Unifier: A Unified Retriever for Large-Scale Retrieval

Tao Shen
Australian AI Institute, FEIT
University of Technology Sydney
Sydney, Australia
tao.shen@uts.edu.au

Xiubo Geng
Microsoft
Beijing, China
xigeng@microsoft.com

Chongyang Tao
Microsoft
Beijing, China
chotao@microsoft.com

Can Xu
Microsoft
Beijing, China
caxu@microsoft.com

Guodong Long
Australian AI Institute, FEIT
University of Technology Sydney
Sydney, Australia
guodong.long@uts.edu.au

Kai Zhang
The Ohio State University
Columbus, OH, United States
zhang.13253@osu.edu

Daxin Jiang
Microsoft
Beijing, China
djiang@microsoft.com

ABSTRACT
Large-scale retrieval is to recall relevant documents from a huge collection given a query. It relies on representation learning to embed documents and queries into a common semantic encoding space. According to the encoding space, recent retrieval methods based on pre-trained language models (PLM) can be coarsely categorized into either dense-vector or lexicon-based paradigms. These two paradigms unveil the PLMs’ representation capability in different granularities, i.e., global sequence-level compression and local word-level contexts, respectively. Inspired by their complementary global-local contextualization and distinct representing views, we propose a new learning framework, Unifier, which unifies dense-vector and lexicon-based retrieval in one model with a dual-representing capability. Experiments on passage retrieval benchmarks verify its effectiveness in both paradigms. A uni-retrieval scheme is further presented with even better retrieval quality. We lastly evaluate the model on BEIR benchmark to verify its transferability.

CCS CONCEPTS
• Information systems → Novelty in information retrieval; Rank aggregation; Similarity measures; Language models.

KEYWORDS
deep representation learning; pre-trained language model; neural encoder; hybrid retrieval

ACM Reference Format:
Tao Shen, Xiubo Geng, Chongyang Tao, Can Xu, Guodong Long, Kai Zhang, and Daxin Jiang. 2023. Unifier: A Unified Retriever for Large-Scale Retrieval. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’23), August 6–10, 2023, Long Beach, CA, USA. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3580305.3599927

1 INTRODUCTION
Large-scale retrieval aims to efficiently fetch all relevant documents for a given query from a large-scale collection with millions or billions of entries. It plays indispensable roles as a prerequisite for a broad spectrum of downstream tasks, e.g., information retrieval [2], open-domain question answering [3]. To make online large-scale retrieval possible, the common practice is to represent queries and documents by an encoder in a Siamese manner (i.e., Bi-Encoder, BE) [39]. So, its success depends heavily on a powerful encoder by effective representation learning.

Advanced by pre-trained language models (PLM), e.g., BERT [9], recent works propose to learn PLM-based encoders for large-scale retrieval, which are coarsely grouped into two paradigms in light of their encoding spaces with different focuses of representation granularity. That is, dense-vector encoding methods leverage sequence-level compressive representations that embedded into dense semantic space [14, 24, 51, 54], whereas lexicon-based encoding methods make the best of word-level contextual representations by considering either high concurrence [36] or coordinate terms [12] in PLMs. To gather the powers of both worlds, some pioneering works propose hybrid methods to achieve a sweet point between dense-vector and lexicon-based methods for better retrieval quality. They focus on interactions of predicted scores between the two paradigms.

Nonetheless, such surface interactions - score aggregations [25], direct co-training [16], and logits distillations [5] - cannot fully exploit the benefits of the two paradigms - regardless of their complementary contextual features and distinct representation views. Specifically, as for contextual features, the dense-vector models

1 An entry can be passage, document, etc., and we take document for demonstrations.
focus more on sequence-level global embeddings against information bottleneck [13, 14, 31], whereas the lexicon-based models focus on word-level local contextual embeddings for precise lexicon-weighting [10, 11, 36]. Aligning the two retrieval paradigms more closely is likely to benefit each other since global-local contexts are proven complementary in general representation learning [1, 44]. As for representing views, relying on distinct encoding spaces, the two retrieval paradigms are proven to provide different views in terms of query-document relevance [15, 16, 25]. Such a sort of ‘dual views’ has been proven pivotal in many previous cooperative learning works [4, 17, 18, 27], which provides a great opportunity to bridge the two retrieval paradigms. Consequently, without any in-depth interactions, neither the single (dense/lexicon) nor the hybrid retrieval model can be optimal.

Motivated by the above, we propose a brand-new learning framework, Unified Retriever (Unifie), for in-depth mutual benefits of both dense-vector and lexicon-based retrieval. On the one hand, we present a neural encoder with dual representing modules for Unifie, which is compatible with both retrieval paradigms. Built upon an underlying-tied contextualization that empowers consistent semantics sharing, a local-enhanced sequence representation module is presented to learn a dense-vector representation model. Meantime, a global-aware lexicon weighting module considering both the global- and local-context is proposed for a lexicon-based representation. On the other hand, we propose a new self-learning strategy, called dual-consistency learning, upon our unified encoder. Besides a basic contrastive learning objective, we first exploit the unified dual representing modules by mining diverse hard negatives for self-adversarial within the Unifie. Furthermore, we present a self-regularization method based on list-wise agreements from the dual views for better consistency and generalization.

After being trained, Unifie performs large-scale retrieval via either its lexicon representation by efficient inverted index or dense vectors by parallelizable dot-product. Moreover, empowered by our Unifie, we present a fast yet effective retrieval scheme, uni-retrieval, to gather the powers of both worlds, where the lexicon retrieval is followed by a candidate-constrained dense scoring. Empirically, we evaluate Unifie on both passage retrieval benchmarks to check its effectiveness and the BEIR benchmark [48] with twelve datasets (e.g., Natural Questions, HotpotQA) to verify its transferability.

2 RELATED WORK

PLM-based Retriever. Built upon PLMs, recent works propose to learn encoders for large-scale retrieval, which are coarsely grouped into two paradigms in light of their encoding spaces with different focuses of representation granularity: (i) Dense-vector encoding methods directly represent a document/query as a low-dimension sequence-level dense vector \( \mathbf{u} \in \mathbb{R}^d \) (\( d \) is embedding size and usually small, e.g., 768). And the relevance score between a document and a query is calculated by dot-product or cosine similarity [14, 24, 51, 54]. (ii) Lexicon-based encoding methods make the best of word-level contextualization by considering either high concurrence [36] or coordinate terms [12] in PLMs. It first weights all vocabulary lexicons for each word of a document/query based on the contexts, leading to a high-dimension sparse vector \( \mathbf{v} \in \mathbb{R}^{|V|} \) (\( |V| \) is the vocabulary size and usually large, e.g., 30k). The text is then denoted by aggregating over all the lexicons in a sparse manner. Lastly, the relevance is calculated by lexical-based matching metrics (e.g., BM25 [42]). In contrast, we unify the two paradigms into one sophisticated encoder for better consistency within PLMs, leading to complementary information and superior performance.

Hybrid Retriever. Some works propose to bridge the gap between dense and lexicon for a sweet spot between performance and efficiency. A direct method is to aggregate scores of the two paradigms [25], but resulting in standalone learning and sub-optimal quality. Similar to our work, CLEAR [16] uses a dense-vector model to complement the lexicon-based BM25 model, but without feature interactions and sophisticated learning. Sharing inspiration with our uni-retrieval scheme, COIL [15] equips a simple lexicon-based retrieval with dense operations over word-level contextual embeddings. Unifie differs in not only our lexicon representations jointly learned for in-depth mutual benefits but also sequence-level dense operations involved for memory/computation-efficiency. Lastly, SPARC [26] distills ranking orders from a lexicon model (BM25) into a dense model as a companion of the original dense vector, which is distinct to our motivation.

