Enhancing Sparse Retrieval via Unsupervised Learning

Xueguang Ma[*]
University of Waterloo
Waterloo, Canada
x93ma@uwaterloo.ca

Hengxin Fun
Amazon Alexa AI
Seattle, United States
hengfun@gmail.com

Xusen Yin
Amazon Alexa AI
Seattle, United States
yinxusen@gmail.com

Antonio Mallia
Amazon Alexa AI
Seattle, United States
me@antoniomallia.it

Jimmy Lin
University of Waterloo
Waterloo, Canada
jimmylin@uwaterloo.ca

ABSTRACT

Recent work has shown that neural retrieval models excel at text ranking tasks in a supervised setting when given large amounts of manually labeled training data. However, it remains an open question how to train *unsupervised* retrieval models that are more effective than baselines such as BM25. While some progress has been made in unsupervised *dense* retrieval models within a bi-encoder architecture, unsupervised *sparse* retrieval models remain unexplored. We propose BM26, to our knowledge the first such model, which is trained in an unsupervised manner without the need for any human relevance judgments. Evaluations with multiple test collections show that BM26 performs on par with BM25 and outperforms Contriever, the current state-of-the-art unsupervised dense retriever. We further demonstrate two promising avenues to enhance lexical retrieval: First, we can combine BM25 and BM26 using simple vector concatenation to yield an unsupervised hybrid BM51 model that significantly improves over BM25 alone. Second, we can enhance supervised sparse models such as SPLADE with improved initialization using BM26, yielding significant improvements in in-domain and zero-shot retrieval effectiveness.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**.

KEYWORDS

Neural IR, Sparse Retrieval, Unsupervised Learning

ACM Reference Format:
Xueguang Ma, Hengxin Fun, Xusen Yin, Antonio Mallia, and Jimmy Lin. 2023. Enhancing Sparse Retrieval via Unsupervised Learning. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region (SIGIR-AP '23), November 26–28, 2023, Beijing, China.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3624918.3625334

[*]Work partly done while interning at Amazon Alexa AI.

1 INTRODUCTION

Traditional text retrieval methods such as TF–IDF and BM25 treat documents as "bags of words" and assign term weights using a heuristic function [27]. In general, these methods can be characterized as *unsupervised* lexical-matching models.[1] Although such methods date back many decades, they remain strong baselines [18] in various ranking tasks [1, 2], even in the age of deep neural networks [19]. Deployed in popular search platforms such as Elasticsearch, traditional lexical retrieval methods such as BM25 are widely used in industry for real-world search applications due to their robustness to domain and query variations.

There has been much recent progress in neural retrieval models that adopt a bi-encoder architecture to encode queries and documents independently into a representation space [14] using pretrained language models such as BERT [5]. These representations can comprise either dense low-dimensional vectors [7, 14, 20, 29] or sparse high-dimensional vectors [6, 21, 23, 32]. Various models have been shown to be more effective than BM25 under the in-domain supervised learning setting for various document retrieval tasks. That is, given a sufficient amount of labeled training data comprising query–document pairs that have been (manually) judged for relevance, there is no doubt that we can train highly effective models. However, whether we can obtain an effective neural retrieval model that performs better than BM25 in an *unsupervised* setting remains an open question [11, 28].

Existing work on unsupervised neural retrieval models have focused on dense retrievers such as ICT [16], Contriever [11], and cpt-text [24]. At a high level, these models demonstrate how to craft pseudo relevant query–document pairs and how to obtain a large negative candidate pool, two important factors in the effectiveness of unsupervised dense retrievers. To date, though, we are not aware of any unsupervised model demonstrating effectiveness that is unequivocally better than BM25.

We note that in the evolution of unsupervised retrieval models, unsupervised dense retrievers introduce two main innovations compared to a traditional heuristic model such as BM25, which is used as a point of reference: (1) they change heuristic weighting functions to deep neural networks, and (2) they change sparse lexical representations to dense semantic representations. Although changing the representation space gives such models more freedom to fit target labels, they lose the ability to perform exact lexical matches, which

[1]While it is possible to tune parameters (k_1 and b for BM25) with training data, in this work we simply adopt default parameters ($k_1 = 0.9$ and $b = 0.4$) in all experiments.

are more robust to noisy data and domain shifts. Moreover, sparse models are amenable to efficient retrieval using standard inverted indexes, a well-established technology with decades of research that has produced sophisticated query evaluation techniques. Thus, we hypothesize that unsupervised retrieval based on sparse lexical representations has an advantage in terms of robustness compared to those that learn dense semantic representations. Our chain of reasoning is easy to see if we understand BM25 as a bi-encoder model with an unsupervised (i.e., heuristic) encoder [17]. Given this starting point, we propose a method to train a sparse retrieval model in an unsupervised manner. We call our model BM26 (i.e., what comes after BM25).

Our experiments show that BM26 alone is overall more effective than the existing state-of-the-art unsupervised dense retriever Contriever [11] and performs on par with BM25 in terms of effectiveness across a broad range of modern test collections. Furthermore, a retrieval model based on the simple concatenation of BM25 and BM26 representations significantly and consistently outperforms BM25 alone. We call this model BM51 (since $25+26 = 51$). A key feature is that it remains a lexical retrieval model and thus is compatible with retrieval infrastructure based on inverted indexes. This provides a major advantage over dense–sparse hybrid systems, which require separate (approximate) nearest neighbor search libraries for efficient top-k retrieval. We also explore unsupervised sparse retrieval using token expansion to reduce token mismatch issues in lexical matching. These variants, BM26e and BM51e, show potential for further improvements in effectiveness. Finally, we demonstrate that our unsupervised model serves as a better initialization point for supervised sparse models such as SPLADE.

