LoSparse: Low-dimensional Sparse Retrieval Model

Kohei Uno[1], Shogo Kisa[1] and Yohei Iseki[1]

[1]LY Corporation, Tokyo, Japan

Abstract

Two-stage retrieval has become a standard paradigm in Neural Information Retrieval, where an initial retrieval stage efficiently selects candidate documents, followed by a more sophisticated re-ranking model. While dense retrieval methods leveraging pre-trained models have achieved strong performance, sparse retrieval has recently attracted attention due to its advantages in exact lexical matching, indexing efficiency, and interpretability. However, existing sparse retrieval models often generate extremely high-dimensional representations, as their embedding dimensionality typically corresponds to the tokenizer's vocabulary size. This results in substantial computational and memory overhead during both training and inference. Moreover, such high-dimensional representations are difficult to integrate with partial matching techniques, such as the T-overlap query. In this work, we propose LoSparse: Low-dimensional Sparse Retrieval Model, which improves efficiency through low-dimensional sparse representations while enabling flexible retrieval control using the T-overlap query.

Keywords

Sparse representations, Embedding, Indexing, E-commerce

1. Introduction

Fine-tuning large pre-trained language models such as BERT [1] and large language models (LLMs) [2] has achieved remarkable performance across a wide range of natural language processing (NLP) tasks. In Information Retrieval, neural ranking models based on pre-trained encoders [7, 8] have become widely adopted. Text retrieval for natural language queries typically follows a two-stage retrieval paradigm, where a large document collection is first searched efficiently, and the retrieved candidates are subsequently re-ranked by a more sophisticated model.

In the first-stage retrieval, bag-of-words (BoW) approaches such as TF–IDF and BM25 remain widely used. Although BoW models are fast and interpretable, they rely on exact lexical matching, which often leads to query–document mismatch and recall loss when there is no lexical overlap between the query and the document. To address this limitation, vector-based semantic retrieval methods leveraging Word2Vec [3], Transformer architectures [4], and contextual embeddings have been proposed to capture semantic similarity beyond surface-level word overlap.

Vector-based retrieval methods can be broadly categorized into dense retrieval and sparse retrieval. Both involve two main steps: (1) generating embeddings for documents and queries, and (2) retrieving candidates based on similarity between these embeddings. Dense retrieval represents queries and documents as low-dimensional dense vectors derived from pre-trained models [6]. Similarity is typically computed in the embedding space, and for efficiency, Approximate Nearest Neighbor (ANN) search techniques such as IVF, HNSW, and PQ [9, 10, 11] are commonly used. However, compared to exhaustive brute-force retrieval, ANN methods can incur noticeable degradation in accuracy.

In contrast, sparse retrieval encodes queries and documents as high-dimensional sparse vectors. Models such as SPLADE [24, 25, 26] generate sparse representations whose dimensionality corresponds to the tokenizer's vocabulary size, preserving the interpretability and lexical alignment of BoW models while introducing semantic expansion through deep learning. These methods enable efficient AND/OR-style retrieval using inverted indexes, maintaining both high speed and interpretability.

Despite these advantages, existing sparse retrieval models face the challenge of extremely high-dimensional embeddings, often on the order of tens or hundreds of thousands of dimensions. Such representations impose substantial computational and memory overhead during both training and

ECOM'26: SIGIR Workshop on eCommerce, Jul 24, 2026, Melbourne, Australia

kohuno@lycorp.co.jp (K. Uno); shokisa@lycorp.co.jp (S. Kisa); yiseki@lycorp.co.jp (Y. Iseki)

inference, and make it difficult to integrate with partial matching schemes such as T-overlap query [12, 13, 14]. The T-overlap query retrieves documents that satisfy at least k out of n conditions, enabling flexible partial matching even when not all query terms appear in a document. Many commercial full-text search engines implement this mechanism as Minimum Should Match (MSM) search[1].

In this paper, we propose LoSparse: Low-dimensional Sparse Retrieval Model, which achieves efficient and accurate retrieval using low-dimensional sparse embeddings in conjunction with the T-overlap query. We introduce a Binary Loss [28, 29, 30] alongside the standard ranking loss used in contrastive learning. This encourages the vector components to become discretized, improving compatibility with the partial-matching mechanism of T-overlap query. Furthermore, LoSparse can be trained from scratch, without relying on large pre-trained language models. Unlike traditional sparse retrieval approaches, LoSparse's embedding dimensions do not correspond to specific vocabulary tokens, resulting in lower lexical interpretability. Nevertheless, by leveraging low-dimensional sparse representations, LoSparse significantly improves computational efficiency and memory utilization, making it a practical and scalable alternative for real-world retrieval systems.

2. Related Work

Classical bag-of-words (BoW) approaches, such as TF–IDF and BM25, remain widely used in retrieval systems due to their computational efficiency. However, these methods rely solely on exact lexical matching, which limits their ability to capture semantic similarity between queries and documents. To address this limitation, several approaches have been proposed to combine the interpretability and efficiency of BoW models with the semantic representation power of neural models.

One line of research focuses on learning context-dependent term importance. DeepCT (Deep Contextualized Term Weighting) [16] learns contextualized term weights using BERT, assigning importance scores to words based on their surrounding context. This allows frequent but less informative words to receive lower weights, while contextually salient terms are assigned higher scores, resulting in more accurate relevance estimation compared to BM25. However, DeepCT can only assign weights to terms that already appear in a document and cannot introduce new vocabulary terms. Consequently, it fails to match documents containing semantically related but lexically different expressions.

To overcome this limitation, vocabulary expansion approaches have been proposed, where the model predicts additional semantically related terms not originally present in the text and incorporates them into sparse vector representations. A representative example is Doc2Query [20], which generates likely query terms for each document using a pre-trained language model and appends them to the document text. DeeperImpact [17] extends this idea by assigning BERT-based importance scores to the generated terms and indexing them together with the original terms, thereby improving recall over traditional BoW models. EPIC [22] propagates learned term importance to semantically similar words, whereas TILDE (Term Independent Likelihood moDEl) [21] independently estimates the probability of each term being relevant to a given query using BERT-based scoring.

