KALE: Using a K-Sparse Projector for Lexical Expansion

Luís Borges
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, USA
lborges@andrew.cmu.edu

Bruno Martins
IST and INESC-ID
University of Lisbon
Lisbon, Portugal
bruno.martins@tecnico.ulisboa.pt

Jamie Callan
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, USA
callan@andrew.cmu.edu

ABSTRACT

Recent research has proposed retrieval approaches based on sparse representations and inverted indexes, with terms produced by neural language models and leveraging the advantages from both neural retrieval and lexical matching. This paper proposes KALE, a new lightweight method of this family that uses a small model with a k-sparse projector to convert dense representations into a sparse set of entries from a latent vocabulary. The KALE vocabulary captures semantic concepts than perform well when used in isolation, and perform better when extending the original lexical vocabulary, this way improving first-stage retrieval accuracy. Experiments with the MSMARCOv1 passage retrieval dataset, the TREC Deep Learning dataset, and BEIR datasets, examined the effectiveness of KALE under varying conditions. Results show that the KALE terms can replace the original lexical vocabulary, with gains in accuracy and efficiency. Combining KALE with the original lexical vocabulary, or with other learned terms, can further improve retrieval accuracy with only a modest increase in computational cost.

CCS CONCEPTS

• **Information systems** → **Query representation**; **Document representation**; *Retrieval effectiveness*; *Retrieval efficiency*.

KEYWORDS

Neural Information Retrieval, Learned Sparse Representations, Efficiency in Neural Retrieval.

ACM Reference Format:
Luís Borges, Bruno Martins, and Jamie Callan. 2023. KALE: Using a K-Sparse Projector for Lexical Expansion. In *Proceedings of the 2023 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '23), July 23, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3578337.3605131

1 INTRODUCTION

Neural retrieval approaches, based on the computation of dense vector representations for documents and queries, tend to outperform methods based on sparse representations [26, 41]. Dense methods typically perform a re-ranking of a small set of results obtained from a first-stage retriever, or they use in-memory indexes for performing either exhaustive or Approximate Nearest-Neighbour (ANN) searches over the document collection. On the other hand, methods based on sparse representations use inverted indexes for efficient retrieval over large collections stored in disk, and these particular indexing and query processing strategies have been extensively studied within the community [37, 43]. It is therefore of interest to consider ways of computing sparse representations with the same type of neural approaches that support the better performing dense representations, envisioning indexing with inverted indexes.

The dual encoder architecture is a popular method of converting text to dense representations [8, 9, 14, 20, 21, 30–32, 39]. Queries and documents are processed with a Transformer encoder model (e.g., BERT), and a similarity between the resulting dense vectors can then be computed. The current research focus with dense retrieval lies on the choice of the negative examples for training [21, 32, 39], knowledge distillation [20, 21], better training methodologies [9, 30, 32], and larger models [27]. Dense representations are typically stored in memory and incur not only in large memory requirements, but also in larger search times, since queries are compared to every document in the collection. On the other hand, modern sparse approaches generally leverage Transformer models to process the text and create new sparse representations, which involve reweighting existing document terms, expanding the document with new terms, or both. These methods either project the dense representations into a known lexical vocabulary, e.g. the BERT wordpiece vocabulary [7, 24], or into new high dimensional vocabulary spaces [12], often with a larger vocabulary size than BERT. State-of-the-art learned sparse representations typically perform reweighting/expansion over the BERT vocabulary, but are limited to the concepts captured by the aforementioned set of terms. Additionally, expanding queries/documents can become expensive, in terms of query latency and index size. Approaches that project dense representations into high dimensional spaces usually underperform reweighting/expansion techniques, and are used in isolation, replacing the existing English vocabulary.

This paper presents KALE (**K**-sp**A**rse Projector for **L**exical **E**xpansion), a simple and fast approach based on a typical dual-encoder that produces sparse representations in a new vocabulary space. Instead of relying on the BERT vocabulary, a new vocabulary is created, with the goal of allowing a neural model to generate important semantic concepts from its training dataset. Ideally, these terms should capture concepts beyond those from the lexical vocabulary, and can be used either as a replacement of the original English vocabulary, or as an addition to existing representations.

KALE consists of a simple encoder, leveraging a DistilBERT model, in order to generate sparse representations of the input text. In order to train the model, a frozen teacher distills knowledge into

ICTIR '23, July 23, 2023, Taipei, Taiwan.

ACM ISBN 979-8-4007-0073-6/23/07…$15.00
https://doi.org/10.1145/3578337.3605131

Dense approaches	MRR@10
(A) DPR [14]	0.311
(B) ANCE [39]	0.330
(C) TCT-ColBERT [20]	0.364
(D) Condenser [8]	0.366
(E) RocketQA [30]	0.370
(F) TCT-ColBERTv2 [21]	0.375
(G) CoCondenser [9]	0.382
(H) RocketQAv2 [32]	0.388
(I) MASTER [42]	0.415
(J) RetroMAEv2 [38]	0.426
(K) ABS [2] + CoCond. [9]*	0.447

Sparse approaches	MRR@10
(a) BM25 [33]	0.187
(b) DeepCT [6]	0.244
(c) DocT5Query [28]	0.277
(d) CCSA [18]	0.289
(e) UHD-BERT [12]	0.300
(f) SPLADE [7]	0.322
(g) DocT5Query– [10]	0.323
(h) DeepImpact [24]	0.326
(i) SpaDE [3]	0.355
(j) EfficientSPLADE [17]	0.380
(k) LexMAEv2 [34]	0.426

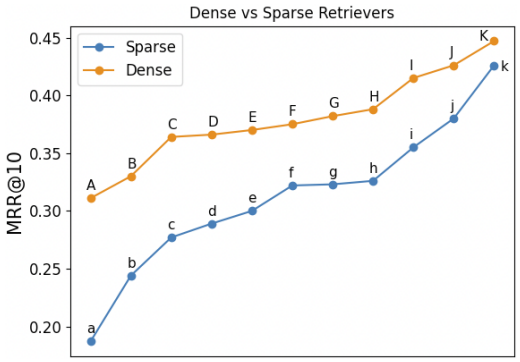

Figure 1: Experimental results for the methods presented in Section 2. Evaluation is performed on MSMARCOv1, and the reported metric is the official MSMARCOv1 metric of MRR@10. The rightmost plot shows each of the retrievers in the tables, with the *x* axis roughly corresponding to a time dimension (i.e., methods on the right are usually more recent, although that is not always the case). The symbol * denotes a submission to the MSMARCOv1 passage ranking leaderboard, which includes further improvements not covered in the original paper.

a learnable DistilBERT. This learnable language model then outputs a vector to be further sparsified and indexed, either in isolation or together with existing lexical vocabularies.

