Dense Retrieval with Entity Views

Hai Dang Tran
Max Planck Institute for Informatics
Saarbrucken, Germany
hatran@mpi-inf.mpg.de

Andrew Yates
University of Amsterdam
Amsterdam, Netherlands
Max Planck Institute for Informatics
Saarbrucken, Germany
a.c.yates@uva.nl

ABSTRACT

Pre-trained language models like BERT have been demonstrated to be both effective and efficient ranking methods when combined with approximate nearest neighbor search, which can quickly match dense representations of queries and documents. However, pre-trained language models alone do not fully capture information about uncommon entities. In this work, we investigate methods for enriching dense query and document representations with entity information from an external source. Our proposed method identifies groups of entities in a text and encodes them into a dense vector representation, which is then used to enrich BERT's vector representation of the text. To handle documents that contain many loosely-related entities, we devise a strategy for creating multiple entity representations that reflect different views of a document. For example, a document about a scientist may cover aspects of her personal life and recent work, which correspond to different views of the entity. In an evaluation on MS MARCO benchmarks, we find that enriching query and document representations in this way yields substantial increases in effectiveness.

CCS CONCEPTS

• **Information systems** → **Information retrieval**.

KEYWORDS

Ad Hoc Ranking; Dense Retrieval; Entity Representations

ACM Reference Format:
Hai Dang Tran and Andrew Yates. 2022. Dense Retrieval with Entity Views. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557285

1 INTRODUCTION

Search with dense representations has quickly become an effective alternative to statistical methods that use an inverted index (e.g., BM25) [23]. Ranking with dense representations is efficient with the support of approximate nearest neighbor search [19], which uses a specialized index to quickly identify the document representations (approximately) closest to a query representation. These representations are produced by encoding queries and documents (during an indexing step) as dense vectors using a pre-trained language model (PLM) like BERT [15]. To give a concrete example of their success, dense retrieval approaches have performed well on the TREC Deep Learning collections [9–11] in terms of both efficiency and effectiveness. For example, a BERT-based model trained with topic-aware sampling (TAS) [18] improves over a tuned BM25 approach by at least 35% nDCG@10 on the 2019 and 2020 TREC Deep Learning track while having low query latency.

Limitations of the state-of-the-art. However, work has found that the PLM models powering such approaches do not capture full information about real world entities [17]. Especially for uncommon entities, this limitation becomes more clear since PLMs both have difficulty disambiguating entities with similar surface forms [17] and are less likely to have seen information about these entities during pretraining. Even the most powerful PLMs cannot capture information about an entity that was not present in the data or that emerged after the PLM was produced.

Approach. These limitations motivate approaches for encoding information about entities outside of the PLM itself. We investigate techniques for enriching the representations of query and document text using such entity representations. In our approach, a textual (query or document) representation from a PLM is combined with an entity representation derived from embeddings of the entities present in the input (query or document) text. We leverage external entity embeddings [48] as a lightweight way to incorporate information about entities. Rather than being tied to a PLM, these embeddings are created and stored independently.

Method. A document may contain many entities; we find that the straightforward approach of creating a single vector representation of all a document's entities does not perform well. Instead, we propose a multiple representation approach in which we create several entity representations with respect to different clusters of entities in a text. This can be understood as creating different views of a document that focus on different groups of entities. These entity views are then combined with a textual representation of the document and indexed for approximate nearest neighbor (ANN) search. We find that this is an effective and efficient strategy: enriching representations from a strong BERT-based model (TAS) yields significant improvements on the TREC Deep Learning [11] and DL-HARD [30] and MS MARCO [8] datasets.

Example. Figure 1 illustrates the entity views present in a short passage about the scientist Lilli Hornig. The first view is a generic one that captures Lilli Hornig without focusing on any specific aspect. The second and third views capture entities related to Lilli's scientific work on the Manhattan project, while the remaining

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557285

two views capture entities related to Lilli's studies at Bryn Mawr College. Our approach captures these different views in order to match different aspects of an entity. For example, the query *Lilli Hornig's work at Los Alamos* better matches the second and third views related to the Manhattan project, whereas a query about Lilli's personal life or education better matches the last two views.

Contributions. Our salient contributions are:

- a novel approach for improving search by enriching query and document representations with entity views that represent different groups of the entities in a document,
- exploration and analyses of alternative approaches for enriching query and document representations with entities,
- an extensive evaluations of the proposed approach and baselines on four datasets, with effectiveness improvements up to a 12% increase in nDCG with a TAS BERT base,
- and a release of our code [1] for reproductions of our approach.

2 RELATED WORK

We describe two lines of related work: efforts to leverage deep learning to create improved ranking methods and efforts to use entities to improve ranking.

Neural IR. State-of-the-art neural approaches to ranking can be divided into two categories: bi-encoder methods focus on building separate vector representations of queries and documents that are compared to calculate relevance scores (e.g., [18, 20, 21, 46, 49]), whereas cross-encoder methods aim to measure the similarity between query and document directly by taking them both as inputs (e.g., [2, 13, 22, 29]). Though highly effective, the cross-encoder approach is inefficient because all candidate documents must be processed at inference time. The bi-encoder approach ("dense retrieval") can be highly efficient when coupled with ANN search [19], because an ANN index can quickly identify the most relevant documents for a given query vector (i.e., those documents with the highest inner products or cosine similarities). Dense document representations are computed and stored in an ANN index at indexing time, which is independent of the query. In this work, we follow the bi-encoder approach to make use of its superior efficiency.

Sharing the same idea of building representations for documents at indexing time and being independent from query with bi-encoder, ColBERT [21] extends from text-level to term-level vector representations, where each term in the query or document has one representation. While effective, this approach introduces a new inefficiency: storing a vector representation of each document term requires substantially more disk space than storing a single representation for the entire document. To fix this issue, a representation compression technique is implemented to cut down space requirement in ColBERTv2 [37].

Leveraging entities in IR. With the development of knowledge bases (KBs) like Yago [38], DBpedia [3], Freebase [5], and Wikidata [40], several lines of research have investigated how these resources can be leveraged to make information systems smarter. Prior to neural methods, researchers have investigated a variety of ways to make use of facts and entities from KBs. For example, queries can be extended with entity descriptions [47], a range of features can be derived from a KB and used as features with a learning to rank method [14], and entities can be used to create vector representations in which each dimension corresponds to an entity [25]. Balog [4] provides an extensive survey of pre-neural approaches leveraging entities.