Contributions. In summary, we view this work as having three main contributions:

- First, we introduce an unsupervised sparse retrieval model that we call BM26. This, to our knowledge, represents the first successful attempt to learn a neural lexical retriever in a completely unsupervised manner. BM26 is more effective than existing dense retrieval models in the unsupervised setting.
- Second, we demonstrate how to combine BM25 and BM26 via simple vector concatenation to create BM51, a novel lexical retrieval model that is also unsupervised and remains compatible with inverted indexes. This method has the potential to improve "cold start" systems where no labeled data are available using only lexical matching.
- Finally, we show that supervised sparse models can be enhanced by using our unsupervised model as initialization.

2 BACKGROUND AND RELATED WORK

2.1 Dense Retrieval Models

Throughout this paper we assume the standard definition of the *ad hoc* retrieval problem, where given a corpus $C = \{D_1, D_2, \cdots, D_n\}$ comprised of an arbitrarily large collection of documents and a query Q, the system's task is to return a top-k ranking of documents that maximizes some metric of quality such as nDCG, MRR, etc. The term "documents" here is used generically to refer to the unit of retrieval, even though in actuality the system may be retrieving passages or other units of content (e.g., images).

Dense Passage Retriever (DPR) [14], which we take as an exemplar of a popular and large class of models known as dense retrieval models, adopts a bi-encoder structure to encode queries and documents separately into low-dimensional (e.g., 768 dimensions) dense vector representation as follows:

$$\mathbf{E}_Q = \text{Encoder}_Q(Q), \mathbf{E}_D = \text{Encoder}_D(D)$$

where the encoders are initialized with a pretrained language model such as BERT [5]. The query representation $\mathbf{E}_Q$ and the document representation $\mathbf{E}_D$ are taken from the last layer output of the `[CLS]` token of the corresponding encoder. The relevance between a query and a document is measured by the dot product of their representations, $\text{Sim}(Q, D) = \langle \mathbf{E}_Q, \mathbf{E}_D \rangle$.

Dense retrieval models are typically trained (more precisely, their underlying transformers are fine-tuned) using large amounts of supervised data comprising human-labeled query–document pairs. For DPR, during training, given a query Q, a labeled relevant document D^+, and n non-relevant documents $D_1^-, D_2^-, ...D_n^-$, the model is optimized by contrastive learning using infoNCE loss:

$$\begin{aligned}
&\mathcal{L}(Q, D^+, D_1^-, D_2^-, \cdots, D_n^-) \\
&= -\log p(D = D^+ \mid Q) \\
&= -\log \frac{\exp(\text{Sim}(Q, D^+))}{\exp(\text{Sim}(Q, D^+)) + \sum_{i=1}^{n} \exp(\text{Sim}(Q, D_i^-))}.
\end{aligned}$$

Once the model has been trained, the document encoder can be applied to generate document representations for every document in the corpus (as a preprocessing step). At retrieval time, inference is applied to the query to generate the query representation, and the top-k most similar documents (in terms of dot products) are retrieved. This is operationalized as a nearest neighbor search problem in dense vector space, which can be accomplished by existing libraries such as Faiss [12].

2.2 Unsupervised Dense Retrieval Models

Training dense retrieval models requires large amounts of labeled data. Recently, however, researchers have begun to explore training dense retrieval models in an *unsupervised* manner. The main challenge is how to automatically generate positive *pseudo* query–document pairs on which a model can be trained.

Two existing methods of creating such pseudo pairs are the Inverse Cloze Task (ICT) [16] and Independent Cropping (IC) [11]. Given a text span S composed of a sequence of tokens $\{t_1, t_2, \cdots, t_n\}$, ICT randomly samples a sub-span M from S as the pseudo query and uses the rest of the sequence $S \setminus M$ as the pseudo document to form a positive query–document pair. The Independent Cropping method introduced by the recent Contriever model samples two random sub-spans S_1, S_2 (with replacement) to be the pseudo query and document respectively. This has been shown to be another simple yet promising way to create positive query–document pairs. The Contriever experiments found that IC achieves better effectiveness than ICT for unsupervised dense retriever training [11].

Another challenge of training unsupervised dense retrievers is how to find hard negative documents for each query. Training with hard negatives can improve the discriminability of the model [29]. In a supervised learning setting, the hard negatives for a query, in the simplest case, can be sampled from negative documents in the top results of an existing ranker (e.g., BM25), although the community has since developed much more sophisticated techniques. However, in the unsupervised setting, the most common approach to obtain "harder" negative documents is to increase the negative candidates pool for a larger chance of the model encountering hard negatives. Two existing methods to achieve this are increasing the batch size with in-batch negative training [24] or caching a large negative representation queue with momentum update [10, 11].

