Learned Sparse Retrieval with Vector Quantization

Sai Sreenivas Chintha saisreenivas@umass.edu
Durga Prasad Maram dmaram@umass.edu
Harshitha Kolukuluru hkolukuluru@umass.edu
Sairohith Yanamala syanamala@umass.edu

1 Introduction

Efficient and effective retrieval of relevant documents is a fundamental challenge in information retrieval (IR). The traditional retrieval methods can be broadly categorized into **lexical matching** and **semantic matching**. Lexical retrieval methods such as BM25, rely mainly on exact keyword matching and leverage inverted indices for efficient search. While these methods are computationally efficient, they often suffer from the **vocabulary mismatch problem**, where relevant documents might be overlooked if they do not contain the exact query terms. On the other hand, **semantic retrieval methods** take advantage of dense embeddings to capture deeper contextual meanings, enabling more robust matching. However, these methods require **high-dimensional vector searches**, making them computationally expensive for accurate large-scale retrieval despite advances like Approximate Nearest Neighbor (ANN) search techniques.

Recent approaches attempt to bridge this gap by learning sparse representations from dense embeddings, where words in the vocabulary are reweighted, dropped, or substituted with related terms to improve retrieval performance. While these methods retain the efficiency of lexical retrieval, they remain constrained by vocabulary-based indexing. In this project, we propose a **novel method for learning sparse representations in the latent semantic space using vector quantization (VQ)**. Instead of mapping dense embeddings to vocabulary words, we cluster them into discrete **semantic tokens**, forming a learned codebook representation. By encoding documents and queries as sequences of cluster indices, we transform retrieval into an efficient matching problem using fast lexical-style techniques such as inverted indexing over semantic tokens.

This approach offers several advantages: (1) it captures the semantic relationships at a deeper level while maintaining high retrieval efficiency, (2) it reduces the memory and computational expenses by replacing the dense vector queries with sparse index lookups, and (3) it enables learning topics in semantic space rather than projecting embeddings into the vocabulary (as in the current LSR approaches), in order for the model to learn more contextualized, meaningful relations. Through the use of VQ-based sparse representations, this work seeks to leverage benefits of both neural and lexical IR approaches.

2 Related work

Information retrieval (IR) methods have evolved significantly in recent times, shifting from traditional lexical matching techniques to neural-based semantic matching models.

Traditional IR systems are predominantly based on lexical matching, with methods such as BM25 (Robertson and Zaragoza, 2009) and TF-IDF ranking documents based on strict term match and term frequency-inverse document frequency weight. Such methods are computationally inexpensive owing to their reliance on inverted indexing but they often perform poorly with respect to synonymy and vocabulary mismatch problems (Furnas et al., 1987).

To address these issues, neural IR models were used which use dense representations of text to capture semantic relations between words. Although early approaches like Word2Vec (Mikolov et al., 2013), and GloVe (Pennington et al., 2014) employed word embeddings to do the same, more recent retrieval models rely mostly on contextualized embeddings from transformers such as BERT (Devlin et al., 2019). More notable neural retrieval models include Dense Passage Retrieval (DPR)

(Karpukhin et al., 2020), which uses a bi-encoder-based model to represent(embed) queries and documents in the same space, ColBERT (Khattab and Zaharia, 2020), which improves efficiency by computing token-level similarities.

Despite being semantically rich, these dense retrieval models often suffer from high latency and memory overhead issues, requiring the application of Approximate Nearest Neighbor (ANN) search-based techniques like FAISS (Douze et al., 2025) and HNSW (Malkov and Yashunin, 2018) to alleviate the retrieval time. Nevertheless, ANN search remains computationally expensive, especially for large-scale retrieval tasks.

Sparse Representation Learning for Retrieval: In order to combine the efficacy of inverted indexes with the semantic benefits of dense models, more recent works have attempted to generate sparse representations of words from their dense embeddings. These methods mainly try to achieve lexicon-style representations while simultaneously preserving the semantic representations. In this line of work, SPLADE (Formal et al., 2021b) introduced a self-learned expansion procedure where the transformer model predicts sparse weights over the vocabulary tokens, using L1 regularization to enforce sparsity. Deep Impact (Mallia et al., 2021) and uniCOIL (Lin and Ma, 2021) enhance sparse representations by re-weighting term importance rather than simply expanding the queries. In Contriever (Izacard et al., 2022) and retroMAE(Xiao et al., 2022) use contrastive learning and masked autoencoding to enhance sparse retrieval capabilities.

