НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
январь–февраль 2025 Том 25 № 1 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
January–February 2025 Vol. 25 No 1 http://ntv.ifmo.ru/en/
ISSN 2226-1494 (print) ISSN 2500-0373 (online)

doi: 10.17586/2226-1494-2025-25-1-61-67

Efficient sparse retrieval through embedding-based inverted index construction

Viacheslav Yu. Dobrynin[1]✉, Roman K. Abramovich[2], Alexey V. Platonov[3]

[1,2,3] ITMO University, Saint Petersburg, 197101, Russian Federation
[1] Shift Lab LTD, London, W3 7XS, Great Britain
[2] Payler Ltd, London, E14 4QA, Great Britain

[1] vidobrynin@itmo.ru✉, https://orcid.org/0009-0004-3056-8403
[2] asmetliness24237@gmail.com, https://orcid.org/0009-0005-5397-2772
[3] avplatonov@itmo.ru, https://orcid.org/0000-0002-8485-1296

Abstract
Modern search engines use a two-stage architecture for efficient and high-quality search over large volumes of data. In the first stage, simple and fast algorithms like BM25 are applied, while in the second stage, more precise but resource-intensive methods methods, such as deep neural networks, are employed. Although this approach yields good results, it is fundamentally limited in quality due to the vocabulary mismatch problem inherent in the simple algorithms of the first stage. To address this issue, we propose an algorithm for constructing an inverted index using vector representations combining the advantages of both stages: the efficiency of the inverted index and the high search quality of vector models. In our work, we suggest creating a vector index that preserves the various semantic meanings of vocabulary tokens. For each token, we identify the documents in which it is used, and then cluster its contextualized embeddings. The centroids of the resulting clusters represent different semantic meanings of the tokens. This process forms an extended vocabulary which is used to build the inverted index. During index construction, similarity scores between each semantic meaning of a token and documents are calculated which are then used in the search process. This approach reduces the number of computations required for similarity estimation in real-time. Searching the inverted index first requires finding keys in the vector index, helping to solve the vocabulary mismatch problem. The operation of the algorithm is demonstrated on a search task within the SciFact dataset. It is shown that the proposed method achieves high search quality with low memory requirements. The proposed algorithm demonstrates high search quality, while maintaining a compact vector index whose size remains constant and depends only on the size of the vocabulary. The main drawback of the algorithm is the need to use a deep neural network to generate vector representations of queries during the search process which slows down this stage. Finding ways to address this issue and accelerate the search process represents a direction for future research.

Keywords
inverted index, vocabulary mismatch problem, neural networks, vector representations, clusterization

For citation: Dobrynin V.Yu., Abramovich R.K., Platonov A.V. Efficient sparse retrieval through embedding-based inverted index construction. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2025, vol. 25, no. 1, pp. 61–67 doi: 10.17586/2226-1494-2025-25-1-61-67

УДК 004.89

Эффективный разреженный поиск с помощью построения инвертированного индекса на основе эмбеддингов

Вячеслав Юрьевич Добрынин[1]✉, Роман Константинович Абрамович[2], Алексей Владимирович Платонов[3]

[1,2,3] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация
[1] Shift Lab LTD, Лондон, W3 7XS, Великобритания
[2] Payler Ltd, Лондон, E14 4QA, Великобритания

[1] vidobrynin@itmo.ru✉, https://orcid.org/0009-0004-3056-8403
[2] asmetliness24237@gmail.com, https://orcid.org/0009-0005-5397-2772
[3] avplatonov@itmo.ru, https://orcid.org/0000-0002-8485-1296

