Multi-Aspect Dense Retrieval

Weize Kong
weize@google.com
Google

Swaraj Khadanga
khadanga@google.com
Google

Cheng Li
chgli@google.com
Google

Shaleen Kumar Gupta
shaleeng@google.com
Google

Mingyang Zhang
mingyang@google.com
Google

Wensong Xu
asong@google.com
Google

Michael Bendersky
bemike@google.com
Google

ABSTRACT

Prior work in Dense Retrieval usually encodes queries and documents using single-vector representations (also called embeddings) and performs retrieval in the embedding space using approximate nearest neighbor search. This paradigm enables efficient semantic retrieval. However, the single-vector representations can be ineffective at capturing different aspects of the queries and documents in relevance matching, especially for some vertical domains. For example, in e-commerce search, these aspects could be *category*, *brand* and *color*. Given a query "white nike socks", a Dense Retrieval model may mistakenly retrieve some "white *adidas* socks" while missing out the intended brand. We propose to explicitly represent multiple aspects using one embedding per aspect. We introduce an aspect prediction task to teach the model to capture aspect information with particular aspect embeddings. We design a lightweight network to fuse the aspect embeddings for representing queries and documents. Our evaluation using an e-commerce dataset shows impressive improvements over strong Dense Retrieval baselines. We also discover that the proposed aspect embeddings can enhance the interpretability of Dense Retrieval models as a byproduct.

CCS CONCEPTS

• **Information systems** → **Document representation**; **Query representation**; **Retrieval models and ranking**.

KEYWORDS

Dense Retrieval; Multi-Aspect; Multi-Task Learning

ACM Reference Format:
Weize Kong, Swaraj Khadanga, Cheng Li, Shaleen Kumar Gupta, Mingyang Zhang, Wensong Xu, and Michael Bendersky. 2022. Multi-Aspect Dense Retrieval. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3534678.3539137

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08.
https://doi.org/10.1145/3534678.3539137

1 INTRODUCTION

Instead of relying on lexical-based representations as in keyword search, Dense Retrieval represents queries and documents as dense vectors, also called embeddings. It usually employs a bi-encoder design [15, 22, 28] to encode a query and a document *independently* and then computes a relevance score using some similarity functions, e.g., cosine, between the query embedding and document embedding. With fast approximate nearest neighbor search [2, 11], this paradigm enables efficient semantic retrieval as one can precompute document embeddings and index them offline [22].

One critical limitation of Dense Retrieval is that the *single-vector* embeddings can be inadequate at capturing different aspects of the query and the document for relevance matching [23], especially in some vertical domains. For example, in e-commerce search, these aspects could be *category*, *brand* and *color*. Given a query such as "white nike socks", a Dense Retrieval model may mistakenly retrieve some "white nike *shoes*" or "white *adidas* socks", while missing out the intended category or brand. Similarly, in people search [14, 27], Dense Retrieval models may fail to capture the intended *location*, *company* or *education* aspect of a person.

To address this limitation, we propose to explicitly represent multiple aspects for both queries and documents, using one embedding per aspect. To implement this idea, we design a Multi-Aspect Dense Retrieval model (MADRM), illustrated in Figure 1.

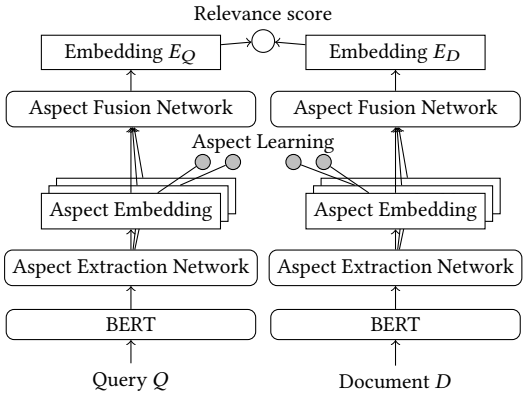

Figure 1: Multi-Aspect Dense Retrieval model (MADRM).

MADRM contains three major innovations. First, we design an Aspect Extraction Network based on Attention [32] to extract embeddings for multiple aspects, called aspect embeddings. Second, we introduce an aspect prediction task for training the aspect embeddings. We call this Aspect Learning. This is a critical step which not only teaches the model domain knowledge but also guides the model to represent each aspect (e.g., *category* or *brand*) with a particular aspect embedding. Lastly, we design lightweight Aspect Fusion Networks to combine multiple aspect embeddings together as the final query or document embedding for fast approximate nearest neighbor search (ANNS). This helps to save index space and reduce indexing/querying complexity in ANNS.

MADRM provides several advantages over a regular bi-encoder Dense Retrieval model. First, MADRM can more effectively capture multiple aspects for relevance prediction and improve retrieval quality, as shown by our experiments and case studies on an e-commerce dataset (Section 9). Second, with Aspect Learning, MADRM also learns domain knowledge in addition to general language semantics. For example, it could effectively recognize *category*, *brand* and *color* for the e-commence domain (Table 5). Last, but equally importantly, since we explicitly represent aspects for queries and documents, our model offers enhanced interpretability as demonstrated by our case study (Section 9.4).

In addition, the aforementioned improvements do not come at the cost of indexing, querying and model complexity. For indexing/querying complexity, since we fuse the multiple aspect embeddings as the final query/document embedding, MADRM does not add additional cost compared with other regular bi-encoder models. In contrast, the prior work [20, 30] keeps embeddings for each query and document tokens. As a result, its ANNS index size grows linearly with the text sequence length. For model complexity, compared to the BERT based bi-encoder baseline (Section 8.2), our MADRM implementation uses *less* parameters and compute in serving (Section 7.2). This is achieved by implementing MADRM with one-less Transformer layer [32] than the baseline, and then stacking the Aspect Extraction Network and Aspect Fusion Network as the last layers.

We summarize our research contributions as follows:

- We propose the Multi-Aspect Dense Retrieval model (MADRM) – a novel Dense Retrieval model that explicitly represents different aspects of a query and a document using multiple embeddings.
- We design the Aspect Extract Network and the Aspect Fusion Networks for MADRM to effectively extract and fuse aspect embeddings.
- We introduce the aspect prediction task for MADRM. It not only teaches the model domain knowledge, improves the retrieval quality via multi-task learning, but also guides the model to represent each aspect with a particular aspect embedding.
- We conduct experiments on an e-commence dataset to test the effectiveness of MADRM as well as each proposed component. The results show impressive improvements over strong Dense Retrieval and multi-task learning baselines. We perform a case study to understand our model behavior, which also demonstrates the interpretability of MADRM.

