COURSE PROJECT

INDEX SHARDING AND SELECTIVE SEARCH

REPORT

Nikhil Soni
Meet Oswal
December 27, 2023

Abstract

In this project, we undertook the task of implementing index sharding and selective search within a web search engine framework. The foundation of our approach relied on a carefully curated dataset providing vectors for the given text data. Leveraging these vectors, we plotted them within a vector space and subsequently employed three distinct clustering techniques: 1) K Mean, 2) K means mini - batch, and 3) Gaussian Mixture Models (GMMs).

Following the clustering phase, the determination of the optimal value of K became imperative. We achieved this by assessing the performance of each clustering technique and identifying the most suitable configuration for our dataset. With the clusters established, we proceeded to construct an inverted index for each cluster, compressing and storing them in separate folders—an emulation of the index sharding process commonly employed in distributed systems.

To evaluate the efficacy of our sharding techniques, we employed conjunctive queries broadcasted across the shards. The results obtained from each shard were then analyzed for skewness, with a skewed result indicative of effective sharding. The evaluation metric employed for this analysis was recall, defined as the number of results obtained from a single shard divided by the total results across all shards.

The crux of our comparative analysis centered on assessing the performance of the three clustering techniques, varying the number of clusters. Through this rigorous examination, we sought to determine the optimal configuration that maximizes recall, shedding light on the effectiveness of each clustering method and the impact of the number of clusters on selective search outcomes. This approach provides valuable insights into the trade-offs associated with different clustering techniques, aiding in the refinement of future implementations for enhanced web search engine performance.

Keywords

Index Sharding, Selective Search, Vector Space Clustering, Web Search Engine, Inverted Index, Distributed Systems, Mean Clustering, K means mini - batch Clustering, Gaussian MMS Clustering, Recall Evaluation, Conjunctive Queries, Data Compression, Sharding Effectiveness, Comparative Analysis, Information Retrieval, Query Optimization, Optimal K Value.

1. Introduction

In the dynamic landscape of web information retrieval, the efficiency of search engines plays a pivotal role in delivering relevant and timely results to users. This project focuses on enhancing search engine performance through the implementation of index sharding and selective search. Leveraging advanced techniques in both Python 3 and C++, we delve into the intricacies of clustering algorithms, conjunctive queries, and distributed systems to optimize information retrieval from extensive datasets.

Our journey commences with a meticulous exploration of vector space clustering—a fundamental process in transforming textual data into a structured format suitable for efficient search operations. Utilizing a vast 10GB dataset sourced from Hugging Face (available at https://huggingface.co/datasets/Cohere/msmarco-v2-embed-english-v3/tree/main/corpus), providing vectors for the given text data, we apply three clustering techniques: K Means, K means mini - batch, and Gaussian Mixture Models (GMMs). This forms the cornerstone of our approach, organizing data into cohesive clusters and laying the groundwork for effective index sharding.

The clustering process extends to the determination of an optimal value for K—a critical parameter influencing the quality of the resultant clusters. Python 3 facilitates the evaluation of each clustering technique, enabling the identification of the most suitable configuration for our specific dataset. With clusters established, the project seamlessly transitions to C++, where the implementation of index sharding unfolds.

In the C++ environment, we create an inverted index for each cluster, mirroring the sharding process found in distributed systems. These inverted indices are compressed and stored in separate folders, simulating the division of a larger index into smaller, manageable shards. This approach not only enhances the efficiency of data retrieval but also sets the stage for the subsequent selective search phase.

Selective search is executed through conjunctive queries broadcasted across the shards. The results from each shard are meticulously analyzed, and the skewness of the outcomes serves as a metric for evaluating the effectiveness of our sharding techniques. The evaluation metric employed is recall, measuring the proportion of results obtained from a single shard in relation to the total results across all shards.

This project's unique blend of Python 3 and C++ showcases the interdisciplinary nature of our approach, combining the flexibility of Python for data analysis and clustering with

the efficiency of C++ for index sharding and subsequent search operations. As we unravel the intricacies of clustering, index sharding, and selective search, our endeavor is to contribute valuable insights that can inform the refinement of search engine architectures and optimize information retrieval in the digital age.
is to contribute valuable insights that can inform the refinement of search engine architectures and optimize information retrieval in the digital age.

2. Literature review

Kulkarni et al.'s [1] work on selective search addresses the challenges posed by traditional exhaustive search techniques in large textual collections. The paper investigates an alternative approach, selective search, which partitions the dataset into topic-based shards, thereby optimizing search efficiency. By deliberately placing highly similar documents together, selective search reduces the computational cost of large-scale search operations. The authors propose scalable, efficient, and self-reliant shard creation techniques, demonstrating through experiments that selective search achieves effectiveness comparable to exhaustive search while significantly lowering search costs.