Аннотация
Введение. Современные поисковые системы используют двухэтапную архитектуру для эффективного и качественного поиска по большим объемам данных. На первом этапе применяются простые и быстрые алгоритмы, такие как BM25, а на втором — более точные, но ресурсоемкие методы, например глубокие нейронные сети. Несмотря на то, что такой подход показывает хорошие результаты, он фундаментально ограничен по качеству из-за проблемы несовпадения словарей, что присуще простым алгоритмам первого этапа. **Метод.** Для решения проблемы ограничений качества поиска, в настоящей работе предлагается алгоритм построения инвертированного индекса с использованием векторных представлений. Представленный подход объединяет преимущества обоих этапов: эффективность инвертированного индекса и высокое качество поиска при использовании векторных моделей. Предложено создание векторного индекса, сохраняющего различные семантические значения токенов словаря. Для каждого токена определяются документы, в которых он используется, после чего его контекстуализированные эмбеддинги кластеризуются. Центроиды полученных кластеров представляют различные семантические значения токенов. Таким образом, формируется расширенный словарь, который применяется для построения инвертированного индекса. При построении индекса вычисляются оценки близости между каждым семантическим значением токена и документами, что затем используется в процессе поиска. Это позволяет сократить количество вычислений для оценки близости в режиме реального времени. Поиск по инвертированному индексу требует нахождения ключей в векторном индексе, что позволяет решить проблему несовпадения словарей. **Основные результаты.** Работа алгоритма продемонстрирована на задаче поиска в наборе данных SciFact. Показано, что предлагаемый метод обеспечивает высокое качество поиска при низких требованиях к объему памяти. **Обсуждение.** Разработанный алгоритм демонстрирует высокое качество поиска, при этом он поддерживает компактный векторный индекс, размер которого остается неизменным и определяется исключительно размерами словаря. Основным недостатком алгоритма является необходимость использования глубокой нейронной сети для генерации векторных представлений запроса в процессе поиска, что замедляет этот этап. Поиск путей для решения данной проблемы и сокращения времени поиска представляет собой направление дальнейших исследований.

Ключевые слова
инвертированный индекс, проблема несоответствия словарей, нейронные сети, векторные представления, кластеризация

Ссылка для цитирования: Добрынин В.Ю., Абрамович Р.К., Платонов А.В. Эффективный разреженный поиск с помощью построения инвертированного индекса на основе эмбеддингов // Научно-технический вестник информационных технологий, механики и оптики. 2025. Т. 25, № 1. С. 61–67 (на англ. яз.). doi: 10.17586/2226-1494-2025-25-1-61-67

Introduction

Modern search systems typically use a two-stage architecture to balance between speed and search quality while working with massive volumes of data such as the entire Internet. At the first stage, simple and fast algorithms are used to reduce the number of candidates to hundreds or thousands. At the second stage, these candidate documents are re-ranked using more complex models that provide high-quality results but require by an order of magnitude more processing power. Consequently, those models cannot be efficiently applied to the entire dataset due to performance constraints. However, this approach has limitations in search quality, as the algorithm at the first stage can miss relevant documents by not accounting for the semantics of the texts.

nd-to-end solutions, such as Contextualized Late Interaction over Bidirectional Encoder Representations from Transformers (BERT) (ColBERT), implement search using a single stage, allowing for significantly improved search quality. For example, Best Matching 25 (BM25), which is often used as a first-stage ranking model, achieves a Mean Reciprocal Rank (MRR)@10 of 19.5 on the Microsoft Machine Reading Comprehension (MS MARCO) dataset, whereas ColBERT (end-to-end) achieves an MRR@10 of 36.7 [1]. The main innovation in the ColBERT architecture is the late interaction mechanism which independently encodes query and document tokens into vector representations that are used for computing relevance scores. This approach allows us to independently pre-compute document embeddings at the offline stage and store them in a vector index for further retrieval. However, it also requires a significant amount of resources to store the indexed documents and process incoming queries, making it challenging to apply to large-scale datasets.

To address this problem, we propose implementing a single-stage architecture algorithm that uses an inverted index as the primary structure for indexing and searching. However, unlike classical approaches like BM25, our method constructs an inverted index using deep neural networks, allowing for a more precise capture of the context of tokens and their relevance to the documents.

In this paper, we present a new method that leverages the efficiency of an inverted index while maintaining the search quality of vector models. Our approach combines the advantages of inverted index structure with the scoring provided by deep learning models that deeply understand token semantics.

Related works

The Sparse Neural Ranking Model (SNRM), introduced by Zamani et al. in 2018 [2], was one of the first works that tried to integrate deep neural networks with traditional inverted indexing. By using sparsity constraints, SNRM is trained to generate high-dimensional sparse embeddings for both queries and documents, which can be later used to construct an inverted index. This model was able to significantly improve search quality, but, at the same time, it has certain limitations related to its architecture, such as loss of token interpretability, fixed dimensionality of output vectors, and the need to process queries through the model which significantly increases computational resources at query time.

SparTerm, introduced by Bai et al. in 2020 [3], was designed to improve traditional sparse term-based representations by using deep models like BERT [4]. By generating dense contextualized embeddings that capture the semantics of each term and using a gating controller to sparsify the resulting vectors, SparTerm is able to construct an inverted index using original vocabulary terms. By doing so, SparTerm improves semantic matching in the inverted index, while keeping the interpretability and efficiency of classical methods.