Bottleneck-based Learning. For neural designs, our encoder is similar to several recent representation learning works, e.g., SEED-Encoder [31], Condenser [13], coCondenser [14], and DiffCSE [6], but they focus on the bottleneck of sequence-level dense vectors. For example, SEED-Encoder, Condenser, and CoCondenser enhance their dense capabilities by emphasizing the sequence-level bottleneck vector and weakening the word-level language modeling heads, while DiffCSE makes the learned sentence embedding sensitive to the difference between the original sentence and an edited sentence by a word-level discriminator. With distinct motivations and targets, we fully exploit both the dense-vector bottleneck and the word-level representation learning in a PLM for their mutual benefits. These are on the basis of not only the shared neural modules but also structure-facilitated self-learning strategies (see the next section). But, as discussed in our experiments, our model can still benefit from these prior works via parameter initializations.

Instance-dependent Prompt. Our model also shares high-level inspiration with recent instance-dependent prompt learning methods [22, 50]. They introduce a trainable component to generate prompts based on each input example. Such generated prompts can provide complementary features to the original input for a better prediction quality. Analogously, our sequence-level dense vector can be seen as a sort of ‘soft-prompt’ for the sparse lexicon-based representation module, resulting in the superiority of our lexicon-based retrieval, which will be discussed in experiments. In addition, the ‘soft-prompt’ in our Unifie also serves as crucial outputs in a unified retrieval system.

Reranker-taught Retriever. Distilling the scores from a reranker into a retriever is proven promising [10, 19, 20]. In light of this, recent works propose to jointly optimize a retriever and a reranker: RocketQA v2 [41] is proposed to achieve their agreements with reranker-filtered hard negatives, while AR2 [56] is to learn them in an adversarial fashion where the retriever is regarded as a generator and the reranker as a discriminator. In contrast to reranker-retriever co-training, we resort to in-depth sharing from the bottom (i.e.,

4788
features) to the top (i.e., self-learning) merely within a retriever, with no need for extra overheads of reranker training. Meanwhile, our unified structure also uniquely enables it to learn from more diverse hard negatives mined by its dual representing modules.

3 METHODOLOGY

Task Definition. Given a collection with numerous documents (i.e., \( D = \{ d_i \}_{i=1}^{|D|} \)) and a textual query \( q \) from users, a retriever aims to fetch a list of text pieces \( \bar{D}_q \) to contain all relevant ones. Generally, this is based on a relevance score between \( q \) and every document \( d_i \) in a Siamese manner, i.e., \( \langle \text{Enc}(q), \text{Enc}(d_i) \rangle \), where \( \text{Enc} \) is an arbitrary representation model (e.g., Bag-of-Words and neural encoders) and \( \langle \cdot, \cdot \rangle \) denotes a lightweight relevance metric (e.g., BM25 and dot-product).

3.1 General Retriever Learning Framework

To ground a method, we first introduce a contrastive learning framework to train a retrieval model (Figure 1). For supervision data in retriever training, differing from traditional categorical tasks, only query-document tuples (i.e., \((q, d_i^p)\)) are given as positive pairs. Hence, given a \( q \), a method needs to sample a set of negatives \( \mathbb{N}_q = \{ d_{q}^n \}_{i=1}^M \) from \( D \), and trains the retriever on tuples of \((q, d_q^n, \mathbb{N}_q)\). \( M \) is the number of negatives. If no confusion is caused, we omit the subscript ‘\( q \)’ for a specific query in the remaining.

Formally, given \( q \) and \( \forall d \in \{ d^+ \} \cup \mathbb{N}, \) an encoder, \( \text{Enc}(\cdot; \theta) \), is applied to them individually to produce their embeddings, i.e., \( \text{Enc}(q; \theta) \) and \( \text{Enc}(d; \theta) \), where the encoder is parameterized by \( \theta \) if applicable. It is noteworthy we tie the query encoder with the document encoder in our work for simplicity. Then, a relevance metric is applied to each pair of the embeddings of the query and each document. Thus, a probability distribution over the documents \((d^+) \cup \mathbb{N})\) can be defined as

\[
P := P(d \mid q, (d^+) \cup \mathbb{N}; \theta) = \frac{\exp(\langle \text{Enc}(q; \theta), \text{Enc}(d; \theta) \rangle)}{\sum_{d' \in (d^+) \cup \mathbb{N}} \exp(\langle \text{Enc}(q; \theta), \text{Enc}(d'; \theta) \rangle)},
\]

where \( \forall d \in \{ d^+ \} \cup \mathbb{N} \). Lastly, a contrastive learning loss to optimize the encoder \( \theta \) is

\[
L_\theta = -\log P(d = d^+ \mid q, (d^+) \cup \mathbb{N}; \theta).
\]

3.2 Neural Encoder in Unifier

We present an encoder (see Figure 2) for Unifier for dense-vector and lexicon-based retrieval.

**Underlying-tied Contextualization.** We first propose to share the low-level textual feature extractor between both representing paradigms. Although the two paradigms are focused on different representation granularities, sharing their underlying contextualization module can still facilitate semantic knowledge transfer between the two paradigms. As such, they can learn consistent semantic and syntactic knowledge towards the same retrieval targets, especially the salient lexicon-based features transferred to dense vectors. Formally, we leverage a multi-layer Transformer [49] encoder to produce word-level (token-level) contextualized embeddings, i.e.,

\[
H(x) = \text{TransfEm-Enc}([\text{CLS}]x[\text{SEP}]; \theta^{(ctx)}),
\]

where \( \forall x \in \{ q \} \cup \{ d^+ \} \cup \mathbb{N} \), and [CLS] [SEP] are special tokens by following PLMs [9, 30]. \( H(x) = [h_{\text{CLS}}(x), h_1(x), \ldots, h_n(x), h_{\text{SEP}}(x)] \) are resulting embeddings, and \( n \) is the number of words in \( x \).

**Local-enhanced Sequence Representation.** On top of the embeddings with enhanced local contexts, we then present a representing module to produce sequence-level dense vectors. For this purpose, we apply another multi-layer Transformer encoder to \( H(x) \), followed by a pooler to derive a sequence-level vector. This can be written as

\[
u(x) = \text{Pool}((\text{TransfEm-Enc}(H(x); \theta^{(den)}))),
\]

where this module is parameterized by \( \theta^{(den)} \) untied with \( \theta^{(ctx)} \). Pool(\) gets a sequence-level dense vector by taking the embedding of special token [CLS], and the resulting \( u(x) \in \mathbb{R}^e \) denotes a global dense representation of the input text \( x \), which is used for dense-vector retrieval.