The proposed approach was tested on the MSMARCOv1 [1], TREC DL [4, 5], and BEIR [36] datasets for passage retrieval. The experimental results confirm that the KALE vocabulary captures distinct concepts not always present in the lexical terms of the document. The new terms, on their own, significantly outperform a BM25 baseline based on the original lexical terms. When appended to the lexical terms, or to the representations generated by other sparse retrievers, KALE is consistently able to provide a further increase in effectiveness, at a small price in terms of query latency.

Section 2 reviews the state-of-the-art in terms of both dense and sparse retrieval models. Section 3 describes our KALE model. Section 4 elaborates on the datasets and evaluation methods, while Section 5 presents and discusses our experimental results. Section 6 reflects on some limitations of KALE, and finally Section 7 presents the conclusions and ideas for future work.

2 RELATED WORK

This section presents an overview of relevant related work, covering both dense and sparse representations. Figure 1 aggregates state-of-the-art results associated to those two directions.

2.1 Retrieval with Dense Representations

A popular architecture for dense retrieval, known as the dual-encoder architecture, uses a BERT-based model to independently encode the query and the document into real valued vectors. The document vectors can be pre-computed, indexed in memory, and searched with libraries such as FAISS [13]. Dual-encoders contrast with the slower cross-encoders, with the latter jointly encoding queries and passages, and being commonly used for re-ranking.

Recent work is focused on optimizing the choice of negative examples for loss computation, distilling knowledge from better performing models, or employing better (pre-)training methodologies. For instance, DPR [14] uses in-batch negatives, and BM25 negatives. ANCE [39] chooses hard negatives by performing ANN search on the representations derived from its current model state.

One of the first knowledge distillation methods [11] had a cross-encoder computing scores between queries and documents, using them to train a faster student model with a margin loss to ensure the differences between scores were similar between student and teacher. Later, the same authors [20] released TCT-ColBERT, consisting of a dual encoder student, and a ColBERT [15] teacher. The teacher computes scores for the positive documents and in-batch negatives, and the student is trained with a loss composed of a cross entropy component, together with a Kullback-Leibler (KL) divergence between the distributions of student and teacher scores. In subsequent work, Lin et al. [21] further improved TCT-ColBERT with hard negative mining.

Regarding more nuanced training methodologies, RocketQA [30] trains a dual encoder with cross-batch negatives along multiple GPUs, uses a cross-encoder to sample hard negatives with a high confidence score, and augments the training corpus by having a cross-encoder create high confidence positive and negative documents, in response to a set of unlabeled queries. The follow-up work, RocketQAv2 [32], jointly optimized document retrieval with a dual encoder, with further re-ranking with a cross-encoder.

The current state-of-the-art involves target corpus pre-training before fine-tuning. Gao et al. [8] proposed the Condenser, i.e., a Transformer architecture able to better aggregate information into the `[CLS]` token representations from a BERT model, followed by CoCondenser [9], i.e. a strategy where the Condenser is additionally pre-trained with a contrastive loss. Recently, Xiao et al. [23, 38] proposed RetroMAE, composed of a BERT encoder and a shallow Transformer decoder. In this approach, the input text is masked before being fed to the model, with the decoder input being masked more aggressively. Given the representation from the encoder and the highly masked input text, the shallow decoder attempts to reconstruct the original text, forcing a higher quality from the representation output by the encoder. Following the same idea of bottlenecked masked auto-encoding, Zhou et al. [42] proposed

MASTER, which extends this idea into a multi-task setting, with a task-specific decoder for every task.

2.2 Retrieval with Sparse Representations

With sparse retrieval, queries and documents are represented with sparse vectors, usually with size of the lexical vocabulary and where the values different than zero occur on the dimensions corresponding to the vocabulary terms that are present in the text. Those values represent the importance of the term in that text, and common approaches include Term Frequency (TF) weights, or combinations with Inverse Document Frequency (TF-IDF) together with length normalization. The documents are indexed and searched through in-disk inverted indexes, extensively studied in the literature.

Dai et al. [6] proposed DeepCT, using BERT to estimate the term frequencies of sparse representations, afterwards doing ranked retrieval with the BM25 [33] approach. Still, one of issues with BM25 and DeepCT is the vocabulary mismatch problem, where different words with the same meaning never match.

Lin et al. [29] proposed Doc2Query and later DocT5Query [28], which use a language model to generate queries to which the documents might be relevant, later using them to expand the documents. This does both re-weighting and expansion with new vocabulary. Recently, Gospodinov et al. [10] filtered the queries from DocT5Query with a cross-encoder before expansion, getting a strong performance boost. Another step forward was proposed by Mallia et al. [24], first expanding the original documents with DocT5Query, and then computing impact scores for every term in the expanded passage. These impacts are stored in the index, and the score between a query and a document is their dot product.

Choi et al. [3] proposed SpaDE, which uses a BERT encoder to learn term weighting, and another BERT encoder with a MLM head to learn expansion. The term weighting encoder creates a score for every input token via a MLP from its hidden states, and max-pools the scores to obtain a term weighting vector. The term expansion encoder is analogous, leveraging the MLM head together with a top-K operation to ensure sparsity.

The Composite Code Sparse Auto-Encoder [18] model takes as input dense vectors, generating a corresponding sparse representation and attempting to reconstruct the original vectors from the sparse representation with a reconstruction loss, using a gumbel softmax activation function to create the sparse vectors. The sparse vectors are further converted to sparse composite codes of dimensionality 65,536, which are then indexed by an inverted index and searched with a counting scoring function.

UHD-BERT also makes use of sparse vectors with a high dimensionality (i.e., 81,920), arguing for a higher expressive power of representations with bigger dimensionalities. Queries and documents are encoded by a shared-weight BERT model. The outputs of every layer within BERT are then sparsified with a top-K operation, together with max-pooling to create one single vector, which the authors call bucket. The final scoring operation between a query and a document is the sum of the dot products between query and document buckets from every BERT layer, and the buckets are indexed with an inverted index.

An approach bearing similarities to the present work is the Standalone Neural Ranking Model (SNRM) from Zamani et al. [40].