The Sparse Lexical And Expansion (SPLADE) model by Formal et al. [5], builds upon SparTerm by simplifying its architecture. The main idea is that instead of using a gating controller to achieve sparsity, the authors would employ a log-saturation function and a sparsifying regularization at the training stage to induce sparsity in the output vectors, thus addressing one of the key limitations of SparTerm, allowing for end-to-end training and reducing computational complexity.

ColBERT [1] and ColBERTv2 [6] can be considered an alternative approach to generating sparse vectors, as instead of building an inverted index, it focuses on algorithmic optimizations to reduce computational resources required for search. In this work, the authors introduce the concept of “late interaction” which separates the encoding of queries and documents from computing relevance scores between them. By employing an Approximate Nearest Neighbor (ANN) index with the Facebook AI Similarity Search [7] library and vector compression techniques like Product Quantization [8], combined with offline indexing, ColBERT allows for significantly reducing resources required for storing and processing search queries. However, despite achieving high search quality and requiring significantly fewer resources than traditional vector search methods, ColBERT still requires more computational resources during query time and greater storage space for document embeddings than inverted index models.

The model SparseEmbed [9] was inspired by both SPLADE and ColBERT. By generating sparse vectors using the same approach as SPLADE and storing dense embeddings for each input token, SparseEmbed constructs an inverted index with original vocabulary terms, where the values stored in the index are the dense representations of the tokens. At search time, it uses dense embeddings of activated tokens to compute relevance scores efficiently. This approach improves context capture compared to SPLADE by using dense representations and is more efficient than ColBERT as it requires linear time relative to the number of activated terms rather than quadratic time. SparseEmbed achieved an MRR@10 score of 39.2 on the MS MARCO dataset, which is slightly the score of 39.7 achieved by ColBERTv2. However, it still requires storing dense vectors for each token and document and computing contextualized embeddings at query time, increasing the computational resources needed during search.

The work Sparse Transformer Matching (SPARTA) [10] offers an efficient neural ranking method that addresses the limitations of dense vector search in open-domain question answering. Unlike similar models that rely entirely on dense embeddings, SPARTA learns sparse representations that can be implemented as an inverted index, allowing for scalable retrieval without the need for expensive ANN search. SPARTA captures fine-grained relevance information by focusing on token-level interactions between queries and documents, allowing for high-quality matching while maintaining efficiency. This approach significantly improves retrieval performance compared to dense models and achieves state-of-the-art results across multiple open-domain question answering tasks.

This paper is a continuation of the algorithm proposed in [11]. The main improvement is in the way we select contextualized embeddings from documents. Instead of computing context based on a sliding window algorithm as in our earlier approach, we now utilize contextualized vector representations of tokens within the entire document. This change allows us to perform more effective clustering and capture different semantic meanings of words. By constructing an inverted index using these embeddings, we address the resource-intensive computations required during search, achieving efficiency comparable to traditional methods while capturing the semantic richness highlighted in models like SPARTA and SPLADE.

In the next sections we describe the whole algorithm with the new upgrades.

Description of the Proposed Algorithm

The usage of transformer models allows for significantly improved search quality due to their ability to capture complex semantic relationships between tokens in the text. However, these models also require substantial computational resources, making them far less practical to use directly in large-scale search systems.

In contrast, the usage of inverted indexes is a standard practice in modern production search systems due to their scalability, high performance, and relatively low memory usage.

Inverted indexes are able to efficiently retrieve documents based on exact query term matching, however, when queries and documents use different words with similar meanings, traditional inverted indexes fail to retrieve relevant documents, which is known as the vocabulary mismatch problem.

To address this problem, we propose a novel method of inverted index construction, where the index terms are selected based on their semantic similarity, rather than exact term matching. By doing so, we aim to resolve the vocabulary mismatch problem by incorporating semantic understanding into the index building process.

Our method involves training a compact vector index that contains embeddings representing different semantic meanings for each token in the vocabulary. To generate these embeddings, we cluster the contextualized token embeddings across all documents in the dataset. This allows us to capture the various contexts in which a token appears, effectively representing its multiple semantic meanings.

Index construction

As search systems face the requirement to work with massive data volumes, the efficient implementation of the indexing stage is a crucial concern. Caching is one of the most common optimization techniques to speed up data processing, and the one that we adopted to optimize our indexing algorithm.