**Global-aware Lexicon Weighting.** Lastly, to achieve lexicon-based retrieval, we adapt a recent SParse Lexical AnD Expansion Model (SPLADE) [10] into our neural encoder. SPLADE is a lexicon-weighting retrieval model which learns sparse expansion for each word in query/document \( x \) via the MLM head of PLMs and sparse regularization. Differing from the original SPLADE, our lexicon-based representing module not only shares its underlying feature extractor with a dense model but strengthens its hidden states by the global vector \( u(x) \) above. The intuition is that, similar to text decoding with a bottleneck hidden state, the global context serves as high-level constraints (e.g., concepts/topics) to guide word-level operations [13, 31, 47]. In particular, the word-level contextualization embeddings passed into this module are manipulated as

![Figure 2: The encoder in Unifier.](image-url)
we follow [12] to use dot-product of lexicon-weighting representation for $x$ can be derived by

$$\sigma^{(x)} = \log(1 + \text{Max-Pool}([\text{ReLU}(\mathbf{W}^{(e)} \text{ Transf-Menc}(\mathbf{H}^{(x)}); \theta^{(\text{mlm})})]),$$

where, $\theta^{(\text{mlm})}$ parameterizes a multi-layer Transformer encoder, $\mathbf{W}^{(e)} \in \mathbb{R}^{V \times V}$ denotes the transpose of word embedding matrix as the MLM head, $|V|$ denotes the vocabulary size. $\theta^{(\text{lex})} = \{\theta^{(cxt)}; \theta^{(\text{den})}\}$ parameterizes this module, and $\theta^{(x)} \in \mathbb{R}^{|V|}$ is a sparse lexicon-based representation of $x$. And its sparsity is regularized by FLOPS [37] as in [10]. Here, the saturation function $\log(1 + \text{Max-Pool}(\cdot))$ prevents some terms from dominating.

In summary, given a text $x$, Unifieer produces two embeddings via its dual representing modules:

$$u^{(x)} := \text{Uni-Den}(x; \theta^{(\text{den})}), \quad \sigma^{(x)} := \text{Uni-Lex}(x; \theta^{(\text{lex})}),$$

where $\Theta^{(\text{den})} = \{\theta^{(cxt)}; \theta^{(\text{den})}\}$ and $\Theta^{(\text{lex})} = \{\theta^{(cxt)}; \theta^{(\text{den})}; \theta^{(\text{lex})}\}$. Hence, $u^{(x)} \in \mathbb{R}^r$ denotes a dense vector and $\sigma^{(x)} \in \mathbb{R}^{|V|}$ denotes a sparse lexicon-based embedding.

### 3.3 Dual-Consistency Learning for Unifieer

To maximize our encoder’s representing capacity, we propose a self-learning strategy, called dual-consistency learning (Figure 3). The ‘dual-consistency’ denotes learning the dual representing modules to achieve consistency in a unified model via negative samples and module predictions.

**Basic Training Objective.** To learn the encoder, a straightforward way is applying the contrastive learning loss defined in Eq.(1-2) to our dual representing modules. That is,

$$L^{(\text{con})} = - \log P(d = d^+ | q, \{d^+\} \cup \mathbb{N}; \Theta^{(\text{den})}) - \log P(d = d^+ | q, \{d^+\} \cup \mathbb{N}; \Theta^{(\text{lex})}),$$

where the former is for dense-vector retrieval while the latter is for lexicon-based retrieval. Towards the same retrieval target, the model is prone to learn consistent semantic and syntactic features via complementing the global-local granularity of the two retrieval paradigms. Due to the non-differentiability of lexicon-based metrics, we follow [12] to use dot-product of lexicon-weighting representation during training but resort to a lexicon matching system [53] with quantization during indexing&retrieval. (see Appx. 3.4 for details) Note that $\theta^{(\text{den})}$ would not be optimized w.r.t. the losses on top of the lexicon-based module. As for the query’s negatives $\mathbb{N}$ of in Eq.(7), they are initially sampled by a BM25 retrieval system at the warmup stage [14, 54], denoted as $\mathbb{N}^{(\text{bm25})} = \{d \mid d = \text{P}(d | q, \mathbb{D}\setminus(d^+); \text{BM25})\}$, where $\mathbb{D}\setminus(d^+)$ denotes all documents in the collection $\mathbb{D}$ except the positive $d^+$ for query $q$.

**Negative-bridged Self-Adversarial.** However, it is verified that learning a retriever based solely on BM25 negatives cannot perform competitively [51, 54]. Thereby, previous works propose to sample hard negatives by the best-so-far retriever for continual training [14, 54], a.k.a. self-adversarial learning [46]. In our pilot experiments, we found the two retrieval paradigms can provide distinct hard negatives (> 40% top-retrieved candidates are different) to ensure diversity after a combination. This motivates us to make the best of the hard negatives sampled by our dual representing modules: hard negatives sampled from one module can be applied to both itself and its counterpart in one unified framework. This can be regarded as a sort of self-distillation as both distilling samples (i.e., document mined from the collection) and distilling labels (i.e., negative label only) are sourced from one unified model. So, we first sample two sets of negatives from the dual-representing modules:

$$\mathbb{N}^{(\text{den})} = \{d \mid d = \text{P}(d | q, \mathbb{D}\setminus(d^+); \Theta^{(\text{den})})\},$$
$$\mathbb{N}^{(\text{lex})} = \{d \mid d = \text{P}(d | q, \mathbb{D}\setminus(d^+); \Theta^{(\text{lex})})\},$$

where our Unifieer was trained with $\mathbb{N}^{(\text{bm25})}$ at warmup stage, Next, we upgrade $\mathbb{N}$ in Eq.(7) from $\mathbb{N}^{(\text{bm25})}$ at warmup stage to $\mathbb{N}^{(\text{den})} \cup \mathbb{N}^{(\text{lex})}$, and then perform a continual learning stage.

**Agreement-based Self-Regularization.** We lastly present a self-regularization method for Unifieer. Its goal is to achieve an agreement from different views through our dual representing modules. Such an agreement-based self-regularization has been proven effective in both retrieval model training (via retriever-reranker agreements for consistent results [41, 56]) and general representation learning (via agreements from various perturbation-based views for better generalization [4, 17, 27]). It is stronger than the contrastive learning in Eq.(7) as the agreement is learned by a KL divergence: $$L^{(\text{reg})} = \text{KL}(\mathcal{P}(d|q,d^+;\text{Uni-Den});\mathbb{N}^{(\text{den}})) \quad \text{and} \quad \| \mathcal{P}(d|q,d^+;\text{Uni-Lex}) \|_{\Theta^{(\text{lex})}}$$

**Overall Training Pipeline.** In line with [14], we lastly follow a simple three-step pipeline to learn our retriever on the basis of the proposed training objectives and hard negatives: (i) **Warmup Stage:** Initialized by a pre-trained model, Unifieer is updated w.r.t. Eq.(7) + $\lambda$ FLOPS (by following [10] for sparsity), with BM25 negatives $\mathbb{N}^{(\text{bm25})}$. (ii) **Hard Negative Mining:** According to the warmup Unifieer, static hard negatives, $\mathbb{N}^{(\text{den})}$ and $\mathbb{N}^{(\text{lex})}$, are sampled by Eq.(8). (iii) **Continual Learning Stage:** Continual with the warm-up-ed Unifieer, the model is finally optimized on $\mathbb{N}^{(\text{den})} \cup \mathbb{N}^{(\text{lex})}$ w.r.t. a direct addition of Eq.(7&9) + $\lambda$ FLOPS.