3 METHODS

3.1 BM25 as a Bi-Encoder

The starting point of our approach is the representation learning framework for IR articulated by Lin [17]. Recall that given a query Q and a document D, the BM25 ranking model computes the following similarity score:

$$\mathrm{Sim}(Q,D) = \sum_{i=1}^{n} \mathrm{IDF}(q_i) \cdot \frac{\mathrm{TF}(q_i,D)\cdot(k_1+1)}{\mathrm{TF}(q_i,D)+k_1\cdot(1-b+b\cdot\frac{|D|}{avgdl})}$$

where n is the number of tokens in the query and q_i is the i-th token. This formula can be abstracted as:

$$\mathrm{Sim}(Q,D) = \sum_{i=1}^{n} f(q_i,D)$$

where f is a function of token statistics of q_i in D and other global statistics. Instead of summing across the tokens in the query, we can rewrite the formula as a summation across all terms in the entire vocabulary space, where $\mathbb{1}$ is an indicator function for a term appearing in the query:

$$\mathrm{Sim}(Q,D) = \sum_{j=1}^{|V|} \mathbb{1}(v_j,Q)\cdot f(v_j,D)$$

This is in fact equivalent to the dot product of the query representation and the document representation in a vector space:

$$\mathrm{Sim}(Q,D) = \langle \mathbf{E}_Q, \mathbf{E}_D \rangle$$

This formulation is exactly equivalent to a bi-encoder architecture that underlies recent transformer-based dense retrieval models such as DPR [14]. This generic design, however, admits a number of parametric differences, which can be characterized as design choices we can make:

- Basis of the representation vectors: They can be dense (i.e., latent semantic dimensions learned by transformers) or sparse (i.e., the document vocabulary space).
- How weights are assigned: The weight of each dimension can be assigned by a heuristic (i.e., unsupervised) function or a function that has been learned (for example, a transformer).

In other words, BM25 adopts a bi-encoder structure where the query and document representations are generated independently by heuristic function "encoders" that operate on the document vocabulary space. Similarly, dense retrieval models—exemplified by DPR [14]—fall into the category of generating dense semantic representations using encoders that are learned in a supervised setting. Many existing models have demonstrated the effectiveness of a *supervised* approach to learning encoders that operate on the document vocabulary space, e.g., DeepCT [4], DeepImpact [23], uniCOIL [21], and SPLADE [6]. Among these existing models, we adopt the architecture proposed in uniCOIL [21] for training our *unsupervised* lexical retrieval model as its similarity function has the same parametric form as BM25. However, we change the heuristic functions of both query and document "encoders" into neural models. This design also provides a natural point of comparison: Can we build an unsupervised sparse retrieval model that outperforms BM25 utilizing pretrained transformers such as BERT?

3.2 BM26: An Unsupervised Sparse Bi-Encoder

Given the general framework introduced above, BM26 can be characterized as an unsupervised sparse retrieval model. The basis of the vector representation is the BERT token space, and the vector weights are assigned by a transformer that has been fine-tuned *without any human-labeled relevance judgments.*

In our model, the similarity between a query Q and a document D is computed by:

$$\mathrm{Sim}(Q,D) = \sum_{j=1}^{|V|} w_{v_j,Q}\cdot w_{v_j,D}$$

$$w_{v_j,Q} = \mathrm{ReLU}(P\cdot\mathrm{BERT}(Q)_{v_j}) \text{ if } v_j \in Q \text{ else } 0$$

$$w_{v_j,D} = \mathrm{ReLU}(P\cdot\mathrm{BERT}(D)_{v_j}) \text{ if } v_j \in D \text{ else } 0$$

where $\mathrm{BERT}(\cdot)_{v_j}$ is the contextual token representation from the last layer of BERT for token v_j and P is a linear projection that maps the token representation into a scalar weight. Note that if v_j occurs in a query or a document multiple times, we select the maximum scalar weight for v_j as the weight of the token in the text. Exactly as above, this formula is equivalent to the dot product of the query and the document representation in a vector space with a basis defined by the vocabulary: $\mathrm{Sim}(Q,D) = \langle \mathbf{E}_Q, \mathbf{E}_D \rangle$.

Following Contriever, we use independent cropping to create pseudo-positive pairs for contrastive learning. However, we choose not to use the MoCo [10] method to create a large negative candidate pool, as the representations in the pool are not generated by the latest model parameters. As an alternative, we increase the number of negative examples by increasing the batch size to a large number (16384 in our experiments). To achieve such a large batch size with limited GPU memory, we adopt the Gradient Caching [9] method. The training objective for our unsupervised model is the same as training a dense retriever, i.e., using the infoNCE loss.

3.3 BM25 ⊕ BM26 = BM51

Since BM25 and BM26 are both unsupervised sparse retrieval models, we can combine their results to create a hybrid retrieval model that itself remains in the space of unsupervised methods. While linear combination of retrieval scores or reciprocal rank fusion are popular methods for fusion, we create a novel "sparse–sparse" hybrid by vector concatenation in this work. We name this model "BM51" as it is the fusion of BM25 and BM26 (25+26=51).

Specifically, the query representation and document representation of the BM51 retrieval model are computed by:

$$\mathbf{E}_{Q_{\mathrm{BM51}}} = \mathbf{E}_{Q_{\mathrm{BM25}}} \oplus \alpha\cdot\mathbf{E}_{Q_{\mathrm{BM26}}}$$

$$\mathbf{E}_{D_{\mathrm{BM51}}} = \mathbf{E}_{D_{\mathrm{BM25}}} \oplus \beta\cdot\mathbf{E}_{D_{\mathrm{BM26}}}$$

where $\oplus$ represents the vector concatenation operation and α, β adjust the relative contributions of each representation to the final model. Since the token weights from BM25 and BM26 are on different scales, we first normalize the token weights from the two models into integers in the range(0, 256); i.e., these become, in essence, impact scores. For simplicity and to retain the unsupervised setup, we keep $\alpha = \beta = 1$ after normalizing the token weights, which means that BM25 and BM26 are weighted equally. The BM51 query–document similarity is also computed as a dot product.