In the context of neural IR methods, researchers have primarily investigated how entities can be used by pre-BERT ranking methods. Researchers have investigated how pre-trained entity embeddings can be used in an interaction-based ranking method [43, 45] and how they can be jointly learned with word embeddings [26]. We employ a kernel pooling component following these approaches [43, 45]. Crucially, these approaches are interaction-based and cannot be applied directly to dense retrieval. Contemporaneous work [7, 16] has demonstrated that interaction-based (cross-encoder) PLMs can benefit from entity representations in the context of entity search.

ERNIE [39] is a prominent PLM that incorporates entity knowledge through one of its pretraining tasks (i.e., predicting the surface form of a masked entity). ERNIE has been applied to ranking with strong results as part of the larger RocketQA [34] method. While there are many effective performance enhancement techniques implemented in RocketQA, in the current work we focus on studying how entity knowledge specifically can improve dense retrieval. To do this, we fine-tune the model used in RocketQA (i.e., ERNIE) for search and evaluate its performance, finding that our entity enrichment approach is complementary to ERNIE's entity knowledge.

Other approaches have been proposed for incorporating entity knowledge into a pre-trained language model in a general setting (not for ranking) [24, 32, 50]. Injecting entities into BERT has yielded significant improvements in NLP tasks such as relation extraction and entity typing [32, 33] or the GLUE [41] benchmark [50]. While these approaches can be used to add information about entities to a pre-trained language model, they suffer from the same issues with entities that were not seen during training (discussed in the introduction). Motivated by the success of prior work, we focus on approaches that overcome these drawbacks by creating entity-representations independently of a pre-trained language model.

3 BACKGROUND

Textual Representation. Pre-trained language models like BERT can encode the semantic meaning of text as a dense vector representation. This forms the basis for the bi-encoder approach of ranking with dense representations, which is the basis for the textual representations we enrich in our approach. Specifically, we build on a distilled BERT-based [36] model trained with topic aware sampling (TAS) [18], which ensures that queries in the same batch are similar to each other (and thus that hard negative examples are chosen). We choose this TAS BERT model to support text understanding for two reasons. First, this approach was demonstrated to have state-of-the-art effectiveness while being highly efficient, because it is compatible with ANN search. Second, this approach is faster to train and to process passages at indexing time than larger BERT variant, while still being effective. To create textual representations of queries and passages, we follow the TAS BERT process [18].

Definition 3.1 (Textual Representation). Given a query or passage as text t, the textual representation $R_{text}(t)$ of t is formed by passing

[1] See https://github.com/haidangtran1989/EVA and https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/neural-ir

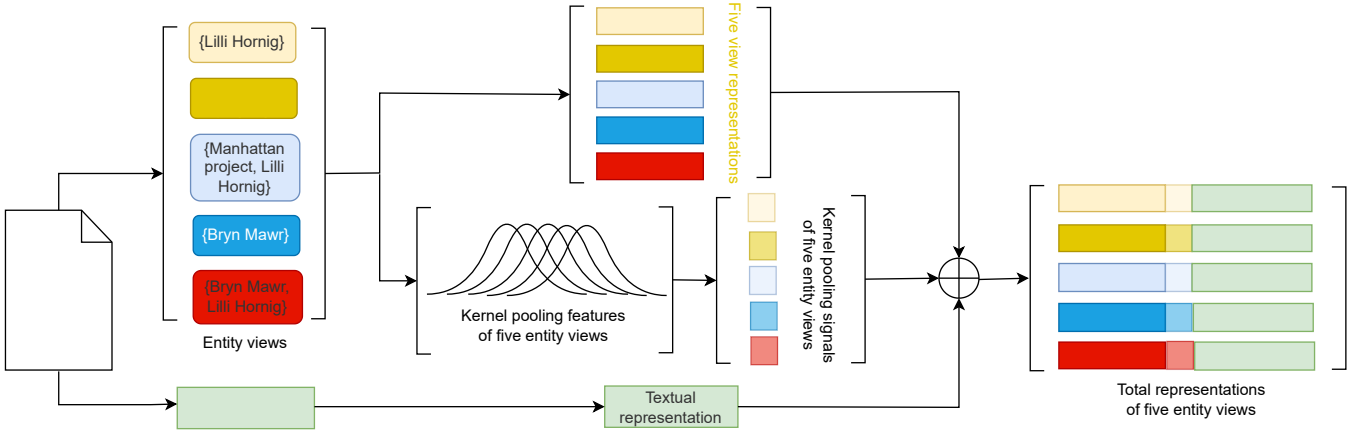

Figure 1: Overview of EVA with multiple representations. Entity clusters such as {Lilli Hornig}, {Manhattan project}, {Manhattan project, Lilli Hornig}, {Bryn Mawr} and {Bryn Mawr, Lilli Hornig} can be understood as different entity views of the passage. EVA generates one total representation for each view, which enriches a textual representation with the entities present.

t to a PLM: $R_{text}(t) = \text{PLM}_{CLS}(t)$, where PLM can be TAS or any other BERT-like model.

Entities and Similarity. We leverage pretrained wikipedia2vec embeddings [48] to represent entities and words. Similarity between two embeddings is calculated with cosine similarity. We say two entities are related to each other if their similarity is above some predefined threshold. Later, we will build clusters of related entities, which are subsets of one or more related entities in a given passage.

Kernel Pooling. Given a set of entities X and a passage p, let Y be a set of entities and non stop-words in p. The importance degree of X in context p can be estimated by the number of high similarities between elements of X and Y. The kernel pooling approach [44], which has also been applied to entity/word similarities [43, 45], can be leveraged to detect whether X is important to p. To do so, it is necessary to define the relatedness matrix between X and Y:

Definition 3.2 (Relatedness Matrix). Given the above sets X and Y, their relatedness matrix $T \in \mathbb{R}^{|X|\times|Y|}$ is defined as $T_{i,j} = sim(X_i, Y_j)$ where X_i and Y_j are i-th and j-th elements of X and Y, respectively.

The ranking score of this KNRM approach [44] can be built based on the relatedness matrix. Following the original work:

Definition 3.3 (KNRM Score). The KNRM score $S_{knrm}(X, p)$ of X and p is defined as follows:

$$S_{knrm}(X, p) = tanh(w_{knrm}^T \phi(X, p) + b_{knrm})$$

$$\phi(X, p) = \sum_{i=1}^{|X|} log\vec{Z}(R_i)$$

$$\vec{Z}(R_i) = \langle Z_1(R_i), ..., Z_K(R_i)\rangle$$

$$Z_l(R_i) = \sum_{j=1}^{|Y|} exp(-\frac{(T_{i,j} - \mu_l)^2}{2\sigma_l^2})$$

where K, μ and σ are hyper-parameters (see Section 5) while w_{knrm} and b_{knrm} are a learned vector and bias.