### 3.4 Retrieval Schemes

**Inference of Lexicon-based Retrieval.** During the inference of large-scale retrieval, there are some differences between dense-vector and lexicon-based retrieval methods. As in Eq.(1), we use the
Table 1: Passage retrieval results on MS-Marco Dev and TREC Deep Learning 2019. †Refer to Table 2. 'coCon': coCondenser that continually pre-trained BERT in unsupervised manner. ‘Reranker taught’: distillation from a reranker (see §2).

| Method                  | Pre-trained model | Reranker taught | Hard negs | Mul Repr | MRR@10 | R@100 | R@1k | R@100 | nDCG@10 |
|-------------------------|-------------------|-----------------|-----------|----------|---------|-------|------|-------|---------|
| **Dense-vector Retriever** |                   |                 |           |          |         |       |      |       |         |
| ANCE [51]               | RoBERTa_base      | ✓               |           |          | 33.8    | 86.2  | 96.0 | 44.5  | 65.4    |
| ADORE [54]              | RoBERTa_base      | ✓               |           |          | 34.7    | 87.6  | 97.8 | 47.3  | 68.3    |
| TAS-B [20]              | DistilBERT        | ✓               |           |          | 34.7    | 97.8  | -    | -     | 71.2    |
| TCT-ColBERT [29]        | BERT_base         | ✓               | ✓         |          | 39.9    | 97.0  | -    | -     | 71.9    |
| Condenser [13]          | Condenser_base    | ✓               | ✓         |          | 36.6    | 97.4  | -    | -     | -       |
| coCondenser [14]        | coCon_base        | ✓               | ✓         |          | 38.2    | 98.4  | -    | -     | -       |
| CoBERTv1 [24]           | BERT_base         | ✓               | ✓         | ✓        | 39.7    | 98.4  | -    | -     | -       |
| CoBERTv2 [43]           | BERT_base         | ✓               | ✓         | ✓        | 37.9    | -     | -    | -     | -       |
| PAIR [40]               | ERNIE_base        | ✓               |           |          | 37.9    | -     | -    | -     | -       |
| RocketQA [38]           | ERNIE_base        | ✓               |           |          | 37.0    | -     | -    | -     | -       |
| RocketQAv2 [41]         | ERNIE_base        | ✓               | ✓         |          | 38.8    | -     | -    | -     | -       |
| AR2 [56]                | coCon_base        | ✓               | ✓         |          | 39.5    | -     | -    | -     | -       |
| **Lexicon-base or Sparse Retriever** |                   |                 |           |          |         |       |      |       |         |
| DeepCT [8]              | BERT_base         | ✓               |           |          | 34.0    | 86.4  | 96.5 | 49.2  | 66.8    |
| RepCONC [55]            | RoBERTa_base      | ✓               |           |          | 34.0    | -     | -    | 68.4  | -       |
| SPLADE-max [10]         | DistilBERT        | ✓               |           |          | 32.2    | -     | -    | 66.7  | 72.9    |
| SPLADE-doc [10]         | DistilBERT        | ✓               |           |          | 32.2    | 94.6  | -    | -     | 72.9    |
| DistisSPLADE-max [10]   | DistilBERT        | ✓               |           |          | 36.8    | 97.9  | -    | -     | 72.9    |
| SelfDistil [11]         | DistilBERT        | ✓               | ✓         |          | 36.8    | -     | -    | 72.3  | 72.1    |
| EnsembleDistil [11]     | DistilBERT        | ✓               | ✓         |          | 36.9    | 97.9  | -    | -     | 72.1    |
| Co-SelfDistil [11]      | coCon_base        | ✓               | ✓         | ✓        | 37.5    | -     | -    | 73.0  | 73.0    |
| Co-EnsembleDistil [11]  | coCon_base        | ✓               | ✓         | ✓        | 38.0    | 98.2  | -    | -     | 73.2    |
| **Hybrid Retriever**    |                   |                 |           |          |         |       |      |       |         |
| CLEAR [16]              | BERT_base         | ✓               |           |          | 33.8    | -     | -    | 69.9  | -       |
| COIL-full [15]          | BERT_base         | ✓               |           |          | 35.5    | -     | -    | 70.4  | -       |
| **Uniﬁerlexicon (warmup)** | coCon_base        | ✓               |           |          | 37.2    | 90.1  | 97.8 | 50.1  | 69.7    |
| Uniﬁer dense (warmup)  | coCon_base        | ✓               |           |          | 36.1    | 87.7  | 96.6 | 44.6  | 63.9    |
| Uniﬁer uni-retrieval (warmup) | coCon_base | ✓               |           |          | 38.3    | 90.8  | 98.0 | 50.6  | 70.2    |
| Uniﬁerlexicon          | coCon_base        | ✓               |           |          | 39.7    | 91.2  | 98.1 | 53.4  | 73.3    |
| Uniﬁer dense           | coCon_base        | ✓               |           |          | 38.8    | 90.3  | 97.6 | 50.2  | 71.1    |
| Uniﬁer uni-retrieval   | coCon_base        | ✓               | ✓         |          | 40.7    | 92.0  | 98.4 | 53.8  | 73.8    |

Uni-retrieval Scheme. As in Figure 2, our model is fully compatible with the previous two retrieval paradigms. In addition, we present a uni-retrieval scheme for fast yet effective large-scale retrieval. Instead of adding their scores [11, 25] from twice-retrieval with heavy overheads, we pipelineize the retrieval procedure: given q, our lexicon-based retrieval under an inverted file system is to retrieve top-K documents from D. Then, our dense-vector retrieval is then applied to the constrained candidates for dense scores. The final retrieval results are according to a simple addition of the two scores. We use ‘addition’ as our combination baseline for its generality and explore more advanced methods in §4.4. And, due to fast dense-vector dot-product calculations on top-K documents, uni-retrieval’s latency is almost equal to single lexicon-based retrieval.

4 EXPERIMENT

Datasets & Metrics. In line with [10], we use popular passage retrieval datasets, MS-Marco [34], with official queries (no augmentations [41]), and report for MS-Marco Dev set and TREC Deep Learning 2019 set [7]. Following previous works, we report
MRR@10 (M@10) and Recall@1/50/100/1K² for MS-MarcovDev, and report nDCG@10 and R@100 for TREC Deep Learning 2019. Besides, we also transfer our model trained on MS-Marcov to the BEIR benchmark [48] to evaluate its generalizability, where nDCG@10 is reported. We take 12 datasets (i.e., TREC-COVID, NFCorpus, NQ, HotpotQA, PiQA, ArguAna, Touche-2020, DBPedia, Scidocs, Fever, Climate-FEVER, and SciFact) in the BEIR benchmark as they are widely-used across most previous papers.

**Experimental Setups.** As stated in §3.3, we take a 2-stage learning scheme [14]. We use coCondenser-marco [14] (unsupervised continual pre-training from BERT-base [9]) as our initialization as it shares a similar neural structure and has potential for promising performance [11, 14, 56]. \( \theta_\text{ctx}(z) \), \( \theta_\text{den} \), and \( \theta_\text{lex}(x) \) correspond to Transformer layers of 6, 6, and 2, respectively, where max length is 128 and warmup ratio is 5%. At warmup stage, batch size of queries is 16, each with 1 positive document and 15 negatives, learning rate is \( 210^{-3} \), the random seed is fixed to 42. And loss weight of FLOPS [37] is set to 0.0016 since we want the model sparser than SPLADE [10] (0.0008). At continual learning stage, batch size is 12 to enable each module with 15 negatives. And learning rate is reduced to 1/3 of the original, and the random seed is changed to 22 for a new data feeding order. And the loss weight of FLOPS is lifted to 0.0024. We did not tune the hyperparameters. In retrieval phase, we set \( K=2048 \) in our uni-retrieval, and also compare other choices in our analysis. All experiments are run on a single A100 GPU. Our codes are released at https://github.com/taoshen58/UnifieR.