2 RELATED WORK

Neural information retrieval [24] is concerned with utilizing deep neural networks for retrieving and ranking an ordered list of documents from a corpus given a query. The combination of neural networks, especially Transformers [32], and self-supervised pretraining methods, has produced positive results in many domains, e.g., e-commerce [35, 36] and social networks [17]. Among Transformer based models, BERT [9] is a popular choice [22, 28]. Most applications of BERT for neural information retrieval follow a retrieve-and-rerank setup. Specifically in the retrieval stage, candidate documents are retrieved from the corpus using keyword search based on a scoring function like BM25 [29]. In the reranking stage, the BERT model is used to compute the relevance of the given query and the retrieved documents. Examples that follow such procedure are the monoBERT model and the duoBERT model [25].

Recently, Dense Retrieval, the focus of this work, has become an emerging area of research. Instead of relying on the traditional keyword search, queries and documents are encoded as densified embeddings, or vectors of fixed-width, and retrieval is performed in the embedding space using nearest neighbor search of embedding vectors [22]. A bi-encoder design is usually employed [16] to encode a query and a document independently, which reduces retrieval latency by precomputing document embeddings and indexing them offline. The DSSM model [15] is one of the earliest models for Dense Retrieval. It uses two deep fully-connected networks as encoders. More sophisticated encoders have been explored, e.g., CNN [13], RNN [26] and BERT [4]. Bi-encoders usually underperform cross-encoders, which consider the interactions between queries and documents and thus is not scalable for retrieval. A common strategy to improve bi-encoders is to distill a cross-encoder into a bi-encoder [4]. A body of work has investigated how to mine hard negative examples to make the bi-encoders more robust [10, 19, 33].

Some studies focus on improving the design of bi-encoders for better retrieval performance. Our work falls into this direction. The work most relevant to us considers employing multiple vectors to represent queries or documents [20, 23]. The vectors of a document can interact with the vectors of a query after retrieval, improving the expressiveness of bi-encoders [23]. These methods utilize multi-vector representations without any explicit assumptions about the semantics each vector could capture. In contrast, our work explicitly considers the existence of the multiple aspects in relevance matching. In addition, our proposed method only outputs a single embedding for each document using our lightweight Aspect Fusion Networks. This significantly saves the space to index documents – for comparison, the index size grows linearly with the document length in the previous work [20, 30].

Another line of related research is multi-task learning, which aims to leverage information contained in multiple related tasks to help the model generalize better. This paradigm is widely applied to various domains like computer vision [37], bioinformatics [34] and natural language processing [7]. Multi-task learning is relatively less explored in information retrieval. In web search, different markets or countries are treated as separate tasks and gradient boosting based methods have been proposed to jointly learn these tasks [3, 5]. Zhang et al. [36] learn the tasks of query-title similarity and query

taxonomy classification simultaneously for e-commerce search, which is included as a baseline in our experiments.

Since our model explicitly considers the aspects present in the documents and queries, it could enhance the interpretability of the model as a byproduct. A few retrieval models in the literature have been designed to enhance the interpretability as well. Hofstätter et al. [12] compute soft-match counts over the contextualized embeddings of each query-document token pair, which provides insights into model decisions. Leonhardt et al. [21] predict query-document relevance based on the selected sentences from a document, which could be treated as an explanation. The EXS system in [31] employs a post-hoc explanation method by training a simple explanation model to approximate the ranking model.

3 OVERVIEW

We first provide an overview of our Multi-Aspect Dense Retrieval Model (**MADRM**). We illustrate MADRM's architecture in Figure 1. As a Dense Retrieval model, MADRM takes in a query Q and a document D as input, and outputs a score for relevance matching. The query $Q = (q_1, ..., q_{|Q|})$ and document $D = (d_1, ..., d_{|D|})$ usually contain a sequence of tokens as input features. The relevance score is computed by first independently encoding Q and D as dense representations, also called embeddings. More formally, we denote the query embedding as $E_Q \in \mathbb{R}^H$ and document embedding as $E_D \in \mathbb{R}^H$, where H is the embedding dimension or hidden size. The relevance score is then computed as the dot product or cosine similarity between E_Q and E_D. We list the frequently used notations in Table 1.

Table 1: Frequently used notations.

Notation	Description
Q, D	The query, document in the retrieval task
E_Q, E_D	The query, document embedding
$A = \{a_i\}$	A set of aspects, e.g., {*category, brand, color*}
a_i or a	A particular aspect. We omit the subscript i when it is clear from the context, i.e., a
E_{a_i} or E_a	The aspect embedding for either query's or document's aspect a.
$\mathtt{Q}, \mathtt{K}, \mathtt{V}$	The query, key, value input to the `Attention` layer
$W^{\mathtt{Q}}, W^{\mathtt{K}}, W^{\mathtt{V}}$	The projection parameters in the `Attention` layer
$\mathcal{V}_a$	The aspect vocabulary for aspect a, e.g., $\mathcal{V}_{color}$ = {`black`, `white`, ...}
$\mathcal{A}_a$	The set of aspect value annotations for aspect a, e.g., {`Sandals`, `Shoes`}
U_a	The aspect embedding table for aspect a
w_a	The aspect weight for aspect a

As illustrated in Figure 1, to capture different aspects, MADRM encodes Q as follows: it first passes Q through a BERT encoder [9]. Other Transformer encoders [32] are also applicable here; from BERT's sequence output, MADRM then uses an Aspect Extraction Network (Section 4) to extract embeddings for each aspect, called "aspect embeddings"; the aspect embeddings are then trained using available aspect annotations. This process is called Aspect Learning (Section 5); finally, MADRM combines the multiple aspect embeddings together as the query embedding E_Q using an Aspect Fusion Network (Section 6). MADRM encodes D in the same way as Q, except that the two encoders don't have to share the same network parameters. Thus, we only describe our model structures using the query encoder as an example in Section 4, 5 and 6.

4 ASPECT EXTRACTION NETWORK (AEN)

We illustrate the Aspect Extraction Network (AEN) in Figure 2. We describe AEN's structure in this section and defer the description of the Aspect Learning process to Section 5.