Another major research direction involves learning high-dimensional sparse embeddings in the vocabulary space. Models such as SparTerm [18] and SPLADE (Sparse Lexical and Expansion Model for First-Stage Ranking) [24, 25, 26] encode queries and documents into sparse lexical vectors using Transformer-based architectures. These models assign activation weights to each token, enabling the generation of weighted BoW representations that include semantically expanded terms not explicitly present in the original text. Such methods retain the interpretability of BoW approaches while achieving strong performance in semantic matching.

Our proposed method, LoSparse, also learns sparse vector representations for queries and documents. In contrast to existing models that depend on extremely high-dimensional vocabulary-based embeddings, LoSparse employs compact low-dimensional sparse representations, improving computational efficiency and scalability while maintaining retrieval effectiveness. By emphasizing model compactness and

[1]Lucene-based systems, such as Apache Solr and Elasticsearch, commonly refer to this approach as Minimum Should Match (MSM) search.

supporting flexible matching through T-overlap query, LoSparse offers a lightweight yet effective alternative to existing sparse retrieval frameworks.

3. Model

In this section, we briefly review SPLADE, which is closely related to our approach. We then introduce the proposed method, LoSparse: Low-dimensional Sparse Retrieval Model, and explain how sparse vector retrieval is performed using the T-overlap query.

3.1. SPLADE Model

We begin by describing the SPLADE (Sparse Lexical and Expansion Model for First-Stage Ranking) [24, 25, 26], a representative model in sparse retrieval.

Model Architecture. In SPLADE, sparse vector representations are computed as term importance scores using the Masked Language Modeling (MLM) head of BERT. Let the vocabulary size be denoted as $|V|$, and the tokenized input sequence as $t = (t_1, t_2, \dots, t_T)$. The token-level embeddings obtained from BERT are represented by $h = (h_1, h_2, \dots, h_T)$. For the i-th input token, the importance score corresponding to the j-th vocabulary term is defined as:

$$w_{ij} = \text{transform}(h_i)^\top E_j + b_j, \quad j \in \{1, \dots, |V|\}, \tag{1}$$

where E_j denotes the BERT input embedding of the j-th vocabulary term, b_j is the bias term, and transform$(\cdot)$ is a linear layer with layer normalization and the GeLU activation function. The computation of w_{ij} corresponds to that of the MLM head in BERT and can be expressed as:

$$\text{transform}(x) = \text{LN}(\text{GeLU}(x^\top W_{\text{linear}} + b_{\text{linear}})). \tag{2}$$

Finally, the overall importance score w_j for the j-th vocabulary term is obtained by applying max pooling over all input tokens as follows:

$$w_j = \max_{1 \le i \le T} \log\left(1 + \text{ReLU}(w_{ij})\right). \tag{3}$$

Loss. Let $s(q, d)$ denote the similarity score computed as the dot product between the sparse vector representations of a query q and a document d. For a query q_i, let d_i^+ denote a positive document, d_i^- a negative document, and d_{ij}^- in-batch negatives. The ranking loss $\mathscr{L}_{\text{rank}}$, based on contrastive learning, is defined as:

$$\mathscr{L}_{\text{rank}} = -\log \frac{e^{s(q_i, d_i^+)}}{e^{s(q_i, d_i^+)} + e^{s(q_i, d_i^-)} + \sum_j e^{s(q_i, d_{ij}^-)}} \tag{4}$$

To encourage sparsity, SPLADE employs the FLOPS regularization loss. Given a batch of N documents d_i and their predicted activation values w_j, let $\bar{a}_j = \frac{1}{N} \sum_{i=1}^{N} w_j^{(d_i)}$ denote the mean activation for the j-th vocabulary term. The FLOPS loss is then defined as:

$$\mathscr{L}_{\text{FLOPS}} = \sum_{j \in V} \bar{a}_j^2 = \sum_{j \in V} \left(\frac{1}{N} \sum_{i=1}^{N} w_j^{(d_i)} \right)^2 \tag{5}$$

It is well known that linguistic data follow a Zipfian distribution. When L_1 regularization is used to induce sparsity, this frequency imbalance often biases the learned representations toward frequent tokens. In contrast, FLOPS regularization encourages a more balanced and distributed sparse representation by mitigating over-reliance on high-frequency terms. Finally, the overall training objective is defined as a weighted sum of the ranking loss and the FLOPS regularization for queries and documents. Let λ_q and λ_d denote the regularization coefficients for queries and documents, respectively, and let $\mathscr{L}_{\text{FLOPS}}^q$ and $\mathscr{L}_{\text{FLOPS}}^d$ represent their corresponding FLOPS losses. The total loss is formulated as:

$$\mathscr{L} = \mathscr{L}_{\text{rank}} + \lambda_q \mathscr{L}_{\text{FLOPS}}^q + \lambda_d \mathscr{L}_{\text{FLOPS}}^d. \tag{6}$$

3.2. LoSparse: Low-dimensional Sparse Retrieval Model

This section describes the proposed LoSparse: Low-dimensional Sparse Retrieval Model.