### 4.1 Main Evaluation

**MS-MarcovDev.** As in Table 1&2, our framework achieves new state-of-the-art metrics on most metrics. Our dense-vector retrieval surpasses previous methods without distillations from rankers, while our lexicon-based retrieval pushes the best sparse method to a new level, especially in MRR@10 (+1.4%). Empowered by our unified structure, the uni-retrieval scheme can achieve 40.7% MRR@10. Although R@1K is approaching its ceiling across recent works, we notice Unifie is less competitive than AR2 (-0.2%) in Table 2, as the latter involves a costly reranker in training for better generalization. And please see §4.4 for our rerank-taught results.

### TREC Deep Learning 2019.

As listed in Table 1, our retrieval method, with either single (dense/lexicon) or unified representation, achieves a state-of-the-art or competitive retrieval quality. Specifically, compared to the previous best method, called TAS-B, our model lifts MRR@10 and nDCG@10 by 6.9% and 2.6%, respectively.

### BEIR Benchmark.

Table 3 shows in-domain evaluation and zero-shot transfer on BEIR (see §8.1). It is observed that, with outstanding in-domain inference ability, our model also delivers comparable transferability among the retrievers with similar training settings (i.e., comparable models o/w reranker distillations). But, we found our model suffers from inferior generalization ability compared to the models with MSE-based reranker distillation [10, 43]. And a small model with distillation (e.g., DistilSPLADE) even beats the models with billions of parameters (e.g., GTR-XXL). The potential reasons are two-fold: i) distilling a reranker to the retriever has been proven to produce more generalizable scores than a bi-encoder [32] and ii) the initialization of Unifie, coCondenser, has been pre-trained on Marco collection, reducing its generalization.

### 4.2 Further Analysis

**Comparison to Ensemble Models.** As in Table 4, we report the numbers to compare our uni-retrieval scheme with ensemble models. Even if we only need once large-scale retrieval followed by a small amount of dot-product calculation, the model still surpasses its competitors. Meantime, we found both uni-retrieval and ensemble are bounded by the worse participant. For example, even if we use a SPLADE with MRR@10 of 39.3 for 'Ensemble/Uni-scheme

\[ \text{...} \]

---

1 We follow official evaluation metrics at https://github.com/castorini/anserini. But, we found 2 kinds of Recall@N on MS-Marcov in recent papers, i.e., official all-positive-macro recall and one-positive-enough recall (see §A for details). Thereby, we report the former by default but list the latter separately for fair comparisons.

---

**Table 3:** Retrieval nDCG@10 results on BEIR with 12 out-of-domain datasets, and 1 in-domain dataset. Avg is mean nDCG over 12 datasets and Best is how many datasets a method achieves best. DocT5query [36], CoIBERT [24], CoIBERT-v2 [43], DistilSPLADE [10].

| Method          | M@10 | R@100 |
|-----------------|------|-------|
| RocketQA [38]   | 37.0 | 85.5  |
| Pair [40]       | 37.9 | 86.4  |
| RocketQA-v2     | 38.8 | 86.2  |
| AR2             | 39.5 | 87.8  |
| Unifie_lexicon  | 39.7 | 87.6  |
| Unifie_dense    | 38.8 | 86.3  |
| Unifie_retrieval| 40.7 | 88.2  |

---

**Table 4:** Comparison with ensemble and hybrid retrievers. We operate on the best SPLADE model (MRR@10=38.5) with the best coCondenser (MRR@10=38.2). An ensemble of four SPLADE models.

| Method               | M@10 | R@100 | Avg | Best |
|----------------------|------|-------|-----|------|
| RocketQA             | 44.3 | -     | 44.3| 44.3 |
| Pair                 | 42.4 | 0     | 42.4| 42.4 |
| RocketQA-v2          | 47.0 | N/A   | 47.0| 47.0 |
| CoIBERT              | 47.0 | N/A   | 47.0| 47.0 |
| GenQ [48]            | 39.8 | 1     | 39.8| 39.8 |
| TAS-B [20]           | 40.4 | 0     | 40.4| 40.4 |
| Contriever [21]      | 44.3 | 4     | 44.3| 44.3 |
| Unifie_retrieval     | 44.5 | 4     | 44.5| 44.5 |
| Ensemble of Best     | 40.3 | 26.1  | 40.3| 26.1 |
| Ensemble of SPLADE²  | 40.4 | 26.5  | 40.4| 26.5 |
| COL-full (hybrid)    | 35.5 | -     | 35.5| -    |

---

2We follow official evaluation metrics at https://github.com/castorini/anserini. But, we found 2 kinds of Recall@N on MS-Marcov in recent papers, i.e., official all-positive-macro recall and one-positive-enough recall (see §A for details). Thereby, we report the former by default but list the latter separately for fair comparisons.
of Best’, the performance did not show a remarkable gain. This suggests us to look for a better aggregation method in the future.

Ablation of Neural Structure. To verify the correctness of each module design, we conduct an ablation study on the neural structure of the encoder (§3.2) in Table 5. This must be performed at the warmup step as the second stage is continual from the warmup. It is observed that, either removing the global information from the lexicon-based module or discarding in-depth inter-paradigm interactions (i.e., learning independently) degrades the model dramatically. Surprisingly, removing the global also diminishes dense performance. A potential reason is that, such a change makes the fine-tuning inconsistent with its initializing pre-trained model, co-Condenser, leading to corrupted representing capability.

Table 5: Ablation of the encoder on MS-Marco Dev.

| Methods               | Lexicon-based | Dense-vector |
|-----------------------|---------------|--------------|
|                       | M@10 | R@100 | M@10 | R@100 |
| Unifie (warmup)       | 37.2  | 90.1  | 36.1  | 87.7  |
| w/o sharing Global    | 36.1  | 89.8  | 35.2  | 87.2  |
| w/o in-depth Interact | 36.1  | 89.3  | 35.7  | 89.7  |

Study on Learning Strategy. Furthermore, we conduct another study on the learning strategies (§3.3) in Table 6. This is performed at the continual training stage. The table shows that, ablating the negative-bridged self-adversarial (self-adv) and the agreement-based self-regularization (self-reg) has a minor effect on lexicon-based retrieval but is remarkable on dense-vector one. This is because the former is already far stronger than the latter. Thereby, both self-adv and self-reg can be regarded as a sort of (self-)distillation from lexicon knowledge to the dense-vector module. An upper bound of performance. Interestingly, optimizing for each of the representing modules can improve both retrieval paradigms (i.e., lexicon and dense). This confirms that optimizing one module can benefit the other, attributed to complementary representations and the consistent learning target.

| Methods | Lexicon-based | Dense-vector |
|---------|---------------|--------------|
|         | M@10 | R@100 | M@10 | R@100 |
| Unifie  | 39.7  | 91.2  | 38.8  | 90.3  |
| w/o Self-adv | 39.6  | 91.5  | 38.2  | 90.3  |
| w/o Self-adv-reg | 39.5  | 91.3  | 37.9  | 90.1  |

Table 6: Learning strategy for continual training on MS-Marco Dev.