These approaches have proved effective for improving retrieval efficiency but they still rely on mapping the dense embeddings to predefined fixed vocabulary words, which limits their flexibility and generalization capabilities. In this project, we aim to differ from this idea and want to explore the sparsity within the latent space of words rather than in the vocabulary space.

Vector Quantization for Discrete Representations: Vector Quantization (VQ) has been an extensively used technique for compression and discrete representation learning and has proven effective in tasks like image generation (VQ-VAE (van den Oord et al., 2018)), speech processing(Baevski et al., 2020), and discrete latent space modeling. The fundamental idea of VQ is to map continuous embeddings into a learned set of discrete codebook vectors, hence reducing the memory requirements while simultaneously preserving valuable information.

Though VQ has found use in generative modeling, its application to retrieval is still mostly untapped. Some studies have investigated clustering-based retrieval such as hierarchical clustering of document embeddings (Xu and Jiang, 2022), but they still do not provide us with structured, tokenizable representations for efficient indexing. Our proposed method leverages a VQ-based codebook learning to create a sequence of cluster indices – a sparse representation in a radically different form.

Clustering based Retrieval: Clustering-based structures have been explored in the works of topic modeling (e.g., LDA (Blei et al., 2003)) and document clustering (Aggarwal and Zhai, 2012) but these methods however are often not dynamically adaptive and are also not usually intended for large-scale retrieval applications.

Recent works on semantic tokenization like DeepCluster of visual features(Caron et al., 2019) suggest that clustering embeddings into discrete tokens can improve model interpretability. Our work further plans to extend this idea by treating clusters as semantic tokens and using inverted indexing with BM25 over these tokens instead of raw vocabulary words.

3 Our approach

To address the limitations of both lexical and dense retrieval methods, we propose a novel **Learned Sparse Retrieval with Vector Quantization (LSR-VQ)** framework.

3.1 Dense to Sparse Transformation with Vector Quantization

Unlike traditional sparse retrieval methods that operate over vocabulary terms, our approach discretizes dense embeddings into a sequence of learned cluster indices, forming a sparse representation in a latent semantic space. We explore the bi-encoder variant for dense representation learning.

3.1.1 Generating Dense Representations

Bi-Encoder: The query and document are passed separately through an encoder model (eg: BERT), trained using contrastive loss (ranking loss) to learn semantically aligned embeddings.

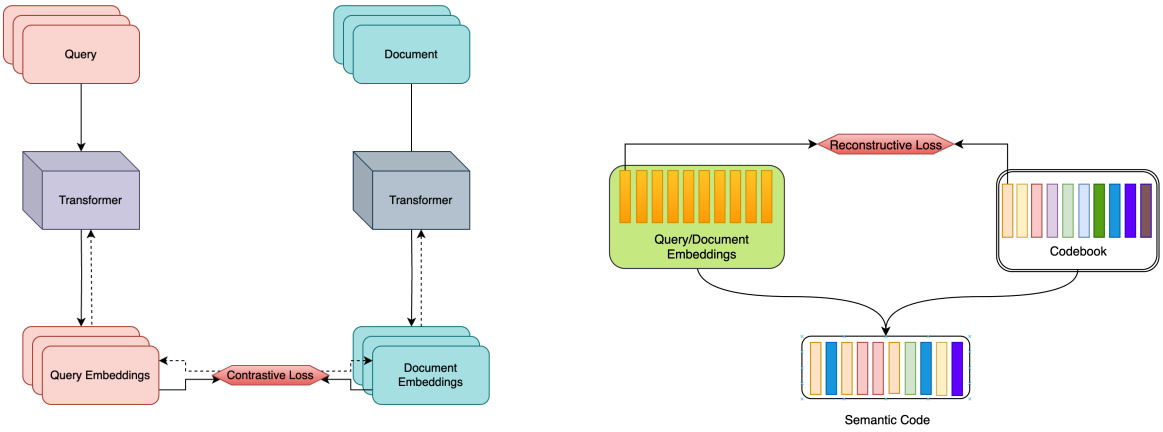

Figure 1: LSR with bi-encoder

Contrastive loss - Given a query q_i in a batch, a positive document d_i^+, a (hard) negative document d_i^- (e.g., coming from BM25 sampling), and a set of negative documents in the batch (positive documents from other queries) $\{d_{i,j}^-\}_j$,