As we build both a compact vector index and an inverted index using vector representations extracted from documents, it is a logical step to cache those representations for both processes. Our vector representations are obtained using a deep neural network based on the transformer architecture. More specifically, it is a bidirectional encoder that considers each token context by looking at both preceding and following tokens. This allows for a deeper semantic understanding of words depending on their context, which is a critical factor for calculating the relevance score between tokens and documents.

The result of this preparation stage is a collection that contains mappings of document identifiers to their corresponding contextualized embeddings, represented as pairs (*doc_id*, *contextualized_embs*), which is later used for obtaining semantic clusters for tokens, building a vector index and an inverted index.

The index construction is divided into two main stages: first, we use vector representations and clustering to build an expanded vocabulary that captures different semantic contexts of tokens, and second, we use this expanded vocabulary to construct the inverted index.

The core idea of the proposed approach is to construct a fixed Hierarchical Navigable Small World (HNSW) [12] index that contains vector representations of all vocabulary tokens in their various semantic contexts across indexed documents. This index allows us to efficiently distinct different token contexts at both indexing and query time. Since a token can have multiple meanings depending on its usage, capturing these variations is essential for semantic search.

The first step is quite similar to building a classical inverted index. For each token, we collect and store all the document ids in which this token appears, ending up with a map of token id to the list of document ids. This collection will later be used to gather contextualized embeddings for each token across different contexts.

Then, for each token in the map, the following steps are performed:

1. **Collect Contextualized Embeddings** from all the documents that the token appears in. As the token might have different semantic meanings based on the context, by collecting token embeddings from all documents we make sure to account for all of them. At this step, we use the (*doc_id*, *contextualized_embs*) map prepared at the preparation step to speed up the process and avoid re-calculating embeddings for each document over and over.
2. **Clustering:** As soon as we have all token embeddings from each document it appears in, we perform clustering on these embeddings using the *k*-means++ algorithm [13]. As a result, we get a small set of cluster centroids that group similar embeddings and represent different semantic meanings that this token has.
3. **Store semantic centroids:** Finally, we store the resulting centroids to the HNSW index, where they can be later used for building an inverted index. With each centroid, we also store the associated metadata required for further steps: a source token and a unique cluster identifier (*token_id*, *cluster_id*).
   This process is depicted in Fig. 1.

The constructed HNSW index enables efficient nearest neighbor search based on semantic similarity during both indexing and query processing. By using those centroids to build an inverted index we address the vocabulary mismatch problem and allow for a search based on semantics and thus more agile, but still interpretable as we save corresponding token and cluster identifiers with each embedding in both vector and inverted index.

At this step, we combine the results from all the previous steps in order to construct the inverted index.

To do so, we iterate over the collection of document ids mapped to their contextualized embeddings generated at the preparation step. For each contextualized token embedding *e* in the document, we search for its nearest semantic centroids from the HNSW index built at the previous step.

The relevance score between contextualized embeddings of a document and retrieved semantic centroids is calculated using the MaxSim operator as defined in [1]. Thus, in our work, most of the computations of the "late interaction" mechanism are performed at the indexing stage, which speeds up the search process compared to ColBERT. Finally, these relevance scores are then stored in the inverted index along with the document ids, where the keys are represented as pairs of token and cluster identifiers (*token_id*, *cluster_id*).

The whole inverted index construction process is presented in Fig. 2.

This algorithm allows us to effectively expand the vocabulary of the inverted index based on a deep semantic understanding of the source texts. The vocabulary mismatch problem is addressed by searching for nearest semantic clusters for each token, allowing for including semantic clusters from different tokens to the resulting posting list even if they do not appear in the document. The inverted index remains efficient and interpretable, as it still relies on tokens from the original vocabulary augmented with cluster identifiers representing different semantic meanings.