4.3 Efficiency Analysis

FLOPS analysis. To view sparsity-efficacy trade-off, we vary the loss weight λ for FLOPS sparse regularization [37]. As in Figure 5, with λ exponentially increasing, document FLOPS decreases linearly, improving the efficiency of our framework. The descending of lexicon-based efficacy is not remarkable when FLOPS > 4 and then becomes notable with the growth of λ. Fortunately, this will not affect the dense representation in terms of dense-vector retrieval.

Uni-Retrieval Hyperparameter. In uni-retrieval scheme, a hyperparam K is used to control computation overheads of dense dot-product. As illustrated in Figure 6, ‘K=0’ denotes lexicon-only retrieval in Unifie. The table shows that Unifie reaches an MRR@10 ceiling when K is set to a de facto number, i.e., 1000. Then, the upper bound of R@1000 is reached when K=2048. After that, the two metrics cannot be observed with any changes.
Table 8: UnifieR-lex v.s. QPS by Top-N lexicon sparsifying. The QPS is calculated on a CPU machine with pre-embedded queries, and ORG denotes non-sparsified UnifieR.

| Top-N | QPS  | M@10 | R@100 | Remark             |
|-------|------|------|-------|--------------------|
| BM25  | 449  | 19.3 | 69.0  |                    |
| ORG   | 50   | 41.3 | 92.3  | Not sparsified     |
| 75    | 129  | 40.8 | 91.5  | ▼ Index as Sparse as BM25 |
| 50    | 188  | 40.4 | 91.1  |                    |
| 25    | 343  | 38.4 | 89.0  |                    |
| 20    | 446  | 37.5 | 87.6  | ▼ Infer as Faster than BM25 |
| 15    | 537  | 36.2 | 86.0  |                    |
| 10    | 693  | 33.6 | 82.2  |                    |
| 8     | 911  | 31.9 | 79.8  |                    |
| 4     | 954  | 25.5 | 70.0  |                    |
| 2     | 1144 | 16.2 | 53.2  |                    |
| 1     | 1376 | 1.8  | 22.3  |                    |

Table 9: Stage 1 of UnifieR with query-side gating.

| Method               | M@10 | R@100 |
|----------------------|------|-------|
| UnifieR-uni (warmup) | 38.3 | 90.8  |
| + query-side gating  | 39.2 | 91.2  |

Latency Analysis. Besides the un-intuitive FLOPS numbers, we also exhibit the latency (measured by ‘query-per-second’ – QPS) of UnifieR. Basically, our UnifieR is bottlenecked by its lexicon head in terms of inference speed as aforementioned, so we would like to dive into the controllable sparsity of UnifieR-lexicon. Note that, to reserve a large room for further sparsifying, we leverage the reranker-taught UnifieR-lexicon as shown in Table 10, whose MRR@10 is 41.3%. Then, we adopt a simple but effective sparsifying method – top-N [52] – but in the index-building process only. As a result, we show the performance of our UnifieR-lexicon with N decreasing in Table 8. It is shown only the Top-4 tokens kept for each passage can deliver very competitive results with faster speed than BM25.

Table 10: Reranker-taught UnifieR v.s. previous state-of-the-art (SoTA) models (i.e., Dense [56], Lexicon[11], Multi-Vec [43]).

| Methods               | Dense-vector | Lexicon-based | Uni/Multi-Vec |
|-----------------------|--------------|---------------|---------------|
|                       | M@10 R@100   | M@10 R@100    | M@10 R@100    |
| Previous SoTA         | 39.5         | -             | 37.5          |
| UnifieR               | 38.8         | 90.3          | 39.7          |
| UnifieR (distill)     | 40.5         | 91.6          | 41.3          |

4.4 Exploration of Advanced Architecture

Query-side Gating Mechanism. As it is too rough to directly add the scores of the two retrieval paradigms, we incorporate a recent inspiration of mix-of-expert (MoE) to enhance the combination of the two paradigms. As in illustrated in §B.2, we leveraged a gating mechanism to switch UnifieR between dense and lexicon, based solely on the semantics of queries. The reasons for “solely on queries” are two-fold: i) the analyses in §4.5 show that the type of queries affects models a lot and ii) the dependency on queries only will not affect the indexing procedure for large-scale collections, leading to zero extra inference overheads. After this gating mechanism in the warmup stage of UnifieR training where the gate’s optimization is based on the relevance score of uni-retrieval. As listed in Table 9, a remarkable improvement is observed with such a query-side gating mechanism (+0.9% MRR@10 and +0.4% R@100).

Reranker-taught UnifieR. Although the UnifieR in Table 1 & 2 seems significant in terms of performance improvement, it’s noteworthy that the comparisons are unfair because UnifieR didn’t use a re-ranker (a strong but heavy cross-encoder) as a teacher for knowledge distillation (see ‘Reranker taught’ in Table 1). To make the comparisons fairer, we first trained a re-ranker based on UnifieR’s hard negatives and then used a KL loss for distillation in the Continual Training Stage (as illustrated in Figure 8 of §B.3). As listed in Table 10, it is shown that i) our proposed UnifieR is compatible with ‘Reranker taught’ scheme and consistently brings 1%+ improvement, and ii) UnifieR outperforms its strong competitors by large margins (2.0%+).

4.5 Qualitative Analysis

Case Study. As shown in Table 11, we list two queries coupled with the ranking results from five retrieval systems. Those are from three groups, i.e., i) previous state-of-the-art dense-vector and lexicon-based retrieval models, ii) the dense-vector and lexicon-based retrieval modules from our UnifieR, and iii) uni-retrieval scheme by our UnifieR. As demonstrated in the first query of the table, ‘Indep-lex’ achieves a very poor performance, where the positive passage is ranked as 94. Via exhibiting its top-1 passage, the error is possibly caused by the confusion between the ‘weather’ for a specific day and ‘weather’ for a period (i.e., climate). This is
Table 11: Case study on MS-Marcos Dev set. 'Passage+' denotes positive passage of the corresponding query. 'Indep-den' denotes a well-trained state-of-the-art dense-vector retrieval model with static hard negatives (i.e., coCondenser [14], M@10=38.2) while 'Indep-lex' denotes a well-trained state-of-the-art lexicon-based retrieval model with static hard negatives (i.e., SPLADE [11], M@10=38.3).