Model Architecture. In SPLADE, importance scores for each token are computed based on BERT's Masked Language Modeling (MLM) head, resulting in an embedding dimensionality equal to the vocabulary size $|V|$. In contrast, the proposed method generates embeddings of arbitrary dimensionality D, allowing for a more compact representation. Given a tokenized input sequence $t = (t_1, t_2, \dots, t_T)$, let the token-level embeddings obtained from BERT be $h = (h_1, h_2, \dots, h_T)$. Instead of computing the importance score w_{ij} for the j-th vocabulary term in BERT corresponding to the i-th token, we introduce D pseudo-tokens. For the i-th input token, the importance score of the j-th pseudo-token ($j \in \{1, \dots, D\}$) is computed as:

$$\tilde{w}_{ij} = h_i^\top \tilde{E}_j + \tilde{b}_j, \tag{7}$$

where $\tilde{E}_j$ denotes the embedding of the j-th pseudo-token, and $\tilde{b}_j$ is its bias term. The overall importance score for the j-th pseudo-token is then obtained via max pooling across all tokens:

$$\tilde{w}_j = \max_{1 \le i \le T} \log\left(1 + \mathrm{ReLU}(\tilde{w}_{ij})\right). \tag{8}$$

To control the sparsity of the embedding representation, a Top-K filtering operation can be applied to retain only the K largest activation values:

$$\tilde{w}^K = \mathrm{TopK}(\tilde{w}), \tag{9}$$

where $\tilde{w}^K$ represents the sparse vector after filtering. This operation helps improve retrieval efficiency and allows dynamic adjustment of the number of retrieved documents during search based on the sparsity level of the representation. An overview of the proposed model architecture is shown in Figure 1. LoSparse adopts a two-tower architecture, consisting of a query tower and a document tower that are encoded independently but share the same embedding space.

Ranking Loss. As the ranking loss $\mathscr{L}_{\mathrm{rank}}$, we adopt CircleLoss [27], which has demonstrated strong performance across a wide range of contrastive learning tasks. Negative samples are generated using in-batch negative (IBN) sampling, a widely used strategy in contrastive learning frameworks that has been empirically shown to yield high retrieval performance.

$$\mathscr{L}_{\mathrm{rank}} = \log\Big[1 + \sum_{j=1}^{L} \exp(\gamma \alpha_n^j (s_n^j - \Delta_n)) \sum_{i=1}^{K} \exp(-\gamma \alpha_p^i (s_p^i - \Delta_p))\Big] \tag{10}$$

$$\begin{cases} \alpha_p^i = [O_p - s_p^i]_+ \\ \alpha_n^i = [s_n^i - O_n]_+ \end{cases} \tag{11}$$

Let $s_p^i = (x_p^i)^T x / \|x_p^i\| \|x\|$ and $s_n^j = (x_n^j)^T x / \|x_n^j\| \|x\|$ denote the cosine similarities between an anchor x and the i-th positive example $x_p^i \in \mathscr{P}$, and the j-th negative example $x_n^j \in \mathscr{N}$, respectively. The numbers of positive and negative samples are denoted by $K = |\mathscr{P}|$ and $L = |\mathscr{N}|$, respectively. γ is a scaling factor, and Δ_n, Δ_p denote the decision margins. The operator $[\cdot]_+$ truncates negative values to zero. O_p and O_n represent the optimal similarity values for positive and negative samples, respectively, ensuring that α_p^i and α_n^i remain non-negative.

Binary Loss. In our retrieval framework, we employ the T-overlap query. The T-overlap query determines whether a query matches a document based on whether the number of overlapping nonzero indices between their sparse vectors exceeds a predefined threshold T. Since this retrieval mechanism only considers whether each index is nonzero or not, the matching process is inherently discrete. However, when the model is trained solely with the ranking loss, the optimization objective emphasizes

the real-valued magnitudes of sparse embeddings. This leads to a mismatch between the continuous optimization objective during training and the discrete retrieval behavior during inference with the T-overlap query. To bridge this gap, we introduce a Binary Loss based on the Hamming distance between binarized embeddings. Given a D-dimensional sparse vector $\tilde{w}$ produced by the model, we define a hash layer [28, 29, 30] that outputs a binarized vector $z \in \{0,1\}^D$:

$$z = \text{sign}(\tilde{w}) \tag{12}$$

where $\text{sign}(\cdot)$ is defined as $\text{sign}(z_i) = 1\ (\tilde{w}_i > 0),\ \text{sign}(z_i) = 0\ (\tilde{w}_i \leq 0),\ i = 1, \ldots, D$. Because the sign function yields zero gradients during back-propagation, direct optimization is infeasible. To address this issue, we use the tanh function to progressively approximate binarization during training:

$$\tilde{z} = \tanh(\beta \tilde{w}), \tag{13}$$

where β is a scaling coefficient computed as $\beta = \sqrt{\gamma \cdot \text{epoch} + 1}$, with γ being a hyperparameter. As training progresses, β increases, making the tanh function closer to the sign function; in the limit $\beta \to \infty$, the transformation becomes equivalent to the sign operation. We define $\mathscr{L}_{\text{cand}}$ as the Binary Loss, computed based on the Hamming distance as follows:

$$\mathscr{L}_{\text{cand}} = \sum_{j=1}^{n} \max\left(0, -\left(s(\tilde{z}_{q_i}, \tilde{z}_{d_i^+}) - s(\tilde{z}_{q_i}, \tilde{z}_{d_{ij}^-})\right) + \alpha\right) \tag{14}$$

Here, for a query q_i, d_i^+ denotes the positive document and d_{ij}^- represents the j-th negative document among n in-batch negatives. The parameter α specifies the margin. The distance between two binary vectors z_i and z_j is computed as $\text{dist}_{\text{binary}}(z_i, z_j) = \text{const.} - s(z_i, z_j)$, where $s(z_i, z_j)$ represents their dot-product similarity.

Regularization Loss. To promote sparsity in the learned vector representations, we use L_1 regularization, defined as $\mathscr{L}_{\text{reg}} = |x|_1$. In the proposed method, the dimensionality of the sparse vectors D is relatively small. Moreover, these D dimensions correspond to pseudo-tokens rather than actual vocabulary items, making the model less susceptible to the frequency bias commonly observed in natural language data that follows a Zipfian distribution. While SPLADE employs a FLOPS loss to address such frequency bias across vocabulary terms, our model inherently avoids this issue due to the use of pseudo-tokens. Therefore, a simple L_1 regularization term is sufficient to produce effective and stable sparse representations in LoSparse.