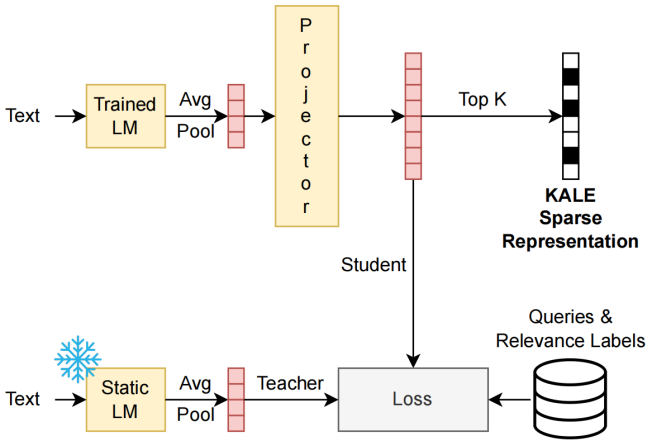

Figure 2: The KALE architecture. Dense vectors are colored in red, while sparse vectors are black and white. The dense vectors derived from the DistilBERT Language Models (LMs) have dimensionality 768. After going through the projector, dense vectors have dimensionality |V| (i.e., the size of the KALE vocabulary). The terms to be indexed are denoted as the KALE sparse representation.

SNRM aggregates *n*-gram representations into a high-dimensional sparse representation (i.e., up to a maximum of 20k terms), therefore also generating a latent vocabulary. The documents are stored in an inverted index and, at query time, documents are retrieved via the dot product between the latent vocabulary weights of query and document vectors. The authors did not evaluate on MSMARCO.

Recently, SPLADE models [7, 17] became increasingly popular. In SPLADE [7], for each term in the input, the MLM head from BERT is used to compute scores for |V| terms, where |V| is the size of the BERT vocabulary. Afterwards, the resulting vectors are max-pooled to create a final representation. An improved version named EfficientSPLADE [17] incorporated multiple improvements: a distillation component; using separate encoders for queries and documents, specifically with a smaller query encoder; using an L1 regularization term for query representations; and performing middle-training before fine-tuning on the retrieval task.

The current state-of-the-art for sparse retrieval was proposed by Shen et al. [34], with a system named LexMAE (Lexicon Bottlenecked Masked Autoencoder). LexMAE is fairly analogous to its dense counterpart named RetroMAE, with an encoder and a shallow decoder trying to reconstruct a heavily masked version of the input text. The encoder first generates a sequence of LM logit vectors of the size of the vocabulary, which is then max-pooled to generate an intermediate representation. This representation is multiplied by the embedding matrix from the encoder, hence creating a bag-of-words bottleneck to serve as additional context for the decoder to reconstruct the aggressively masked input.

3 THE KALE APPROACH

KALE receives as input a piece of text, and returns a sparse representation of the input over a new vocabulary of index terms. Ideally, the sparse representations should be aligned with existing dense representations, which offer good retrieval performance. This is

achieved with a teacher model, which is kept fixed, and with a loss component that aligns the teacher with a learnable student. Additionally, the new index terms should be equally distributed over the corpus, avoiding the skews that are typical of natural vocabularies, and promoting a more efficient search over the inverted index. For this purpose, an equipartitioning component is included in the loss function, which promotes equal weight distribution among the dimensions of the generated KALE representations.

The KALE architecture is composed of two Language Models (LMs), where one LM is learnable, and the other LM is initialized with a fine-tuned checkpoint and has its weights kept frozen. This last LM was not specifically trained to be compatible with sparse index terms. Additionally, both LMs consider vector representations of fixed size. With that in mind, the fine-tuned LM serves as the Teacher LM, producing the dense vectors to which the learnable dense representations are going to be aligned to, through distillation in the loss function. In order to have flexibility regarding different vocabulary sizes other than the LM hidden size, KALE leverages a projector, which consists of a single feed-forward layer, projecting the learned LM vectors to a different vocabulary size (e.g., 1024 terms). The main components in KALE are therefore a learnable DistilBERT model, which is referred to as the Trained Language Model (Trained LM), a teacher model (Teacher LM), and a projector layer. At inference time, a TopK operation sparsifies the output of the projector, outputting an indexable sparse representation. Figure 2 provides an overview of the approach.

KALE gets as input the text of a query/document, and returns a sparse vector representation of its input. First, the Trained LM generates a dense vector representation of the input text, which is the average-pooling of the sequence of its internal hidden states. Alternative dense representations, considering either max-pooling or the `[CLS]` token representation, did not provide any additional benefits. In parallel, the same textual input is fed to the Teacher LM, which generates a dense vector representation, also through an average-pooling operation. The average-pooled dense vector from the Teacher LM serves as target for the learnable dense representations to align to. The learnable representation of the input goes through the projector, which creates a new vector of the size of the vocabulary. The activation within the projector is the ReLU activation function, which naturally already enforces a degree of sparsity, by setting all negative values to zero. Also, the ReLU ensures positivity of the vector values, which is necessary for the TopK operation to function as expected (e.g., negative values with a high absolute value would never be extracted). The representation from the projector then acts as a student in the ranking loss, and is also fed to the equipartitioning component.

At inference time, each query and document is augmented with a set of new terms, which requires KALE to output a sparse representation for each input text. Given the dense vector output by the projector, sparsification is achieved through a TopK operation. The TopK operation grabs a vector, maintains its top K values, and zeroes out the remaining dimensions. The sparsified vector is the KALE sparse representation of the input text, ready to be indexed and searched. The resulting sparse dimensions are indexed with a constant term frequency weight. Different weighting schemes did not provide additional benefits. A K hyperparameter is set for the queries, while another is set for the passages. These hyperparameters are kept fixed. A dynamic K value for every query/passage did not provide significative improvements.

The loss function is a weighted sum of two components. The first component aligns the teacher with the student model, and is the Multi-Margin MSE (M3SE) proposed by Menon et al. [25]. Given a set of student scores $\mathbf{s}$ and teacher scores $\mathbf{t}$, the M3SE is defined as:

$$\text{M3SE}(\mathbf{t}, \mathbf{s}) = \sum_{i \in R} ((t_i - t_{j*}) - (s_i - s_{j*}))^2 + \sum_{j \in N} [s_j - s_{j*}]_+^2, \quad (1)$$

where R is the set of relevant passages for a training query, N is the set of non-relevant passages, and $j*$ is the index of the negative passage with the highest teacher score. This approach enforces a correct margin between the relevant passages and the highest scoring negatives. For the remaining negatives, it suffices to have a lower score than $j*$. The student scores and teacher scores are computed with the cosine similarity. For each query, a relevant passage is fetched from the annotated data. A hard negative is retrieved from the top 50 passages returned from BM25 for that query. The relevant passages of the other queries in the batch serve as in-batch negatives. One aspect to note is that although the KALE sparse representation is the vector to be indexed, initial experiments showed that it was beneficial to connect the dense representation output by the projector into the ranking loss, instead of directly connecting the sparsified vector. Since TopK only extracts a small number of dimensions, the sparse vector likely had insufficient information to provide stable learning compared to the dense vector, and thus this dense vector was kept as the student.