| Query ID:1088347/ | weather in new york city ny |
|------------------|-----------------------------|
| Passage ID:7094280/ | Title: - Body: New York, NY - Weather forecast from TheWeather.com. Weather conditions with updates on temperature, humidity, wind speed, snow, pressure, etc. for New York, New York Today: Cloudy skies with light rain, with a maximum temperature of 72C and a minimum temperature of 52C. |
| Rank | Indep-den: 1; Indep-lex: 94; Unifie\_den: 1; Unifie\_lex: 3; Unifie\_uni: 1 |
| Retrieved | Indep-lex’s 1st. ID:65839/ Title: New York City - Best Time To Go & When to Go Body: Weather: Spring in New York City is the best time to be in the city, without doubt. Spring usually means less humidity and temps between 50-80 degrees, though June occasionally sees a 90 degree day. An occasional humidity soaked heat wave can strike, but it usually feels nice the first time around. |
| | Indep-lex’s 2nd. ID:4835773/ Title: Climate of New York Body: Weather: Unlike the vast majority of the state, New York City features a humid subtropical climate (Koppen Cfa), with temperatures 5-7 degrees Fahrenheit (3-4 degrees Celsius) warmer overnight than surrounding areas. In an effort to fight this warming, roofs of buildings are being painted white across the city in an effort to increase the reflection of solar energy, or albedo. |
| | Unifie\_lex’s 1st. ID:65839/ Title: New York City - Best Time To Go & When to Go Body: Weather: Spring in New York City is the best time to be in the city, without doubt. Spring usually means less humidity and temps between 50-80 degrees, though June occasionally sees a 90 degree day. An occasional humidity soaked heat wave can strike, but it usually feels nice the first time around. |
| | Unifie\_lex’s 2nd. ID:8819213/ Title: New York City - Best Time To Go & When to Go Body: Weather: Spring in New York City is the best time to be in the city, without doubt. Spring usually means less humidity and temps between 50-80 degrees, though June occasionally sees a 90 degree day. |
| Query ID:391101/ | idiotsguides tai chi |
|------------------|-----------------------------|
| Passage ID:7668258/ | Title: - Body: Bill is the author of The Complete Idiot’s Guide to T’ai Chi & Qigong (4th edition), and his newest upcoming books, The Tao of Tai Chi, and The Gospel of Science, in which he paints a vision of vast global benefit as mind-body sciences spread across the planet. |
| Rank | Indep-den: 41; Indep-lex: 1; Unifie\_den: 10; Unifie\_lex: 1; Unifie\_uni: 1 |
| Retrieved | Indep-den’s 1st. ID:1663205/ Title: - Body: Tai chi. Tai chi (simplified Chinese: ; traditional Chinese: ; pinyin: chi, an abbreviation of ; is an internal Chinese martial art (Chinese: ; pinyin: ) practiced for both its defense training and its health benefits. |
| | Indep-den’s 2nd. ID:3449438/ Title: - Body: Tai chi: A gentle way to fight stress Body: Tai chi is an ancient Chinese tradition that, today, is practiced as a graceful form of exercise. It involves a series of movements performed in a slow, focused manner and accompanied by deep breathing. Tai chi, also called tai chi chuan, is a noncompetitive, self-paced system of gentle physical exercise and stretching. |
| | Unifie\_den’s 1st. ID:2294942/ Title: WHAT IS TAI CHI? Body: The Chinese characters for Tai Chi Chuan can be translated as the ‘Supreme Ultimate Force’. The notion of ‘supreme ultimate’ is often associated with the Chinese concept of yin-yang, the notion that one can see a dynamic duality (male/female, active/passive, dark/light, forceful/yielding, etc.) in all things. |
| | Unifie\_den’s 2nd. ID:3449442/ Title: - Body: What is Tai Chi? Body: What is Tai Chi? In China, and increasingly throughout the rest of the world, tai chi is recognized for its power to instill and maintain good health and fitness in people of all ages. Tai chi aims to bring balance to body, mind and spirit through specifically designed movements, natural breathing and a calm state of mind. It is easily recognized by its slow, captivating and mesmerizing movements. It represents a way of life, helping people meet day to day challenges while remaining calm and relaxed. |

because the ‘weather’ as a pivot word in both contexts receives large weights, making the distinguishing very hard. Although our Unifie\_lex can lift the positive from 94 to 3 by our carefully designed unified model, it still suffers from confusion. Meantime, it is observed that both dense-vector methods perform well since they rely on latent semantic contextualization, less focusing on a specific word. As shown in the second query of the table, the strange word, ‘idotsguides’ makes both dense-vector models less competent. On the contrary, the lexicon-based method can handle this case perfectly. It is still noteworthy that our Unifie\_den can also outperform the vanilla one, ‘Indep-den’, by lifting 31 (41→10) ranking position. This is attributed to our consistent feature learning, which bridges the gap of heterogeneity between dense-vector and lexicon-based retrieval. These two cases also support the previous claim that the two representing ways can provide distinct views of query-document relevance. Furthermore, despite varying performance across different paradigms, our uni-retrieval scheme consistently performs well as it is an aggregation of both. Please see §B.4 for our further case study on error analysis.

Limitation. The main limitations of this work are i) PLM Compatibility: due to the special encoder design, Unifie can only be initialized from a limited number of pre-trained models, and ii) Additional Infrastructure: in spite of the almost same retriever latency as traditional lexicon-based retrieval, Unifie requires extra computation infrastructure for indexing and storing both dense and sparse embeddings of all documents in the collection.

5 CONCLUSION

We present a brand-new learning framework, dubbed Unifie, to unify dense-vector and lexicon-based representing paradigms for large-scale retrieval. It improves the two paradigms by a carefully designed neural encoder to fully exploit the representing capability of pre-trained language models. Its capability is further strengthened by our proposed dual-consistency learning with self-adversarial and -regularization. Moreover, the uni-retrieval scheme and the advanced architectures upon our encoder are presented to achieve more. Experiments on several benchmarks verify the effectiveness and versatility of our framework.
A EXPLANATION OF TWO RECALL METRICS

Regarding R@N metric, we found there are two kinds of calculating ways, and we strictly follow the official evaluation one at https://github.com/usnistgov/trec_eval and https://github.com/castorini/anserini, which is defined as

\[
\text{Marco-Recall@N} = \frac{1}{|Q|} \sum_{q \in Q} \frac{\sum_{d \in D^+} 1_{d \in \text{top-K}}} {\min(N, |D^+|)},
\]

(10)

where there may be multiple positive documents \( D^+ \in D \), \( Q \) denotes the test queries and \( D \) denotes top-K document candidates by a retrieval system. We also call this metric all-positive-macro Recall@N. On the other hand, another recall calculation method following DPR [23] is defined as

\[
\text{DPR-Recall@N} = \frac{1}{|Q|} \sum_{q \in Q} 1_{\exists d \in \text{top-K} \land d \in D^+},
\]

(11)

which we call one-positive-enough Recall@N. Therefore, The official (all-positive-macro) Recall@N is usually less than DPR (one-positive-enough) Recall@N, and the smaller \( N \), the more obvious.

B SUPPLEMENTARY EXPERIMENT SUPPORTS

B.1 BEIR Details

Please refer to Table 12 for detailed results on BEIR benchmark with 12 datasets. It is noteworthy that applying a retriever trained on legacy data (e.g., MS-MARCO labeled in 2016) to the latest topics (e.g., TREC-COVID after 2020) will likely be vulnerable to distribution shifts over time. Because rare or/and brand-new words are usually present over time, we argue that the proposed UnifieR would suffer less from the out-of-vocab (OOV) problem during the distribution shifts. Basically, as we adopted the BERT tokenizer with WordPiece techniques [45], our neural retriever can still model the input text without any interference: Consistent with BERT [9] and coCondender [14] pre-training, some rare words will be split into smaller units (e.g., word ‘idiotsguide’ split into ‘idiots’ and ‘##guide’) before being fed into the Transformer architecture, so obvious OOV problem will not arise. In addition, many previous works [10–12, 19, 48, 57] found that lexicon-based methods can achieve better zero-shot retrieval performance because “lexicon overlap” is an essential feature for them to calculate query-document relevance scores, aligning closely with the goal of first-stage retrieval. Therefore, inheriting the advantages from previous lexicon-based methods, our lexicon retrieval head could alleviate the distribution shifts over time.

B.2 Illustration of Query-side Gating

We illustrate the query-side gating mechanism in Figure 7, which leverages a gating mechanism to dynamically combine lexicon and dense embeddings only at the query side.

B.3 Reranker-taught Pipeline

In contrast to the normal two-stage training pipeline in Figure 3, we present our reranker-taught pipeline in Figure 8.