A detail worth noting: the vector bases of the individual representation vectors from BM25 and BM26 are different, since they are determined by the tokenizer used to process the text. In our implementation of BM25, the vocabulary space is defined by a Lucene analyzer (which, in the case of the MS MARCO passage collection, contains 2.7M unique tokens). In contrast, BM26 operates in the vocabulary space of BERT subwords, which contains 30,522 unique tokens. Thus, in our implementation, the vocabulary size of BM51 is only marginally bigger than that of BM25.

A key feature of our fusion-via-vector-concatenation approach is that BM51 remains a lexical-matching model. This means that top-k retrieval can take advantage of existing infrastructure built around inverted indexes. In addition, vector concatenation presents an advantage over other popular fusion techniques such as the linear combination of scores or reciprocal rank fusion, both of which involve performing retrieval twice and post-processing two ranked lists. In other words, BM51 can serve as a drop-in replacement for BM25. That is, it can serve as a first-stage ranker to provide candidates for further reranking, it can be further combined with supervised models, etc.

3.4 Adding Token Expansion

BM26 adopts the same parametric form as BM25, which means the lexical representation is restricted to tokens that appear in the original text. However, vocabulary mismatch is common in the lexical retrieval setting. SPLADE [6] utilizes the MLM layer of BERT to perform expansion for tokens in the original text: the purpose here is to assign weights to tokens that have close meanings to the existing tokens.

SPLADE was originally designed for the supervised learning setting. We adopt its architecture here to achieve token expansion for *unsupervised* lexical representation learning. Following the same naming convention, we denote this unsupervised sparse retrieval model with expansion as BM26e. Specifically, the similarity between the pseudo query–document pairs is computed by:

$$\mathrm{Sim}(Q, D) = \sum_{j=1}^{|V|} w_{v_j,Q} \cdot w_{v_j,D}$$

$$w_{v_j,Q} = \max_{i \in |Q|} \log(1 + \mathrm{ReLU}(\mathrm{MLM}(\mathrm{BERT}(Q))_{v_{i,j}}))$$

$$w_{v_j,D} = \max_{i \in |D|} \log(1 + \mathrm{ReLU}(\mathrm{MLM}(\mathrm{BERT}(D))_{v_{i,j}}))$$

where the term impact of a token v_j from the vocabulary space V is determined by the max-pooled MLM layer output, corresponding to the token, across all input tokens in a query Q or document D. During training, additional FLOPS loss [25] is added to the infoNCE loss to control the sparsity of query and document expansion:

$$\ell_{\mathrm{FLOPS}} = \sum_{j \in V} \overline{w}_j^2$$

This is calculated as the sum of the squares of the average impacts of each token within a batch. The FLOPS loss for the query and the document are weighted by λ_q and λ_d. We set both to 0.001 in all our experiments for BM26e.

Similar to the way we build BM51 = BM25 $\oplus$ BM26, we assemble the "with expansion" version BM51e = BM25 $\oplus$ BM26e. We follow the same vector concatenation and token weights processing as described in Section 3.3.

4 ENHANCING UNSUPERVISED RETRIEVAL

4.1 Experimental Setup

4.1.1 Data. Following the data crafting method in Contriever, we use Independent Cropping on the CCNet corpus [3], an English corpus with 700 million documents extracted from web crawls, to create the unsupervised training data. Specifically, we concatenate all the natural documents in CCNet together and treat each 500-token split as a document. Each training pair is obtained by independent cropping from a random span of 256 tokens in a document. We also apply 10% random token deletion on both the pseudo query and the pseudo document as data augmentation.

4.1.2 Models. We train BM26 and BM26e following the methods proposed in Sections 3.2 and 3.4. We use `bert-base-uncased` as initialization and train 10k steps with batch size 16384. Our model is trained on a single AWS EC2 instance with 8× A100 40 GB GPUs, with code based on the open-source toolkit Tevatron [8].

Our retrieval experiments are performed using the Pyserini IR toolkit [18] using the "fake words" trick, which is a common technique that allows sparse retrieval models to transparently reuse inverted indexes and existing query evaluation machinery [22]. For each document vector, Pyserini (internally) generates a "fake document" where a token is repeated a number of times equal to its (quantized) weight; i.e., if a token gets a weight of 5, the fake document contains five copies of the token. As part of the normal indexing process, these weights are written to the term frequency position in the postings of a standard inverted index. Inner product can be implemented as a similarity function that computes the product between the query "term frequency" and the document "term frequency" (i.e., both are in actuality the original weights assigned in the respective vectors). BM51 is implemented exactly in the manner described in Section 3.3: We materialize the document vectors from BM25 and BM26 and then feed the concatenated vectors back into Pyserini for indexing and retrieval.

Finally, we evaluate two variants that allow us to precisely attribute differences in model effectiveness:

- BM25$_{\mathrm{wp}}$: This represents the BM25 weighting function applied directly to wordpiece tokens. That is, BM25$_{\mathrm{wp}}$ and BM26 share exactly the same vector basis, except BM26 weights are computed by a transformer model.
- BM25′: This represents BM25$\oplus$BM25$_{\mathrm{wp}}$, or the concatenation of "standard" BM25 and BM25$_{\mathrm{wp}}$ vectors. That is, BM25′ and BM51 share exactly the same vector basis, differing only in how the BM26 portion of the weights are assigned.

