Enhancing Late Interaction with Informative Entities for Passage Retrieval

Jinyuan Fang, Zaiqiao Meng, and Craig Macdonald

University of Glasgow, Glasgow, UK
j.fang.2@research.gla.ac.uk
zaiqiao.meng, craig.macdonald@glasgow.ac.uk

Abstract. Passage retrieval aims to rank a set of passages based on their relevance to a query. The ColBERT model, which is an efficient yet effective retrieval model, employs a late interaction approach for relevance estimation. This approach first independently encodes queries and passages and then leverages an interaction step to capture the fine-grained similarities between the queries and the passages. Despite the effectiveness of this approach, it might have difficulty in fully understanding the relationships among entities within the queries and the passages, due to the tokenization of entities into multiple tokens. To this end, we propose ColLUKE, an entity-aware contextualised late interaction model, to enhance the late interaction approach by explicitly leveraging the entity information. Specifically, we first leverage an entity linking tool to identify the entities within the queries and the passages. We then use LUKE, an entity-aware pretrained language model, to independently encode queries and passages into dense embeddings by taking the corresponding texts and the entities within the texts as inputs. The dense embeddings are passed to an interaction function for relevance estimation. We evaluate the performance of our ColLUKE on five different query sets, which include four in-domain query sets and one out-of-domain query set. Experimental results show that our ColLUKE outperforms ColBERT by up to 10.59% on the in-domain TREC DL-HARD query set and up to 11.89% on the out-of-domain HotPotQA query set.

Keywords: Passage Retrieval · Bi-Encoders · Late Interaction.

1 Introduction

The passage retrieval task aims to rank a set of passages from a corpus based on their relevance to a given query [7]. This retrieval process can help users quickly identify and access relevant information within a large collection of passages, making it a crucial component of search engines [4] and various natural language processing applications [11, 13]. With the development of pretrained language models (PLMs) such as BERT [6] and RoBERTa [19], bi-encoder (dense retrieval) models that leverage PLMs as the backbone models have gained significant attention [41]. These models use PLMs to independently encode queries and passages into dense embeddings, and estimate the relevance between a query

and a passage based on the similarity between their corresponding embeddings. The advantage of bi-encoders is that they allow building the index of the passages by pre-computing embeddings of all the passages within a corpus [13]. Once the index is established, bi-encoders leverage the approximate nearest neighbour (ANN) search [12] to retrieve top-K relevant passages. The **late interaction** approach was proposed in ColBERT [14], and enhances over single-representation bi-encoders by encoding and storing the token-level embeddings of passages at indexing time using BERT and then employs an interaction step to capture the fine-grained token-level similarities between queries and passages. The late interaction approach offers the advantage of building a corpus index while capturing the interactions between queries and passages for improved ranking performance.

Despite ColBERT has demonstrated superior performance compared with other bi-encoders [2, 3], it might have difficulty in fully understanding the relationships among entities within the queries and the passages [31]. This is due to the fact that the encoder model in ColBERT, i.e., BERT, employs tokenization to break down the surface forms of entities into multiple tokens, making it challenging to capture the semantics of these entities [37]. Therefore, the dense embeddings generated by ColBERT may not fully capture the entity-related information, leading to sub-optimal ranking performance for queries that require such entity-level semantic information. Moreover, existing works also show that explicitly incorporating the entity information can either improve the recall of sparse retrieval method [30] or enhance the ranking performance of bi-encoder [31] and cross-encoder [7]. However, it remains unexplored how to explicitly leverage the entity information to enhance the late interaction approach.

Therefore, in this paper, we aim to enhance the performance of the late interaction approach by enriching the dense embeddings with entity information within the queries and the passages. In order to fully understand the entity information within the texts, we employ a pretrained language model LUKE (Language Understanding with Knowledge-based Embedding) [37] as the backbone model. Compared with vanilla PLMs such as BERT [6], LUKE incorporates an entity embedding matrix in addition to the token embedding matrix to enhance the model's understanding of entities within the texts. Moreover, it leverages an entity-aware self-attention mechanism to learn entity-aware contextualised token embeddings. The learned embeddings can capture the semantic relationships among both tokens and entities. In our entity-aware late interaction model, we first identify the entities within the queries and the passages using a fast end-to-end entity linking tool ELQ (Entity Linking model for Questions) [16], which is able to identify the Wikipedia entities within the texts. Subsequently, we use LUKE to learn entity-aware contextualised token embeddings of queries and passages by taking the corresponding texts and the entities within the texts as inputs. Finally, we use the late interaction approach to estimate the relevance between queries and passages by calculating the maximum similarity (maxsim) scores of their token embeddings. Since our model is based on LUKE and incorporates a late interaction approach, we term our model as ColLUKE. We train our ColLUKE model on the MS MARCO (Train) [23] and evaluate the

performance on five different query sets, including four in-domain query sets: MS MARCO (Dev) [23], TREC DL-2019 [2], TREC DL-2020 [3], TREC DL-HARD [22] query sets, and one out-of-domain query set: HotPotQA [39]. The results show that leveraging the entity information can enhance the performance of the late interaction approach and that our ColLUKE achieves better or comparable performance compared with the state-of-the-art ColBERT model.