KNRM can be viewed as creating differentiable histograms. This approach builds kernel pooling feature $\phi(X, p)$ by measuring how elements in R are distributed around and close to K different levels of similarities μ. We incorporate this kernel pooling feature in our approach to prioritize passages where query entities are important.

4 METHODOLOGY

Given a collection of passages and a query, our goal is to identify the top relevant passages for the query. To do so, we propose the EVA method, which stands for entity views in dense retrieval. The overview of EVA is described in Figure 1. We use a novel multiple entity view technique to convert each passage into several total representations, which each emphasize a different view of the passage by focusing on different entity clusters. For example, the five views in Figure 1 highlight different aspects of Lilli Hornig, such as her work (*Manhattan project*) and personal life (*Bryn Mawr*). We calculate one total representation for each view, consisting of:

- a textual representation encoding the semantic meaning of a passage using a BERT-based encoder,
- an entity representation encoding information about the entities in the entity view,
- and, optionally, a kernel pooling signal.

While each passage may have multiple entity views (and thus multiple representations), for queries we use only one entity cluster containing all query entities. EVA follows the common bi-encoder approach where a relevance score is calculated as the dot product of two total representations from query and passage. To ensure the efficiency of EVA, we leverage an ANN search library [19] optimized for the bi-encoder approach. In Section 4.1, we begin by building a single representation per passage. In Section 4.2 we expand from single to multiple representations and discuss in detail.

4.1 Single Representation

Existing bi-encoder approaches produce relevance scores by considering the semantic matching between query and document representations; we additionally consider separate representations of

Beispiel für den Algorithmus vorzeigen

the entities in the texts and a KNRM signal. That is, our approach involves combining an entity representation and KNRM score with a textual representation obtained from a pretrained language model like TAS BERT. Together with semantic understanding from this BERT model, entity representation provides knowledge about real world entities. Besides, KNRM score supports ranking model to detect whether query entities are important to the passage.

Single Entity Representation. The single representation approach produces one entity representation for a given query and one entity representation for each passage. In both cases, we identify entities present in the text (using an entity linker) and subsequently obtain the pretrained embedding for each entity. Creating an entity representation for the query is straightforward since queries are relatively short and focused.

Definition 4.1 (Query Entity Representation). Let $E_{all}(q)$ be the set of all entities mentioned in query q. The query entity representation $R_{all}(q)$ is then the average embedding of entities in $E_{all}(q)$.

The most straightforward approach for creating an entity representation of a passage is to include all entities present in the passage, which is analogous to how $R_{all}(q)$ is created:

Definition 4.2 (Query-independent Passage Entity Representation). After identifying the set $E_{all}(p)$ of all entities mentioned in p, the query-independent passage entity representation $R_{all}(p)$ is the average embedding of entities in $E_{all}(p)$.

In this query-independent approach, the average pooling may not be focused on the query's information need if p includes entities in many topics. There also may be wrong entity detection in the passage p, which potentially results in noise in the entity representation. This passage representation is used in EVA Single.

Query-aware Total Representation. Alternatively, we can make the assumption that q is known in order to quantify the effect of excluding off-topic entities. With this approach, we select only the entities in p that are close to those in q before performing average pooling. This approach enables a passage's entity representation to focus on the query and ignore unrelated entities.

Definition 4.3 (Query-aware Passage Entity Representation). Let $E_{focus}(p)$ be the set of passage entities which have maximum similarity with query entities. The query-aware passage entity representation $R_{focus}(p)$ is average embedding of entities in $E_{focus}(p)$. This procedure is described in Algorithm 1. The parameter α is a predefined threshold to decide whether a query entity has any close entity in the passage.

To reuse the query-aware passage entity representation, we still need to assume query q is known in the rest of this section. We will propose a solution to remove this assumption in Section 4.2.

Before being integrated with a textual representation, the entity representation is projected using a learned matrix W_{entity} to create a transformed entity representation:

Definition 4.4 (Transformed Entity Representation). We define the transformed entity representation $R_{trans}(t)$ of text t as:

$$R_{trans}(t)^T = \begin{cases} R_{all}(t)^T W_{entity} & \text{if } t \text{ is query} \\ R_{focus}(t)^T W_{entity} & \text{if } t \text{ is passage} \end{cases}$$

Input: Query q and passage p
Output: Query entity representation for q and query-aware passage entity representation for p

$E_{all}(q)$ = set of entities in q
$R_{all}(q)$ = average embedding of entities in $E_{all}(q)$
$E_{focus}(p) = \{\}$
for e_q *in* $E_{all}(q)$ **do**
  e_p = entity in p having the maximum Cosine similarity *max* with e_q
  if $max > \alpha$ **then**
    $E_{focus}(p) = E_{focus}(p) \cup e_p$
  end
end
$R_{focus}(p)$ = average embedding of entities in $E_{focus}(p)$
return $R_{all}(q), R_{focus}(p)$

Für jede Entity aus der Query suche Entity mit der maximalen Similarity aus Passage. Falls der Wert der Similarity > alpha, dann füge Entity aus Passage zum Set hinzu

Algorithm 1: Query-aware passage entity representation

To create a query-aware total representation of a query or passage, the text's transformed entity representation can be combined with its textual representation as follows.

Definition 4.5 (Query-aware Total Representation). Formally, query-aware total representation $R_{total}(t)$ of query or passage t is:

$$R_{total}(t) = R_{text}(t) \oplus R_{trans}(t)$$

where $\oplus$ is concatenation operator that we use to combine textual and transformed entity representations.

The method which uses this total representation for ranking is EVA Single-QA. In later experiments (Section 6.3) we additionally consider other integration approaches for combining the textual and transformed entity representations (e.g., summation rather than concatenation). With the goal of using kernel pooling to detect the importance of query entities in the passages, we aim to combine KNRM with $R_{total}(t)$.