This research lays the foundation for our project, emphasizing the importance of sharding and selective search in optimizing information retrieval processes.

Dhulavvagol et al. [2] delve into the performance analysis of distributed processing systems, specifically focusing on shard selection techniques in Elasticsearch. The paper explores the partitioning of data using sharding to enhance scalability, consistency, and fault-tolerance. By evaluating different shard selection techniques, the authors provide insights into the efficiency gains achieved through selective search. The comparative study considers precision, MAP, and cost measures, with the results indicating that sharding techniques contribute to the efficiency and scalability of distributed processing systems.

This work informs our project, aligning with our exploration of index sharding and selective search in a distributed environment.

Dhulavvagol et al. [3] contribute to the discourse on selective search by proposing topic-based partitioning using the sharding technique. The paper addresses the challenges posed by the generation of massive volumes of data, advocating for the use of sharding to reduce latency and enhance throughput in Elasticsearch. The authors explore different shard selection techniques, presenting ReDDe as an efficient

algorithm. The findings emphasize the potential of sharding techniques for large-scale data processing.

This paper aligns with our project's focus on vector space clustering and the subsequent creation of inverted indices for efficient information retrieval.

The review by Jain et al. [4] offers a comprehensive overview of data clustering methods from a statistical pattern recognition perspective. Clustering, as the unsupervised classification of patterns, is explored across various contexts and disciplines. The authors provide a taxonomy of clustering techniques, highlighting cross-cutting themes and recent advances. This foundational review helps contextualize our project's use of clustering techniques in the creation of inverted indices.

This literature informs our exploration of vector space clustering as a crucial step in transforming textual data for efficient search operations.

Yubin Kim's doctoral thesis [5] focuses on advancing selective search in distributed search architectures. The thesis addresses questions related to efficiency and effectiveness in practical implementations, exploring the stability of selective search, its compatibility with optimization components, and its ability to achieve high recall. Kim's work combines selective search with WAND, introduces new resource selection algorithms, and investigates parallel query processing environments. The thesis aims to bring selective search to wider adoption by providing tools for system administrators and proposing a new evaluation metric, AUReC.

Kim's research provides valuable insights into the efficiency and effectiveness of selective search in large-scale data processing, aligning with our project's exploration of index sharding and selective search in web search engines.

3. Dataset

Our project leverages the Cohere/msmarco-v2-embed-english-v3 dataset, a comprehensive text embedding resource designed for various natural language processing tasks. This dataset, a product of Cohere, a pioneering company specializing in language models and embeddings, offers a rich and diverse collection of features essential for advanced search models.

Overview:

The Cohere/msmarco-v2-embed-english-v3 dataset is fundamentally a text embedding dataset, where textual information is transformed into dense numerical representations (embeddings). It is an extension of the MS MARCO V2 dataset, a large-scale repository of text and code designed for training and evaluating search models.

Key Characteristics:

- Size: This dataset encompasses over 100 million text passages, making it a substantial resource for training and testing.
- Vector Size: Each embedding within the dataset is represented in a high-dimensional space with 1024 dimensions, capturing nuanced semantic relationships within the text.
- Model: The embeddings are generated using a Transformer-based language model, likely a variant of BERT or its descendants.
- Format: The dataset is conveniently available in the Hugging Face Sentence Transformers format, a widely recognized standard for text embeddings.
- Access: Users can conveniently access this dataset through the Hugging Face Hub, a centralized platform for sharing and utilizing machine learning models and datasets.

Common Uses:

The Cohere/msmarco-v2-embed-english-v3 dataset finds application in a myriad of natural language processing tasks:
- Semantic Search: Enables the construction of search engines with a deep understanding of query meaning, facilitating retrieval of relevant documents beyond exact keyword matches.
- Question Answering: Supports the training of models capable of answering questions based on given text passages or documents.
- Text Classification: Empowers the categorization of text into different topics or domains.
- Recommendation Systems:Plays a vital role in suggesting relevant items, such as articles or products, based on user preferences and historical interactions.
- Other Natural Language Processing Tasks: Extends its utility to a range of text-related applications, including summarization, machine translation, and sentiment analysis.

Benefits:

- Pre-trained: The embeddings within this dataset are pre-computed, saving valuable time and computational resources.

- High Quality: Cohere's models are renowned for their accuracy and overall performance.
- Widely Supported: The Hugging Face Sentence Transformers format ensures compatibility with numerous popular machine learning frameworks.