Our contributions can be summarised as: (1) We propose ColLUKE, an entity-enhanced late interaction model for the passage retrieval task. (2) Experimental results show that our ColLUKE can outperform ColBERT, which does not use entity information, by up to 10.59% on the in-domain TREC DL-HARD query set and up to 11.89% on the out-of-domain HotPotQA query set.

2 Related Work

Neural Ranking Models. Recently, neural ranking models that leverage PLMs as the backbone models have gained significant attention. Existing neural ranking models for passage retrieval can be roughly divided into two categories: cross-encoder models [10, 17, 24, 25, 45] and bi-encoder models [13, 14, 21, 36, 42]. Cross-encoders, such as monoBERT [25] and monoT5 [24], take both queries and passages as inputs and directly output relevance scores between the queries and the passages using a classifier. Since cross-encoders can capture semantic interactions between queries and passages with the self-attention mechanism, they achieve state-of-the-art performance on the passage retrieval task [41]. Vanilla cross-encoders use cross-entropy loss for training, recent works also propose to use hard negatives [26] or pairwise ranking loss [45] to improve the performance. One issue with cross-encoders is that they are computationally expensive as they require combining the sequence of queries and passages as inputs. Therefore, it is impractical to use cross-encoders to estimate the relevance scores between a query and all the passages in the corpus. The cross-encoders are usually used in a re-ranking pipeline, where a set of candidate relevant passages are first retrieved with an inexpensive retrieval method such as BM25 [27, 28] and then these candidate passages are re-ranked using the cross-encoders.

In contrast, bi-encoders models, such as DPR [13] and ANCE [36], are more computationally efficient. These models learn separate representations for queries and passages with PLMs, and estimate relevance scores between queries and passages through the inner product of their contextualised representations [13]. One advantage of bi-encoders is that they allow pre-computing and storing the embeddings of passages in the corpus, and therefore alleviate the burden of computing passage embeddings during retrieval. Moreover, when coupled with the ANN search [12], it would be more efficient for bi-encoders to retrieve top-K relevant passages. However, the performance of bi-encoders is usually not as good as the cross-encoders since there are no interactions between queries and passages. To address this issue, the late interaction approach proposed in ColBERT [14] first independently encodes queries and passages and then uses an interaction step to capture the fine-grained similarities between queries and pas-

sages. The Col* models in [34] extend ColBERT to other PLMs. Compared with existing late interaction models, our ColLUKE leverages the entity information within queries and passages to enhance the model's understanding of queries and passages, thereby leading to improved ranking performance.

Entity-Enhanced Neural Ranking Models. Entities have been used to enhance the performance of some ranking tasks, such as question answering [5, 40, 43, 44], recommendation [32, 33, 38] and entity retrieval [1, 8]. They have also been used to improve the performance of the passage retrieval task [9, 18, 20, 30, 31, 35]. For example, Gonçalves et al. [9] proposes to combine bag-of-words (BoW) representations with bag-of-entities (BoE) representations for sparse retrieval. Xiong et al. [35] introduces a word-entity duet framework for ad-hoc retrieval, which utilises word-based and entity-based representations as ranking features and designs an attention-based model AttR-Duet for retrieval. Liu et al. [20] extend the work of AttR-Duet and replace the attention-based model with the interaction-based neural ranking model. Recently, Shehata et al. [30] propose to incorporate entity information into the sparse retrievers to improve the recall performance at the first stage of the re-ranking pipeline. Moreover, the EVA model [31] proposes to enrich the query and passage representations with the entity information. Specifically, they leverage kernel functions to create multiple entity representations that reflect different views of a passage, which are then combined with the token embeddings for relevance estimation. Compared with previous works, our ColLUKE leverages pretrained entity embeddings and entity-token cross-attention to enrich the representations of queries and passages.

3 Preliminaries

In this section, we first introduce the task formulation and then introduce the LUKE model, which serves as the foundation of our proposed ranking model.

Task Formulation. The passage retrieval task aims to rank a set of passages according to their relevance to a given query. In this paper, we focus on leveraging the late interaction approach [14] to address this task. Moreover, our goal is to enhance the performance of the late interaction approach by enriching the dense embeddings with entity information within the queries and the passages.