Overall Loss. The overall training objective is defined as a linear combination of three components: the ranking loss $\mathscr{L}_{\text{rank}}$, the binary loss $\mathscr{L}_{\text{cand}}$, and the regularization loss $\mathscr{L}_{\text{reg}}$. Let λ_1 denote the weighting coefficient for the binary loss, and λ_2^q and λ_2^d represent the regularization coefficients for the query and document vectors, respectively. $\mathscr{L}_{\text{reg}}^q$ and $\mathscr{L}_{\text{reg}}^d$ denote the regularization losses for the query and document representations.

$$\mathscr{L} = \mathscr{L}_{\text{rank}} + \lambda_1 \mathscr{L}_{\text{cand}} + \lambda_2^q \mathscr{L}_{\text{reg}}^q + \lambda_2^d \mathscr{L}_{\text{reg}}^d \tag{15}$$

3.3. Candidate Generation

T-overlap Query. We describe the T-overlap query, which serves as the retrieval mechanism for the obtained sparse vector representations. Given a binary representation of N bits, $b_1, \ldots, b_N$, the T-threshold function $\vartheta(T, \{b_1, \ldots, b_N\})$ returns True if at least T bits are equal to 1, and False otherwise:

$$\vartheta(T, \{b_1, \ldots, b_N\}) = \begin{cases} \text{True}, & \text{if at least T bits are 1} \\ \text{False}, & \text{otherwise} \end{cases} \tag{16}$$

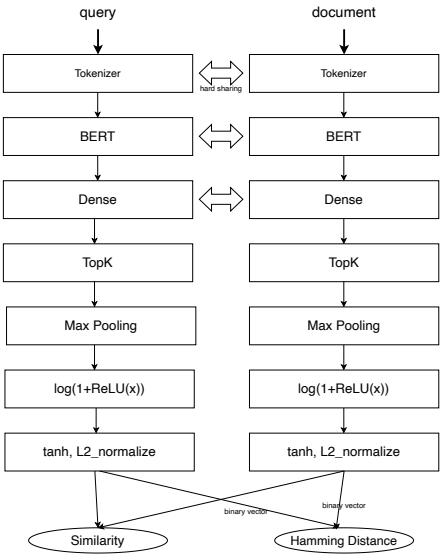

Figure 1: Model Architecture LoSparse

When $T = 1$, this function is equivalent to a logical OR operation ($b_1 \vee b_2 \vee ... \vee b_N$), and when $T = N$, it corresponds to a logical AND operation ($b_1 \wedge b_2 \wedge ... \wedge b_N$). Here, $\wedge$ and $\vee$ denote logical AND and logical OR, respectively. For cases where $2 \leq T \leq N - 1$, this condition is commonly referred to as the T-overlap, T-occurrence, or T-threshold query [12, 13, 14]. In this paper, we refer to it simply as the T-overlap query.

Candidate Generation. Let the sparse vector representations of a query and a document be denoted by $q \in \mathbb{R}^D$ and $d \in \mathbb{R}^D$, respectively, where D is the dimensionality of the sparse vectors. The binary indicators corresponding to the nonzero indices of these vectors are defined as $z^{(q)} = \text{sign}(q)$ and $z^{(d)} = \text{sign}(d)$, where $z^{(q)}, z^{(d)} \in \{0, 1\}^D$. The logical AND operation between the two binary vectors is expressed as $z^{(q \wedge d)} = z^{(q)} \wedge z^{(d)}$, where $z^{(q \wedge d)}$ is a D-bit binary vector whose i-th bit is 1 if the corresponding dimension is nonzero in both the query and the document vectors, and 0 otherwise. The retrieval condition is specified by the function $\vartheta(T, z^{(q \wedge d)})$, which considers a document relevant if the number of overlapping nonzero indices between the query and the document is at least T. By adjusting the parameter T, the number of retrieved candidates can be flexibly controlled. When either the query or document vectors contain a large number of nonzero indices, the number of matched documents tends to increase, which can impose additional computational load on the retrieval engine. To address this issue, we optionally limit the number of nonzero indices by retaining only the Top-K elements with the largest activation values. Specifically, for the query and document sparse vectors q and d, we retain the top $K^{(q)}$ and $K^{(d)}$ elements, respectively, prior to performing retrieval.

4. Experiment

We conducted two experiments to verify the effectiveness of the proposed method: (1) a numerical experiment using the publicly available Amazon ESCI dataset [31], and (2) an online A/B test on Yahoo! JAPAN Shopping[2].

[2]https://shopping.yahoo.co.jp/

4.1. Open Dataset Experiment

4.1.1. Dataset

We performed numerical experiments using the publicly available Amazon ESCI dataset [31]. The Amazon ESCI dataset is a large-scale shopping query dataset constructed from Amazon's search queries and search results. It contains approximately 130,000 unique queries and 2.6 million manually labeled query–product pairs, covering English, Japanese, and Spanish. It thus serves as a multilingual benchmark for evaluating search accuracy and relevance classification.

We utilized the subset of the Amazon ESCI dataset consisting of English-language query–document pairs labeled as Exact (E), Substitute (S), or Complement (C). A total of 1,147,326 pairs were used for training and 126,270 pairs for validation. For model input, the query column in the dataset was used as the query input, and the product_title column was used as the document input. During evaluation, retrieval was performed over a document collection containing 1,215,854 product titles. For testing, 5,000 queries were randomly sampled from the 22,458 test queries provided in the dataset. Retrieval performance was evaluated by comparing model accuracy under this query–document retrieval setting.