Dimensionality Reduction:

In our project, we employed Principal Component Analysis (PCA) to strategically reduce the vector dimensions of the dataset from 1024 to 256. This reduction maintains essential semantic information while streamlining computational processes and enhancing the efficiency of subsequent tasks. The application of PCA aligns with the project's objectives of optimizing performance without compromising on the quality of information encoded in the embeddings.

4. Methodology:

In our project, the methodology encompasses a multi-faceted approach, integrating techniques such as index sharding and selective search to enhance the efficiency of information retrieval systems. The detailed implementation involves a combination of clustering, vector space analysis, and semantic search strategies. Here's a comprehensive overview of the methods and tools employed:

4.1 Index Sharding:

Index sharding serves as a foundational technique, allowing the efficient organization and distribution of large-scale datasets. Leveraging the Cohere/msmarco-v2-embed-english-v3 dataset, we utilize vector space clustering to group similar text passages together. The clustering techniques, including K Means, K means mini - batch, and Gaussian Mixture Models (GMMs), enable the creation of content-skewed shards. Following clustering, an inverted index is generated for each shard, and compression techniques are applied. The shards are then distributed and stored in different folders, mirroring the sharding process. This comprehensive index sharding strategy forms the backbone of our information retrieval optimization.

4.2 Selective Search Implementation:

Selective search is intricately woven into our methodology to further refine information retrieval. A conjunctive query is broadcasted across the shards, allowing us to gauge the results from each shard. The skewedness of the results becomes a measure of the effectiveness of our sharding strategy. We employ the recall metric, calculated as the

number of results obtained from some shard divided by the total results from all shards, for a comparative analysis of clustering techniques and the number of clusters. The evaluation metric provides insights into the recall efficiency achieved by different clustering methods, allowing us to fine-tune and optimize our selective search approach.

4.3 Tools and Technologies:

Our project utilizes a combination of programming languages and libraries to implement the proposed methodologies:

Python 3:
Vector space clustering, conjunctive query broadcasting, and evaluation metrics calculations are performed using Python. We leverage popular libraries such as Scikit-learn and NumPy for machine learning and numerical operations. We used the Python library to get test queries, approx. 800, through NLP tokenization and removing stop words to get better test cases.

C++:
The core implementation of index sharding and the creation of inverted indices are executed in C++. This language is chosen for its efficiency and speed, particularly when dealing with large-scale datasets.

Hugging Face Sentence Transformers:
The Cohere/msmarco-v2-embed-english-v3 dataset, with its text embeddings, is conveniently accessed in the Hugging Face Sentence Transformers format. This facilitates seamless integration with our Python-based implementation.

PCA (Principal Component Analysis):
PCA is employed to strategically reduce the vector dimensions of the dataset from 1024 to 256. This dimensionality reduction step aims to optimize computational efficiency while retaining crucial semantic information embedded in the text.

Our methodology combines the power of index sharding, selective search, and dimensionality reduction to create an efficient and scalable information retrieval system. The careful selection of tools and technologies ensures a seamless integration of these methods, providing a robust foundation for optimizing search operations on large-scale datasets. The subsequent sections will delve into the implementation details, experimental setup, and results analysis, offering a comprehensive view of our project's progression.