Method	Rep.	MS MARCO		NQ		TriviaQA	
	sparse/dense	MRR@10	R@1k	Top20	Top100	Top20	Top100
(1) BM25	sparse	18.4	85.3	62.9	78.3	76.4	83.2
(2) BM25_{wp}	sparse	17.5	82.6	63.2	77.8	73.8	81.4
(3) $\text{BM25}' = \text{BM25} \oplus \text{BM25}_{\text{wp}}$	sparse	18.9	86.9	64.4	79.4	76.5	83.5
(4) BM26	sparse	19.0$^{\blacktriangle}$	88.6	63.9$^{\blacktriangle}$	78.8	74.3$^{\blacktriangledown}$	82.5
(5) BM26e	sparse	18.1	92.1	70.0$^{\blacktriangle}$	83.3	78.7$^{\blacktriangle}$	85.2
(6) BM51 = BM25⊕BM26	sparse	**22.2**$^{\blacktriangle}$	92.0	68.1$^{\blacktriangle}$	81.6	77.6$^{\blacktriangle}$	83.9
(7) BM51e = BM25⊕BM26e	sparse	20.2$^{\blacktriangle}$	**93.5**	**71.8**$^{\blacktriangle}$	**84.2**	**79.3**$^{\blacktriangle}$	**85.5**
(a) Contriever	dense	16.0	88.1	67.8	82.1	74.2	83.2
(b) cpt-text-S	dense	19.9	-	65.5	77.2	75.1	81.7

Table 1: Retrieval effectiveness of BM26 and BM51 compared to baselines and other unsupervised retrieval models on MS MARCO, NQ, and TriviaQA. We performed significance tests comparing BM26 and BM51 to BM25, indicated by $^{\blacktriangle}/^{\blacktriangledown}$.

	(1) BM25 sparse	(2) BM25_{wp} sparse	(3) BM25′ sparse	(4) BM26 sparse	(5) BM26e sparse	(6) BM51 sparse	(7) BM51e sparse	(a) Contriever dense	(b) cpt-text-S dense
Arguana	39.7	36.4	39.0	38.1$^{\blacktriangledown}$	42.2$^{\blacktriangle}$	41.7$^{\blacktriangle}$	**43.1**$^{\blacktriangle}$	37.9	38.7
Climate-FEVER	16.5	15.8	17.1	15.8	18.9$^{\blacktriangle}$	18.9$^{\blacktriangle}$	**20.0**$^{\blacktriangle}$	15.5	15.8
DBpedia	31.8	28.4	31.1	28.6$^{\blacktriangledown}$	33.0	34.0$^{\blacktriangle}$	**36.5**$^{\blacktriangle}$	29.3	27.2
FEVER	65.1	65.8	67.3	74.4$^{\blacktriangle}$	75.1$^{\blacktriangle}$	75.2$^{\blacktriangle}$	**77.9**$^{\blacktriangle}$	68.2	57.1
FiQA	23.6	21.8	24.2	25.8$^{\blacktriangle}$	28.9$^{\blacktriangle}$	28.3$^{\blacktriangle}$	30.1$^{\blacktriangle}$	24.5	**34.1**
HotpotQA	63.3	59.3	63.4	63.6	64.0$^{\blacktriangle}$	**68.2**$^{\blacktriangle}$	67.9$^{\blacktriangle}$	48.1	51.5
NFCorpus	32.2	31.4	32.8	32.4	33.3	34.2$^{\blacktriangle}$	**34.7**$^{\blacktriangle}$	31.7	32.0
NQ	30.6	30.5	31.8	27.9$^{\blacktriangledown}$	29.9	33.9$^{\blacktriangle}$	**33.4**$^{\blacktriangle}$	25.4	-
Quora	78.9	73.0	78.0	77.5$^{\blacktriangledown}$	82.4$^{\blacktriangle}$	82.3$^{\blacktriangle}$	**83.5**$^{\blacktriangle}$	**83.5**	68.1
SCIDOCS	14.9	13.8	15.0	15.0	16.0$^{\blacktriangle}$	15.9$^{\blacktriangle}$	**16.8**$^{\blacktriangle}$	14.9	-
Scifact	67.9	67.2	69.4	66.7	67.9	69.8$^{\blacktriangle}$	**70.9**$^{\blacktriangle}$	64.9	65.4
TREC-COVID	59.5	56.5	60.0	54.4	34.9$^{\blacktriangledown}$	**65.6**	53.9	27.4	52.9
Touche-2020	44.2	46.6	**50.2**	26.1$^{\blacktriangledown}$	23.0$^{\blacktriangledown}$	34.7$^{\blacktriangledown}$	28.6$^{\blacktriangledown}$	19.3	21.0
Avg.	43.7	42.0	44.7	42.0	42.3	**46.4**	46.0	37.7	-

Table 2: Retrieval effectiveness (nDCG@10) of BM26 and BM51 compared to baselines and other unsupervised retrieval models on BEIR. We performed significance tests comparing BM26 and BM51 to BM25, indicated by $^{\blacktriangle}/^{\blacktriangledown}$.

4.1.3 Evaluation. We evaluate our models on the following three datasets that draw from diverse domains:

- MS MARCO Passage Ranking [1]: a web search dataset with 8.8 million passages and 6980 queries in the development set for evaluation. Effectiveness is measured by MRR@10 and Recall@1k.
- NaturalQuestions [15] and TriviaQA [13]: open-domain question answering datasets that use English Wikipedia as the corpus. Effectiveness is measured by top-20/100 retrieval accuracy.
- BEIR [28]: a collection of datasets for zero-shot evaluation of search and related tasks. We evaluate our models on the 13 publicly available datasets; effectiveness is measured by nDCG@10.