Formally, for a query $q = \{k_1, k_2, \ldots, k_n\}$ that contains n words and a passage $p = \{w_1, w_2, \ldots, w_m\}$ that contains m words, we denote the entities in the query as $e_q = \{h_1, h_2, \ldots h_a\}$ and the entities in the passage as $e_p = \{t_1, t_2, \ldots, t_b\}$. Given the query and the passage as well as their entities, our goal is to learn an encoder model to estimate the relevance between them:

$$\text{Relevance}(q, p) = Sim\left(Enc_\theta(q, e_q), Enc_\theta(p, e_p)\right), \tag{1}$$

where $Enc_\theta(\cdot)$ denotes the encoder model with parameter θ that outputs the contextualised token embeddings of the inputs, $sim(\cdot)$ denotes a similarity function (e.g. inner product). For each query, we calculate the relevance of all the

passages in the corpus respective to the query using Equation (1). These passages are then ranked in the descending order of their relevance scores. The top-K passages are selected as the retrieval results for the given query q.

LUKE Model. In order to effectively leverage the entity information within texts for passage retrieval, we use the LUKE model as the base model. LUKE is a bidirectional transformer-based pretrained language model designed for entity-related natural language tasks [37]. Compared with other PLMs such as BERT [6] and RoBERTa [19], LUKE additionally uses entities appearing in the text as inputs, thus enhancing its capacity to handle entity-related information. The input format of LUKE can be denoted as "[CLS] text [SEP] entities". Moreover, LUKE regards an entity, which may contain multiple words such as "United Kingdom", as a single entity token and introduces a set of entity embeddings for these entity tokens. The entity embeddings and the word embeddings are jointly learned during the pretraining. Since LUKE deals with two types of tokens, i.e., words and entities, it also introduces an entity-aware self-attention mechanism to calculate the attention between tokens of different types. Specifically, the attention score a_{ij} between two token embeddings $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is computed as:

$$a_{ij} = \begin{cases} (\mathbf{Q}\boldsymbol{x}_i)^\top \mathbf{K}\boldsymbol{x}_j, & \text{if both } \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ are words} \\ (\mathbf{Q}_{w2e}\boldsymbol{x}_i)^\top \mathbf{K}\boldsymbol{x}_j, & \text{if } \boldsymbol{x}_i \text{ is word and } \boldsymbol{x}_j \text{ is entity} \\ (\mathbf{Q}_{e2w}\boldsymbol{x}_i)^\top \mathbf{K}\boldsymbol{x}_j, & \text{if } \boldsymbol{x}_i \text{ is entity and } \boldsymbol{x}_j \text{ is word} \\ (\mathbf{Q}_{e2e}\boldsymbol{x}_i)^\top \mathbf{K}\boldsymbol{x}_j, & \text{if both } \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ are entities} \end{cases} \tag{2}$$

where $\mathbf{Q}$, $\mathbf{Q}_{w2e}$, $\mathbf{Q}_{e2w}$ and $\mathbf{Q}_{e2e}$ represent the query matrices between different types of tokens and $\mathbf{K}$ is the key matrix for computing the self-attention.

The pretraining task of LUKE is similar to that of the RoBERTa model, where it employs the masked language model (MLM), i.e., predicting the masked tokens in the inputs, to learn the model parameters. In addition to MLM, LUKE also masks some entity tokens and predicts the masked entity tokens to learn entity embeddings. The LUKE model is pretrained on an entity-annotated corpus obtained from Wikipedia [37].

4 Methodology

In this section, we introduce the details of our proposed ColLUKE model. In general, our ColLUKE first leverages an entity linking tool to identify the entities within the queries and passages and then uses the LUKE model to encode queries and passages into dense embeddings, which are passed to an interaction function for relevance estimation. Figure 1 shows the overview of ColLUKE. We begin by introducing the entity linking tool to obtain the entities in Section 4.1 and then introduce our entity-enhanced late interaction model in Section 4.2.

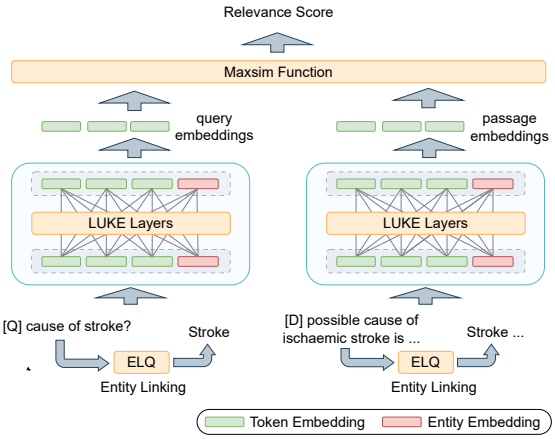

Fig. 1. Overall Framework of our ColLUKE.