4.1.2. Training

We trained three types of models: (1) a dense retrieval model, (2) the SPLADE model, and (3) the proposed LoSparse model. All models were implemented using PyTorch [32] and Hugging Face Transformers [33], and trained on an NVIDIA A100 GPU (80 GB) environment. For both the dense retrieval model and LoSparse, we conducted two training setups: (1) training from scratch using an LSTM-based model (7.3M parameters), and (2) fine-tuning a BERT-based model. SPLADE was trained by fine-tuning the same BERT-based model for comparison. The base model used in all experiments was google-bert/bert-base-uncased [35], which has 110M parameters. The embedding dimensionality was set to 256 for the dense retrieval model, $D = 1024$ for LoSparse, and 30,522 for SPLADE, corresponding to the vocabulary size of the underlying BERT tokenizer. The training hyperparameters for LoSparse are listed in Table 1. For the query-side sparse vectors, we applied a Top-K constraint with $K^{(q)} = 10$, while no such constraint was applied to the document-side vectors ($K^{(d)} = D$). Due to the query-side Top-K constraint, the L_1 regularization weight was set to $\lambda_2^q = 0.0$, and L_1 regularization was applied only to the document-side sparse vectors. Several hyperparameters were tuned through parameter search, and the model used for evaluation corresponds to the best-performing configuration obtained from this search.

Table 1
Training parameters for the LoSparse model (LSTM-based / BERT-based)

Parameter	LSTM-based	BERT-based
Model architecture	1-layer BiLSTM	BERT-based
Total parameters	7.3M	110M
Optimizer	Adam	AdamW
Learning rate	0.005	0.0001
Batch size / Num negatives	2048 / 2048	256 / 256
Binary loss weight	λ_1=[0.01, 0.05, 0.08]	λ_1=[0.05, 0.08, 0.10, 0.12]
Gradient clipping	2.0	2.0
Warm-up epochs	4	4
Top-K constraint (query)	$K^{(q)} = 10$	$K^{(q)} = 10$
Top-K constraint (document)	$K^{(d)} = D$ (no constraint)	$K^{(d)} = D$ (no constraint)
L1 regularization (query side)	$\lambda_2^q = 0.0$	$\lambda_2^q = 0.0$
L1 regularization (document side)	$\lambda_2^d = [0.01, 0.05, 0.08]$	$\lambda_2^d = [0.003, 0.005, 0.008]$
Embedding dimension	$D = 1024$	$D = 1024$

4.1.3. Retrieval

This section describes the retrieval methods used in the evaluation. For dense vector retrieval, we evaluated two approaches: (1) Brute-force search over the entire document set, and (2) Cluster brute-force search after performing k-means clustering on the dense vectors. For sparse vector retrieval, we performed searches using the T-overlap query based on the nonzero indices obtained from the query and document sparse vectors. The T-overlap query condition T was defined as $T = 0.5N_{\text{nonzero}} + M$, where N_{nonzero} is the number of nonzero indices in the query sparse vector and M is an integer parameter that controls the number of retrieved candidates. By adjusting M, we can flexibly control the retrieval breadth. For the query-side sparse vectors, the maximum number of nonzero indices N_{nonzero} was limited to 10, while for the document-side sparse vectors, no such limitation was imposed. Retrieval was then conducted using the defined T-overlap query condition.

4.1.4. Evaluation

Evaluation was conducted based on Recall@1000 and MRR@10 for each label in the Amazon ESCI Dataset. We also compared the average number of retrieved candidates per query (Mean number of candidates) and the number of zero-match queries (zero matches) as additional indicators of retrieval quality and coverage.

Dense vector and LoSparse (LSTM-based). Table 2 shows the comparison results between the dense vector model and LoSparse, both based on an LSTM architecture. When performing brute-force retrieval over the entire document collection, the dense model achieved high Recall. However, when using cluster brute-force retrieval after k-means clustering, the accuracy significantly dropped. For LoSparse, we conducted the T-overlap query retrieval with the condition $T = 0.5N + 3$, where N denotes the number of nonzero indices in the query sparse vector. Although the Recall was lower than that of the dense brute-force retrieval, it was higher than the ANN-based clustered retrieval method. When reducing the number of clusters in dense retrieval, Recall tends to increase as each cluster contains more documents. For example, with 700 clusters, the average number of documents per cluster was 1,734. Even under these conditions, LoSparse, with an average of 1,011 retrieved documents per query, achieved higher Recall. LoSparse used 1024-dimensional sparse vectors, with the average number of nonzero elements being 9.895 for query vectors and 49.913 for document vectors. In contrast, dense vectors require all 256 dimensions to be stored for both queries and documents, meaning that LoSparse requires significantly less memory for vector storage while maintaining competitive accuracy.

Dense vector and LoSparse (BERT-based). Next, we compared the dense vector model and LoSparse, both based on BERT fine-tuning. The results are presented in Table 3. As with the LSTM-based models, brute-force retrieval over all documents using dense vectors yielded high accuracy. However, when using cluster brute-force retrieval after k-means clustering, the accuracy dropped noticeably. In contrast, retrieval using sparse vectors with the T-overlap query achieved higher Recall than the dense vector ANN-based methods. For example, in the dense model, each cluster contained 1,734 documents, whereas LoSparse, with an average of 1,526 retrieved documents per query, achieved better Recall performance.

Table 2
Comparison between Dense Vector and LoSparse (LSTM-based)

model	R@1000_E	MRR@10_E	Mean number of candidates	zero matches
Dense LSTM Brute-force	0.7959	0.4143	-	-
Dense LSTM k-means(cluster10000)	0.3632	0.3054	121.5	0
Dense LSTM k-means(cluster700)	0.4744	0.3132	1736.9	0
LoSparse LSTM T-overlap query($0.5N$+3)	**0.5415**	0.3516	1011.3	7

Table 3
Comparison between Dense Vector and LoSparse (BERT-based)

model	R@1000_E	MRR@10_E	Mean number of candidates	zero matches
Dense BERT Brute-force	0.8201	0.3794	-	-
Dense BERT k-means(cluster700)	0.5665	0.3301	1736.9	0
LoSparse BERT T-overlap query(0.5N+3)	**0.6669**	0.3936	1526.3	4