Definition 4.6 (Kernel Pooling Signal). Given a set of entities X, we define kernel pooling signal $S_{kps}(X, t)$ of X with the text t as:

$$S_{kps}(X, t) = \begin{cases} 1 & \text{if } t \text{ is query} \\ S_{knrm}(X, t) & \text{if } t \text{ is passage} \end{cases}$$

Definition 4.7 (Query-aware Total Representation with Kernel Pooling). The query-aware total representation with kernel pooling $R_{total\_knrm}(t)$ of text t is:

$$R_{total\_knrm}(t) = R_{total}(t) \oplus S_{kps}(E_{all}(q), t)$$

This version of total representation is leveraged in EVA Single-QA-KNRM. To perform ranking, the dot product $\otimes$ of the total representations for query and passage are used as relevance scores $S_{rel\_knrm}(q, p)$:

$$S_{rel\_knrm}(q, p) = R_{total\_knrm}(q) \otimes R_{total\_knrm}(p) =$$
$$(R_{text}(q) \otimes R_{text}(p)) + (R_{trans}(q) \otimes R_{trans}(p)) + S_{knrm}(E_{all}(q), p)$$

This formula indicates why one is used as the kernel pooling signal forthe query: we want to add a query entities-passage KNRM score to the final ranking score of Single-QA-KNRM.

The formulas for total representation and relevance score describe our variant EVA Single-QA ranking models. Entity embeddings obtained from Wikipedia2vec are frozen in our model. Meanwhile the weights of components such as TAS BERT, W_{entity}, w_{knrm}, b_{knrm} are co-learned to produce the total representation in method EVA Single-QA-KNRM. In EVA Single-QA, only TAS BERT and W_{entity} are co-learned to build ranking model. Training details and hyperparameters are presented later in Section 5.3.

4.2 Multiple Representations

Thus far we have assumed that only a single passage entity representation is created. This means that we either face the off-topic entity issue as in EVA Single or the representation must be created at query time as in EVA Single-QA. In this section we describe an approach to create multiple entity representations for each passage, which mitigates the issue of including off-topic entities without requiring that representations are built at query time.

Multiple Entity Views. Observe that entities in $E_{all}(q)$ should be related to each other since q should describe a coherent information need. Besides, from Algorithm 1 it can be seen that every entity in a relevant passage's $E_{focus}(p)$ is similar to at least one entity in $E_{all}(q)$, which results in relatedness among entities in $E_{focus}(p)$ itself. Thus it is not necessary to wait until q is known to identify a set of entities $E_{focus}(p)$. Instead, we can find all possible clusters in the passage p at indexing time and the query-aware $E_{focus}(p)$ should be close to one of them.

Let M be the upper bound for $|E_{all}(q)|$. In Algorithm 1 each entity in $E_{all}(q)$ corresponds to at most one entity in $E_{focus}(p)$, so $|E_{all}(q)| \geq |E_{focus}(p)| \Rightarrow M \geq |E_{focus}(p)|$. When analyzing training dataset in Section 5.2, it can be seen that in more than 99% queries, the number of entities is upper bounded by 2, thus $M = 2$ is a reasonable default. Since M is a small value, we can quickly enumerate every subset C of entities in p with size $l \leq M$ and check whether C is a cluster. The entity cluster C can be leveraged to replace $E_{focus}(p)$ for building total representations; this key idea allows us to remove the known query assumption.

Bsp.

Example 4.8. In the short passage "Lilli Hornig worked in Manhattan project, studied in Bryn Mawr", there are three entities: *Lilli Hornig, Bryn Mawr* and *Manhattan project*. In this example, with parameter $M = 2$, five entity clusters of maximum size 2 can be formed such as *{Lilli Hornig}*, *{Bryn Mawr}*, *{Manhattan project}*, *{Lilli Hornig, Manhattan project}* and *{Lilli Hornig, Bryn Mawr}*. Note that one entity may belong to many different clusters, e.g. *{Lilli Hornig}* and *{Lilli Hornig, Bryn Mawr}*. However, *{Bryn Mawr, Manhattan project}* is not a cluster because *Bryn Mawr* and *Manhattan project* are not related to each other.

Kurz: Wähle Paare von Entities aus Passage und prüfe auf Similarity. Falls > alpha, dann gilt das Cluster als relevant

While we can re-use the total representations of queries from Section 4.1, passage representations need to be computed in a way that is independent of the query.

Definition 4.9 (Transformed Cluster Representation). Given passage p and an entity cluster C in p, let $R_{cluster}(C)$ be an average embedding of entities in C. The transformed cluster representation $R_{trans\_cluster}(C)$ of C is then:

$$R_{trans\_cluster}(C)^T = R_{cluster}(C)^T W_{entity}$$

where W_{entity} is the learnt matrix after the training for method EVA Single-QA in Section 4.1.

Definition 4.10 (Cluster Total Representation). Given passage p and an entity cluster C in p, we define the cluster total representation $R_{total\_cluster}(C, p)$ of passage p with cluster C as:

$$R_{total\_cluster}(C, p) = R_{text}(p) \oplus R_{trans\_cluster}(C)$$

The corresponding query total representation is the same as in EVA Single-QA. Thus, as described, we have a single total representation per query and multiple total representations per each passage (one per cluster). For simplicity, the weights in TAS BERT and W_{entity} are all reused after the training for EVA Single-QA, without further fine-tuning. Detailed steps for generating multiple total representations are described in Algorithm 2, which are conducted at indexing time. After the cluster total representations are available, they are added to ANN index.

Input: Passage p
Output: Multiple cluster total representations of p
$E_{all}(p)$ = set of all entities in p
$clusters = \{\}$
for *every non-empty subset* $C \subset E_{all}(p)$ *with size* $l \leq M$ **do**
  if $l == 1$ *or (every pair of entities in C has Cosine similarity* $> \beta$*)* **then**
    $clusters = clusters \cup C$
  end
end
$total\_reps = \{\}$
for C *in clusters* **do**
  $R_{total\_cluster}(C, p)$ = cluster total representation of p with cluster C
  $total\_reps = total\_reps \cup R_{total\_cluster}(C, p)$
end
return $total\_reps$

Algorithm 2: Multiple cluster total representations of passage

Intuitively Algorithm 2 scans through the passage and finds entity clusters, which can be understood as different document views that emphasize different entities. The step of building total representation per entity cluster, at its core, is a plan to maximize the chance that query entities are matched against relevant passages. Compared to using a single query-aware representation, ranking time is significantly improved because time-consuming representation generation (including heavy BERT inference) is moved from query time to index time. Besides, our approach expands from one to multiple representations, which tackles a fundamental capacity issue in representation learning [28].

Cluster total representation $R_{total\_cluster}$ is used in **EVA Multi**, which is one of our main methods. After ANN search in EVA Multi, there can be many top ranked total representations of the same passage. In this case, we deduplicate by selecting the one with the highest relevance score for the final ranking.