4.1 Entity Linking

In order to enrich the dense embeddings with the entity information, we need to first identify the entities within queries and passages. To achieve this, we employ a fast end-to-end entity linking model system [16], called ELQ, which is designed to link entity mentions in a text to their corresponding Wikipedia entities. Specifically, ELQ jointly performs *mention detection*, i.e., identifying the mention boundaries of entities within the text, and *entity disambiguation*, i.e., linking the mentions to Wikipedia entities. ELQ also employs a bi-encoder architecture, comprising two separate encoders: the entity encoder and the question encoder. First, ELQ leverages the entity encoder to pre-compute the embeddings for all the Wikipedia entities. The entity encoder uses a BERT model to encode the surface forms of the entities and their short descriptions. The embeddings of the [CLS] tokens are considered as the embeddings of their corresponding entities. Subsequently, given a text, the question encoder also leverages a BERT model to generate token embeddings. It then employs two linear classifiers to calculate the probabilities of each token being the start and the end of an entity mention. After identifying the mention spans within the text, ELQ computes the embeddings for each mention by averaging the embeddings of tokens within that mention. Finally, it uses the inner product to measure the similarity between each mention embedding and all the entity embeddings, ultimately linking each mention in the text to an entity that exhibits the highest similarity.

We apply the ELQ model to identify the entities for both the queries and the passages. Moreover, in order to mitigate potential errors caused by the ELQ model, we introduce a threshold λ to regulate the accuracy of entity linking. Specifically, we retain the entity linking results only when the similarities between the mention embeddings and the linked entity embeddings exceed the predefined threshold λ. Otherwise, if the similarities are below this threshold, we discard the corresponding entity linking results. This thresholding mechanism

can help refine the accuracy of the entity linking process. In practice, we set the threshold λ as 4.5 as we found it consistently achieves satisfactory performance.

4.2 Entity-Enhanced Late Interaction Model

After identifying the entities within the queries and the passages, we encode each query or passage into contextualised token embeddings. In our model, we employ a single LUKE model as both the query encoder and the passage encoder to learn entity-aware dense embeddings.

Query Encoder. Given a query q, we tokenize it into tokens using the LUKE-based BPE tokenizer [29]. Similar to ColBERT, We add a prefix "[Q]" to the query to indicate that the current sequence is a query. This prefix is placed right after the [CLS] token. Moreover, we also leverage the *query augmentation* method to expand existing queries. Specifically, when the number of tokens within a query is fewer than a predefined threshold N_q, we augment it by padding it with LUKE's special [MASK] tokens until it reaches the predefined length N_q. Otherwise, we truncate the tokens and only retain the first N_q tokens. The padded sequence of query tokens as well as the query entities are then passed into the LUKE model to compute contextualised token embeddings. Moreover, our query encoder further passes these token embeddings through a linear layer without activations. The linear layer aims to control the dimension of query embeddings. Formally, given a query $q = \{k_1, k_2, \ldots, k_n\}$ with query entities $e_q = \{h_1, h_2, \ldots, h_a\}$, the query embeddings are computed as:

$$\mathbf{E}_q = \text{Linear}\left(\text{LUKE}\left(\text{“[Q] } k_1, \ldots k_n \text{ [MASK]...[MASK]”}, \text{“}h_1 \ldots h_a\text{”}\right)\right), \tag{3}$$

where $\text{LUKE}(\cdot, \cdot)$ denotes the LUKE model that takes both tokens and entities as inputs and outputs the entity-aware token embeddings.

Passage Encoder. Given a passage p, we tokenize it into tokens and add a prefix "[D]" right after the [CLS] token to indicate that the current sequence is a passage. Different from queries, we do not use the [MASK] token to pad the sequence to a predefined length. The resulting passage tokens as well as the passage entities are passed through the LUKE model to obtain the contextualised token embeddings. Similar to the query encoder, we use the same linear layer to control the dimension of the token embeddings. Formally, given a passage $p = \{w_1, w_2, \ldots, w_m\}$ and passage entities $e_p = \{t_1, t_2, \ldots, t_b\}$, the passage embeddings are computed as:

$$\mathbf{E}_p = \text{Linear}\left(\text{LUKE}\left(\text{“[D] } w_1, \ldots w_m\text{”}, \text{“}t_1 \ldots t_b\text{”}\right)\right). \tag{4}$$

Relevance Estimation. After obtaining the query embeddings and the passage embeddings, we leverage the maximum similarity (Maxsim) function proposed in [14] to estimate the relevance between a query and a passage:

$$\text{Relevance}(p, q) = \sum_{i \in \{1, \ldots, |\mathbf{E}_q|\}} \max_{j \in \{1, \ldots, |\mathbf{E}_p|\}} \mathbf{E}_{q_i}^{\top} \mathbf{E}_{p_j}, \tag{5}$$

where $|\mathbf{E}_q|$ and $|\mathbf{E}_p|$ denote the number of tokens in the query and the passage respectively. For a given query and a collection of passages, we compute the maxsim score of each passage with respect to the query using Equation (5) and then rank these passages based on the maxsim scores in the descending order. The top-K passages are selected as the relevant passages of the given query.