The regularization term promoting equipartitioning takes as input the same vectors that go into the M3SE loss. Given the dense vectors, the sum of the weights in each dimension is computed, as well as the sum of all the weights in the batch. Dividing these values returns the current weight distribution across the dimensions. Ideally, the distribution should be uniform, which translates to:

$$\mathbf{dims} = \sum_{i}^{|B|} \mathbf{B}_i, \; weight = \sum_{v}^{|V|} dims_v, \quad (2)$$

$$\text{Equipartition}(\mathbf{B}) = \text{KL}\left(\frac{\mathbf{dims}}{weight}, \frac{\mathbf{1}}{D}\right) + \text{KL}\left(\frac{\mathbf{1}}{D}, \frac{\mathbf{dims}}{weight}\right). \quad (3)$$

In the previous equations, KL is the Kullback-Leibler divergence, $|B|$ is the batch size, and $|V|$ is the vocabulary size, which matches the dimensionality of the input dense vectors. Vectors and matrices are boldfaced, while scalars are italicized. The term $\frac{\mathbf{1}}{D}$ denotes a uniform distribution over the D dimensions.

With equipartitioning, the KALE loss is the sum of the two individual components:

$$\text{Loss} = \text{M3SE} + \text{Equipartition}. \quad (4)$$

4 DATASETS AND EVALUATION METHODS

KALE was trained on the MSMARCOv1 [1] dataset. About 800.000 training queries have been released, while approximately 100.000 queries were reserved for model development. The passage collection comprises more than 8 million passages. There is on average

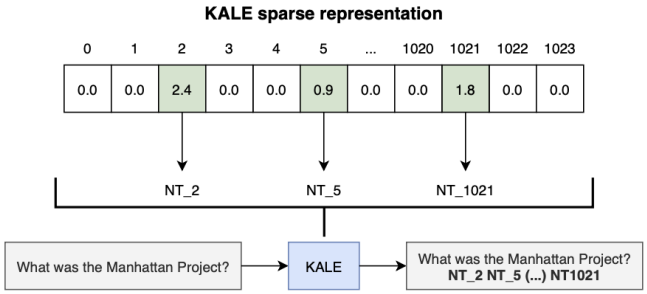

Figure 3: Generating new terms from the KALE sparse representations, and then expanding the input text with the extracted terms. The augmented passages are then ready to be indexed, while expanded queries can be searched over the resulting index. NT stands for New Term.

one relevant passage for each query. Besides MSMARCOv1, evaluation is done on the TREC DL19 [4] and TREC DL20 [5] judged sets of queries - composed of 43 and 54 queries, respectively - which are also searched on the MSMARCOv1 passage collection. The retrieval metrics are Recall@10 for both datasets, MRR@10 for MSMARCOv1, and NDCG@10 for TREC DL.

The learned vocabulary was further evaluated on out-of-domain data, making use of the BEIR benchmark [36]. BEIR is a collection of 18 datasets from multiple text retrieval tasks over multiple domains. Four of those datasets are not public. The tasks range from biomedical information retrieval to citation prediction. In this article, the BEIR benchmarks were divided into search tasks and semantic relatedness tasks, and NDCG@10 is the evaluation metric.

The experiments mostly relied on BM25 scoring, with the exception of impact indexes. After KALE is trained and a sparse representation of a query/passage is obtained, KALE creates a series of artificial terms for the dimensions in which the KALE sparse representation is different than zero, appending each term to the text of the query or passage, with a constant term frequency. The texts can be composed either by the original representations only (what is referred to as BM25), or by the original terms reweighted/expanded by another method (e.g., DeepCT, or DocT5Query). The BM25 hyperparameters were kept at default ($k_1 = 0.9$, $b = 0.4$). Methods with BM25 scoring are expanded with TF=1 for KALE terms. For impact indexes (i.e., DeepImpact and EfficientSPLADE), the artificial vocabulary is also indexed with a constant impact. For DeepImpact, KALE terms have an impact of 10. Regarding EfficientSPLADE, the generated vocabulary is indexed with an impact of 30. Figure 3 illustrates this process, for TF=1, and a vocabulary size of 1,024. Once the corpus is expanded, it can be indexed and stored in disk. At query time, KALE expands the query, and searches the index.

KALE was implemented with the Pytorch library. The teacher LM was kept fixed as the dual-encoder checkpoint from Sentence Transformers [31] named *msmarco-distilbert-cos-v5*. Models were trained during 20 epochs, with a maximum learning rate of 1e-4, and a linear learning rate scheduler. An epoch was considered as a whole pass through the entire set of 500k training queries. Indexing and BM25 search were done with the Pyserini toolkit [19], with 12 threads and a batch size of 64.

5 EXPERIMENTAL EVALUATION

This section presents and discusses experimental results, guided by several research issues to be assessed, namely: (1) the effect of different vocabulary sizes; (2) the effect of the number of artificial terms added to the queries/documents; (3) the usefulness of complementing existing learned sparse retrievers with the KALE vocabulary; (4) the distribution of the generated term posting lists; and (5) the out-of-domain performance of the proposed method.

5.1 Evaluating the Vocabulary Size

KALE argues that the generated vocabulary is able to capture semantic concepts that existing lexical terms might not be able to clearly capture. Smaller vocabularies should imply more abstract concepts, which are perhaps not expressive enough to accurately represent a document, at least on their own. As the vocabulary size increases, KALE should have more expressive power to accurately extract the important concepts from the corpus, leading to better effectiveness. Ideally, higher vocabulary sizes should also result in a smaller average posting list size, and consequent faster search.

This subsection presents experiments varying the size of the KALE vocabulary, with a fixed number of 16 artificial terms for the query, and 64 artificial terms for the passage (i.e., relatively small values in both cases). Two settings were tested, namely one where only the artificial vocabulary was indexed and searched (i.e., KALE only), and another where the lexical English vocabulary was expanded with the KALE terms, and search was performed with BM25 (i.e., BM25+KALE). The average query latency for MSMARCO queries, in milliseconds, was also measured.

Table 1 presents the results from the aforementioned experiments, divided into four table blocks. The first block features the lexical baseline BM25, together with the Teacher LM. This shows not only the effectiveness drop when distilling teacher knowledge into KALE, but also the efficiency gain.

The second block changes the vocabulary size in the KALE only scenario. Regarding accuracy, every vocabulary size either equalled or outperformed BM25 with the original English terms. Both accuracy and efficiency improved as the vocabulary size increased. For sufficiently large sizes, KALE was faster than BM25. Besides outperforming BM25, the KALE vocabulary performed closely to DocT5Query, despite using a smaller DistilBERT model instead of the heavier T5 backbone. Significant improvements on accuracy halted after 8,192 KALE terms, likely given that the number of query and passage terms from KALE was being kept fixed. Larger vocabularies may require more terms to accurately describe the contents of a query/document since, in that setting, KALE is trained to encode its input over a more fine-grained set of concepts.