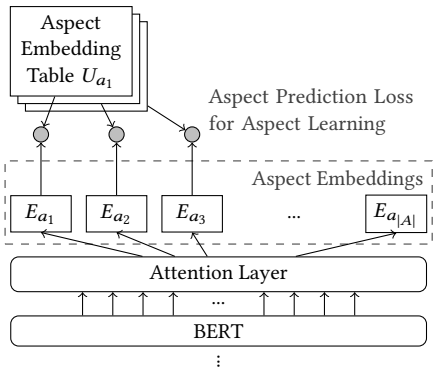

Figure 2: Aspect Extraction Network (AEN). Section 4 and Section 5 respectively describe the network structure and its Aspect Learning process.

AEN takes in BERT's sequence output, $\phi_{BERT}(Q) \in \mathbb{R}^{|Q| \times H}$ and outputs embeddings for each aspects, called **aspect embeddings**. We denote the set of aspects as $A = \{a_i\}_{i=1}^{|A|}$ and the corresponding aspect embeddings as $E_{a_i}(Q) \in \mathbb{R}^H$ for $i = 1, ..., |A|$. For brevity, when it is clear from the context, we omit the subscript i in a_i as well as the query input Q. For example, we use E_a for $E_{a_i}(Q)$ and ϕ_{BERT} for $\phi_{BERT}(Q)$. To facilitate derivation, we pack E_a for all $a \in A$ as a matrix and denote it as $E_A \in \mathbb{R}^{|A| \times H}$.

We compute E_A using Attention as follows,

$$E_A = \mathtt{Attention}(\mathtt{Q}W^{\mathtt{Q}}, \phi_{BERT}W^{\mathtt{K}}, \phi_{BERT}W^{\mathtt{V}}), \tag{1}$$

$$\mathtt{Attention}(\mathtt{Q}, \mathtt{K}, \mathtt{V}) = \mathtt{softmax}(\frac{\mathtt{Q}\mathtt{K}^T}{\sqrt{H}})\mathtt{V}. \tag{2}$$

`Attention` in Equation 2 is as defined in the Transformer paper [32], where H is our hidden size or embedding dimensions. In Equation 1, ϕ_{BERT} is the BERT sequence output as mentioned above, $W^{\mathtt{Q}}, W^{\mathtt{K}}, W^{\mathtt{V}} \in \mathbb{R}^{H \times H}$ are the projection parameters. $\mathtt{Q} \in \mathbb{R}^{|A| \times H}$ is our query[1] used in the Attention layer. Different from the Attention query used in BERT [9] which is the previous layer's sequence output, our Attention query $\mathtt{Q}$ is an input-independent trainable parameter. This enables the model to attend to different parts of the input sequence for representing each aspect. We illustrate this idea in Table 2, which shows the attention scores predicted by our model over query "ugg sandals" for aspect *category* and *brand*. The model can correctly attend to *sandals* for extracting the *category* aspect embedding and attend to *ugg* for the *brand* aspect embedding.

[1]Note the distinction between the Attention query $\mathtt{Q}$ and the query Q in the retrieval task.

Table 2: Attention scores over input [u, ##gg, sandals] for aspects *category* and *brand*.

Token	[CLS]	u	##gg	sandals	[SEP]
Category	0.000	0.1220	0.1415	0.5870	0.1495
Brand	0.000	0.3672	0.2976	0.1557	0.1796

We want to highlight three points for AEN. First, AEN is computationally more efficient than the Attention layers used in Transformers (See Section 7.2). Second, when computing E_A in Equation 1, we mask the CLS token to avoid attending to it, since the CLS encoding may already mix information from different aspects. Last, aside from those explicit aspects (Section 5) like *category* and *brand* in Table 2, we also add a special aspect, denoted ***OTHER***, to capture the remaining important information that may not be covered by the explicit aspects.

5 ASPECT LEARNING

Without any special learning, the aspect embeddings only represent **latent aspects**. That is, the represented aspects are not explicit, and the model has no control over which aspect or what type of information a particular aspect embedding represents.

We address this issue via, what we called, **Aspect Learning**. Specifically, we can train the aspect embeddings by predicting the corresponding aspect annotations when they are available. For instance, given a query "ugg sandals", its category annotation (Shoes) and its brand annotation (UGG), we can train the aspect embedding E_{a_i} and E_{a_j} by using them to predict the query's category and brand respectively. We call this **aspect prediction task**. These aspect annotations could be obtained at large scale using some NLP tools for certain domains (e.g., e-commerce). We acknowledge that aspect annotations may not be available in some scenarios, thus we also test our model without Aspect Learning (Section 8.2). In this case, the aspect embeddings only represent latent, rather than explicit, aspects.

We compute the **aspect prediction loss** based on softmax cross entropy loss, similar to MLM loss [9]. As illustrated in Figure 2, given an aspect $a \in A$ (e.g., *brand*), we predict the probability $P(x)$ for each aspect value x (e.g., UGG, Nike) in the aspect vocabulary $\mathcal{V}_a$ using $\texttt{softmax}(E_a U_a^T + b_a)$, where E_a is the aspect embedding, $U_a \in \mathbb{R}^{|\mathcal{V}_a| \times H}$ is the **aspect embedding table** and $b_a \in \mathbb{R}^{|\mathcal{V}_a|}$ is the bias term. Note U_a and b_a are free trainable parameters, initialized randomly and shared between the query and document encoders. They are auxiliary parameters for training only – they are only used for computing the loss and are not used during inference. We find the aspect embedding tables can capture interesting aspect semantics as demonstrated in Table 9 from our case study (Section 9.4).

Given a set of aspect annotations $\mathcal{A}_a$ (ground truth labels for aspect a), the aspect prediction (AP) loss for aspect a is then computed as,

$$\mathcal{L}_{AP}^{a} = -\frac{1}{|\mathcal{A}_a|} \sum_{x \in \mathcal{A}_a} \log\left(P(x)\right), \tag{3}$$

where $\mathcal{A}_a$ denotes the set of the aspect annotations. Note that some aspects may have multiple annotations. For example, a query can be annotated with both category Sandals and Shoes, i.e., $\mathcal{A}_{category} =$ {Sandals, Shoes}. Equation 3 simply aggregates the losses for all the annotations by averaging.