5 Experimental Setup

In this section, we first introduce the research questions guiding the remainder of the paper in Section 5.1. Subsequently, we introduce the datasets and the baselines used in our experiments in Section 5.3 and Section 5.3, respectively. Finally, we introduce the training and hyperparameter details in Section 5.4.

5.1 Research Questions

We aim to investigate the following research questions: (**RQ1**): How does the proposed ColLUKE perform compared with the baselines? (**RQ2**): Do the entities help to improve the ranking performance? (**RQ3**): What types of queries can benefit from our entity-enhanced ColLUKE model?

5.2 Datasets

We use the following datasets in our experiments: (1) **MS MARCO Passage** [23]: It is a widely used dataset for the passage retrieval task. The corpus has around 8.8 million passages. Its training set has approximately 530K (query, relevant passage) pairs over 503K unique queries. We use the MS MARCO training set to train our model. Moreover, we also evaluate on its development (Dev) set, which contains 7,437 query-passage pairs over 6,980 unique queries. (2) **TREC DL-2019/2020** [2, 3]: These two datasets are commonly used test sets for evaluating the retrieval model's performance. They share the same corpus as the MS MARCO Passage dataset but with human-evaluated query-passage relevance labels. There are 43 queries and 54 queries in the TREC DL-2019 and DL-2020 datasets, respectively. (3) **TREC DL-HARD** [22]: This is also a test set designed to evaluate retrieval models on complex queries. The dataset has 50 queries, where 23 queries come from TREC DL-2019/2020 that are considered to be challenging, and the other 27 queries are newly and independently assessed by the authors. (4) **HotPotQA** [39]: The aforementioned test query sets can be considered as belonging to the same domain as the training data since they use the same MS MARCO corpus. To evaluate models' performance in out-of-domain scenarios, we develop a new query set from the HotPotQA dataset. HotPotQA is a question answering dataset where each question requires finding and reasoning over multiple documents to answer. We use the questions in its test set as the query set and employ the provided Wikipedia corpus as the retrieval corpus. Following the DPR [13] setting, we consider a passage being relevant to a question if and only if the passage contains the answer to the question. Therefore,

Table 1. End-to-end retrieval results of our ColLUKE and baselines (* indicates p-value < 0.05 compared with ColRoBERTa).

Model	DL-HARD	MS MARCO (Dev)	HotPotQA	DL-2019	DL-2020
	nDCG@10	MRR@10	ACC@10	nDCG@10	nDCG@10
BM25	0.2743	0.1806	0.6017	0.4795	0.4936
ColBERT	0.3633	0.3586	0.6094	**0.7066**	**0.6895**
ColRoBERTa	0.3574	0.3511	0.5904	0.6828	0.6640
ColLUKE	**0.4018***	**0.3698***	**0.6819***	0.7047	0.6854

Table 2. Ablation study (* indicate p<0.05 compared with ColLUKE).

Model	DL-HARD	MS MARCO (Dev)	HotPotQA
	nDCG@10	MRR@10	ACC@10
ColLUKE	**0.4018**	**0.3698**	**0.7344**
w/o entity	0.3723*	0.3514*	0.6712*
append entity	0.3818*	0.3586*	0.6898*

in order to evaluate the performance of queries, we exclude questions for which the answers cannot be found within the corpus. As a result, we obtained 6,502 queries and the corpus has approximately 5.2M passages.

5.3 Baselines

We compare against the following baselines: (1) **BM25** [28]: This is a sparse retrieval model, which estimates relevance based on the term frequency (TF) and the inverse document frequency (IDF). (2) **ColBERT** [14]: ColBERT is a BERT-based late interaction model. It uses a BERT model to encode queries and passages. (3) **ColRoBERTa**: The difference between this model and ColBERT is that it uses a RoBERTa model instead of BERT to encode queries and passages.

5.4 Training and Hyperparameter Details

In our experiments, we use LUKE-base as the encoder model. Following previous work [13], we use cross entropy loss with in-batch negative sampling to train the model. We set the maximum length of queries and passages as 32 and 180, respectively and set the batch size as 32. During training, we use AdamW [15] with a 1% linearly scheduled warmup as the optimizer. The model is trained with a learning rate of 5e-6 for 200K steps. We follow the end-to-end ranking pipeline [14] to evaluate the performance. Specifically, we first build an index for the corpus and then use the ANN search to retrieve the top-K relevant passages for each query. Following previous work [25], we report MRR@10 on the MS MARCO Dev set and nDCG@10 on the TREC DL-2019, DL-2020 and DL-HARD datasets, which are the official evaluation metrics for these query sets. For the HotPotQA dataset, we follow the DPR [13] setting and report the

Table 3. Queries that exhibit improved performance (nDCG@10) when using our ColLUKE in comparison with ColBERT on the DL-HARD dataset.