Kernel Pooling Integration. With the aim to prioritize ranking of passages dedicated to query entities, we integrate KNRM into EVA Multi. To do that, we use the following total representation:

Definition 4.11 (Cluster Total Representation with KNRM). Given passage p and an entity cluster C in p, we define the cluster total representation with KNRM $R_{total\_cluster\_knrm}(C, p)$ of passage p and cluster C as follows:

$$R_{total\_cluster\_knrm}(C, p) = R_{total\_cluster}(C, p) \oplus S_{kps}(C, p)$$

where weights of TAS BERT, W_{entity}, w_{knrm}, b_{knrm} are all re-used after EVA Single-QA-KNRM training. The corresponding query total representation formula is the same as in EVA Single-QA-KNRM.

Note that this type of total representation calculates the kernel pooling signal using a cluster C rather than all query entities. The total representation is independent from a query and can be built at index time. However, this formula prioritises the ranking of passages in which cluster C is important. To ensure that query entities are dedicated in top ranked passages, we need to choose the cluster in which the entities are close to query entities. Such idea motivates the definition of cluster-query entities attention:

Definition 4.12 (Cluster-Query Entities Attention). Given a query and an entity cluster C, C attends to query entities if each entity in C has high similarity (above a predefined threshold α) with at least one query entity and vice versa.

With this attention technique, in the final search result we retain only top ranked total representations $R_{total\_cluster\_knrm}(C, p)$ of cluster C and passage p from ANN search such that C attends to query entities. If many total representations of the same passage fulfill this condition, then we obviously pick the one with the highest relevance score. The combination of EVA Multi and KNRM and cluster-query entities attention forms **EVA Multi-KNRM**, which is our second main method in this work.

5 EXPERIMENTAL SETUP

We implement our approach in TensorFlow [1] and initialize our models from the TAS BERT checkpoint [18]. Entities are represented by 100-dimension embeddings obtained from Wikipedia2vec [48]. We use FAISS [19] to index and perform approximate nearest neighbor search on the passage representations produced by our models. In the rest of this section, we describe in detail the data, training procedure, and hyperparameters used in our experiments. We will release our code and experimental outputs upon publication.

5.1 Datasets

We evaluate our approach using four datasets built on top of the MS MARCO passage collection[2]: the MS MARCO development queries, the TREC Deep Learning (DL) Track 2019 [11], DL 2020 [9], and DL HARD [30]. These datasets consist of queries issued to a Web search engine and related passages extracted from Web pages. We build upon the Dexter [6] entity linker in order to annotate texts with entities, i.e. detecting and linking entities in texts to Wikipedia. The original Dexter implementation is static and extracts entities based on their commonness. We additionally enable context awareness capability to improve entity disambiguation in Dexter. We use this improved version to automatically annotate all the queries and passages used in our experiments. On the TREC DL 19, DL 20, and DL HARD datasets, graded relevance judgments are used with a four-point scale where 0 is non-relevant and 3 is perfectly relevant. Following the TREC Deep Learning track, we calculate nDCG@10 using these graded judgments and calculate MRR@10 and MAP@1000 using binarized judgments where only labels of 2 and 3 are treated as relevant.

[2]We do not consider the Deep Learning 2021 dataset, which uses a different collection.

# Entities	Training Queries		Testing Queries	
	Count	Fraction	Count	Fraction
0	130353	0.435	3442	0.483
1	149073	0.497	3232	0.454
2	19207	0.064	416	0.058
3+	1367	0.004	37	0.005
Total	300000		7127	
Average	0.640		0.587	

Table 1: Summary statistics of the queries.

# Entities	Training Passages		Collection Passages	
	Count	Fraction	Count	Fraction
0-2	201932	0.337	3309263	0.375
3-5	261200	0.435	3731425	0.422
6-7	82416	0.137	1103501	0.125
8+	54452	0.091	697634	0.078
Total	600000		8841823	
Average	3.87		3.63	

Table 2: Summary statistics of the passage collection.

If the document frequency of an entity on Wikipedia is high (i.e. bigger than a threshold $\delta = 3000$) then such entity is common, otherwise it is uncommon. Annotating common entities is not needed in this work since it is likely that BERT was already trained on the corresponding entity mentions. Thus we only keep annotations of **uncommon** entities in training and testing data, so that the entity representations and computations are not wasted on common entities that BERT is likely to recognize.

5.2 Entity Summary Statistics

We analyze entity detection for queries and passages in the above datasets by calculating the number of unique entities per query or passage text. Note that we only retain uncommon entities when analyzing and doing experiments. Table 1 shows summary statistics for the MS MARCO training queries and the test queries from TREC DL 19, TREC DL 20, DL HARD and MS MARCO development set. It can be seen from the training data that most queries have a single entity detected, with a small number of queries having two or more entities. Based on these statistics we decide to choose $M = 2$ to generate multiple cluster total representations.

Table 2 shows summary statistics on the MS MARCO passages. Most passages have from 1 to 7 entities, and on average approximately 4 entities. This average number affects the average number of cluster total representations per passage. In Algorithm 2, we have parameters $\beta = 0.9$ and $M = 2$, which results in 3.7 representations per passage on average after indexing.

5.3 Training and ANN Settings

The above TAS BERT is 6 layer DistilBERT [36], which has 768 dimensions, and was pretrained on the MS MARCO dataset. The entity vector has 100 dimensions and size of the learned matrix W_{entity} is 100×100 in our setup.

We use AdamW [27] optimizer to train our EVA models. The initial learning rate is 3×10^{-5} and the warm up takes 3% of the learning process. These models are trained using 4 Quadro RTX 8000 GPUs in parallel with mixed mode precision where each GPU handles batch size of 128. Our training is in two epochs with 300 thousand training triples from MS MARCO and pairwise hinge loss [42]. For example, with EVA Single-QA-KNRM our loss function is $\sum max\{0, 1 - S_{rel\_knrm}(q, pos) + S_{rel\_knrm}(q, neg)\}$ where q, pos, neg are respectively query, positive passage, negative passage of a particular training triple.

For kernel pooling hyper-parameters, we set the number $K = 6$ with similarity levels $\mu_1 = 0.1$, $\mu_2 = 0.3$, $\mu_3 = 0.5$, $\mu_4 = 0.7$, $\mu_5 = 0.9$ and $\mu_6 = 1.0$. Besides, $\sigma_1 = ... = \sigma_4 = \sigma_5 = 0.1$ for soft matches and $\sigma_6 = 0.001$ for capturing hard exact matches.

Our approach and dense retrieval baselines are indexed with FAISS. These models share the same FAISS settings: the number of clusters *nlist* when building index is $4 \times \sqrt{N}$ where N is the total number of vectors in such index (this is the recommended default). Employing these approximations reduces metrics compared to using an exact index as in some of the original works (e.g., from 0.340 to 0.334 MRR on MS MARCO Dev with TAS BERT).