Table 4
Comparison between SPLADE and LoSparse

model	R@1000_E	R@1000_S	R@1000_C	R@1000_I	R@1000_ESC	MRR@10_ESC	Mean number of candidates	zero matches
SPLADE (0.5N+2)	0.6600	0.4606	0.4116	0.2780	0.5981	0.5106	1334.0	127
LoSparse (0.5N+3)	**0.6669**	**0.4926**	0.3624	0.2908	**0.6108**	0.4372	1526.3	4

Table 5
Results under Different T-overlap Query Conditions

model	T-overlap condition	R@1000_E	R@1000_S	R@1000_C	R@1000_I	R@1000_ESC	MRR@10_ESC	Mean number of candidates	zero matches
SPLADE	0.5N+3	0.5197	0.3156	0.2781	0.1780	0.4573	0.4909	435.4	382
SPLADE	0.5N+2	0.6600	0.4606	0.4116	0.2780	0.5981	0.5106	1334.0	127
SPLADE	0.5N+1	0.7540	0.5774	0.5104	0.3625	0.6973	0.5169	3657.8	28
SPLADE	1	0.8303	0.6851	0.6280	0.4631	0.7842	0.5175	375633.9	0
LoSparse	0.5N+4	0.5226	0.3258	0.2145	0.1792	0.4605	0.4409	502.2	54
LoSparse	0.5N+3	0.6669	0.4926	0.3624	0.2908	0.6108	0.4372	1526.3	4
LoSparse	0.5N+2	0.7593	0.6117	0.4712	0.3741	0.7105	0.4334	4063.0	0
LoSparse	1	0.8032	0.6809	0.5706	0.4330	0.7648	0.4313	718305.2	0

Table 6
Impact of Binary Loss Weight on Retrieval Performance

model	binary loss weight	R@1000_E	R@1000_S	R@1000_C	R@1000_I	R@1000_ESC	MRR@10_ESC	Mean number of candidates	zero matches
LoSparse(0.5N+3)	0.12	0.6669	0.4926	0.3624	0.2908	0.6108	0.4372	1526.3	4
LoSparse(0.5N+3)	0.05	0.6336	0.4411	0.3291	0.2593	0.5726	0.4502	907.3	5
LoSparse(0.5N+3)	0.0	0.5413	0.3538	0.2686	0.2103	0.4805	0.4545	319.2	21

SPLADE and LoSparse. To compare models that employ sparse vector representations, we conducted an evaluation between SPLADE and the proposed LoSparse. For both models, retrieval was performed using the T-overlap query, where the threshold parameter was determined by the number of nonzero indices in the query vector, denoted as N. Specifically, the condition was set to $T = 0.5N + 2$ for SPLADE and $T = 0.5N + 3$ for LoSparse. The results for different T-overlap query conditions are discussed in the following subsection. The comparison results are shown in Table 4. LoSparse achieved a higher Recall than SPLADE. Among 5000 queries, LoSparse produced only 4 zero-match cases, whereas SPLADE resulted in 127 zero matches. This indicates that LoSparse successfully retrieved relevant documents without increasing the number of zero matches, suggesting that LoSparse performs better in T-overlap query–based retrieval.

T-overlap query condition. The results obtained by varying the T-overlap query conditions are summarized in Table 5 and Figure 2. The T-overlap query threshold was defined as $T = 0.5N_{\text{nonzero}} + M$, where N_{nonzero} represents the number of nonzero indices in the query's sparse vector. When the condition is $T = 1$, it corresponds to a logical OR search.

When $T = 1$ (OR search), both SPLADE and LoSparse achieve high Recall with zero zero-match cases. As the threshold T increases (i.e., the retrieval condition becomes stricter), the average number of retrieved documents decreases, and Recall declines correspondingly. At $T = 1$, SPLADE slightly outperforms LoSparse; however, as T becomes more restrictive, LoSparse shows higher Recall than SPLADE. Regarding zero matches, SPLADE exhibits a notable increase in zero-match cases as the threshold becomes stricter, whereas LoSparse maintains a lower increase, demonstrating that LoSparse is more robust to performance degradation under stricter T-overlap query conditions.

Influence of Binary Loss. We further investigated the effect of the Binary Loss weight λ_1 on retrieval performance while keeping all other training parameters fixed. The results are shown in Table 6. The experimental setup used an L1 regularization weight of 0.003, number of negatives = 256, and learning