$$L_{\text{rank}} = -\log \frac{e^{s(q_i,d_i^+)}}{e^{s(q_i,d_i^+)} + e^{s(q_i,d_i^-)} + \sum_j e^{s(q_i,d_{i,j}^-)}}$$

We then extract fixed dense embeddings either from:

1. The [CLS] token (global representation), or
2. The final hidden states of input text tokens (for richer representations).

3.1.2 Learning Sparse Representations

Method 1: Finetune Encoder and Train Codebook Separately

1. The selected bi-encoder is first fine-tuned on query-document pairs.
2. A codebook of K cluster centroids is then learned in the latent space.
3. For each of the hidden embeddings $h_1, h_2, ...h_k$, we compute the nearest codebook vector e_i, producing a sequence of cluster indices (semantic code).
4. The codebook is optimized using the reconstruction loss:

$$L_{\text{rec}} = \sum_i ||h_i - e_i||_2^2$$

Method 2: Jointly Train Encoder and Codebook

1. We jointly train both the encoder (pretrained) and the codebook vectors.
2. The training procedure for the codebook remains the same as in Method 1, where both the encoder parameters and codebook are updated.

$$L = L_{rank} + L_{rec}$$

The overall architecture of this approach based on bi-encoder is illustrated in Figure 1.

This joint training might be unstable, requiring us to resort to techniques such as alternating between aligning the dense vectors in the latent space with gradient descent and updating the codebook vectors in different iterations.

Experiments : We will also implement the above methods with following variations:

- Alternative objective functions (eg: commitment loss (van den Oord et al., 2018)).
- Add heads over grouped hidden vectors from the last embedding layer and perform VQ on these head outputs.
- Explore hierarchical codebooks to further optimize semantic vector representations.
- Investigate the possibility of using **cross-encoder** architecture to get more context rich embeddings for quantization. The challenge here is the inability to build the index before inference as the cross-encoder gives joint representation.
- Possible extension to **multi-modal** retrieval depending on time.

Once trained, these learned sparse representations can be used for retrieval via IOU-based ranking or BM25 over semantic tokens.

3.2 Building Inverted Index & Retrieval

Once we obtain the semantic code, we construct an inverted index to enable efficient retrieval. Each codebook vector acts as an index term, similar to words in traditional lexical retrieval.

3.2.1 Building the Inverted Index

We construct an inverted index where each **codebook vector** (semantic token) points to the documents in which it appears. This enables fast lookup and retrieval using sparse matching techniques.

3.2.2 Retrieval Methods

IOU-based Ranking: Given a query's semantic code, we rank documents based on the Intersection over Union (IOU) score between their codes.
Lexical Matching: Since the inverted index treats codebook vectors as tokens, we apply BM25 over these semantic tokens.

3.3 Baseline algorithms

To validate the effectiveness of our approach, we plan to compare it against the following baselines: BM25 (Lexical Baseline) (Robertson and Zaragoza, 2009), Dense bi-encoder with Approximate Nearest Neighbor (ANN) Search (eg: Contriever (Lei et al., 2023)) and, existing Learned Sparse Retrieval methods (eg: SPLADE(Formal et al., 2021b),SPLADEv2(Formal et al., 2021a))

4 Schedule

1. Pre-processing the datasets (2 weeks)
2. Building the baseline models (1 week)
3. Implementation (5 weeks)
4. Conducting ablation studies and evaluating on other datasets (1 week).
5. Working on the report and result analysis (1 week)

We plan to work on all the tasks together. To ensure a thorough evaluation, we have allocated time for both model analysis and final report preparation. If time permits and our approach proves effective, we will explore experiments with **multimodal data** as well.