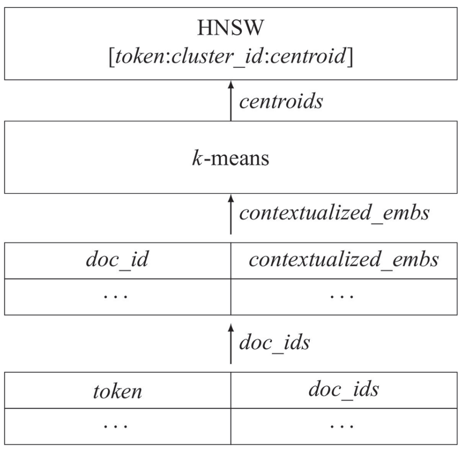

Fig. 1. Vector index construction process

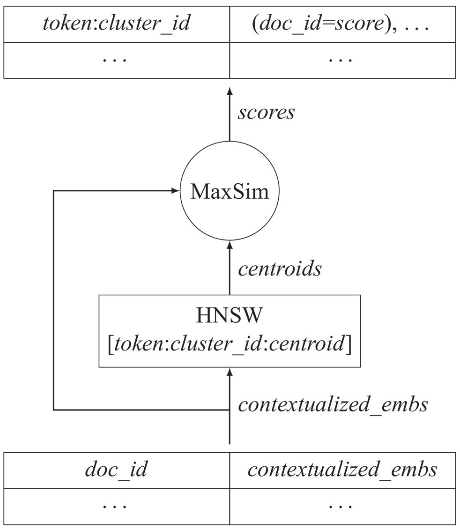

Fig. 2. Inverted index construction process

Search process

The search process begins by encoding the query with the same encoder used at the indexing stage (e.g., a pretrained transformer model like BERT). Then, for each contextualized embedding of the query tokens, we use a vector index to search for the nearest neighbor tokens. These nearest tokens represent the semantic variations of the query tokens captured during the clustering step.

The retrieved tokens and cluster ids are then used to look up the list of documents and their associated scores from the inverted index. These scores represent the relevance between the document and the semantic cluster of the token.

To compute the overall relevance score for each document, we aggregate the scores from all matching tokens by summing them:

$$score_{query,doc} = \Sigma_{token \in query} \Sigma_{cluster\_id} score_{token:cluster\_id,doc},$$

where $score_{token:cluster\_id,doc}$ is the relevance score between the document and the specific semantic variation of the token.

Finally, we sort the documents in descending order based on their aggregated scores. This results in a ranked list of documents ordered by their relevance to the query. This method leverages the semantic understanding captured during the indexing process, allowing for effective retrieval even when there is a vocabulary mismatch between the query and the documents.

The search process is presented in Fig. 3.

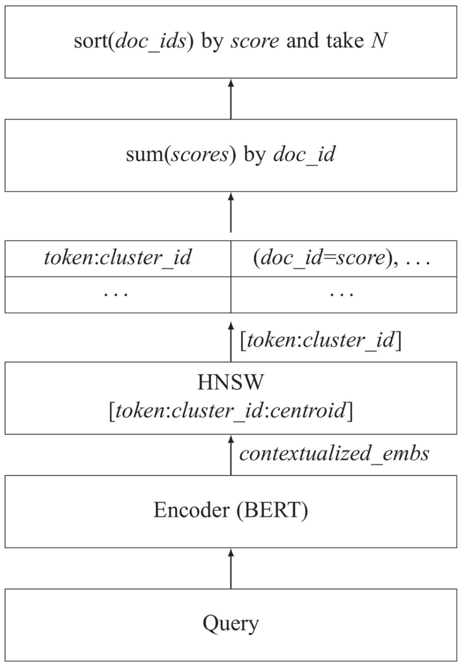

Fig. 3. Search process for a given query

The BIER framework provides several standard metrics that can be used to evaluate search quality, such as Normalized Discounted Cumulative Gain (NDCG), Mean Average Precision, MRR, Precision and Recall, etc. To assess our model, we've primarily focused on NDCG and MRR metrics which are commonly used to measure the ranking quality of retrieval systems.

As the dense model that would convert our documents into contextualized embeddings, we used the sentence-transformers/all-MiniLM-L6-v2, which is a highly efficient transformer model optimized for semantic search. However, while this model balances between speed and embedding quality, it has a considerable limitation, as the maximum input sequence length is restricted to 256 tokens.

We chose the HNSW algorithm for vector search due to its high search quality and low search latency in comparison to other approximate algorithms. However, while HNSW requires considerable memory resources, this does not limit our algorithm since the index size depends on the number of terms in the vocabulary rather than the number of documents in the corpus, which keeps memory usage optimal.

For comparison, we considered the SPARTA, HNSW, and ColBERTv2 algorithms. The results of the comparison are shown in Table.