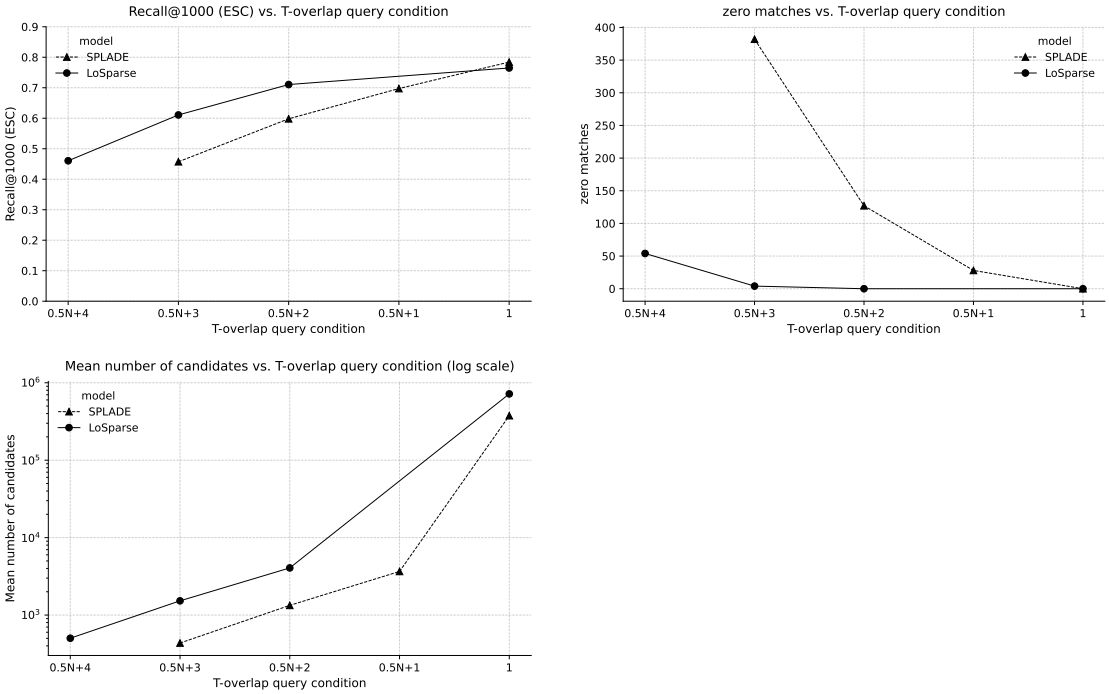

Figure 2: Relationship among T-overlap Query Condition, Recall@1000_ESC, zero matches, and Number of Retrieved Candidates

rate = 0.0001. The T-overlap query condition was fixed at $T = 0.5N_{\text{nonzero}} + 3$. As λ_1 decreases toward zero, the average number of retrieved documents decreases, while the number of zero matches increases. Conversely, as λ_1 increases, Recall improves, but MRR@10 decreases, indicating that the model shifts its focus from re-ranking precision to optimizing for T-overlap query retrieval accuracy. When $\lambda_1 = 0$, the model effectively relies only on the Ranking Loss, leading to a mismatch between the embedding optimized for loss minimization during training and the embedding required for high-accuracy T-overlap retrieval. By incorporating the Binary Loss term, the model learns embeddings that are better aligned with the discrete matching behavior of T-overlap query, resulting in improved retrieval performance.

4.2. Online Experiment

We conducted an online A/B test on Yahoo! JAPAN Shopping to compare dense retrieval and LoSparse models, both trained on data collected from the same service. For the dense retrieval setup, dense vectors were clustered using k-means, and search was performed using NGT (Neighborhood Graph and Tree for Indexing High-dimensional Data) [36]. For the sparse retrieval setup, we used 1024-dimensional sparse vectors and performed retrieval based on T-overlap query. In both settings, we employed a hybrid retrieval approach that combined keyword-based BoW matching with vector-based retrieval. The A/B test was conducted over a period of two weeks. The results showed a significant increase in both the number of clicks and the number of orders for documents retrieved by the vector-based search. Moreover, overall Click-Through Rate (CTR) and Conversion Rate (CVR) improved when combining keyword and vector retrieval, confirming the effectiveness of the proposed model and the T-overlap query-based sparse retrieval. In addition, we observed that the proposed sparse retrieval approach, LoSparse, introduced minimal computational overhead compared with dense ANN (Approximate Nearest Neighbor) methods.

5. Conclusion

In this work, we proposed LoSparse, a retrieval model that leverages low-dimensional sparse representations to simultaneously improve computational efficiency and retrieval accuracy through T-overlap query. Experiments on public benchmark datasets demonstrated that the proposed method achieves higher retrieval accuracy than existing high-dimensional sparse retrieval models under T-overlap–based matching, validating the effectiveness of LoSparse. Moreover, LoSparse outperformed dense ANN-based retrieval methods in terms of retrieval precision under identical experimental conditions. We also showed that the proposed model can be trained from scratch using lightweight architectures such as LSTMs, without relying on large pre-trained language models, while still achieving competitive performance. Finally, an online A/B test on Yahoo! JAPAN Shopping confirmed that the proposed sparse retrieval approach with T-overlap query outperforms conventional dense ANN-based retrieval methods in both retrieval effectiveness and user engagement metrics, further demonstrating the practical effectiveness and scalability of the proposed approach.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT (GPT-5) in order to: Grammar and spelling check, paraphrase and reword. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publications content.

References

[1] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*, 2019. URL:https://arxiv.org/abs/1810.04805

[2] Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power, Improving Language Understanding by Generative Pre-Training. , 2018. URL:https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[3] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781v3*, 2013. URL:https://arxiv.org/abs/1301.3781v32

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. *arXiv preprint arXiv:1706.03762*, 2017. URL:https://arxiv.org/abs/1706.03762

[5] Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, Russell Power, Semi-supervised sequence tagging with bidirectional language models *arXiv:1705.00108* URL:https://arxiv.org/abs/1705.00108

[6] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, Wen-tau Yih. Dense Passage Retrieval for Open-Domain Question Answering *In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 6769-6781.* URL:https://arxiv.org/abs/2004.049067

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

[8] Rodrigo Nogueira, Kyunghyun Cho. Passage Re-ranking with BERT *arXiv:1901.04085* URL:https://arxiv.org/abs/1901.04085

[9] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, Hervé Jégou. The Faiss library *arXiv:2401.08281* URL:https://arxiv.org/abs/2401.08281

[10] Yu. A. Malkov, D. A. Yashunin. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs *arXiv:1603.09320* URL:https://arxiv.org/abs/1603.09320

[11] Herve Jégou, Matthijs Douze, Cordelia Schmid. Product Quantization for Nearest Neighbor Search *IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011* URL:https://inria.hal.science/inria-00514462v2/document

[12] Owen Kaser, Daniel Lemire, Compressed bitmap indexes: beyond unions and intersections *arXiv:1402.4466* URL:https://arxiv.org/abs/1402.4466

[13] Chen Li, Jiaheng Lu, Yiming Lu, Efficient Merging and Filtering Algorithms for Approximate String Searches *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, ICDE'08, IEEE Computer Society: Washington, DC, USA, 2008; 257–266, doi:10.1109/ICDE.2008.4497434* URL:https://ics.uci.edu/~chenli/pub/icde08-stringsearch.pdf