5 Data

We plan to train and evaluate our models on the following datasets,
MS MARCO (Bajaj et al., 2016): We take the passage ranking dataset in the full ranking setting. This dataset comprises approximately 8.8M passages and hundreds of thousands of training queries, with 1.1 relevant passages per query. For evaluation, we will use the development set, which approximately includes 6,900 queries. (https://microsoft.github.io/msmarco/Datasets)
TREC DL 2023: Contains high-quality judgments from fine-grained human annotations. We will evaluate our model's performance on the dev set of the passage ranking dataset. (https://microsoft.github.io/msmarco/TREC-Deep-Learning)

6 Tools

We will use PyTorch for implementation and the HuggingFace library for transformer models. Additionally, we will utilize the pytrec_eval tool (https://github.com/cvangysel/pytrec_eval) to compute evaluation metrics such as NDCG, Precision, and Recall. GPUs will be required for this work, and we plan to use Google Colab for that. We also plan to purchase Colab Pro to access advanced GPUs like the A100 for enhanced compute performance.

7 AI Disclosure

- Did you use any AI assistance to complete this proposal? If so, please also specify what AI you used.
  - Yes

If you answered yes to the above question, please complete the following as well:

- If you used a large language model to assist you, please paste *all* of the prompts that you used below. Add a separate bullet for each prompt, and specify which part of the proposal is associated with which prompt.
  - Is the grammar in "[our text]" correct?
  - Check Grammar "[our text]"
- **Free response:** For each section or paragraph for which you used assistance, describe your overall experience with the AI. How helpful was it? Did it just directly give you a good output, or did you have to edit it? Was its output ever obviously wrong or irrelevant? Did you use it to generate new text, check your own ideas, or rewrite text?
  - It was quite useful. It helped us correct our grammar and suggested new words to replace those that were sometimes un-

consciously overused. Occasionally, it suggested new words when it wasn't really necessary, but it was nice to have synonyms readily available. Yes, it did help us generate new text and rewrite a few words in certain scenarios.

References

Aggarwal, C. C. and Zhai, C. (2012). *A Survey of Text Clustering Algorithms*, pages 77–128. Springer US, Boston, MA.

Baevski, A., Zhou, H., Mohamed, A., and Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations.

Bajaj, P., Campos, D., Craswell, N., Deng, L., Gao, J., Liu, X., Majumder, R., McNamara, A., Mitra, B., Nguyen, T., et al. (2016). Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Caron, M., Bojanowski, P., Joulin, A., and Douze, M. (2019). Deep clustering for unsupervised learning of visual features.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L., and Jégou, H. (2025). The faiss library.

Formal, T., Lassance, C., Piwowarski, B., and Clinchant, S. (2021a). SPLADE v2: Sparse lexical and expansion model for information retrieval. *CoRR*, abs/2109.10086.

Formal, T., Piwowarski, B., and Clinchant, S. (2021b). Splade: Sparse lexical and expansion model for first stage ranking.

Furnas, G. W., Landauer, T. K., Gomez, L. M., and Dumais, S. T. (1987). The vocabulary problem in human-system communication. *Commun. ACM*, 30(11):964–971.

Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., and Grave, E. (2022). Unsupervised dense information retrieval with contrastive learning.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and tau Yih, W. (2020). Dense passage retrieval for open-domain question answering.

Khattab, O. and Zaharia, M. (2020). Colbert: Efficient and effective passage search via contextualized late interaction over bert.

Lei, Y., Ding, L., Cao, Y., Zan, C., Yates, A., and Tao, D. (2023). Unsupervised dense retrieval with relevance-aware contrastive pre-training. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10932–10940, Toronto, Canada. Association for Computational Linguistics.

Lin, J. and Ma, X. (2021). A few brief notes on deepimpact, coil, and a conceptual framework for information retrieval techniques.

Malkov, Y. A. and Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs.

Mallia, A., Khattab, O., Tonellotto, N., and Suel, T. (2021). Learning passage impacts for inverted indexes.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3:333–389.

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. (2018). Neural discrete representation learning.

Xiao, S., Liu, Z., Shao, Y., and Cao, Z. (2022). Retromae: Pre-training retrieval-oriented language models via masked auto-encoder.

Xu, T. and Jiang, J. (2022). A graph adaptive density peaks clustering algorithm for automatic centroid selection and effective aggregation. *Expert Systems with Applications*, 195:116539.