Qid	Query	Entities	Improved
315637	how much does it cost to go to alabama university.	University of Alabama	0.4608
273695	how long will methadone stay in your system.	Methadone	0.2537
527433	types of dysarthria from cerebral palsy	Dysarthria, Cerebral palsy	0.2457
174463	dog day afternoon meaning	Dog Day Afternoon	0.1512
801118	what is supplemental security income used for	Supplemental Security Income	0.1044

Table 4. Queries that exhibit degraded performance (nDCG@10) when using our ColLUKE in comparison with ColBERT on the DL-HARD dataset.

Qid	Query	Entities	Degraded
177604	eating foods that are considered warm.	Food	0.5156
794429	what is sculpture shape space	-	0.1911
1056416	who was the highest career passer rating in the nfl	Passer rating	0.1696
1103153	who is thomas m cooley	-	0.1072
87452	causes of military suicide	Suicide attack	0.0276

top-k accuracies (ACC@k), which denotes the percentage of the top-k retrieved documents that contain the answer. We set k as 10 in our experiments.

6 Results and Analysis

6.1 RQ1: Overall Performance

In order to evaluate the performance of our ColLUKE on the passage retrieval task, we compare ColLUKE with BM25 and other bi-encoders in an end-to-end setting. The evaluation results are provided in Table 1, which shows the performance of different models on different query sets. The results show that our ColLUKE can achieve the best performance on three out of five datasets: DL-HARD, MS MARCO Dev set, HotPotQA, and has comparable performance with the best model on the other two datasets. Moreover, compared with Col-RoBERTa, which uses a similar RoBERTa backbone as our model, our ColLUKE achieves significantly better performance on the DL-HARD, MS MARCO Dev set and HotPotQA datasets. Furthermore, when compared with the best baseline model, i.e., ColBERT, on the in-domain query sets, our ColLUKE can improve the performance by 10.59% and 3.12% on the DL-HARD and MS MARCO Dev set, respectively. Regarding the out-of-domain query set, our ColLUKE demonstrates performance improvement of 11.89% on the HotPotQA dataset. Therefore, the experimental results indicate the effectiveness of the proposed ColLUKE model on the passage retrieval task.

6.2 RQ2: Effect of Entities

In order to examine the effect of entities in our model, we introduce two variants of our model: (1) **w/o entities**: In this variant, we remove the entities within both the queries and the passages and only use the tokens to estimate

the relevance. (2) **append entity**: In this variant, we do not use the entity embedding of the LUKE model. Instead, we simply append the surface forms of entities at the end of their corresponding mentions. For example, for a query "how long will methadone stay in your system" with entity "Methadone", the input sequence to the encoder is "how long will methadone **(Methadone)** stay in your system". The results of our model and these two variants are provided in Table 2, which shows their ranking performance on three test query sets: DL-HARD, MS MARCO Dev set and HotPotQA. The results demonstrate that our ColLUKE consistently achieves significantly better performance compared to these two variants on three query sets, which suggests the effectiveness of incorporating contextualised entity embeddings to enrich the dense embeddings of queries and passages, resulting in improved ranking performance. Moreover, when comparing the performance of **append entity** and **w/o entity**, the results show that **append entity** can achieve slightly better performance on three query sets. This result indicates that incorporating the surface forms of entities can also help to improve the ranking performance.

6.3 RQ3: Qualitative Analysis

Finally, we conduct qualitative analysis to understand what types of queries can benefit from our model. Specifically, we calculate the per-query performance of our ColLUKE and ColBERT on the DL-HARD dataset. We then investigate the queries that exhibit improved or degraded performance when using our ColLUKE compared with ColBERT. The improved queries and the degraded queries on the TREC DL-HARD dataset are provided in Table 3 and Table 4 respectively, which include information about the query, query entities and the difference in ranking performance between our model and ColBERT on each query. After a careful examination of these queries, we found that the improved queries are entity-centric, which means that these queries are focused on seeking specific information related to the identified entities. For example, the query "how much does it cost to go to alabama university" aims to retrieve information regarding the tuition fees of the entity "University of Alabama". Moreover, we found that for the queries that exhibit degraded performance, the entities are either incorrectly identified (e.g., Q87452) or missing some or all of them (e.g., Q794429, Q1056416, Q1103153). Therefore, the qualitative analysis indicates that the entity-centric queries can benefit most from our ColLUKE model.