The third block provides a different view, where the artificial terms were combined with the existing English vocabulary. In this setting, every experiment significantly improved over the BM25 baseline, showing the proposed approach to be strong at complementing existing vocabularies. Very small vocabulary sizes (i.e. 512 and 768) were suboptimal, and presumably the concepts were excessively abstract. The effectiveness peak was reached with 1,024 terms, and larger vocabulary sizes performed worse. The reason may again be that larger vocabularies require more terms to accurately describe a query/document, and these hyperparameters were kept fixed in these experiments.

Table 1: Experimental results when varying the size of the KALE vocabulary. The number of terms in the KALE query representation (k_{query}) was set to 16, while $k_{passage}$ was set to 64. BM25+KALE denotes BM25 search with KALE terms, and the original lexical terms. QL denotes the average MSMARCO query latency, and was measured in milliseconds. The teacher model used for distillation (*msmarco-distilbert-cos-v5*) is also included in the first block of results, in order to assess the extent to which accuracy was decreased when generating the sparse representations. The symbol † denotes statistically significant improvement over BM25, for a paired t-test with a p-value of 0.05.

		MSMARCO Dev		TREC DL 19		TREC DL 20		
Method	|V|	MRR@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	QL
BM25	-	0.184	0.379	0.506	0.129	0.480	0.164	17
Teacher	-	0.338†	0.586†	0.680†	0.143	0.645†	0.207†	182
KALE only	512	0.202†	0.400†	0.521	0.103	0.498	0.166	86
KALE only	768	0.223†	0.430†	0.535	0.100	0.507	0.164	73
KALE only	1024	0.231†	0.442†	0.520	0.100	0.518	0.159	63
KALE only	8192	0.252†	0.455†	0.498	0.095	0.538†	0.173	22
KALE only	32768	0.254†	0.456†	0.550	0.105	0.540†	0.169	16
KALE only	65536	0.251†	0.445†	0.535	0.100	0.569†	0.182	14
KALE only	98304	0.254†	0.453†	0.547	0.097	0.562†	0.178	15
KALE only	131072	0.251†	0.450†	0.559	0.106	0.565†	0.167	14
BM25+KALE	512	0.294†	0.565†	0.653†	0.151	0.639†	0.213†	93
BM25+KALE	768	0.308†	0.569†	0.657†	0.149	0.646†	0.219†	75
BM25+KALE	1024	0.309†	0.567†	0.630†	0.133	0.649†	0.206†	72
BM25+KALE	8192	0.306†	0.549†	0.582	0.130	0.626†	0.204†	27
BM25+KALE	32768	0.301†	0.539†	0.615†	0.126	0.627†	0.206†	24
BM25+KALE	65536	0.296†	0.532†	0.592	0.122	0.624†	0.205†	19
BM25+KALE	98304	0.296†	0.535†	0.628†	0.132	0.637†	0.214†	19
BM25+KALE	131072	0.297†	0.533†	0.613	0.112	0.624†	0.194	18
BM25+Teacher	768	0.293†	0.546†	0.651†	0.153	0.606†	0.212	84

The last block of the table answers the question of whether there is a necessity of learning the student LM. The block tested a setting where sparsification, indexing, and consequent search occurred directly over the dense outputs of the Teacher LM, bypassing any additional training. Directly sparsifying the teacher vectors resulted in a lower performance compared to the setting where a student LM is learned and a new vocabulary vector is generated through the projector. This is likely because the teacher LM vectors are real valued (compared to projected vectors from KALE, which go through a ReLU operation), which causes the TopK operation to ignore dimensions with a high absolute value, but negative weight.

The latencies behaved as expected. Every query was expanded with a fixed number of artificial terms, so latency depends on the posting list sizes of the added terms. Passages were also expanded with a constant number of KALE terms, meaning smaller vocabularies cause an increase in the average posting list size, therefore making search slower. Some experiments with larger vocabularies, in the KALE only setting, are more efficient than the original BM25 baseline. This may indicate that the KALE term posting list sizes are reasonably uniform, avoiding typical skews from the English vocabulary. When complementing KALE with BM25, latency increased since queries were made longer, considering terms from both the English and KALE vocabularies.

In summary, increasing the vocabulary size showed to be beneficial, specially in the KALE only scenario. In this setting, a larger vocabulary consistently resulted in efficiency and effectiveness gains, although accuracy flattened. Furthermore, the KALE terms in isolation were able to significantly outperform the BM25 baseline, both in efficiency and effectiveness. When augmenting the existing English vocabulary, larger vocabularies underperformed, possibly given the experimental decision of keeping a fixed number of expansion terms. Nonetheless, KALE showed to be specially effective together with the lexical vocabulary. Moreover, a significant improvement over BM25 was observed independently of the vocabulary size, which indicates that regardless of the nature of the concepts being captured, KALE is consistently able to improve on the lexical vocabulary. Throughout the other tests KALE was therefore considered together with the lexical terms.

5.2 Evaluating the Number of Expansion Terms

When considering different expansion terms in the query/document, a trade-off between efficiency and effectiveness is expected. A higher number of generated query/passage terms should improve the text representations, therefore increasing accuracy. However, adding new terms either causes the inverted lists to become longer (by increasing the number of passage terms), or forces retrieval and search over additional inverted lists (when adding query terms), hence hurting query latency.

This subsection reports tests varying the number of terms to expand the queries and passages, and Table 2 displays the experimental results. In general, an effectiveness stability is observed across the several vocabulary sizes, and the reported intuitions match the results. Regarding latency, an increase in expansion terms, whether in the query or the passages, did lead to an efficiency drop. Increasing the amount of query terms consistently lead

Table 2: Experimental results when changing the number of query and passage terms in the KALE representations. KALE terms were used together with the lexical terms in BM25. QL denotes MSMARCO query latency, measured in milliseconds.