4.2 Results

Table 1 presents results on MS MARCO, NQ and TriviaQA, and Table 2 presents results on the 13 BEIR datasets. In both tables, we perform significance testing using paired t-tests ($p < 0.05$); the symbols $^{\blacktriangle}/^{\blacktriangledown}$ indicate significant differences (better/worse). The BM25 baselines are obtained by following the reproduction guides in the open-source Anserini IR toolkit [31]. The results of Contriever and cpt-text-S are copied from the original papers.

4.2.1 Evaluation of BM26. First, we compare BM26 with BM25. The high level conclusion is that they are roughly on par, which can be seen by comparing row (4) with row (1) in Table 1 on MS MARCO, NQ, and TriviaQA, and by comparing column (4) with column (1) in Table 2 for the BEIR datasets. Despite the statistical significance of some differences, overall the effect sizes are small: less than a point in most cases.

For BM25, the default Lucene analyzer in Anserini generates 2.7M unique tokens for the MS MARCO passage corpus, which is orders of magnitude larger than the vocabulary space of the `bert-base-uncased` tokenizer (30,522 unique subwords). To isolate the effects of this difference, we turn to BM25_{wp}, which applies BM25 term weighting on wordpiece tokens. Thus, BM25_{wp} shares exactly the same representation space as BM26, providing a fair comparison between neural and heuristic encoders. Here, we seen that BM26 outperforms BM25_{wp}. Turning our attention to effectiveness on the BEIR datasets in Table 2, column (4) vs. column (1), we see that BM26 is significantly better than BM25 for two datasets and worse for five datasets. Overall, BM26 is 1.7 points worse than BM25, but BM26 and BM25_{wp} achieve the same level of effectiveness.

Method	MRR@10	Speed
BM25	18.4	100 %
BM51	22.2	34 %
- min-idf=1	22.3	39 %
- min-idf=2	22.2	47 %
- min-idf=3	22.1	70 %
- min-idf=4	19.9	139 %
- min-idf=5	15.9	253 %

Table 3: Relative retrieval performance (in terms of QPS) of BM51 compared against BM25 with different levels of query token filtering based on IDF values.

Comparing BM26 to Contriever and cpt-text-S on the datasets in Table 1, shown in row (a) and row (b), respectively, we see some cases where our method performs better and other cases where our method performs worse. At a high level, it would be fair to characterize effectiveness as "comparable". On BEIR, however, BM26 scores four points higher than Contriever. Although cpt-text-S results are not available for all BEIR datasets, it does appear that BM26 obtains higher scores in most cases. Note that significance tests are not possible here because we do not have access to the raw run files from these models.

4.2.2 *Evaluation of BM51.* Next, we investigate how BM26 helps improve BM25 via our vector concatenation technique. The main comparison condition is between BM51 (= BM25 $\oplus$ BM26) and BM25. We perform significance testing and denote better/worse results exactly in the same manner as above.

From Table 1, we see significant improvements across all three datasets, comparing row (6) to row (1). In the case of MS MARCO, we observe a nearly four point gain, which translates into a 21% relative improvement. From Table 2, comparing column (6) with column (1), we see that BM51 is significantly more effective than BM25 for all BEIR datasets except Tóuche-2020. This corpus contains many long documents, which might be the reason why our model performs worse than any BM25 variant. However, Tóuche-2020 appears to be an outlier, as other existing models [6, 11, 24] also perform poorly. We emphasize that in all cases, our gains are obtained without the use of manual labels.

To untangle the effectiveness improvements due to BM26 from improvements from simply having a hybrid vocabulary base, we can compare BM51 to BM25′ = BM25$\oplus$BM25$_{\text{wp}}$. Recall that the latter represents the concatenation of BM25 and BM25$_{\text{wp}}$: the BM25 components are identical, and as explained above, BM25$_{\text{wp}}$ and BM26 share exactly the same representation space, differing only in how term weights are computed. We see that BM51 consistently outperforms BM25′, which demonstrates that BM26 provides additional relevance signals beyond a simple vocabulary space hybrid.

Comparing against dense retrieval models: We see that BM51 is more effective than Contriever and cpt-text-S across most datasets. On MS MARCO, BM51 outperforms Contriever by over six points, and on BEIR, nine points.

As shown in Table 3, the performance of BM51 (measured in terms of queries-per-second throughput on a single thread) is lower than BM25 due to its denser representation. However, this efficiency can be enhanced while maintaining retrieval effectiveness by filtering out query tokens with low inverse document frequency during the search process (i.e., skipping excessively long inverted lists). As the table indicates, when the minimum inverse document frequency (min-idf) is set to 3 or 4, performance approaches that of BM25, but with BM51 exhibiting superior effectiveness.

4.2.3 *Evaluation of BM26e and BM51e.* Finally, we investigate the effectiveness of BM26e and BM51e, which incorporate token expansion to address the vocabulary mismatch issue. We compare BM26e and BM51e with their non-expanded counterparts, BM26 and BM51, respectively.

In Table 1, comparing row (5) to row (4), we observe that BM26e performs better than BM26 on NQ, and TriviaQA. This indicates that token expansion is beneficial for improving retrieval effectiveness. In Table 2, comparing column (7) with column (6), we find that BM51e is more effective than BM51 on 9 out of the 13 BEIR datasets. These results highlight the value of token expansion in addressing the vocabulary mismatch issue and improving the retrieval effectiveness of unsupervised sparse retrieval models.