7 Conclusion

In this paper, we propose ColLUKE, an entity-aware late interaction model for the passage retrieval task. In order to estimate the relevance between a query and a passage, ColLUKE first use an entity linking tool ELQ to identify the entities within the query and the passage. Subsequently, it uses LUKE to independently encode the query and the passage into entity-aware contextualised token embeddings by taking the texts and the entities within the texts as inputs.

These token embeddings are then passed to an interaction function to estimate the relevance between the query and the passage. We train our ColLUKE on the MS MARCO training set and evaluate its performance on five different query sets, which include four in-domain query sets and one out-of-domain query set. The results indicate that incorporating the entity information into dense embeddings can improve the performance of the late interaction model and that our ColLUKE can outperform ColBERT by up to 10.59% on the in-domain TREC DL-HARD query set and up to 11.89% on the out-of-domain HotPotQA query set. Our case study also shows that our ColLUKE model could benefit more for those entity-centric queries, which suggests its potential practicability in domain-specific (e.g. biomedical) applications.

References

1. Chatterjee, S., Dietz, L.: BERT-ER: Query-specific BERT entity representations for entity ranking. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 1466–1477 (2022)
2. Craswell, N., Mitra, B., Yilmaz, E., Campos, D., Voorhees, E.M.: Overview of the TREC 2019 deep learning track. In: Proceedings of Text REtrieval Conference (2020)
3. Craswell, N., Mitra, B., Yilmaz, E., Campos, D., Voorhees, E.M.: Overview of the TREC 2020 deep learning track. In: Proceedings of Text REtrieval Conference (2021)
4. Croft, W.B., Metzler, D., Strohman, T.: Search engines: Information retrieval in practice, vol. 520. Addison-Wesley Reading (2010)
5. Das, R., Godbole, A., Naik, A., Tower, E., Zaheer, M., Hajishirzi, H., Jia, R., McCallum, A.: Knowledge base question answering by case-based reasoning over subgraphs. In: International Conference on Machine Learning. pp. 4777–4793. PMLR (2022)
6. Devlin, J., Chang, M., Lee, K., Toutanova, K.: 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. pp. 4171–4186 (2019)
7. Fang, J., Meng, Z., Macdonald, C.: KGPR: Knowledge graph enhanced passage ranking. In: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management. pp. 3880–3885 (2023)
8. Gerritse, E.J., Hasibi, F., de Vries, A.P.: Entity-aware transformers for entity search. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 1455–1465 (2022)
9. Gonçalves, G., Magalhães, J., Xiong, C., Callan, J.: Improving ad hoc retrieval with bag of entities. In: Proceedings of the Twenty-Seventh Text REtrieval Conference (2018)
10. Hui, K., Zhuang, H., Chen, T., Qin, Z., Lu, J., Bahri, D., Ma, J., Gupta, J.P., dos Santos, C.N., Tay, Y., Metzler, D.: ED2LM: encoder-decoder to language model for faster document re-ranking inference. In: Findings of the Association for Computational Linguistics. pp. 3747–3758 (2022)
11. Izacard, G., Grave, E.: Leveraging passage retrieval with generative models for open domain question answering. In: Proceedings of the 16th Conference of the

European Chapter of the Association for Computational Linguistics. pp. 874–880 (2021)
12. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with gpus. IEEE Transactions on Big Data **7**(3), 535–547 (2019)
13. Karpukhin, V., Oguz, B., Min, S., Lewis, P.S.H., Wu, L., Edunov, S., Chen, D., Yih, W.: Dense passage retrieval for open-domain question answering. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. pp. 6769–6781 (2020)
14. Khattab, O., Zaharia, M.: ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In: Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval. pp. 39–48 (2020)
15. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations (2015)
16. Li, B.Z., Min, S., Iyer, S., Mehdad, Y., Yih, W.: Efficient one-pass end-to-end entity linking for questions. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. pp. 6433–6441 (2020)
17. Li, C., Yates, A., MacAvaney, S., He, B., Sun, Y.: PARADE: Passage representation aggregation for document reranking. ACM Transactions on Information Systems (2023)
18. Liu, X., Fang, H.: Latent entity space: a novel retrieval approach for entity-bearing queries. Information Retrieval Journal **18**, 473–503 (2015)
19. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: RoBERTa: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
20. Liu, Z., Xiong, C., Sun, M., Liu, Z.: Entity-duet neural ranking: Understanding the role of knowledge graph semantics in neural information retrieval. In: Gurevych, I., Miyao, Y. (eds.) Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics. pp. 2395–2405 (2018)
21. MacAvaney, S., Nardini, F.M., Perego, R., Tonellotto, N., Goharian, N., Frieder, O.: Efficient document re-ranking for transformers by precomputing term representations. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 49–58 (2020)
22. Mackie, I., Dalton, J., Yates, A.: How deep is your learning: the DL-HARD annotated deep learning dataset. In: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 2335–2341 (2021)
23. Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., Deng, L.: MS MARCO: A human generated machine reading comprehension dataset. arXiv preprint arXiv:1611.09268 (2016)
24. Nogueira, R., Jiang, Z., Pradeep, R., Lin, J.: Document ranking with a pretrained sequence-to-sequence model. In: Findings of the Association for Computational Linguistics. pp. 708–718 (2020)
25. Nogueira, R., Yang, W., Cho, K., Lin, J.: Multi-stage document ranking with BERT. arXiv preprint arXiv:1910.14424 (2019)
26. Pradeep, R., Liu, Y., Zhang, X., Li, Y., Yates, A., Lin, J.: Squeezing water from a stone: A bag of tricks for further improving cross-encoder effectiveness for reranking. In: Advances in Information Retrieval: 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10–14, 2022, Proceedings, Part I. pp. 655–670. Springer (2022)