We want to highlight three points for this learning procedure. First, via Aspect Learning, our model not only understands the general language semantics as the other pretrained language models, it also learns some domain knowledge and can recognize aspects for the specific domain (e.g., categories and brands for e-commerce). Our experiment shows our model can perform effectively at predicting the aspects (Table 5). Second, when trained with the aspect prediction loss, the aspect embeddings can represent aspects more explicitly, and we call them "explicit aspects". This could largely enhance model interpretability, as demonstrated in our case study in Table 8. Lastly, the *OTHER* aspect (Section 4) is not trained in Aspect Learning as there are no available annotations for it.

6 ASPECT FUSION NETWORKS (AFNS)

In a lot of scenarios, we need to fuse the multiple aspect embeddings into a single-vector representation, so that we can use approximate nearest neighbor search [2, 11] in Dense Retrieval without additional infrastructure change or cost. This section discusses different Aspect Fusion Networks (AFNs) we designed for this purpose. Again, we only provide details for the query side, as the procedure is identical for the document side.

6.1 Weighted Sum

To fuse the aspect embeddings E_a from all the aspects as the query embedding E_Q, one simple method is to weighted-sum up all the aspect embeddings,

$$E_Q = \sum_{a \in A} w_a \cdot E_a, \qquad w_a = \frac{e^{\gamma_a}}{\sum_{a' \in A} e^{\gamma_a}} \tag{4}$$

where w_a are the aspect weights computed by applying softmax over the trainable parameter γ_a.

However, one drawback with this Weighted Sum AFN is that the aspect weights do not depend on the input, and therefore cannot adapt to different inputs. For example, a query may not even specify the brand, thus we may want to discount the aspect weight for the *brand* aspect embedding for the particular query.

6.2 Presence Weighting

To address the issue above, we design another AFN called "Presence Weighting", which could adjust the aspect weights based on whether the aspects are presented in the query or not.

More specifically, we decompose the aspect weights $w_a(Q)$ into two factors below.

$$w_a(Q) = P_a(Q) \cdot \gamma_a. \tag{5}$$

Before describing the notations above, note that: (a) the aspect weight $w_a(Q)$ now depends on the input text Q; and (b) we $L1$-normalized the aspect weights so that they sum up to one. For the notations in Equation 5, γ_a is an input-independent weight used to capture the aspect importance regardless of aspect presence. $P_a(Q)$ is the probability of the aspect a being presented in the query. We predict that by applying one sigmoid layer on top of the aspect embedding, $P_a(Q) = \texttt{SigmoidLayer}\left(E_a(Q)\right)$. We train this component based on the aspect annotations (Section 5) when they

are available, using a, what we called, **aspect presence prediction task**. Specifically, we assume that if an aspect does not have any annotations then that aspect is not presented in the text input. Based on this aspect presence label, denoted as y_a, we compute the aspect presence prediction (APP) task loss using cross entropy loss,

$$\mathcal{L}^a_{APP} = -y_a \log\left(P_a(Q)\right) - (1 - y_a) \log\left(1 - P_a(Q)\right). \quad (6)$$

6.3 CLS-Gating

Similar to our multi-aspect problem framing, Mixture-of-Experts (MoE) models [18] train multiple experts and then use a softmax gating network for mixture. Inspired by that, we design a gating-based AFN that uses the CLS encoding from BERT as the input to the gating network to fuse aspects, which we call "CLS-Gating". In this case, each aspect embeddings are regarded as the "experts", and the gate-values are equivalent to the aspect weights w_a mentioned above for expert mixture (or aspect fusion).

Specifically, the gate-values or aspect weights are computed by passing the CLS encoding, $E_{\texttt{CLS}}$, through a linear layer to compute logits for each aspects, i.e., $\texttt{Linear}(E_{\texttt{CLS}}) \in \mathbb{R}^{|A|}$. For simplicity, we use γ_a to denote the logit computed for aspect a. The aspect weights are then computed using softmax, $w_a(Q) = \frac{e^{\gamma_a}}{\sum_{a' \in A} e^{\gamma_{a'}}}$. The query embedding is then computed by weighted sum as above (Equation 4).

Note that the other AFNs mentioned above do not *directly* depend on any output from the BERT encoder. For example, the Presence Weighting AFN computes the aspect weights only using E_a as input (with other trainable parameters). CLS-Gating AFN instead directly uses $E_{\texttt{CLS}}$ from BERT, which we find could utilize the information from BERT more efficiently (Section 9.3).

7 MODEL TRAINING & ANALYSIS

This section describes the training procedure for our Multi-Aspect Dense Retrieval Model (MADRM), as well as provides model size and complexity analysis.

7.1 Training

The BERT encoders in MADRM (Figure 1) are initialized from public pretrained checkpoint, we then further pretrain MADRM using a large scale corpus for a specific application domain. For this pretraining, we combine the aspect prediction loss $\mathcal{L}^a_{AP}$ (Equation 3), the aspect presence prediction loss $\mathcal{L}^a_{APP}$ (only applicable if using Presence Weighting AFN) as well as the Masked Language Model (MLM) loss $\mathcal{L}_{MLM}$ [9] as follows for both the query and document side,

$$\mathcal{L}_{MLM} + \sum_{a \in A \setminus \{OTHER\}} \left(\mathcal{L}^a_{AP} + \mathcal{L}^a_{APP}\right). \quad (7)$$

Note that we do not compute any loss for the special *OTHER* aspect (Section 4).

For finetining, we use two losses for the retrieval task: (a) the cross entropy loss $\mathcal{L}_{CE}$ for predicting whether a query-document pair is a relevant pair, where the query-document pairs usually come from some candidate ranking lists; and (b) the in-batch softmax cross entropy loss $\mathcal{L}_{SCE}$, which uses random negatives from the training batch. We also find adding a small amount of aspect prediction loss (and aspect presence prediction loss if applicable) is helpful for the retrieval task, so our finetuning loss becomes,

$$\mathcal{L}_{CE} + \mathcal{L}_{SCE} + \lambda \sum_{a \in A \setminus \{OTHER\}} \mathcal{L}^a_{AP}(Q) + \mathcal{L}^a_{APP}(Q) + \mathcal{L}^a_{AP}(D) + \mathcal{L}^a_{APP}(D), \quad (8)$$

where λ is tuned based on a validation dataset.