			MSMARCO Dev		TREC DL 19		TREC DL 20		
|V|	k_{query}	$k_{passage}$	MRR@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	QL
1024	8	32	0.292	0.548	0.592	0.127	0.614	0.210	31
1024	8	64	0.279	0.538	0.603	0.138	0.622	0.208	36
1024	16	64	0.309	0.567	0.630	0.133	0.649	0.206	72
1024	16	128	0.288	0.555	0.631	0.143	0.606	0.194	86
32768	16	64	0.301	0.539	0.615	0.126	0.627	0.206	24
32768	16	128	0.307	0.554	0.607	0.134	0.655	0.205	27
32768	32	128	0.310	0.555	0.628	0.133	0.639	0.206	41
32768	32	256	0.316	0.566	0.633	0.135	0.657	0.211	52
98304	16	64	0.296	0.535	0.628	0.132	0.637	0.214	19
98304	16	128	0.303	0.542	0.625	0.136	0.656	0.216	19
98304	32	128	0.316	0.557	0.647	0.137	0.636	0.209	21
98304	32	256	0.319	0.564	0.646	0.142	0.639	0.210	26
131072	16	64	0.297	0.533	0.613	0.112	0.624	0.194	18
131072	16	128	0.302	0.548	0.627	0.124	0.644	0.204	20
131072	32	128	0.312	0.553	0.640	0.133	0.660	0.216	22
131072	32	256	0.318	0.564	0.657	0.131	0.673	0.218	25

to a higher accuracy, and increasing the number of passage terms also provided accuracy benefits, with the exception of a vocabulary of 1,024 terms. Leveraging smaller vocabularies and forcing every passage to be expanded with a large number of artificial terms likely introduces noise in the passage representations, since the terms are forced to exist across passages despite bearing no semantic similarity among themselves. Like in previous experiments, KALE showed robustness from an effectiveness perspective, regardless of vocabulary size, query terms, or passage terms.

Overall, the results from these tests confirmed our expectations regarding the trade-offs between accuracy and efficiency. More expansion terms improved effectiveness, with a query latency cost. Given the conclusions from this subsection and from the previous subsection, a set of hyperparameters was chosen for subsequent experiments, in an attempt to balance accuracy and efficiency. Specifically, the vocabulary size was kept at 98,304, k_{query} was set to 32, and $k_{passage}$ was set to 256. This configuration was kept throughout the remainder of the experiments.

5.3 Complementing Different Representations

Previous subsections demonstrated KALE to be compatible with the English lexical vocabulary, through the BM25 retriever. Other learned sparse representations operate over the same or similar vocabularies, with different approaches, e.g., performing lexical expansion, or re-weighting the existing lexical terms. If KALE terms indeed capture different information from that of existing lexical vocabularies, one would expect the KALE vocabulary to be compatible with other types of learned sparse representations.

The next experiments examined whether the terms generated by KALE could complement more advanced lexical representations. Several sparse retrievers, of increasing retrieval performance, were chosen to be augmented with the KALE generated vocabulary. Besides BM25, KALE terms were also tested with DeepCT [6], DocT5Query [28], DeepImpact [24], and EfficientSPLADE [17].

Table 3 presents these results. Each block compares the retrieval accuracy and query latency of the retriever alone (i.e., with the learned lexical vocabulary only), and the same retriever augmented with KALE terms. For every retriever, an effectiveness gain is visible, at a small latency cost. Even with EfficientSPLADE, which already relies heavily on re-weighting and expanding with lexical terms, a statistically significant MRR@10 boost was observed over MSMARCO. This reinforces the claim that the generated terms are able to capture concepts beyond the existing English vocabulary.

Overall, KALE terms were able to complement already strong learned sparse representations, providing accuracy boosts at relatively small efficiency costs. This supports the claim that KALE terms are able to capture semantic information in the corpus that existing sparse representations do not capture as accurately, or do not capture at all.

5.4 Assessing Posting List Size Distribution

Natural vocabularies are typically skewed, and KALE, following previous work such as EfficientSPLADE [17], employs a regularization term in the loss function to ensure a balanced index. Enforcing an equal distribution of document frequencies across all the artificial terms, and the consequent balancing of posting list size, is helpful from an efficiency perspective, avoiding the search of query terms with excessively high posting list sizes. The latencies reported in previous experiments did not hint at any serious unbalance in the posting list sizes, which may be the effect of the regularization term. Still, it is relevant to quantify the effect of regularization, and assess how the KALE vocabulary would be distributed without it.

In this subsection, experiments were conducted with a modified version of KALE, and posting list sizes for the generated vocabulary were plotted. Instead of leveraging both the M3SE loss and the equipartitioning loss, a setting was tested where KALE is trained solely with M3SE. After training KALE in this setting, posting list sizes were plotted, both with and without regularization.

Table 3: Experimental results when complementing other sparse retrievers with the KALE vocabulary. KALE terms were added with TF=1, with the exception of impact indexes. The lexical terms were either untouched (i.e., BM25), or reweighted/expanded by the other retrievers. QL denotes MSMARCO query latency, measured in milliseconds. The symbol † denotes statistically significant improvements over the base retrievers, for a paired t-test with a p-value of 0.05

	MSMARCO Dev		TREC DL 19		TREC DL 20		
Method	MRR@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	QL
BM25	0.184	0.379	0.506	0.129	0.480	0.164	17
BM25+KALE	0.319†	0.564†	0.646†	0.142	0.639†	0.210†	26
DeepCT	0.245	0.481	0.576	0.156	0.550	0.178	17
DeepCT+KALE	0.326†	0.590†	0.681†	0.166	0.650†	0.219	30
DocT5Query	0.274	0.539	0.629	0.159	0.611	0.218	21
DocT5Query+KALE	0.323†	0.574†	0.658	0.145	0.641	0.211	26
DeepImpact	0.326	0.582	0.662	0.152	0.602	0.198	55
DeepImpact+KALE	0.359†	0.625†	0.704†	0.160	0.667†	0.222	84
EfficientSPLADE	0.386	0.671	0.715	0.168	0.718	0.242	43
EfficientSPLADE+KALE	0.389†	0.667	0.720	0.168	0.713	0.239	89

Table 4: Ablation tests with the equipartitioning component. Index size is the disk size of the inverted index, measured in GB. QL denotes query latency, and is measured in ms/query.

Method	MRR@10	Index Size	QL
BM25	0.184	0.72	18
KALE	0.319	3.80	26
KALE w/o Equipartitioning	0.315	3.80	31

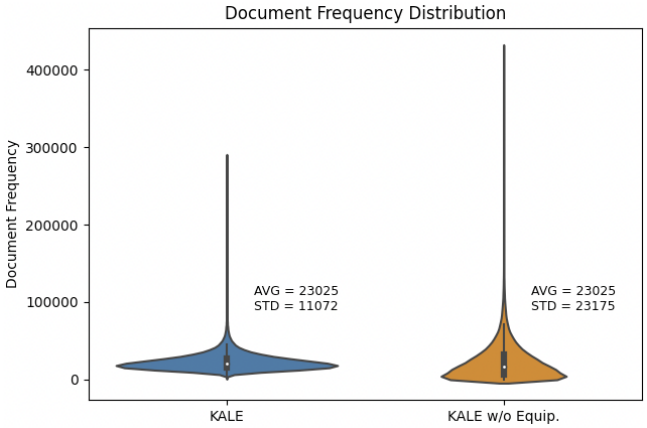

Figure 4: Document frequencies for KALE terms when enabling or disabling the equipartitioning component.