[14] Jérémy Barbay, Claire Kenyon, Deterministic Algorithm for the t-Threshold Set Problem *s. Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'02, Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2002; 390–399.* URL:https://scispace.com/pdf/deterministic-algorithm-for-the-t-threshold-set-problem-2u7aefdw5b.pdf

[15] Jimmy Lin, Xueguang Ma, A Few Brief Notes on DeepImpact, COIL, and a Conceptual Framework for Information Retrieval Techniques *arXiv preprint arXiv:2106.14807*, 2021. URL:https://arxiv.org/abs/2106.14807

[16] Zhuyun Dai, Jamie Callan, Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval *arXiv:1910.10687* URL:https://arxiv.org/abs/1910.10687

[17] Soyuj Basnet, Jerry Gou, Antonio Mallia, Torsten Suel, DeeperImpact: Optimizing Sparse Learned Index Structures *arXiv:2405.17093* URL:https://arxiv.org/abs/2405.17093

[18] Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu, SparTerm: Learning Term-based Sparse Representation for Fast Text Retrieval *arXiv preprint arXiv:2010.00768*, 2020. URL:https://arxiv.org/pdf/2010.00768

[19] Omar Khattab, Matei Zaharia, ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT *In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2020), pages 39–48.* URL:https://arxiv.org/abs/2004.12832

[20] Rodrigo Nogueira and Jimmy Lin, From doc2query to docTTTTTquery URL:https://cs.uwaterloo.ca/~jimmylin/publications/Nogueira_Lin_2019_docTTTTTquery-v2.pdf

[21] Shengyao Zhuang, Guido Zuccon, TILDE: Term Independent Likelihood moDEl for Passage Re-ranking *The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Online, 11-15 July 2021.* URL:https://espace.library.uq.edu.au/view/UQ:b024b10

[22] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, Ophir Frieder, Expansion via Prediction of Importance with Contextualization *arXiv:2004.14245* URL:https://arxiv.org/pdf/2004.14245

[23] Luyu Gao, Zhuyun Dai, Jamie Callan COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List *arXiv:2104.07186* URL:https://arxiv.org/abs/2104.07186

[24] Thibault Formal, Benjamin Piwowarski, Stéphane Clinchant, SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. *arXiv preprint arXiv:2107.05720*, 2021. URL:https://arxiv.org/pdf/2107.05720

[25] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, Stéphane Clinchant, SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval. *arXiv preprint arXiv:2109.10086*, 2021. URL:https://arxiv.org/abs/2109.10086

[26] Carlos Lassance, Hervé Déjean, Thibault Formal, Stéphane Clinchant, SPLADE-v3: New baselines for SPLADE. *arXiv preprint arXiv:2403.06789*, 2024. URL:https://arxiv.org/pdf/2403.06789

[27] Yifan Sun, Changmao Cheng, Yuhan Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, Yichen Wei 1MEGVII Technology 2Beihang University 3Australian National University 4Tsinghua University Circle Loss: A Unified Perspective of Pair Similarity Optimization. *arXiv preprint arXiv:2002.10857*, 2020. URL:https://arxiv.org/abs/2002.10857

[28] Ikuya Yamada, Akari Asai, Hannaneh Hajishirzi, Learning to Hash for Indexing Big Data - A

Survey *Jun Wang, Wei Liu, Sanjiv Kumar, Shih-Fu Chang*, URL:https://arxiv.org/abs/1509.05472

[29] Ikuya Yamada, Akari Asai, Hannaneh Hajishirzi, Hashing based Answer Selection *Proceedings of the AAAI Conference on Artificial Intelligence, 34(05):9330–9337.*, URL:https://arxiv.org/abs/1905.10718

[30] Ikuya Yamada, Akari Asai, Hannaneh Hajishirzi, Efficient Passage Retrieval with Hashing for Open-domain Question Answering. *arXiv preprint arXiv:2106.00882*, 2021. URL:https://arxiv.org/pdf/2106.00882

[31] Chandan K. Reddy, Lluís Màrquez, Fran Valero, Nikhil Rao, Hugo Zaragoza, Sambaran Bandyopadhyay, Arnab Biswas, Anlu Xing, Karthik Subbian, Shopping Queries Dataset: A Large-Scale ESCI Benchmark for Improving Product Search. *arXiv preprint arXiv:2206.06588*, 2022. URL:https://arxiv.org/pdf/2206.06588, URL:https://github.com/amazon-science/esci-data

[32] Adam Paszke,Sam Gross,Francisco Massa,Adam Lerer,James Bradbury,Gregory Chanan,Trevor Killeen,Zeming Lin,Natalia Gimelshein,Luca Antiga,Alban Desmaison,Andreas Köpf,Edward Yang,Zach DeVito,Martin Raison,Alykhan Tejani,Sasank Chilamkurthy,Benoit Steiner,Lu Fang,Junjie Bai,Soumith Chintala, PyTorch: An Imperative Style, High-Performance Deep Learning Library *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, URL:https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

[33] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, Alexander M. Rush HuggingFace's Transformers: State-of-the-art Natural Language Processing *arXiv:1910.03771*, 2020. URL:https://arxiv.org/abs/1910.03771

[34] Amazon Science. Semantic Product Search. *Amazon Science*, 2021. URL:https://assets.amazon.science/95/2f/51e4ebdb4347990fedc1815d7327/semantic-product-search.pdf

[35] Turc, Iulia, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-Read Students Learn Better: On the Importance of Pre-training Compact Models. *arXiv preprint arXiv:1908.08962*, 2019. URL:https://huggingface.co/google-bert/bert-base-uncased, URL:https://github.com/google-research/bert/blob/master/README.md

[36] Masajiro Iwasaki, Daisuke Miyazaki. Optimization of Indexing Based on k-Nearest Neighbor Graph for Proximity Search in High-dimensional Data *arXiv:1810.07355*, 2018. URL:https://arxiv.org/abs/1810.07355, URL:https://github.com/yahoojapan/NGT