B.4 Error Cases

As shown in Table 13, we show two representative cases which our proposed method cannot handle.

i) query hubness: The first case shows a query that cannot be tackled by our UnifieR in any retrieval paradigm. However, it is observed that the top-1 passage retrieved by our model can also be considered as a positive passage, which can answer the query ‘what is a dvt’. These negative passages for the query are false negatives, which are brought by the limited crowd-sourcing labeling procedure. Therefore, the poor performance of our model instead proves that our model is more robust, whereas the independent learning model is overfitting to its false negatives, resulting in seemingly good outputs.

ii) Insufficient representation ability: The second case lists the top-retrieved passages for all five retrieval systems. It is shown that compared to independently learned retrieval models (i.e., ‘Indep-den’ and ‘Indep-lex’), our unified models even perform worse and retrieve less relevant passages (refer to UnifieRden’s 1st). An interesting point is that the ‘Ups’-related passage is retrieved by our UnifieRden since ‘upsell’ is tokenized as ‘ups’ and ‘##ell’. This is highly likely since one single model is required to serve dual representing modules, compromising its representation ability.

Meantime, our uni-retrieval can still improve the ranking performance by combining both of the representing worlds.
What is Upselling? Upselling is a sales technique aimed at persuading customers to purchase a more expensive, upgraded or premium version of the chosen item or other add-ons for the purpose of making a larger sale. eCommerce businesses often combine upselling and cross-selling techniques in an attempt to increase order value and maximize profit.

Table 12: Detailed results (NDCG@10) on BEIR benchmark.

| Methods          | Sparse BM25 | DTSQ | UniCOIL | ColBERT | Dense DPR | ANCE | GenQ | TAS-B | Contriever | Ours |
|------------------|-------------|------|---------|---------|-----------|------|------|-------|------------|------|
| TREC-COVID       | 65.6        | 71.3 | 59.7    | 67.7    | 33.2      | 65.4 | 61.9 | 48.1  | 59.6       | 71.5 |
| NFCorpus         | 32.5        | 32.8 | 32.5    | 30.5    | 18.9      | 23.7 | 31.9 | 31.9  | 32.8       | 32.9 |
| NQ               | 32.9        | 39.9 | 36.2    | 32.4    | 47.4      | 44.6 | 35.8 | 46.3  | 49.8       | 51.4 |
| HotpotQA         | 60.3        | 58.0 | 64.0    | 59.3    | 39.1      | 45.6 | 53.4 | 58.4  | 63.8       | 66.1 |
| FiQA             | 23.6        | 29.1 | 27.0    | 31.7    | 11.2      | 29.5 | 30.8 | 30.0  | 32.9       | 31.1 |
| ArQuAna          | 31.5        | 34.9 | 35.5    | 23.3    | 17.5      | 41.5 | 49.3 | 42.9  | 44.6       | 39.0 |
| Tôûche-2020     | 36.7        | 34.7 | 25.9    | 20.2    | 13.1      | 24.0 | 18.2 | 16.2  | 23.0       | 30.2 |
| Dripedia         | 31.3        | 33.1 | 30.2    | 39.2    | 26.3      | 28.1 | 32.8 | 38.4  | 41.3       | 40.6 |
| Scidocs          | 15.8        | 16.2 | 13.9    | 14.5    | 7.7       | 12.2 | 14.3 | 14.9  | 16.5       | 15.0 |
| Fever            | 75.3        | 71.4 | 72.3    | 77.1    | 56.2      | 66.9 | 70.0 | 75.8  | 69.6       |     |
| Climate-FEVER    | 21.3        | 20.1 | 15.0    | 18.4    | 14.8      | 19.8 | 17.5 | 22.8  | 23.7       | 17.5 |
| SciFact          | 66.5        | 67.5 | 67.4    | 67.1    | 31.8      | 50.7 | 64.4 | 64.3  | 67.7       | 68.6 |
| BEST ON          | 1           | 0    | 0       | 2       | 0         | 0    | 1    | 0     | 4          | 4    |
| AVERAGE          | 41.1        | 42.4 | 40.0    | 41.8    | 26.4      | 37.7 | 39.8 | 40.4  | 44.3       | 44.5 |

Table 13: Error analysis on MS-Marco Dev set.

Query: ID:682365/ Query: what is a dvt?

Passage: ID:7544585/ Title: Deep vein thrombosis. Body: For other uses, see DVT (disambiguation). Deep vein thrombosis, or deep venous thrombosis (DVT), is the formation of a blood clot (thrombus) within a deep vein, most commonly the legs. Nonspecific signs may include pain, swelling, redness, warmth, and engorged superficial veins.

Retrieved UniCoIL’s 1st. ID:5404002/ Title: Definition of 'DVT'. Body: Definition of 'DVT'. DVT is a serious medical condition caused by blood clots in the legs moving up to the lungs. DVT is an abbreviation for ‘deep vein thrombosis’. The results from one of the largest studies yet carried out leave little doubt that DVT is caused by flying.

Retrieved UniCoIL’s 1st. ID:6849252/ Title: What is DVT? Body: What is DVT? Deep vein thrombosis is a blood clot that forms inside a vein, usually deep within your leg. About half a million Americans every year get one, and up to 100,000 die because of it. The danger is that part of the clot can break off and travel through your bloodstream.

Retrieved UniCoIL’s 1st. ID:6849252/ Title: What is DVT? Body: What is DVT? Deep vein thrombosis is a blood clot that forms inside a vein, usually deep within your leg. About half a million Americans every year get one, and up to 100,000 die because of it. The danger is that part of the clot can break off and travel through your bloodstream.

Query: ID:1029124/ Query: what is upsell

Passage: ID:7220016/ Title: Upselling. Body: What is Upselling? Upselling is a sales technique aimed at persuading customers to purchase a more expensive, upgraded or premium version of the chosen item or other add-ons for the purpose of making a larger sale. eCommerce businesses often combine upselling and cross-selling techniques in an attempt to increase order value and maximize profit.

Retrieved Indep-lex’s 1st. ID:6288350/ Title: - Body: If you improve inventory turn but pay more, in freight costs for multiple shipments or your warehouse has to increase their variable costs. to process the additional shipments, the net result may be a loss. 4. An upsell feature on

Retrieved Indep-lex’s 1st. ID:7220016/ Title: Upselling. Body: What is Upselling? Upselling is a sales technique aimed at persuading customers to purchase a more expensive, upgraded or premium version of the chosen item or other add-ons for the purpose of making a larger sale. eCommerce businesses often combine upselling and cross-selling techniques in an attempt to increase order value and maximize profit.

Retrieved Indep-lex’s 1st. ID:4754301/ Title: Upselling: 75 Strategies, Ideas and Examples Body:. Upsell Drip Campaign to upsell B2B/SaaS solutions. What is it? The upsell for B2B/Saas solutions email is meant to add to the services. These emails offer premium services or upgrades for users on paying, free or trial accounts. When is it sent? Upsell emails for B2B/Saas solutions are meant to extend the usability and functionality of the software.

Retrieved Indep-lex’s 1st. ID:4754301/ Title: Upselling: 75 Strategies, Ideas and Examples Body:. Upsell Drip Campaign to upsell B2B/SaaS solutions. What is it? The upsell for B2B/SaaS solutions email is meant to add to the services. These emails offer premium services or upgrades for users on paying, free or trial accounts. When is it sent? Upsell emails for B2B/SaaS solutions are meant to extend the usability and functionality of the software.