Table 4 compares KALE with and without equipartitioning, in terms of effectiveness, query latency, and index size. The results show a statistically significant effect (paired t-test with a p-value of 0.05) when removing equipartitioning. As expected, KALE terms increased the size of the index in disk, and removing equipartitioning did not influence index size (i.e., equipartitioning changes the size distribution within the same 98,304 posting lists, which combine to the same total size). Taking regularization out resulted in an effectiveness drop, together with an increase in query latency. This indicates that the regularization term was useful in enforcing a fairly balanced distribution of posting list sizes, which also contributed to an accuracy increase.

The left violin plot from Figure 4 illustrates the distribution of the term Document Frequency (DF) for the 98,304 KALE terms, while the right plot presents the same data, without equipartitioning. Ideally, all terms should have the DF obtained by dividing the number of documents in the collection with the KALE vocabulary size of 98,304. The average DF for both distributions matched this ideal DF. Removing equipartitioning increased the standard deviation of the DFs, which aligns with the previous expectation that regularization helped balance the DFs, and consequently, decrease search latency.

The previous experiments showed that the equipartitioning component was useful in balancing the posting list sizes of the generated KALE vocabulary, and therefore improve search efficiency. Not considering the regularization component led not only to worse MRR@10, but also to an increase in query latency.

5.5 Experiments with Out-of-Domain Data

This paper claims that the KALE vocabulary captures important concepts in its training corpus. This makes it seem unreasonable to expect the generated terms to retain performance when porting to unrelated domains, since the vocabulary is likely domain-specific. For example, a vocabulary term related to *gardening* may indeed be useful in MSMARCOv1, but still useless in a physics dataset.

Table 5 displays the results of combining KALE terms with BM25, as well as using KALE in isolation, over the BEIR benchmark. KALE performed poorly when evaluated zero-shot on different domains, particularly on the semantic relatedness tasks. Unlike in previous experiments, KALE struggled to outperform BM25, but still mostly outperformed DeepCT, when combined with the English terms. The search tasks are closer in nature with the MSMARCOv1 passage ranking task, and KALE improved over BM25 in one of these datasets. That corpus corresponds to a Question Answering dataset similar to MSMARCOv1, which aligns with the intuitions.

In order to further assess the hypothesis of whether the terms are specific to MSMARCOv1 or not, five random dimensions were sampled, and the training queries were sorted based on their weights for those specific dimensions. The top 20 queries were selected based on the aforementioned weights. Table 6 displays the dimensions, the five lexical terms with a higher frequency on these top queries, and

Table 5: Experimental results on the public BEIR datasets. Results are measured with NDCG@10. Boldface denotes the best scores, underline denotes the second best, and italics denotes cases where KALE improves over BM25.

	BM25	DeepCT	DocT5Query	EfficientSPLADE	KALE+BM25	KALE only
Search Tasks						
DBPedia	0.313	0.177	0.331	**0.405**	0.287	0.245
FIQA	0.236	0.191	0.291	**0.318**	0.221	0.169
HotpotQA	0.603	0.503	0.581	**0.666**	0.498	0.318
NFCorpus	0.325	0.283	0.328	**0.331**	0.258	0.217
NQ	0.329	0.188	0.399	**0.515**	*0.363*	0.318
TREC-COVID	0.656	0.406	**0.713**	0.661	0.516	0.423
Semantic Relatedness Tasks						
Arguana	0.315	0.309	0.349	**0.473**	0.308	0.256
Climate-FEVER	**0.213**	0.066	0.201	0.189	0.192	0.118
FEVER	**0.753**	0.353	0.714	0.749	0.637	0.509
Scidocs	0.158	0.124	**0.162**	0.153	0.112	0.069
Scifacts	0.665	0.631	**0.675**	0.674	0.573	0.305
Touche2020	**0.367**	0.156	0.347	0.270	0.208	0.204

Table 6: Interpretations of 5 different KALE terms. For each dimension, the top 20 queries with the highest weight for that dimension were inspected, and the table reports the top 5 terms with the highest frequencies in those queries.

Dim.	Top 5 terms	Interp.
871	*click*, *iphone*, *itunes*, *open*, *connect*	iphone
3862	*first*, *wedding*, *love*, *day*, *song*	wedding songs
14609	*business*, *cost*, *market*, *make*, *process*	business
29305	*home*, *windows*, *gas*, *oven*, *kitchen*	kitchen
31376	*heart*, *info.*, *symptoms*, *online*, *test*	heart concern

the interpretation of the concept being captured by the corresponding vocabulary term. This experiment supports the expectations on the domain-specific nature of the vocabulary generated by KALE.

In summary, as expected, the generated vocabulary is domain-specific, which is the likely cause of the limited ability of KALE to generalize out-of-domain. Semantic relatedness tasks are fairly different than the retrieval domain KALE was trained on. For search tasks, KALE still showed some improvements, specifically in a Question Answering dataset similar to MSMARCOv1.

6 LIMITATIONS

In spite of the positive results, some limitations should also be highlighted. First, the experimental results seemed to plateau, and larger vocabularies did not lead to large improvements. A potential reason is the fact that KALE applies no term weighting, augmenting the lexical text with new artificial terms and using a constant term frequency. This setting is not novel in the IR community, as classical expansions with controlled vocabularies [16, 35] often employ no term weighting. Experiments with different term frequencies offered marginal improvements over the TF=1 setting in KALE, but this research direction was not extensively pursued.

The teacher model distilled into KALE is also far from being the strongest available dense retrieval model. We did some initial experiments trying to distill knowledge from stronger dense models, but results fell short compared to the reported teacher model. Previous work [22] pointed out how knowledge distillation from strong models is not entirely straightforward - the authors proposed to gradually improve the quality of the teachers and the difficulty of the examples as training progresses. Initial experiments leveraging two teachers instead of one resulted in a very marginal improvement, corroborating the aforementioned previous work.

7 CONCLUSIONS AND FUTURE WORK

This paper presented KALE, a simple model to generate sparse representations from dense vectors, via a simple projector applied on top of the output from a DistilBERT language model, followed by a TopK operation. Sparse representations for an input text are generated over a new vocabulary space, and these representations are then converted to artificial terms, which can be stored in inverted indexes and searched efficiently. KALE was trained and evaluated on MSMARCOv1. We encoded the MSMARCOv1 corpus with KALE, hence generating new terms for every passage in the collection, and then indexed the resulting augmented passages. At query time, KALE generates new query terms, to be searched in conjunction with the existing lexical terms.

Experimental results demonstrate that the proposed vocabulary, on its own, outperforms the lexical English vocabulary with a BM25 baseline. When combined with existing sparse methods, further performance increases are observed across different sparse retrievers, with a small penalty in query latency. KALE can be highly relevant when resources are scarce, since the method is simple and cheap to use, and incurs in only small latency costs. Hopefully, this work can motivate additional research on learned sparse representations. Several questions are left open (e.g., the benefits of general terms against more specific terms, or different term weighting schemes), and future work can attempt to address them.