6 EVALUATION

We evaluate a range of baselines and EVA variants to quantify the impact of enriching representations with multiple entity views. The primary EVA variants are built on top of the state-of-the-art TAS BERT base model, providing a comparison point to show how much incorporating entities can improve ranking. We also include ERNIE Multi, which combines the ERNIE [39] base model with entity views, in order to demonstrate that our approach brings improvements even with a PLM designed to capture entity information.

- **BM25** is leverages exact match features to rank passages [35].
- **TAS BERT** is a baseline that performs ranking using only the underlying BERT-based model that produces textual representations for EVA. This is a state-of-the-art bi-encoder model on the TREC Deep Learning passage collections [18].
- **ANCE** is a state-of-the-art bi-encoder model with a training procedure that iteratively mines hard negative examples from the entire document collection [46].
- **EVA Single** uses *query-independent passage entity representation* to create a single total representation $R_{single\_total}(t)$ of text t (t could be a query or passage):

$$R_{single\_total}(t) = R_{text}(t) \oplus (W^T_{entity} R_{all}(t))$$

  With this method, entity attention technique is disabled, so an average of all entity embeddings is used. The matrix W_{entity} is learned from MS MARCO training data and we index single total representations of passages with FAISS to boost up efficiency.
- **EVA Single-QA** uses *query-aware total representation* to encode passages, which is described in Section 4.1. Given that this approach must compute entity representations at query time, it is implemented as a reranking step after BM25 retrieval stage.
- **EVA Single-QA-KNRM** is a variant of EVA Single-QA where a *query-aware total representation with kernel pooling* is leveraged. It is also used in reranking after BM25 retrieval.
- **EVA Multi** is our main method in Section 4.2 which creates multiple total representations for each passage. We set the parameters $M = 2$ and $\beta = 0.9$. On average the number of total representations for each passage is 3.7.
- **EVA Multi-KNRM** leverages a *kernel pooling signal* in total representation. This method shares the same other settings with EVA Multi and is the second main method in Section 4.2.
- **BM25 + T5 (Zero-Shot)** uses T5 as a cross-encoder [31] to rerank the top 1000 results from a BM25 first-stage ranking.
- **ERNIE Tuned** is initialized as Baidu ERNIE version 2 model [39], which already had knowledge about entities thanks to an entity masking technique. Later we fine-tune this model for search using the same training data as in EVA methods.
- **ERNIE Multi** has the same setting as EVA Multi except that it uses the Baidu ERNIE [39] instead of TAS BERT to encode textual representation. We fine-tune this model using the same training data as the EVA methods.
- **Best Reported** shows the best result from the MS MARCO leaderboard or from the corresponding paper [9, 12, 30].

6.1 Effectiveness

The results of evaluating the above methods are shown in Table 3. The methods are divided into low latency and higher latency groups where the former requires the average query latency on all datasets below 100ms.

RQ1: What is the benefit of enriching BERT's representations with entities? A PLM is unlikely to encode all facts about entities, especially uncommon or emerging entities (e.g., a new movie or product). Thus, knowledge about entities and how important they are in a passage context should be complementary to BERT. This is supported by results in Table 3 where the nDCG@10 of both EVA Multi variants are substantially higher than those of TAS BERT. This indicates that the introduction of entities is a significant improvement. Comparing EVA-Multi with the Multi-KNRM variant, we see that the kernel pooling signal usually results in a slight improvement. This indicates that the multi-view entity representation is the major factor providing improvements over TAS BERT.

We see a similar result when comparing ERNIE Tuned with ERNIE Multi. Incorporating entities with ERNIE Multi consistently improves over the base ERNIE Tuned model across datasets, with significant improvements on all but DL HARD. Note that before being fine-tuned for search task, ERNIE Tuned already contained some knowledge about entities due to its specialized pre-training [39]. This shows that the way we incorporate entity views is complementary to ERNIE's pre-training techniques.

RQ2: Do we need multiple representations per passage? If multiple passage representations are not required (e.g., because all entities in a passage are likely to be related anyway), then we would expect the EVA Single method to perform well. From Table 3, it can be seen that metrics of EVA Single are lower than those of both Multi

Methods	Latency	TREC DL 19			TREC DL 20			DL HARD			MS MARCO Dev		
	(ms)	nDCG	MRR	MAP	nDCG	MRR	MAP	nDCG	MRR	MAP	nDCG	MRR	MAP
Low latency (<100 ms)													
BM25	13	0.506	0.702	0.301	0.480	0.653	0.286	0.285	0.465	0.159	0.228	0.184	0.193
ANCE	25	0.621	0.763	0.361	0.605	0.786	0.373	0.335	0.446	0.193	0.368	0.311	0.317
ERNIE Tuned	29	0.574	0.728	0.326	0.573	0.760	0.348	0.287	0.388	0.163	0.320	0.267	0.274
ERNIE Multi	70	0.669†	0.822	0.422†	0.631†	0.891†	0.394†	0.329	0.452	0.198	0.344†	0.291†	0.296†
TAS BERT	28	0.693	0.835	0.442	0.673	0.812	0.451	0.360	0.472	0.224	0.395	0.334	0.340
EVA Single	40	0.672	0.853	0.429	0.642	0.813	0.428	0.363	0.481	0.224	0.374	0.316	0.322
EVA Multi	76	0.733	0.853	**0.483**	**0.694**	**0.855**†	**0.456**	0.397†	0.521†	0.240	**0.407**†	0.346†	0.350†
EVA Multi-KNRM	74	**0.743**†	**0.879**	0.482	0.680	0.827	0.440	**0.402**†	**0.532**†	**0.253**	0.406†	**0.347**†	**0.351**†
Higher latency (>100 ms)													
EVA Single-QA	2039	0.737	0.862	0.443	**0.701**	**0.856**	0.444	0.389	0.515	0.221	0.402	0.342	0.346
EVA Single-QA-KNRM	3839	**0.747**	**0.874**	**0.447**	0.685	0.838	0.439	0.397	0.534	0.232	0.405	0.347	0.351
BM25 + T5 (Zero-Shot)	5052	0.718	0.865	0.443	0.683	0.837	**0.462**	**0.408**	**0.585**	**0.238**	**0.443**	**0.380**	**0.383**
Best Reported	-	0.765	0.928	0.503	0.803	0.915	0.545	0.408	0.585	0.238	-	0.463	-

Table 3: Ranking effectiveness and efficiency of EVA and baselines on four datasets. The † symbol indicates a significant improvement of bi-encoder models over the corresponding base TAS BERT model or ERNIE Tuned model (paired t-test, p<0.05)

and of Multi-KNRM, which supports the argument that off-topic or noisy entities are a problem for the Single approach. Even when the annotations are perfect, a passage may contain entities that are unrelated to the query, which would dilute the influence of the related entities. EVA Multi mitigates this problem by forming a representation for each cluster of related entities. Indeed, thanks to creating a representation per entity cluster, the chance of matching query entities is higher than EVA Single, leading to higher effectiveness.

