically, we examine three BISON variants and compare the NCG and NDCG results with BERT and BISON. Each variation disables a component while keep others unchanged.

- BISON$_{tw}$: We replace the whole word weight sharing by **t**oken level **w**eight generation, which generates both query and document BM25 score on WordPiece token level.
- BISON$_{us}$: We exclude the combined fields representation from document encoding. Instead, we use an **u**nion **s**egment representation by simply concatenating all fields as one segment.
- BISON$_{idf}$: To prove BM25 is the best prior source for weight estimation, we exploit a variant using **i**nverse **d**ocument **f**requency as weight source.

Table 2: Performance of BISON variants with intrinsic evaluation

Variant	NCG@20	NDCG@1	NDCG@10	NDCG@20
BISON$_{tw}$	0.6557	0.3245	0.5146	0.5387
BISON$_{us}$	0.6692	0.3287	0.5198	0.5374
BISON$_{idf}$	0.6670	0.3266	0.5152	0.5382
BISON	**0.6827**	**0.3361**	**0.5243**	**0.5473**

Following the settings from Section 4.2, we train these three BISON variants with the best efforts. Take the results of BERT and BISON in left part of Table 1 as baselines, results in Table 2 demonstrate that the absence of any component will inevitably jeopardize the performance of BISON. Specifically, by disabling the word level weight sharing, there is nearly no improvement between BISON$_{tw}$' and BERT, indicting computing BM25 on sub-word token level is not feasible. That is the reason why traditional search engine always use BM25 score on natural word level. BISON achieves 0.0277 improvement on NCG compared with BERT while BISON$_{us}$ can only achieve half of that. Therefore, the combined fields representation for multi-field document is a vital factor for document representation learning task. The result of BISON$_{idf}$ is still lower than BISON, it is explainable as IDF is only counted on global documents without distinction across single document instance.

5.2 Efficiency analysis

Model complexity One of the advantages for BISON is that it does not involve any new trainable parameters. Thus it retains the same model complexity with BERT. The only extra work is to generate weight scores for each token which can be well prepared before model training and inference.

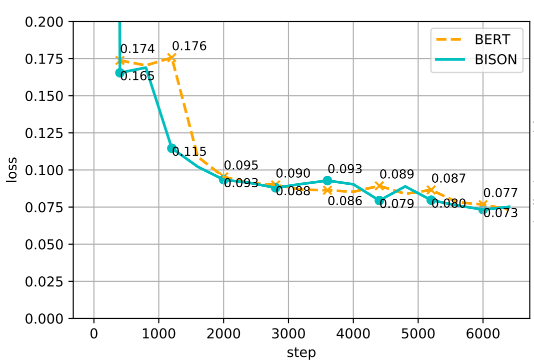

Figure 2: Training loss

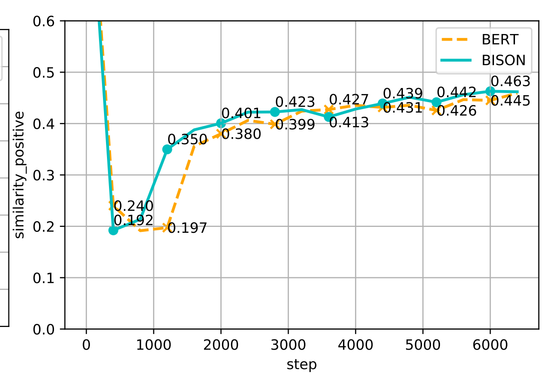

Figure 3: Weighted cosine similarity

Training efficiency People always expect their models to be trained as fast as possible to reduce the training cost. We plot the training loss and weighted cosine similarity (explained in Section 3.6) trend curve of BISON and BERT based on the training experience from MS Marco. According to Figure 2, In the first 2k steps the training loss of BISON decreases significantly faster than BERT. This indicates that BISON is more easier to converge on training data, which could be a big saving when we train document representation on a large scale data set. Practically, we always expect the weighted cosine similarity to be more differentiated to prevent the overlap across different query-document pairs. Thus a large value of weighted cosine similarity is preferred. Figure 3 shows that BISON is superior in enlarging the weighted cosine similarity range rapidly.