The results of our study demonstrate that our model achieves higher search quality compared to SPARTA

Evaluation of the Proposed Algorithm

To train our model, we performed a *k*-means clustering on 300,000 documents from the widely-used ranking dataset MS MARCO [14] passages. The evaluation was done using the Benchmarking Information Retrieval (BIER) framework [15] which provides a variety of datasets and metrics for assessing the quality of search systems. As a dataset for rapid quality validation, we selected the SciFact dataset containing approximately 5,000 documents.

Table. Retrieval results on SciFact dataset

Algorithm	Metrics	
	NDCG@10	MRR@10
SPARTA	0.60	0.57
Our work	0.63	0.60
HNSW	0.64	0.60
ColBERTv2	0.69	0.66

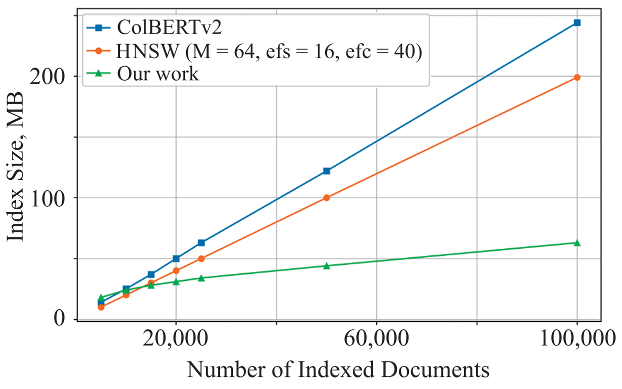

Fig. 4. Index size growth with increasing documents

(conceptually similar algorithm), provides comparable quality to HNSW (one of the best algorithms for ANN search), and falls behind ColBERT (state-of-the-art in search quality). However, due to the use of an inverted index, our model requires less memory than both HNSW and ColBERT, as illustrated in Fig. 4.

In the figure above, only the size of the inverted index for our algorithm is shown. The size of our vector index is constant and, therefore, not included in the calculation.

Discussion of Results

Vector index size

Although our approach requires a separate HNSW index, the way we construct it makes our algorithm more efficient than those of ColBERT and SparseEmbed. In ColBERT, the number of vectors that need to be stored increases linearly with the number of documents in the dataset, as a separate embedding must be saved for each new document, reducing scalability for large datasets. SparseEmbed further exacerbates this issue by storing embeddings of each token within the document as a value in the inverted index, thus requiring to store dozens of embeddings per document.

In contrast, the size of our vector index does not depend directly on the size of the dataset. We store embeddings of different contextual clusters of tokens within our vocabulary. Therefore, the dataset size affects the index size only indirectly: as more documents are added, new contextual meanings of tokens may appear, potentially increasing the number of clusters. This means that for small datasets consisting of a few thousand documents, our approach might require more memory than ColBERT as we would store at least one embedding for each token in the vocabulary. However, as the dataset size grows, the storage requirements for ColBERT and SparseEmbed increase linearly, while in our approach the size of the vector index stabilizes and remains constant as soon as new documents cease to introduce new semantic meanings to existing clusters.

Moreover, as our vector index stores semantic clusters for each token in the vocabulary, as long as our vocabulary does not change, we can train this vector index once and then reuse it in further iterations of indexing and searching processes.

Query-processing overhead

A major limitation of our algorithm is the requirement to process each query through a BERT-like model to obtain the vector representations of its tokens which are subsequently used to retrieve semantically similar clusters from the HNSW index. This step is computationally expensive and might significantly increase the query-time latency. In future iterations of the algorithm, we intend to explore strategies to eliminate this costly model invocation to enhance the efficiency of our model. In the current version, it is still possible to perform search queries using only the inverted index, though it affects the search quality.

Conclusion

In this work, we proposed a novel algorithm for inverted index construction, designed to adapt deep neural models for usage with an efficient data structure widely adopted in production information retrieval systems. By utilizing contextualized vector representations, our approach can effectively address the vocabulary mismatch problem between documents and queries, leading to significant search quality improvements.

A key feature of our algorithm is the construction of a compact vector index used for capturing different semantic meanings of vocabulary tokens. The resulting index has a relatively small size which does not linearly depend on the amount of data in the search system, but rather on the size of the vocabulary, which is a major advantage of our algorithm. Additionally, by utilizing the MaxSim operator during the indexing process, we reduce the computational load of the search process compared to ColBERT while maintaining the advantages of the "late interaction" mechanism in terms of search quality.