Comparing EVA Multi with Single-QA, which creates a single entity representation focused on the query, we see that Multi is an efficient alternative. In the worst case Multi performs only slightly worse than the query-aware method while being substantially more efficient, because it is compatible with ANN search (since entity representations can be formed before the query is received). We discuss these efficiency issues in detail in next section.

RQ3: What is the effectiveness of EVA Multi variants compared to other methods? From Table 3 it can be seen that EVA Multi variants consistently have the highest nDCG@10, MRR@10 and MAP@1000 among low latency methods. Indeed, the Multi variants outperform the underlying TAS BERT model, ANCE, ERNIE Tuned and BM25 across all three datasets. The EVA Multi-KNRM method is competitive with the Best Reported method on DL HARD, though it substantially lags behind on other datasets.

RQ4: What is the impact of entities on different kinds of queries? To see how entities affect various query types, we compare the nDCG@10 from TAS BERT and EVA Multi for specific queries in Table 4. EVA Multi can substantially outperform TAS BERT on queries where detected entities are uncommon (e.g., *mamey* and *tracheids* appear in fewer than 50 Wikipedia articles). The incorporation of these entities can be complementary to BERT's representation. Non-popular entities appear to cover the majority of instances where EVA Multi outperforms TAS BERT; they are also our motivation for combining

Queries	nDCG@10		Change
	TAS	EVA Multi	
Improves			
lps laws definition	0.539	0.879	0.340
tracheids are part of _____.	0.438	0.754	0.316
what is *mamey*	0.566	0.805	0.239
Does not improve			
what is a alm	0.000	0.000	0.000
meaning of *shebang*	0.908	0.845	-0.063

Table 4: Example queries and their metrics

entity and text representations. Obviously, when no entities are returned by the entity linker step, there is no difference between EVA Multi and TAS BERT. For instance, we cannot disambiguate any entity in the query "what is a alm". In some cases where only the wrong entity is detected, such as when *shebang* is recognized as a character sequence in the Unix Shell script rather than an English vocabulary term (last row), using Multi can decrease effectiveness.

6.2 Efficiency Analysis

Along with effectiveness, efficiency is an important factor since search systems need to run at scale. As described above, we avoid the known query assumption in building total representations to enhance scalability of EVA.

RQ5: What is the effect of removing the known query assumption? The known query assumption when building total representations requires BERT-based retrieval methods to invoke BERT inference at ranking time, which is slow. A quick mitigation is to leverage an efficient method such as BM25 for the first phase, and then re-rank the top 1000 results in the following stage (as in Single-QA).

Meanwhile in EVA Multi and EVA Multi-KNRM and Single, the total representations are computed independently from queries at the indexing stage.

From Table 3, we can see the average search time per query (in milliseconds) of EVA methods and baselines on the four datasets. Experiments of all methods are conducted on the same server for fair comparisons. The EVA Single-QA takes more than 2 seconds per query. Meanwhile EVA Multi and EVA Multi-KNRM take about 75 milliseconds across datasets thanks to moving heavy computations such as BERT inference to the indexing stage. EVA Multi variant methods are more than *26 times faster* than EVA Single-QA and EVA Single-QA-KNRM. EVA Multi is slightly slower than ANCE, TAS, ERNIE and EVA Single due to indexing more passage representations, but this yields a significant effectiveness improvement as shown in Table 3. Besides, EVA Multi methods are much faster than methods in the higher latency group.

6.3 Model Settings Selection

Recall from Section 4.1 that we wish to analyze different ways of integrating knowledge with TAS BERT. Along with the proposed concatenation in EVA Single-QA, there are alternatives such as max and sum pooling for the transformed entity representation and textual representation. These alternatives require column dimension of the W_{entity} in Definition 4.4 to be the same as with TAS BERT (i.e., the W_{entity} should project vectors to a 768-dimensional space to be compatible with BERT's representations). All the other training settings among the corresponding models remain the same. Note that we do not use kernel pooling signal in these models.

RQ6: What integration operator should be used? To compare the three operators, we use the corresponding models to infer ranking scores of passages in MS MARCO collection for test queries in MS MARCO development set, and subsequently measure effectiveness such as nDCG@10, MRR@10 and MAP@1000. Regarding search pipeline, BM25 method is used in retrieval stage, subsequently top 1000 passages are reranked by one of the candidate models. Reranking allows us to experiment with different integration operators without re-running expensive indexing procedures. The effectiveness metrics are presented in Table 5 and we can see that concatenation is the clear winner, which outperforms the other two in all metrics. This experiment supports our decision of using concatenation to integrate knowledge to TAS.

Operators	MS MARCO Dev		
	nDCG	MRR	MAP
Sum	0.393	0.335	0.339
Max	0.388	0.330	0.334
Concat	**0.396**	**0.341**	**0.343**

Table 5: Varying aggregation operators

The hyperparameters M and β have an impact on effectiveness and the index size of EVA Multi-KNRM, which we consider in the following question:

RQ7: What is the impact of the hyperparameters M and β? Recall from Section 5.2 that more than 99% training queries have maximum 2 entities, therefore we use a default $M = 2$ in EVA methods. This section explores the performance of EVA Multi-KNRM with different values of M and β. It is not necessary to have parameter β for $M = 1$ since entity cluster has only one entity. Regarding $M = 2$, we use three thresholds 0.9, 0.7 and 0.5 for β. With $M = 3$, only $\beta = 0.5$ is chosen.

Recall that our EVA Multi-KNRM method is designed to maximize the chance that query entities are matched with entity views of relevant passages. With entity views of size more than one, e.g. $M = 2$, passages and queries could potentially be matched based on more than one entity. To focus on these cases, we create a new Dev 2E dataset containing all of the MS MARCO Dev queries with at least two detected entities. Subsequently, we measure the size of index and effectiveness of EVA Multi-KNRM (nDCG@10 and MRR@10) for both MS MARCO Dev and Dev 2E in Table 6.