7.2 Model Size and Complexity

Our MADRM implementation has smaller model size and complexity than the BERT-based bi-encoder model (BiBERT) [22]. Specifically, we implemented MADRM using one-less Transformer layer [32] than BiBERT. In other words, we construct MADRM by replacing the last Transformer layer in BiBERT with the Aspect Extraction Network (AEN) and Aspect Fusion Network (AFN). In fact, the AEN and AFN are analogous to the Multi-Head Self Attention (MHSA) and feedfoward network (FFNN) in Transformers respectively. Thus, we compare AEN and AFN with MHSA and FFNN in this section.

For model size, AEN and AFN combined use at least $2H \times (I - |A|)$ fewer parameters than a Transformer layer, where H is the hidden size, I is the intermediate output size in FFNN and $|A|$ is the number of aspects. Note that we usually have $I \gg |A|$, e.g., $I = 3072$ and $|A| = 4$ in our experiments. Comparing AEN with MHSA, while both use equal-size projection parameters in the Attention layer, AEN in addition introduces the Attention query parameter $\texttt{Q} \in \mathbb{R}^{|A| \times H}$ (Section 4). Aside from that, AEN also adds the aspect embedding tables $U_a \in \mathbb{R}^{|\mathcal{V}_a| \times H}$ for each aspect (Section 5). However, U_a is only used for training when computing the aspect prediction loss, and does not affect model size during inference. Comparing AFNs with FFNN, one of the most compute-expensive AFNs, CLS-Gating AFN, uses a kernel parameter of size $H \times |A|$ in the gating network, while FFNN uses two kernel parameters of total size $2H \times I$.

For time complexity, AEN is also slightly more efficient than MHSA, since AEN only compute encodings for the $|A|$ aspects instead of all the input tokens in the sequence. This is similar to Funnel-transfomer [8]. Note that the number of aspects $|A|$ is usually much smaller than the number of input tokens. It is also easy to see our AFNs are less compute-expensive than FFNN.

8 EXPERIMENTAL SETUP

8.1 Datasets

We use the following proprietary e-commerce data, as we are not able to find public datasets suitable for our experiments [2].

- E-commerce Corpus Dataset (ECD): this corpus contains 9M distinct documents and 8M distinct queries, used for model pretraining and our retrieval evaluation (Section 8.3).
- E-commerce Relevance Dataset (ERD): this dataset contains human relevance judgements for query-document pairs, used for model finetuning and evaluation. We use 39k, 1.6k, 1.9k queries for finetuning, validation and testing respectively. The average query length is 3.4 words. Each query has 8.6 documents with relevance judgements on average. The relevance judgements have grades $rel \in \{0, 1, 2, 3, 4\}$, where 0 means non-relevant and 1/2/3/4 mean fair/good/excellent/perfect relevance respectively.

[2] The closest we can find is the Amazon Product Search dataset [1], but it uses synthetic queries created based on product *categories*. Since our models explicitly capture the *category* aspect, this dataset could bias towards our proposed models over other baselines.

We regard $rel \leq 1$ as negatives and the rest as positives for metrics that require binary labels.

For both datasets, the queries only come with their query text as the input feature, while the documents have titles, descriptions, entities and keywords. In addition to that, the queries and documents are annotated with *category*, *brand* and *color* aspect annotations offline using an in-house high-accuracy NLP tool. Note that these annotations are used as auxiliary labels for model training and evaluation, and are not required for inference. We report the percentage of queries and documents having aspect annotations in Table 3 for each aspect on ECD (the statistics are consistent across the datasets). For example, 96.5% of the queries have at least one category annotation. From ECD, we also collect the aspect vocabularies, used to create the aspect embedding tables in Section 5. The collected vocabulary sizes are 13k, 38k, 152 for *category*, *brand*, *color* aspect respectively after filtering out some low-frequency aspect values.

Table 3: Percentage of queries and documents having aspect annotations for each aspect on ECD.

Query aspects			Document aspects		
Category	Brand	Color	Category	Brand	Color
96.5%	11.8%	5.9%	99.0%	63.1%	15.8%

8.2 Model Implementation

We compare our models with the following bi-encoder baselines, and describe the model implementation below.

- BiBERT [22, 28]: a BERT bi-encoder baseline, using the 12-layer BERT-base as the encoders and computes relevance scores using cosine similarity between the `CLS` encodings. The model is pretrained with MLM loss and finetuned with the two retrieval task losses (Section 7.1).
- MpBERT: an improvement over BiBERT that pools encodings from multiple tokens instead of just the single `CLS` token for text representation [6]. We denote this model as MpBERT, where Mp stands for *m*ulti-token *p*ooling. This is similar to our model, which also fuses multiple aspect embeddings as the final embedding. We implement both models using the same number of tokens/aspects for pooling/fusion using the same AFN. This model is trained in the same way as BiBERT.
- MtBERT: a multi-task (MT) learning baseline. MtBERT uses the same model architecture as BiBERT, but in addition co-trains the `CLS` encoding with the same aspect prediction tasks as our model (specifically, MADRAL) in both pretraining and finetuning. In other words, MtBERT uses the same multi-task objectives as MADRAL, and only differs from MADRAL in the model architecture. This model is similar to the prior work [36], except that we upgrade the TextCNN encoder to BERT encoder and add aspect prediction tasks for all aspects instead of just the *category* aspect for co-training.
- DSSM [15]: a DNN bi-encoder baseline, which is one of the earliest bi-encoder Dense Retrieval model. For fair comparison, we use the BERT wordpiece tokens as input features for DSSM and initialize the embedding table from BiBERT (after BiBERT is being pretrained on ECD). DSSM is then finetuned with the same retrieval task losses.
- MADR and MADRAL: our proposed Multi-Aspect Dense Retrieval models, which add the Aspect Extraction Network (AEN) and Aspect Fusion Network (AFN) on top of the BERT-base encoders (Figure 1). Note that MADR and MADRAL use one-less Transformer layer than BiBERT in our implementation. As a result, they contain fewer parameters than BiBERT (Section 7.2). MADRAL[3] uses Aspect Learning for the *category*, *brand*, *color* aspect. It is pretrained with losses in Equation 7 and finetuned with losses in Equation 8. MADR instead does not use Aspect Learning and therefore does not require any aspect annotations. It is pretrained and finetuned with the same losses as BiBERT, and only differs with BiBERT in terms of the model architecture. Unless mentioned otherwise, we use Presence Weighting AFN in MADR and MADRAL.