5 ENHANCING SUPERVISED RETRIEVAL

5.1 Experimental Setup

5.1.1 *Data.* We use the training set of the MS MARCO passage ranking test collection [1] to train our models. During training, positive labels are taken from human judgments and the in-batch negatives are selected from the BM25 negatives. In-domain retrieval effectiveness is evaluated on the development set, measured by MRR@10 and Recall@1k. Zero-shot retrieval effectiveness is evaluated on the aforementioned BEIR datasets, measured by nDCG@10.

5.1.2 *Models.* BM26e has shown higher unsupervised retrieval effectiveness than BM26, based on results reported in Section 4.2. Therefore, in these experiments, we study the effectiveness of using BM26e as initialization to train SPLADE [6], denoted as SPLADE$_{\text{BM26e}}$. Specifically, we replace the `bert-base-uncased` initialization with the BM26e checkpoint and then fine-tune the model on the MS MARCO passage training set for 3 epochs with batch size 256. Each training example contains 1 positive passage and 7 sampled BM25 hard negative passages. We use the last checkpoint for both in-domain and zero-shot evaluation.

5.1.3 *Baselines.* We compare our SPLADE$_{\text{BM26e}}$ model with other representative supervised sparse models like uniCOIL [21] and the original SPLADE [6]. We have built our own implementation of SPLADE, denoted SPLADE$_{\text{repro}}$, which has the same hyperparameter setting as SPLADE$_{\text{BM26e}}$, but is initialized with the standard `bert-base-uncased` checkpoint. We report results for both SPLADE$_{\text{repro}}$ and the original SPLADE, copied from the authors' paper. While recent studies suggest that the effectiveness of SPLADE and other supervised sparse models can be further enhanced through hard negative mining and distillation from a cross-encoder reranker [6, 20, 26, 29], we have only compared models trained on BM25 hard negatives to maintain a fair comparison.

5.2 Results

5.2.1 *Evaluation of In-Domain Retrieval.* In Table 5, we present in-domain retrieval results on the MS MARCO passage ranking task.

	(1) BM25 sparse	(2) BM51e sparse	(3) uniCOIL sparse	(4) SPLADE sparse	(5) SPLADE$_{\text{repro}}$ sparse	(6) SPLADE$_{\text{BM26e}}$ sparse	(a) CoCondenser dense	(b) Contriever dense
Arguana	39.7	43.1	39.6	43.9	47.4	**53.5**	29.9	44.6
Climate-FEVER	16.5	20.0	18.2	19.9	21.2	21.9	14.4	**23.7**
DBPedia	31.8	36.5	33.8	36.6	38.8	**41.8**	36.3	41.3
FEVER	65.1	77.9	81.2	73.0	78.1	**81.4**	49.5	75.8
FiQA	23.6	30.1	28.9	28.7	30.0	**34.2**	27.6	32.9
HotpotQA	63.3	67.9	66.7	63.6	63.7	**70.7**	56.3	63.8
NFCorpus	32.2	**34.7**	33.3	31.3	33.9	34.6	32.5	32.8
NQ	30.6	33.4	42.5	46.9	48.6	**52.2**	48.7	49.5
Quora	78.9	83.5	66.3	83.5	75.5	84.4	85.6	**86.5**
SCIDOCS	14.9	**16.8**	14.4	14.5	14.5	15.6	13.7	16.5
SciFact	67.9	**70.9**	68.6	62.8	65.8	70.8	61.5	67.7
TREC-COVID	59.5	53.9	64.0	67.3	**72.7**	70.1	71.2	59.6
Touche-2020	**44.2**	28.6	29.8	31.6	28.0	29.8	19.1	23.8
average	43.7	46.0	45.3	46.4	47.6	**50.8**	42.0	47.5

Table 4: Zero-shot retrieval effectiveness of SPLADE initialized with BM26e, compared to baselines and other supervised sparse retrieval models on BEIR. All supervised models are trained with only BM25 hard negatives.

	MRR@10	Recall@1k
(1) BM25	18.4	85.3
(2) BM51e	20.2	93.5
(3) SPLADE	34.0	96.5
(4) SPLADE$_{\text{repro}}$	33.5	97.0
(5) SPLADE$_{\text{BM26e}}$	35.2	98.0

Table 5: In-domain retrieval effectiveness of SPLADE initialized with BM26e, compared to baselines and other supervised sparse retrieval models on MS MARCO passage ranking task.

Top-k Mask	SPLADE$_{\text{repro}}$	SPLADE$_{\text{BM26e}}$
∞	41.5	37.8
512	41.5	44.0
256	42.9	50.3
128	47.4	53.5
64	43.7	51.9

Table 6: Comparison of nDCG@10 of applying different top-k masks to query and document representations from the SPLADE model on the Arguana dataset.

Our SPLADE$_{\text{BM26e}}$ model outperforms the other baseline models in terms of both MRR@10 and Recall@1k, thus illustrating the benefits of our proposed approach for model initialization. The SPLADE$_{\text{BM26e}}$ model achieves an MRR@10 of 35.2, which is a 1.7 point improvement over SPLADE with the default initialization (note that our SPLADE$_{\text{repro}}$ performs slightly worse). These results demonstrate that using our BM26e unsupervised sparse retrieval model to initialize the weights of supervised models can improve effectiveness.