Considering the results for Dev 2E, there is a significant effectiveness increase from $M = 1$ to $M = 2, \beta = 0.5$ in Table 6. This demonstrates the importance of capturing entity views when considering queries that contain multiple entities. In the latter setting more entity clusters are indexed than in the former one, which leads to better chances to capture relevant search results of queries with at least two entities. Only a small fraction (5.8%) of MS MARCO Dev queries contain more than one entity, however, so on this dataset the effectiveness improvement from increasing M is negligible.

???
falsch?

It can be seen that effective measures of EVA Multi-KNRM are on par among five settings while index size grows from 3.6 to 13.5 times bigger in MS MARCO Dev. With the same $\beta = 0.5$, $M = 3$ does not make any difference over $M = 2$, which supports our decision about default value of M after analyzing query entities in training data. All things considered, our default values of $M = 2$ and $\beta = 0.9$ are reasonable and balance a good trade-off between effectiveness and index size on MS MARCO Dev and Dev 2E.

Params		Index	Dev		Dev 2E	
M	β		nDCG	MRR	nDCG	MRR
1	-	×3.6	0.406	0.347	0.236	0.203
2	0.9	×3.7	0.406	0.347	0.236	0.203
2	0.7	×5.0	0.405	0.347	0.234	0.204
2	0.5	×7.8	0.407†	0.349†	0.257†	0.226†
3	0.5	×13.5	0.407†	0.349†	0.256†	0.226†

Table 6: Varying parameters M and β. The † symbol indicates a significant increase over $M = 1$ (paired t-test, $p < 0.05$).

7 CONCLUSION

In this work we presented EVA, an approach for enriching dense query and document representation with multiple views of the entities in them. Results on several test collections indicated EVA's effectiveness, which significantly outperforms state-of-the-art single-vector baselines. Results on a subset of MS MARCO queries containing two or more entities indicate EVA brings further benefits for more complex queries. While we use TAS BERT and ERNIE in our experiments, our approach for enriching dense representations is general and could be combined with any single-vector model.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A system for large-scale machine learning. *OSDI*.

[2] Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Cross-Domain Modeling of Sentence-Level Evidence for Document Retrieval. *EMNLP-IJCNLP*.

[3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A Nucleus for a Web of Open Data, In The Semantic Web. *ISWC*.

[4] Krisztian Balog. 2018. *Entity-oriented search*. Springer Nature.

[5] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. *SIGMOD*.

[6] Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. 2013. Dexter: An Open Source Framework for Entity Linking. *ESAIR*.

[7] Shubham Chatterjee and Laura Dietz. 2022. 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* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 1466–1477. https://doi.org/10.1145/3477495.3531944

[8] Nick Craswell, Bhaskar Mitra, Daniel Campos, Emine Yilmaz, and Jimmy Lin. 2021. MS MARCO: Benchmarking Ranking Models in the Large-Data Regime. *SIGIR*.

[9] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 deep learning track. TREC.

[10] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. 2022. Overview of the TREC 2021 deep learning track. In *Text REtrieval Conference (TREC)*. TREC.

[11] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. TREC.

[12] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. *arXiv:2003.07820* (2020).

[13] Zhuyun Dai and Jamie Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. *SIGIR*.

[14] Jeff Dalton, Laura Dietz, and James Allan. 2014. Entity query feature expansion using knowledge base links. *SIGIR*.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL-HLT*.

[16] Emma J. Gerritse, Faegheh Hasibi, and Arjen P. de Vries. 2022. Entity-Aware Transformers for Entity Search. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 1455–1465. https://doi.org/10.1145/3477495.3531971

[17] Benjamin Heinzerling and Kentaro Inui. 2021. Language Models as Knowledge Bases: On Entity Representations, Storage Capacity, and Paraphrased Queries. *EACL*.

[18] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. *SIGIR*.

[19] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE*.

[20] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. *EMNLP*.

[21] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. *SIGIR*.

[22] Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2020. PARADE: Passage Representation Aggregation for Document Reranking. *arXiv:2008.09093*.

[23] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2020. Pretrained Transformers for Text Ranking: BERT and Beyond. *arXiv:2010.06467*.

[24] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-BERT: Enabling Language Representation with Knowledge Graph. *AAAI*.

[25] Xitong Liu and H. Fang. 2015. Latent entity space: a novel retrieval approach for entity-bearing queries. *Information Retrieval Journal*.

[26] Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2018. Entity-Duet Neural Ranking: Understanding the Role of Knowledge Graph Semantics in Neural Information Retrieval. *ACL*.

[27] Ilya Loshchilov and Frank Hutter. 2017. Fixing Weight Decay Regularization in Adam. *arXiv:1711.05101*.

[28] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2020. Sparse, Dense, and Attentional Representations for Text Retrieval. *TACL*.

[29] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized Embeddings for Document Ranking. *SIGIR*.

[30] Iain Mackie, Jeffrey Dalton, and Andrew Yates. 2021. How Deep is your Learning: the DL-HARD Annotated Deep Learning Dataset. *SIGIR*.

[31] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. *EMNLP*.

[32] Matthew E. Peters, Mark Neumann, Robert L. Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge Enhanced Contextual Word Representations. *EMNLP*.

[33] Nina Pörner, Ulli Waltinger, and Hinrich Schütze. 2019. BERT is Not a Knowledge Base (Yet): Factual Knowledge vs. Name-Based Reasoning in Unsupervised QA. *arXiv:1911.03681v1*.

[34] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. *NAACL*.

[35] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*.

[36] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, A Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. *NIPS*.

[37] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. *To appear at NAACL*.

[38] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. *WWW*.

[39] Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE 2.0: A Continual Pre-training Framework for Language Understanding. *AAAI*.

[40] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A Free Collaborative Knowledgebase. *Commun. ACM*.

[41] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *ICLR*.

[42] J. Weston and C. Watkins. 1999. Support Vector Machines for Multi-Class Pattern Recognition.

[43] Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. 2017. Word-Entity Duet Representations for Document Ranking. *SIGIR*.

[44] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-Hoc Ranking with Kernel Pooling. *SIGIR*.

[45] Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. *WWW*.

[46] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. *ICLR*.

[47] Yang Xu, G. Jones, and Bin Wang. 2009. Query dependent pseudo-relevance feedback based on wikipedia. *SIGIR*.

[48] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation. *CoNLL*.

[49] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. RepBERT: Contextualized Text Embeddings for First-Stage Retrieval. *arXiv:2006.15498*.

[50] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced Language Representation with Informative Entities. *ACL*.