For all the models, we use the query text as query input, truncated at length 32. We concatenate all the document features (Section 8.1) as document input, truncated at length 128. BERT components in all the models are initialized from Google's public pretrained checkpoint. All the models do not share query and document encoder parameters, and are pretrained on ECD for 300k steps and finetuned on ERD for 560 steps with batch size 1024. The hidden size $H = 768$ is inherited from BERT-base. However, the aspect, query and document embeddings are projected to 128 dimensions for all models to make approximate nearest neighbor search efficient. Other model hyperparameters are tuned using the validation dataset.

8.3 Evaluation

We evaluate model performance for the following tasks.

- **Retrieval Task**: this is our primary task. we use ScaNN [11] to perform approximate nearest neighbor search to retrieve ten thousand documents from ECD. We measure Recall@10k using queries and relevance judgements in ERD.
- **Ranking Task**: we also evaluate models' ranking performance by re-ranking the judged documents in ERD for each query. We report NDCG@5, where the gain (before discounting) is defined as $2^{rel} - 1$. In addition to NDCG@5, we also report AUC, which is calculated per query before averaging.
- **Aspect Prediction Task**: this is not our main objective, but we also evaluate aspect prediction performance for relevant models. Specifically, we evaluate accuracy for the top-1 predicted aspect value according to the estimated probability $P(x)$ (Section 5). We denote this metric as Accuracy@1.

We report results on the test set and perform paired t-test with p-value<0.01 for marking statistically significant differences.

9 EXPERIMENTAL RESULTS

9.1 Overall Results

The overall performance is shown in Table 4. Our proposed method MADRAL outperforms all the competing methods by a large margin in all metrics. This suggests that our proposed model architecture and Aspect Learning can indeed bring benefits for retrieval models.

[3]The suffix "AL" in MADRAL stands for Aspect Learning

Table 4: Results for MADRMs and Baselines. Reports relative performance change with respect to BiBERT**. *, †, ¶, ‡, § indicate statistically significant improvement over** DSSM*, BiBERT†, MpBERT¶, MtBERT‡ **and** MADR§ **respectively.**

	Recall@10k	AUC	NDCG@5
DSSM	-14.54%	-6.38%	-3.64%
BiBERT	+0.00%*	+0.00%*	+0.00%*
MpBERT	+2.49%*†	-0.54%*	+0.16%*
MtBERT	+8.56%*†¶§	+0.01%*	+0.57%*
MADR	+5.75%*†¶	+2.17%*†¶‡	+1.54%*†¶‡
MADRAL	+12.44%*†¶‡§	+2.25%*†¶‡	+2.06%*†¶‡

Comparing individual models, we see that BiBERT, which encodes input by BERTs, performs better than DSSM, which encodes input by fully-connected networks. This is in line with previous findings that BERT based encoders generally outperform other types of encoders. As also observed in the previous work [6], MpBERT further improves BiBERT on recall, since it pools encodings of multiple tokens instead of just a single `CLS` token. This is similar to our idea of fusing multiple aspect embeddings. MtBERT is also superior to the vanilla BiBERT. This implies that the aspect prediction task is useful for the retrieval task. The models that jointly learn the two types of tasks could capture commonality from both and generalize better.

MADR is one variant of our model, which shares the same model structure as MADRAL but is not provided with the aspect annotations and thus does not explicitly learn the aspect prediction task. It outperforms BiBERT, but when compared with MpBERT, the improvement seems less significant. This indicates fusing multiple aspect embeddings or pooling multiple token encodings from the Transformer output can more effectively encode text for relevance prediction. MADR also surprisingly performs better than MtBERT on AUC and NDCG, though worse on recall. Considering that MtBERT is provided with the explicit aspect signal while MADR is not, it might be safe to say that the model structure we propose fits the retrieval task well. We suspect that the design of the aspect extraction and fusion imposes prior knowledge on the model, which helps the model automatically capture some latent aspects and generalize better.

With Aspect Learning, MADRAL further improves over MADR, MpBERT and MtBERT, suggesting that both the proposed model structure and the aspect prediction task contribute to the enhancement of the performance. Compared with MADRAL, MtBERT relies on a single embedding, the output representation of the `CLS` token, for multiple prediction tasks. This single embedding design has limited expressiveness. This seems to constrain the model and increases the difficulties of training – we observed that MtBERT's performance on the aspect prediction task improves much slower than that of MADRAL in pretraining. MtBERT's eventual aspect prediction performance is also not as good.

We present the performance on the aspect prediction task in Table 5. We see that our model MADRAL performs much better than MtBERT on almost all aspects, except for the *color* aspect in queries, where both models exhibit an almost perfect performance. This supports our hypothesis that using a single embedding for multi-task learning has limited the ability of MtBERT to handle both tasks simultaneously.

Table 5: Accuracy@1 for the aspect prediction task

	Query aspects			Document aspects		
	Category	Brand	Color	Category	Brand	Color
MtBERT	0.713	0.631	1.000	0.801	0.531	0.991
MADRAL	0.783	0.962	0.990	0.923	0.978	0.999

9.2 Ablation Studies

To further understand the model, we conduct ablation studies by removing one component at a time. The results are summarized in Table 6. Almost all results being negative indicate that most components contribute positively to the performance gain.

Table 6: Ablation study results. Reports relative performance changes with respect to MADRAL**, with † indicating statistically significant changes.**

	Recall@10k	AUC	NDCG@5
Removing different aspects			
MADRAL$^{-\text{category}}$	-3.59%†	-1.23%†	-0.64%
MADRAL$^{-\text{brand}}$	-2.07%†	+0.48%	+0.26%
MADRAL$^{-\text{color}}$	-2.25%†	-0.00%	+0.16%
MADRAL$^{-\text{other}}$	-0.70%	+0.65%	+0.54%
Disabling Aspect Learning on the query or document side			
MADRAL$^{-\text{query}}$	-6.23%†	-0.62%	-0.82%†
MADRAL$^{-\text{doc}}$	-4.16%†	-0.66%	-0.51%
MADRAL	+0.00%	+0.00%	+0.00%
BiBERT	-11.06%	-2.20%	-2.02%

By removing each aspect from MADRAL, it seems like *category*, *brand* and *color* are all important in the domain of e-commerce search, especially the *category* aspect. In contrast, the smaller performance drop of MADRAL$^{-\text{other}}$ in Recall@10k suggests the *OTHER* aspect is less important, compared to the other aspects, which come with aspect annotations for the model to learn from. Note that this observation is domain-specific and our model could adapt based on the datasets of a particular domain.