6 Conclusion

We present BISON, a general framework for multi-fields document search. It learns semantic representation for both query and multi-fields document by integrating BM25 into attention score calculation. It can also handle the discrepancy between natural words and WordPiece tokens with a whole word weight sharing mechanism. Moreover, a combined fields representation is proposed to reduce the multi-fields document encoding to a unified vector. Extensive experiments demonstrate BISON outperforms other frameworks on various document retrieval metrics.

Broader Impact

Our work has the following potential positive impacts to society: Firstly, our framework takes the first and critical step in combining classic feature-based search and semantic search to help real search engine improve document retrieval quality. It leverages prior knowledge into semantic representation in a data-driven way which avoids human sense bias in integration. Furthermore, Given that self-attention has been widely used in various applications like natural language processing, recommender systems, machine translation, etc. Although our work focuses on document recall scenario, it actually shows not only the feasibility, but also the potential to extensively apply weighted self-attention into broader machine learning tasks and thus bring benefits to other fields. At the same time, Some of the prior rule-based knowledge comes from human sense, which should be carefully integrated into deep learning models, as people's prejudgement diverges a lot.

References

[1] Placing search in context: The concept revisited. *ACM Trans. Inf. Syst.*, 20(1):116–131, January 2002.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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

[4] Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. Clueweb09 data set, 2009.

[5] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[6] Zhuyun Dai and Jamie Callan. Context-aware sentence/passage term importance estimation for first stage retrieval. *arXiv preprint arXiv:1910.10687*, 2019.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, June 2019. Association for Computational Linguistics.

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

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2042–2050. Curran Associates, Inc., 2014.

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

[12] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, October 2002.

[13] Jaeyoung Kim, Mostafa El-Khamy, and Jungwon Lee. T-gsa: Transformer with gaussian-weighted self-attention for speech enhancement. *ArXiv*, abs/1910.06762, 2019.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[16] Zhengdong Lu and Hang Li. A deep architecture for matching short texts. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1367–1375. Curran Associates, Inc., 2013.

[17] B. Mitra and N. Craswell. *An Introduction to Neural Information Retrieval*. 2018.

[18] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 1291–1299, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[19] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 2265–2273, Red Hook, NY, USA, 2013. Curran Associates Inc.

[20] Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. Improving document ranking with dual word embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, page 83–84, Republic and Canton of Geneva, CHE, 2016. International World Wide Web Conferences Steering Committee.

[21] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

[22] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*, 2019.

[23] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. Text matching as image recognition. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2793–2799. AAAI Press, 2016.

[24] Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Mike Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. Tf-ranking: Scalable tensorflow library for learning-to-rank. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2970–2978, 2019.

[25] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. Understanding the behaviors of bert in ranking. *arXiv preprint arXiv:1904.07531*, 2019.

[26] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

[27] Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Simple bm25 extension to multiple weighted fields. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, page 42–49, New York, NY, USA, 2004. Association for Computing Machinery.

[28] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at trec-3. In *TREC*, 1994.

[29] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, page 373–374, New York, NY, USA, 2014. Association for Computing Machinery.

[30] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*, 2019.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[32] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[33] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*, 2019.

[34] Wei Yang, Haotian Zhang, and Jimmy Lin. Simple applications of bert for ad hoc document retrieval. *arXiv preprint arXiv:1903.10972*, 2019.

[35] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 2019.

[36] Hamed Zamani, Bhaskar Mitra, Xia Song, Nick Craswell, and Saurabh Tiwary. Neural ranking models with multiple document fields. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, page 700–708, New York, NY, USA, 2018. Association for Computing Machinery.

[37] Hongfei Zhang, Xia Song, Chenyan Xiong, Corby Rosset, Paul Bennett, Nick Craswell, and Saurabh Tiwary. Generic intent representation in web search. In *The 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2019)*. ACM, July 2019.

[38] Shuailiang Zhang, Hai Zhao, Yuwei Wu, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. Dual co-matching network for multi-choice reading comprehension. *arXiv preprint arXiv:1901.09381*, 2019.