ACKNOWLEDGMENTS

This research was supported by the Portuguese Recovery and Resilience Plan through project C645008882-00000055, through Fundação para a Ciência e Tecnologia (FCT) with the Ph.D. scholarship SFRH/BD/150497/2019 under the CMU-PT Program, and through the INESC-ID multi-annual funding from the PIDDAC programme, corresponding to reference UIDB/50021/2020.

REFERENCES

[1] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation at the Annual Conference on Neural Information Processing Systems*.

[2] Donghyun Choi, Myeongcheol Shin, Eunggyun Kim, and Dong Ryeol Shin. 2021. Adaptive batch scheduling for open-domain question answering. *IEEE Access* 9 (2021).

[3] Eunseong Choi, Sunkyung Lee, Minijn Choi, Hyeseon Ko, Young-In Song, and Jongwuk Lee. 2022. SpaDE: Improving sparse representations using a dual document encoder for first-stage retrieval. In *Proceedings of the ACM International Conference on Information & Knowledge Management*.

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

[5] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2021. Overview of the TREC 2020 deep learning track. *arXiv preprint arXiv:2102.07662* (2021).

[6] Zhuyun Dai and Jamie Callan. 2020. Context-aware term weighting for first stage passage retrieval. In *Proceedings of the International ACM SIGIR conference on research and development in Information Retrieval*.

[7] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[8] Luyu Gao and Jamie Callan. 2021. Condenser: a Pre-training Architecture for Dense Retrieval. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

[9] Luyu Gao and Jamie Callan. 2022. Unsupervised Corpus Aware Language Model Pre-training for Dense Passage Retrieval. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

[10] Mitko Gospodinov, Sean MacAvaney, and Craig Macdonald. 2023. Doc2Query: When Less is More. In *Proceedings of the European Conference on Information Retrieval*.

[11] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving efficient neural ranking models with cross-architecture knowledge distillation. *arXiv preprint arXiv:2010.02666* (2020).

[12] Kyoung-Rok Jang, Junmo Kang, Giwon Hong, Sung-Hyon Myaeng, Joohee Park, Taewon Yoon, and Heecheol Seo. 2021. Ultra-High Dimensional Sparse Representations with Binarization for Efficient Text Retrieval. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

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

[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 Conference on Empirical Methods in Natural Language Processing*.

[15] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the International ACM SIGIR conference on research and development in Information Retrieval*.

[16] Margaret EI Kipp. 2011. Controlled vocabularies and tags: An analysis of research methods. *NASKO* 3 (2011), 23–32.

[17] Carlos Lassance and Stéphane Clinchant. 2022. An efficiency study for SPLADE models. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[18] Carlos Lassance, Thibault Formal, and Stéphane Clinchant. 2021. Composite code sparse autoencoders for first stage retrieval. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[19] 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.

[20] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2020. Distilling dense representations for ranking using tightly-coupled teachers. *arXiv preprint arXiv:2010.11386* (2020).

[21] 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 Workshop on Representation Learning for NLP*.

[22] Zhenghao Lin, Yeyun Gong, Xiao Liu, Hang Zhang, Chen Lin, Anlei Dong, Jian Jiao, Jingwen Lu, Daxin Jiang, Rangan Majumder, et al. 2023. PROD: Progressive Distillation for Dense Retrieval. In *Proceedings of the International World Wide Web Conference*.

[23] Zheng Liu and Yingxia Shao. 2022. RetroMAE: Pre-training retrieval-oriented transformers via masked auto-encoder. *arXiv preprint arXiv:2205.12035* (2022).

[24] Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. 2021. Learning passage impacts for inverted indexes. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[25] Aditya Menon, Sadeep Jayasumana, Ankit Singh Rawat, Seungyeon Kim, Sashank Reddi, and Sanjiv Kumar. 2022. In defense of dual-encoders for neural ranking. In *Proceedings of the International Conference on Machine Learning*.

[26] Thong Nguyen, Sean MacAvaney, and Andrew Yates. 2023. A Unified Framework for Learned Sparse Retrieval. In *Proceedings of the European Conference on Information Retrieval*.

[27] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y Zhao, Yi Luan, Keith B Hall, Ming-Wei Chang, et al. 2021. Large dual encoders are generalizable retrievers. *arXiv preprint arXiv:2112.07899* (2021).

[28] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to docTTTTTquery. *Online preprint* 6 (2019).

[29] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375* (2019).

[30] 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 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

[31] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*.

[32] Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. RocketQAv2: A Joint Training Method for Dense Passage Retrieval and Passage Re-ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

[33] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009).

[34] Tao Shen, Xiubo Geng, Chongyang Tao, Can Xu, Xiaolong Huang, Binxing Jiao, Linjun Yang, and Daxin Jiang. 2022. LexMAE: Lexicon-Bottlenecked Pretraining for Large-Scale Retrieval. *arXiv preprint arXiv:2208.14754* (2022).

[35] Barry Smith and Anand Kumar. 2004. Controlled vocabularies in bioinformatics: a case study in the gene ontology. *Drug Discovery Today: BIOSILICO* 2, 6 (2004), 246–252.

[36] 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 *Proceedings of the Conference on Neural Information Processing Systems: Datasets and Benchmarks Track*.

[37] Nicola Tonellotto, Craig Macdonald, Iadh Ounis, et al. 2018. Efficient query processing for scalable web search. *Foundations and Trends® in Information Retrieval* 12, 4-5 (2018), 319–500.

[38] Shitao Xiao and Zheng Liu. 2022. RetroMAE v2: Duplex Masked Auto-Encoder For Pre-Training Retrieval-Oriented Language Models. *arXiv preprint arXiv:2211.08769* (2022).

[39] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *Proceedings of the International Conference on Learning Representations*.

[40] Hamed Zamani, Mostafa Dehghani, W Bruce Croft, Erik Learned-Miller, and Jaap Kamps. 2018. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the ACM International Conference on Information and Knowledge Management*.

[41] Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. 2022. Dense text retrieval based on pretrained language models: A survey. *arXiv preprint arXiv:2211.14876* (2022).

[42] Kun Zhou, Xiao Liu, Yeyun Gong, Wayne Xin Zhao, Daxin Jiang, Nan Duan, and Ji-Rong Wen. 2022. MASTER: Multi-task Pre-trained Bottlenecked Masked Autoencoders are Better Dense Retrievers. *arXiv preprint arXiv:2212.07841* (2022).

[43] Justin Zobel and Alistair Moffat. 2006. Inverted files for text search engines. *ACM computing surveys (CSUR)* 38, 2 (2006), 6–es.