Experimental results show the effectiveness of our approach, achieving strong search quality, which confirms its potential applicability to real-world information retrieval tasks.

References

1. Khattab O., Zaharia M. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. *Proc. of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 39–48. https://doi.org/10.1145/3397271.3401075
2. Zamani H., Dehghani M., Croft W.B., Learned-Miller E., Kamps J. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. *Proc. of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 497–506. https://doi.org/10.1145/3269206.3271800

Литература

1. Khattab O., Zaharia M. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT // Proc. of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020. P. 39–48. https://doi.org/10.1145/3397271.3401075
2. Zamani H., Dehghani M., Croft W.B., Learned-Miller E., Kamps J. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing // Proc. of the 27th ACM International Conference on Information and Knowledge Management. 2018. P. 497–506. https://doi.org/10.1145/3269206.3271800

3. Bai Y., Li X., Wang G., Zhang C., Shang L., Xu J., Wang Z., Wang F., Liu Q. SparTerm: Learning term-based sparse representation for fast text retrieval. *arXiv*, 2020, arXiv:2010.00768. https://doi.org/10.48550/arXiv.2010.00768
4. Devlin J., Chang M.W., Lee K., Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv*, 2018, arXiv:1810.04805v2. https://doi.org/10.48550/arXiv.1810.04805
5. Formal T., Piwowarski B., Clinchant S. SPLADE: Sparse lexical and expansion model for first stage ranking. *Proc. of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 2288–2292. https://doi.org/10.1145/3404835.3463098
6. Santhanam K., Khattab O., Saad-Falcon J., Potts C., Zaharia M. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. *Proc. of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2022, pp. 3715–3734. https://doi.org/10.18653/v1/2022.naacl-main.272
7. Johnson J., Douze M., Jégou H. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 2021, vol. 7, no. 3, pp. 535–547. https://doi.org/10.1109/tbdata.2019.2921572
8. Jégou H., Douze M., Schmid C. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011, vol. 33, no. 1, pp. 117–128. https://doi.org/10.1109/tpami.2010.57
9. Kong W., Dudek J.M., Li C., Zhang M., Bendersky M. SparseEmbed: Learning sparse lexical representations with contextual embeddings for retrieval. *Proc. of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2023. P. 2399–2403. https://doi.org/10.1145/3539618.3592065
10. Tiancheng Z., Lu X., Lee K. SPARTA: Efficient Open-Domain Question Answering via Sparse Transformer Matching Retrieval. *Proc. of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 565–575. https://doi.org/10.18653/v1/2021.naacl-main.47
11. Dobrynin V., Abramovich R., Platonov A. Building a full-text search index using “Transformer” neural network. *Proc. of the 2023 IEEE 17th International Conference on Application of Information and Communication Technologies (AICT)*, 2023. pp. 1–5. https://doi.org/10.1109/aict59525.2023.10313200
12. Malkov Y.A., Yashunin D.A. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020, vol. 42, no. 4, pp. 824–836. https://doi.org/10.1109/tpami.2018.2889473
13. Arthur D., Vassilvitskii S. k-means++: the advantages of careful seeding. *Proc. of the 18th ACM-SIAM Symposium on Discrete Algorithms Mathematics*, 2007, pp. 1027–1035.
14. Bajaj P., Campos D., Craswell N., Deng L., Gao J., Liu X., Majumder R., McNamara A., Mitra B., Nguyen T., Rosenberg M., Song X., Stoica A, Tiwary S., Wang T. MS MARCO: a human generated MAchine Reading COmprehension dataset. *arXiv*, 2016, arXiv:1611.09268. https://doi.org/10.48550/arXiv.1611.09268
15. Thakur N., Nils R., Rücklé A., Srivastava A., Gurevych I. BEIR: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv*, 2021, arXiv:2104.08663. https://doi.org/10.48550/arXiv.2104.08663