Disabling Aspect Learning on the query or document side also leads to decrease of performance. This means that learning on both sides is important. This might be because there is some mismatch between the vocabularies of user queries and documents. Therefore it is important to learn on both sides to understand the different expressions of aspects. Disabling Aspect Learning on the query side (MADRAL$^{-\text{query}}$) has larger performance drop than the document side (MADRAL$^{-\text{doc}}$). This seems to indicate that the query aspect understanding is more critical than that for documents. Specifically, queries are shorter and can be informally or not well formulated, while documents are usually more informative and well structured.

Lastly, our models in the ablation studies still outperform BiBERT. This demonstrates the robustness of our model architecture.

9.3 Aspect Fusion Network Results

In this subsection we examine the effectiveness of different Aspect Fusion Networks, which aim to fuse multiple aspect embeddings into a single one to save the index space. Table 7 lists the results of different fusion mechanisms we have proposed. The simple Weighted Sum method is less effective than other methods, but the performance drop is not very large. By discounting aspects that are not present in the input, Presence Weighting improves over the naive Weighted Sum method on recall. The CLS-Gating approach fuses the embeddings in a similar way as Mixture-of-Experts (MoE) models [18]. By learning to weight embeddings based on the input representation, namely the CLS encoding, CLS-Gating achieves the best performance. This may be explained by CLS-Gating utilizing the information from BERT more efficiently, as it directly takes the CLS encoding as input.

Table 7: Results for different Aspect Fusion Networks. Reports relative performance changes with respect to WeightedSum, with † indicating statistically significant changes.

	Recall@10k	AUC	NDCG@5
Weighted Sum	+0.00%	+0.00%	+0.00%
Presence Weighting	+1.16%†	-0.56%	-0.45%
CLS-Gating	+1.26%†	+0.78%	+0.40%

Table 8: Predictions from MADRAL for an example query-document pair. *Confidence* is the predicted probability in aspect prediction ($P(x)$ in Section 5). *Presence* is the predicted aspect presence probability (Section 6.2). *Weight* is the aspect weights used to fuse aspect embeddings. *Relevance score* is standardized (zero-mean unit variance).

Query: *breville juicer*

	Category	Brand	Color	OTHER
Prediction	Juicers	Breville	Grape	
Confidence	0.979	1.000	0.187	
Presence	1.000	1.000	0.000	0.761
Weight	0.374	0.350	0.000	0.276

Document: *Juicing with the Breville Juice Fountain Extractor: A Simple Steps Brand Cookbook: 101 Superfood Juice Recipes to ...*

	Category	Brand	Color	OTHER
Prediction	Books	Breville	Grape	
Confidence	0.993	0.369	0.151	
Presence	1.000	0.001	0.000	0.993
Weight	0.500	0.000	0.000	0.500

Standardized relevance score

BiBERT: 0.213 MADRAL: -0.469

9.4 Case Studies

To better understand our model as well as to demonstrate its improvements on both effectiveness and interpretability, we show an example query-document pair in Table 8 with different predictions made by MADRAL using Presence Weighting AFN (Section 6.2).

First, our model can make high-quality predictions for aspects as well as aspect presences in this case. Using the query-side predictions as examples, the model correctly predicts the *category* and *brand* to be Juicers and Breville with 1.0 confidence scores, which are estimated by $P(x)$ defined in Section 5. The model also correctly predicts that *color* is not presented in the query – the predicted presence probability is zero. As a result, *color*'s aspect weight, used in aspect fusion, is also zero (see Equation 5). This effectively ignores the *color* aspect embedding in the final query embedding.

Second, this case study demonstrates how our proposed model MADRAL improves over the baselines. The query *breville juicer* is looking for a product in the category of Juicer, while the retrieved document has the category Books. Since there is a high textual similarity between the query and the document, the baseline model BiBERT, which is not trained to explicitly consider aspects, gives a high relevance score to the document. With the explicit learning of the aspects, MADRAL has recognized the mismatch of the category between the query and the document, assigning a much lower score to the document. Note that the relevance scores are standardized with zero-mean unit-variance for comparison.

Lastly, we find our model offers enhanced interpretability, compared to other black-box Dense Retrieval models. As showcased in the example, the intermediate aspect predictions provide a means to inspect whether the model can correctly understand the aspects. Moreover, since the final query and document embedding is a simple weighted sum of the aspect embeddings, one could explain why a high or low relevance score is predicted by the model based on the predicted aspects and the weights that combine them.

Next, we conduct another case study to demonstrate the semantics captured by the aspect embedding tables (U_a in Section 5). In Table 9, we select one example aspect value per aspect, and retrieve their nearest neighbors in the embedding space. We can see the nearest neighbors aspect values are all semantically related to the querying aspect values. For example, for the Bedding category, its related categories are all product categories used for beds and sleeping. This implies that our model is able to position semantically similar aspect values close to each other in the embedding space.

Table 9: Examples of semantically related aspect values found by MADRAL. Lists top-4 nearest categories/brands/colors for Bedding, HP and Pink respectively based on the learned aspect embedding tables U_a (Section 5). Numbers in the parentheses are the dot-products between embeddings.

Category	Brand	Color
(1.00) Bedding	(1.00) HP	(1.00) Pink
(0.626) Comforters	(0.45) Dell	(0.34) Light Pink
(0.563) Bedding Sets	(0.37) Intel	(0.28) Hot Pink
(0.540) Quilts & Bedspreads	(0.35) Canon	(0.25) Rose
(0.513) Sheets	(0.35) Brother	(0.19) Red

10 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a model to represent multiple aspects of a query and a document using multiple embeddings for Dense Retrieval, called Multi-Aspect Dense Retrieval model. We design an Aspect Extraction Network to effectively extract aspect embeddings by attending to different parts of the text input. We also design Aspect Fusion Networks that combine the aspect embeddings as the final query/document embedding based on input-independent weighted sum, aspect presence weighting or a gating network. When aspect annotations are available, we propose an aspect prediction task to teach the model domain knowledge and guide the model to represent aspects explicitly with the aspect embeddings. Our evaluation on an e-commerce dataset shows impressive improvements over strong baselines, including BERT-based bi-encoder and its multi-task learned variant trained with the same signals as our model. Furthermore, our model offers enhanced interpretability compared to other black-box Dense Retrieval models, as demonstrated by the case studies. Lastly, the aforementioned improvements do not come at the cost of indexing, querying and model complexity.