27. Robertson, S.E., Walker, S., Jones, S., Hancock-Beaulieu, M., Gatford, M.: Okapi at TREC-3. In: Proceedings of The Third Text REtrieval Conference, TREC. vol. 500-225, pp. 109–126 (1994)
28. Robertson, S.E., Zaragoza, H.: The probabilistic relevance framework: BM25 and beyond. Foundations and Trends in Information Retrieval **3**(4), 333–389 (2009)
29. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (2016)
30. Shehata, D., Arabzadeh, N., Clarke, C.L.: Early stage sparse retrieval with entity linking. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. pp. 4464–4469 (2022)
31. Tran, H.D., Yates, A.: Dense retrieval with entity views. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. pp. 1955–1964 (2022)
32. Wang, H., Xu, Y., Yang, C., Shi, C., Li, X., Guo, N., Liu, Z.: Knowledge-adaptive contrastive learning for recommendation. In: Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining. pp. 535–543 (2023)
33. Wang, X., He, X., Cao, Y., Liu, M., Chua, T.S.: KGAT: Knowledge graph attention network for recommendation. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 950–958 (2019)
34. Wang, X., Macdonald, C., Tonellotto, N., Ounis, I.: Reproducibility, replicability, and insights into dense multi-representation retrieval models: from colbert to Col*. In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 2552–2561 (2023)
35. Xiong, C., Callan, J., Liu, T.Y.: Word-entity duet representations for document ranking. In: Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval. pp. 763–772 (2017)
36. Xiong, L., Xiong, C., Li, Y., Tang, K., Liu, J., Bennett, P.N., Ahmed, J., Overwijk, A.: Approximate nearest neighbor negative contrastive learning for dense text retrieval. In: 9th International Conference on Learning Representations (2021)
37. Yamada, I., Asai, A., Shindo, H., Takeda, H., Matsumoto, Y.: LUKE: Deep contextualized entity representations with entity-aware self-attention. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. pp. 6442–6454 (2020)
38. Yang, Y., Huang, C., Xia, L., Li, C.: Knowledge graph contrastive learning for recommendation. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 1434–1443 (2022)
39. Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W.W., Salakhutdinov, R., Manning, C.D.: HotpotQA: A dataset for diverse, explainable multi-hop question answering. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. pp. 2369–2380 (2018)
40. Yasunaga, M., Ren, H., Bosselut, A., Liang, P., Leskovec, J.: QA-GNN: reasoning with language models and knowledge graphs for question answering. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 535–546 (2021)
41. Yates, A., Nogueira, R., Lin, J.: Pretrained transformers for text ranking: BERT and beyond. In: Proceedings of the 14th ACM International Conference on web search and data mining. pp. 1154–1156 (2021)
42. Zhan, J., Mao, J., Liu, Y., Guo, J., Zhang, M., Ma, S.: Optimizing dense retrieval model training with hard negatives. In: Proceedings of the 44th International ACM

SIGIR Conference on Research and Development in Information Retrieval. pp. 1503–1512 (2021)
43. Zhang, X., Bosselut, A., Yasunaga, M., Ren, H., Liang, P., Manning, C.D., Leskovec, J.: GreaseLM: Graph reasoning enhanced language models. In: The Tenth International Conference on Learning Representations (2022)
44. Zhang, Y., Dai, H., Kozareva, Z., Smola, A.J., Song, L.: Variational reasoning for question answering with knowledge graph. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence. pp. 6069–6076 (2018)
45. Zhuang, H., Qin, Z., Jagerman, R., Hui, K., Ma, J., Lu, J., Ni, J., Wang, X., Bendersky, M.: RankT5: Fine-tuning t5 for text ranking with ranking losses. arXiv preprint arXiv:2210.10634 (2022)