5.2.2 Evaluation of Zero-Shot Retrieval. In Table 4, we provide zero-shot retrieval effectiveness results on the BEIR datasets. We see that SPLADE$_{\text{BM26e}}$ consistently outperforms other models in most retrieval tasks, thus demonstrating its superior generalizability in a zero-shot retrieval scenario. On average, SPLADE$_{\text{BM26e}}$ achieves an overall score of 50.8, which is 3 points higher than the SPLADE$_{\text{repro}}$ model initialized by `bert-base-uncased` (the original SPLADE is slightly less effective). This further solidifies the effectiveness of our proposed approach in using our unsupervised sparse retrieval model as the initialization of a supervised model for enhancing retrieval effectiveness in a zero-shot setting.

We note that for all datasets in BEIR, with the exception of Arguana, we maintain all token weights for both query and document sparse representations during search. However, for Arguana, which is a dataset designed for passage-to-passage counter-argument retrieval, we discover that applying a top-k token mask [30] with $k = 128$ to sparsify both query and document representations yields significantly higher effectiveness than the default setting, as shown in Table 6. We hypothesize that reducing the k value could help filter out tokens that align with low-level contexts, potentially causing distractions during retrieval. We also believe there is potential to improve asymmetric retrieval by applying different k values to query and document representations. However, these additional explorations are saved for future work.

6 CONCLUSION

In this work, we propose an unsupervised sparse retrieval model called BM26 that adapts techniques originally designed for dense retrieval to a sparse lexical representation space. We find that our unsupervised sparse approach is on par with BM25 but holds great potential to enhance existing lexical retrieval systems. Our unsupervised sparse retriever shines in scenarios where there is a scarcity of relevance judgements. It successfully augments heuristic retrieval methods such as BM25 through the creation of a hybrid vector space. This allows for a "cold start" retrieval system to achieve higher effectiveness than with BM25 alone. Furthermore, given sufficient relevance judgements, our unsupervised sparse model (for instance, BM26e) can serve as better initialization for supervised fine-tuning with human-labeled relevance judgments.

ACKNOWLEDGMENTS

This research was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

REFERENCES

[1] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv:1611.09268v3* (2018).

[2] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada, 1870–1879.

[3] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online, 8440–8451.

[4] Zhuyun Dai and Jamie Callan. 2020. Context-Aware Term Weighting For First Stage Passage Retrieval. In *Proceedings of the 43rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2020)*. 1533–1536.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota, 4171–4186.

[6] Thibault Formal, C. Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval. *arXiv:2109.10086* (2021).

[7] Luyu Gao and Jamie Callan. 2021. Unsupervised Corpus Aware Language Model Pre-training for Dense Passage Retrieval. *arXiv:2108.05540* (2021).

[8] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Tevatron: An Efficient and Flexible Toolkit for Neural Retrieval. In *Proceedings of the 46th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2023)*. Taipei, Taiwan, 3120–3124.

[9] Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. 2021. Scaling Deep Contrastive Learning Batch Size under Memory Limited Setup. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*. 316–321.

[10] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 9726–9735.

[11] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised Dense Information Retrieval with Contrastive Learning. *arXiv:2112.09118* (2021).

[12] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[13] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada, 1601–1611.

[14] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online, 6769–6781.

[15] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics* 7 (2019), 452–466.

[16] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy, 6086–6096.

[17] Jimmy Lin. 2021. A Proposed Conceptual Framework for a Representational Approach to Information Retrieval. *arXiv:2110.01529* (2021).

[18] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*. 2356–2362.

[19] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2021. *Pretrained Transformers for Text Ranking: BERT and Beyond*. Morgan & Claypool Publishers.

[20] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021. In-Batch Negatives for Knowledge Distillation with Tightly-Coupled Teachers for Dense Retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*. Online, 163–173.

[21] Xueguang Ma, Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2022. Document Expansions and Learned Sparse Lexical Representations for MS MARCO V1 and V2. In *Proceedings of the 45th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2022)*. Madrid, Spain, 3187–3197.

[22] Joel Mackenzie, Andrew Trotman, and Jimmy Lin. 2021. Wacky Weights in Learned Sparse Representations and the Revenge of Score-at-a-Time Query Evaluation. *arXiv:2110.11540* (2021).

[23] Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. 2021. Learning Passage Impacts for Inverted Indexes. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*. 1723–1727.

[24] Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David Schnurr, Felipe Petroski Such, Kenny Hsu, Madeleine Thompson, Tabarak Khan, Toki Sherbakov, Joanne Jang, Peter Welinder, and Lilian Weng. 2022. Text and Code Embeddings by Contrastive Pre-Training. *arXiv:2201.10005* (2022).

[25] Biswajit Paria, Chih-Kuan Yeh, Ning Xu, Barnabás Póczos, Pradeep Ravikumar, and Ian En-Hsu Yen. 2020. Minimizing FLOPs to Learn Efficient Sparse Representations. *arXiv:2004.05665* (2020).

[26] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online, 5835–5847.

[27] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389.

[28] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[29] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*.

[30] Jheng-Hong Yang, Xueguang Ma, and Jimmy Lin. 2021. Sparsifying Sparse Representations for Passage Retrieval by Top-k Masking. *arXiv:2112.09628* (2021).

[31] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *Proceedings of the 40th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017)*. Tokyo, Japan, 1253–1256.

[32] Shengyao Zhuang and Guido Zuccon. 2021. Fast Passage Re-ranking with Contextualized Exact Term Matching and Efficient Passage Expansion. *arXiv:2108.08513* (2021).