3. Bai Y., Li X., Wang G., Zhang C., Shang L., Xu J., Wang Z., Wang F., Liu Q. SparTerm: Learning term-based sparse representation for fast text retrieval // arXiv. 2020. arXiv:2010.00768. https://doi.org/10.48550/arXiv.2010.00768
4. Devlin J., Chang M.W., Lee K., Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding // arXiv. 2018. arXiv:1810.04805v2. https://doi.org/10.48550/arXiv.1810.04805
5. Formal T., Piwowarski B., Clinchant S. SPLADE: Sparse lexical and expansion model for first stage ranking // Proc. of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2021. P. 2288–2292. https://doi.org/10.1145/3404835.3463098
6. Santhanam K., Khattab O., Saad-Falcon J., Potts C., Zaharia M. ColBERTv2: Effective and efficient retrieval via lightweight late interaction // Proc. of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2022. P. 3715–3734. https://doi.org/10.18653/v1/2022.naacl-main.272
7. Johnson J., Douze M., Jégou H. Billion-scale similarity search with GPUs // IEEE Transactions on Big Data. 2021. V. 7. N 3. P. 535–547. https://doi.org/10.1109/tbdata.2019.2921572
8. Jégou H., Douze M., Schmid C. Product quantization for nearest neighbor search // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2011. V. 33. N 1. P. 117–128. https://doi.org/10.1109/tpami.2010.57
9. Kong W., Dudek J.M., Li C., Zhang M., Bendersky M. SparseEmbed: Learning sparse lexical representations with contextual embeddings for retrieval // Proc. of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2023. P. 2399–2403. https://doi.org/10.1145/3539618.3592065
10. Tiancheng Z., Lu X., Lee K. SPARTA: Efficient Open-Domain Question Answering via Sparse Transformer Matching Retrieval // Proc. of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2021. P. 565–575. https://doi.org/10.18653/v1/2021.naacl-main.47
11. Dobrynin V., Abramovich R., Platonov A. Building a full-text search index using “Transformer” neural network // Proc. of the 2023 IEEE 17th International Conference on Application of Information and Communication Technologies (AICT). 2023. P. 1–5. https://doi.org/10.1109/aict59525.2023.10313200
12. Malkov Y.A., Yashunin D.A. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2020. V. 42. N 4. P. 824–836. https://doi.org/10.1109/tpami.2018.2889473
13. Arthur D., Vassilvitskii S. k-means++: the advantages of careful seeding // Proc. of the 18th ACM-SIAM Symposium on Discrete Algorithms Mathematics. 2007. P. 1027–1035.
14. Bajaj P., Campos D., Craswell N., Deng L., Gao J., Liu X., Majumder R., McNamara A., Mitra B., Nguyen T., Rosenberg M., Song X., Stoica A, Tiwary S., Wang T. MS MARCO: a human generated MAchine Reading COmprehension dataset // arXiv. 2016. arXiv:1611.09268. https://doi.org/10.48550/arXiv.1611.09268
15. Thakur N., Nils R., Rücklé A., Srivastava A., Gurevych I. BEIR: A heterogenous benchmark for zero-shot evaluation of information retrieval models // arXiv. 2021. arXiv:2104.08663. https://doi.org/10.48550/arXiv.2104.08663

Authors

Viacheslav Yu. Dobrynin — PhD Student, ITMO University, Saint Petersburg, 197101, Russian Federation; Senior Developer, Shift Lab LTD, London, W3 7XS, Great Britain, sc 57223099701, https://orcid.org/0009-0004-3056-8403, vidobrynin@itmo.ru

Roman K. Abramovich — PhD Student, ITMO University, Saint Petersburg, 197101, Russian Federation; Senior Developer, Payler Ltd, London, E14 4QA, Great Britain, sc 58759320100, https://orcid.org/0009-0005-5397-2772, asmetliness24237@gmail.com

Alexey V. Platonov — PhD, Associate Professor, ITMO University, Saint Petersburg, 197101, Russian Federation, sc 57197736275, https://orcid.org/0000-0002-8485-1296, avplatonov@itmo.ru

Авторы

Добрынин Вячеслав Юрьевич — аспирант, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация; старший разработчик, Shift Lab LTD, Лондон, W3 7XS, Великобритания, sc 57223099701, https://orcid.org/0009-0004-3056-8403, vidobrynin@itmo.ru

Абрамович Роман Константинович — аспирант, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация; старший разработчик, Payler Ltd, Лондон, E14 4QA, Великобритания, sc 58759320100, https://orcid.org/0009-0005-5397-2772, asmetliness24237@gmail.com

Платонов Алексей Владимирович — кандидат технических наук, доцент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, sc 57197736275, https://orcid.org/0000-0002-8485-1296, avplatonov@itmo.ru

Received 31.10.2024
Approved after reviewing 10.12.2024
Accepted 22.01.2025

Статья поступила в редакцию 31.10.2024
Одобрена после рецензирования 10.12.2024
Принята к печати 22.01.2025