The idea proposed in this paper opens up a number of directions for future research in Dense Retrieval. One direction is to better leverage the learned multi-embedding representations. Instead of fusing the aspect embeddings, we may preserve them to be used in more expressive similarity functions and design nearest neighbor search algorithms that can work efficiently with them. Another direction is to scale up the number of aspects from a handful to hundreds / thousands, so that the model can understand rich domain attributes, including more subtle ones such as TV panel types (e.g. {LED, OLED}). Instead of computing all the hundreds / thousands of aspect embeddings, we can investigate using MoE [18] to sparsely activate certain aspects and make model efficient for practical use.

REFERENCES

[1] Qingyao Ai, Yongfeng Zhang, Keping Bi, Xu Chen, and W Bruce Croft. Learning a hierarchical embedding model for personalized product search. In *Proc. of SIGIR*, pages 645–654, 2017.

[2] Alexandr Andoni, Piotr Indyk, and Ilya Razenshteyn. Approximate nearest neighbor search in high dimensions. In *Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018*, pages 3287–3318. World Scientific, 2018.

[3] Jing Bai, Ke Zhou, Guirong Xue, Hongyuan Zha, Gordon Sun, Belle Tseng, Zhaohui Zheng, and Yi Chang. Multi-task learning for learning to rank in web search. In *Proc. of CIKM*, pages 1549–1552, 2009.

[4] Oren Barkan, Noam Razin, Itzik Malkiel, Ori Katz, Avi Caciularu, and Noam Koenigstein. Scalable attentive sentence pair modeling via distilled sentence embedding. In *Proc. of AAAI*, volume 34, pages 3235–3242, 2020.

[5] Olivier Chapelle, Pannagadatta Shivaswamy, Srinivas Vadrevu, Kilian Weinberger, Ya Zhang, and Belle Tseng. Multi-task learning for boosting with application to web search ranking. In *Proc. of KDD*, pages 1189–1198, 2010.

[6] Jiecao Chen, Liu Yang, Karthik Raman, Michael Bendersky, Jung-Jung Yeh, Yun Zhou, Marc Najork, Danyang Cai, and Ehsan Emadzadeh. Dipair: Fast and accurate distillation for trillion-scale text matching and pair modeling. *arXiv preprint arXiv:2010.03099*, 2020.

[7] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of ICML*, pages 160–167, 2008.

[8] Zihang Dai, Guokun Lai, Yiming Yang, and Quoc V Le. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *arXiv preprint arXiv:2006.03236*, 2020.

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

[10] Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. Learning dense representations for entity retrieval. *arXiv preprint arXiv:1909.10506*, 2019.

[11] Ruiqi Guo, Sanjiv Kumar, Krzysztof Choromanski, and David Simcha. Quantization based fast inner product search. In *Artificial Intelligence and Statistics*, pages 482–490. PMLR, 2016.

[12] Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. Interpretable & time-budget-constrained contextualization for re-ranking. *arXiv preprint arXiv:2002.01854*, 2020.

[13] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050, 2014.

[14] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. Embedding-based retrieval in facebook search. In *Proc. of KDD*, pages 2553–2561, 2020.

[15] 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 *Proc. of CIKM*, pages 2333–2338, 2013.

[16] Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv preprint arXiv:1905.01969*, 2019.

[17] Ben Ltaifa Ibtihel, Hlaoua Lobna, and Ben Romdhane Lotfi. A deep learning-based ranking approach for microblog retrieval. *Procedia Computer Science*, 159: 352–362, 2019.

[18] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[19] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.

[20] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proc. of SIGIR*, pages 39–48, 2020.

[21] Jurek Leonhardt, Koustav Rudra, and Avishek Anand. Learnt sparsity for effective and interpretable document ranking. *arXiv preprint arXiv:2106.12460*, 2021.

[22] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. Pretrained transformers for text ranking: Bert and beyond. *arXiv preprint arXiv:2010.06467*, 2020.

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

[24] Bhaskar Mitra, Nick Craswell, et al. *An introduction to neural information retrieval*. Now Foundations and Trends, 2018.

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

[26] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and R Ward. Semantic modelling with long-short-term memory for information retrieval. *arXiv preprint arXiv:1412.6629*, 2014.

[27] Rohan Ramanath, Hakan Inan, Gungor Polatkan, Bo Hu, Qi Guo, Cagri Ozcaglar, Xianren Wu, Krishnaram Kenthapadi, and Sahin Cem Geyik. Towards deep and representation learning for talent search at linkedin. In *Proc. of CIKM*, pages 2253–2261, 2018.

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

[29] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. Okapi at trec-3. *Nist Special Publication Sp*, 109:109, 1995.

[30] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *arXiv preprint arXiv:2112.01488*, 2021.

[31] Jaspreet Singh and Avishek Anand. Exs: Explainable search using local model agnostic interpretability. In *Proc. of WSDM*, pages 770–773, 2019.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[33] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*, 2020.

[34] Qian Xu, Sinno Jialin Pan, Hannah Hong Xue, and Qiang Yang. Multitask learning for protein subcellular location prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(3):748–759, 2010.

[35] Shaowei Yao, Jiwei Tan, Xi Chen, Keping Yang, Rong Xiao, Hongbo Deng, and Xiaojun Wan. Learning a product relevance model from click-through data in e-commerce. In *Proc of WWW*, pages 2890–2899, 2021.

[36] Hongchun Zhang, Tianyi Wang, Xiaonan Meng, Yi Hu, and Hao Wang. Improving semantic matching via multi-task learning in e-commerce. In *eCOM@ SIGIR*, 2019.

[37] Tianzhu Zhang, Bernard Ghanem, Si Liu, and Narendra Ahuja. Robust visual tracking via structured multi-task sparse learning. *International journal of computer vision*, 101(2):367–383, 2013.