A Evaluation metrics

A.1 NDCG

Normalized Discounted Cumulative Gain (NDCG) is a widely-accepted measure of ranking quality. To understand NDCG, we need to understand its predecessors: Cumulative Gain (CG) and Discounted Cumulative Gain (DCG). Every query-document pair has a relevance score associated with it. Cumulative Gain is the sum of all the relevance scores in the ranking set.

$$\text{CumulativeGain(CG)} = \sum_{i=1}^{n} relevance_i \tag{9}$$

There is a drawback with Cumulative Gain, which is that it doesnot take position into consideration. DCG fills this gap. The computation involves discounting the relevance score by dividing it with the log of the corresponding position.

$$\text{DiscountedCumulativeGain(DCG)} = \sum_{i=1}^{n} \frac{relevance_i}{log_2(i+1)} \tag{10}$$

DCG seems a good measure at first as it takes position significance into account. However, it is still not complete. Depending on various factors, the number of documents served may vary for every query. Thus, the DCG will vary accordingly. We need a score which has a proper upper and lower bounds so that we can take a mean across all the recommendations score to report a final score. NDCG brings in this normalization. NDCG is then the ratio of DCG of recommended order to DCG of ideal order.

$$\text{NDCG} = \frac{DCG}{iDCG} \tag{11}$$

A.2 NCG

NDCG is a perfect metric for evaluating precise ranking results. However, in the first stage of information retrieval, given there will be a precise model to re-rank in the second stage, we care more about how many positive documents are retrieved within Top N without considering their positions. Hence, NCG is a good choice to mesure the recall quality.

Similar with the normalization in NDCG, NCG is computed as

$$\text{NCG} = \frac{CG}{iCG} \tag{12}$$

A.3 MRR

NDCG/NCG work well when query-document has multiple class positive labels, for those only have single class positive labeling, Mean Reciprocal Rank (MRR) is another choice to evaluate a model's performance with returning a ranked list of documents to queries. For a single query, the Reciprocal Rank is $\frac{1}{rank}$ where **rank** is the position of the highest-ranked answer (1,2,3,..., N for N document returned in a query). If no correct answer was returned in the query, then the reciprocal rank is 0.

For multiple queries Q, the Mean Reciprocal Rank is the mean of the Q reciprocal ranks. The official MRR measure code for MS-MARCO dataset could be found in here[5].

$$\mathrm{MRR} = \frac{1}{Q} \sum_{i=1}^{Q} \frac{1}{\mathrm{rank}_i} \tag{13}$$

B Training details

B.1 Training with Bing's internal data

We train BISON with 8 Tesla V100 GPU, 32 GB memory for each. To best accelerate training, We implement a data parallel training pipeline base on horovod distribute training. What's more, Automatic Mixed Precision is enabled to train with half precision while maintaining the network accuracy. Finally, the training batch size is 500 query-document pairs. We iterate the training by 10 epoches and it takes 5 hours for each epoch.

B.2 Training with MS-MARCO document ranking dataset

The MS-MARCO document ranking dataset[6] has 367,013 queries and the corpus is 3.2 million documents, all binary labels are human-generated. We use the dev set to be our test set, which contains 5,193 queries.

- Training data generation. For every single query, we over-sampled the positive pairs by 10 times and selected the hardest 10 negative pairs from top100 training dataset with initial ranking to generate the 6 million training data. For every single document, we only introduce url and title into training. Raw url in corpus splited by punctuation marks and removed digit firstly, open sourced word break tool - wordninja[7] was applied to slice the munged together words finally. Evaluation data generation follows the same logic of training data generation.
- IDF map file generation. In order to align with the whole dataset, we generate the word-level idf map file from the 3.2M corpus(without body stream) directly by feature extraction module of sklearn rather than reuse the idf map file from Bing index. For any word not in the map file, the default idf value would be 15.3, the maximum idf value represents the most rare word. For [SEP], [CLS] and all punctuation marks, the idf values are set to 1, the minimum idf value represents the most common word.
- Training details. The max token length settings are 20 for query, 30 for url, 30 for title. BISON trained on 8 16GB Tesla V100 GPUs, with 512 training batch size and 5 epochs, learning rate was 8e-5. Horovod distribute training and Automatic Mixed Precision were enabled to accelerate the training too.

[5] `https://github.com/microsoft/MSMARCO-Passage-Ranking/blob/master/ms_marco_eval.py`

[6] `https://microsoft.github.io/TREC-2019-Deep-Learning/`

[7] `https://github.com/keredson/wordninja`