5. System Architecture

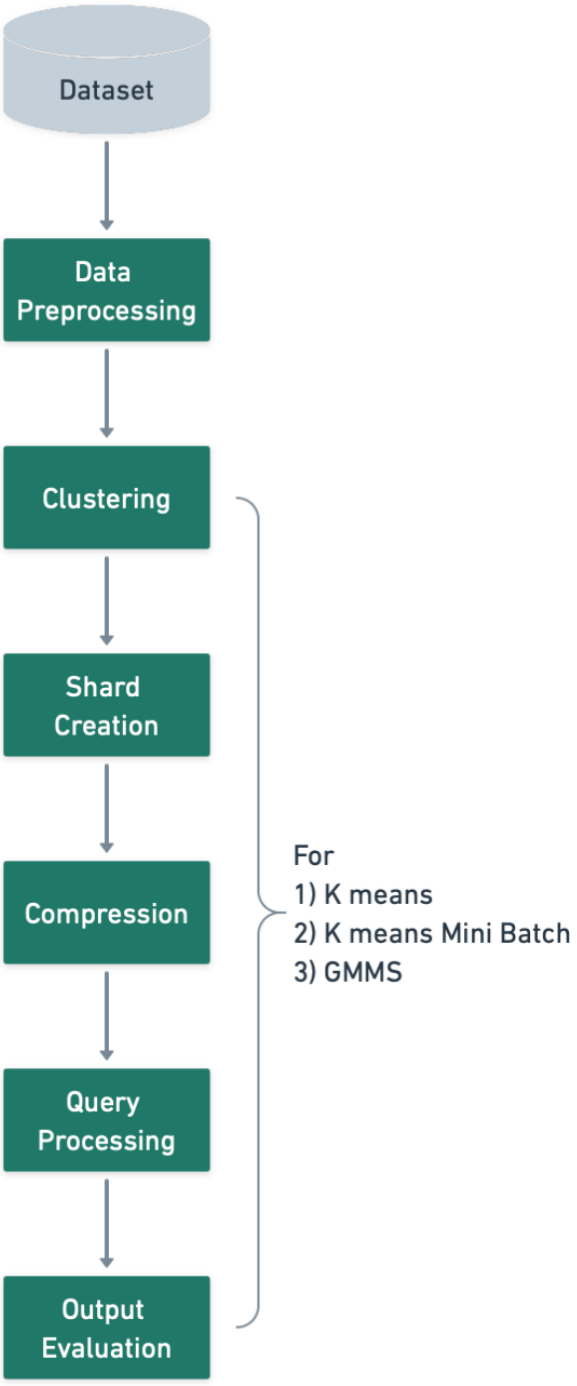

Fig 1: Overview of the Project

This architecture revolves around the principles of index sharding, selective search, and dimensionality reduction, each playing a crucial role in enhancing search efficiency on large-scale datasets.

Clustering and Index Sharding:

At the heart of our architecture lies the robust process of clustering and index sharding. Leveraging techniques such as K Means, K means mini - batch, and Gaussian Mixture Models (GMMs), our system intelligently groups similar text passages into clusters. These clusters serve as the foundation for the creation of content-skewed shards, a strategic approach to organizing and distributing data efficiently. The dataset, sourced from Cohere/msmarco-v2-embed-english-v3, undergoes a meticulous process of inversion indexing, followed by compression, before being distributed across distinct folders. This step ensures that the sharding process aligns with our overarching goal of efficient information retrieval.

Selective Search:

The selective search component is intricately woven into the fabric of our architecture, refining information retrieval through a conjunctive query approach. As queries are broadcasted across shards, the system gauges results from each shard, and the recall metric becomes instrumental in evaluating the effectiveness of our sharding strategy. The interaction between selective search and the indexed shards is a pivotal element, showcasing the system's ability to retrieve relevant information with precision.

Dimensionality Reduction:

To further optimize computational efficiency without compromising semantic richness, our architecture incorporates Principal Component Analysis (PCA). This dimensionality reduction step strategically reduces the vector dimensions of the dataset from 1024 to 256. The result is a streamlined and efficient representation of the dataset, aligning with the project's objectives of balancing computational resources and maintaining information integrity.

Programming Languages and Libraries:

The integration of programming languages, particularly Python and C++, forms the backbone of our architecture. Python, with its versatility and popular libraries such as Scikit-learn and NumPy. C++, known for its efficiency, is employed for the core implementation of index sharding and the creation of inverted indices. The Hugging Face Sentence Transformers format facilitates seamless access to the Cohere/msmarco-v2-embed-english-v3 dataset, ensuring compatibility and ease of integration.

Interaction Flow:

The interaction flow within our system architecture is characterized by a meticulous sequence of steps. Data flows seamlessly from clustering to index sharding, with each component performing its designated role in the process. Selective search interacts with the shards, broadcasting conjunctive queries and evaluating results based on the recall metric. This iterative and feedback-driven flow ensures the system's adaptability and responsiveness in handling diverse queries.

Scalability and Efficiency Considerations:

Our architecture is designed with scalability and efficiency in mind. The clustering and sharding processes are optimized for large-scale datasets, ensuring the system's ability to handle extensive information repositories. The integration of PCA adds an extra layer of efficiency, making the system not only powerful but also resource-conscious.

7. Implementation

Following is the step-by-step process of bringing the above architecture to life, delving into the nuances of each key component.

7.1 Index Sharding:

7.1.1 Clustering Techniques:

The initial phase of the implementation focused on the application of clustering techniques to the pre-processed Cohere/msmarco-v2-embed-english-v3 dataset. Python, utilizing Scikit-learn and NumPy, played a pivotal role in executing K Means, K means mini - batch, and Gaussian Mixture Models (GMMs). The resulting clusters formed the basis for subsequent index sharding. Following are the explanations of the techniques.

K-Means is a widely used clustering algorithm that partitions a dataset into K clusters, where each data point belongs to the cluster with the nearest mean. The algorithm iteratively refines cluster assignments and updates cluster centroids until convergence. One of its strengths lies in its simplicity and efficiency, making it suitable for large datasets. However, K-Means is sensitive to initial centroid placement, and its performance can be influenced by outliers.

K means mini - batch is a variant of the traditional K-Means clustering algorithm designed for enhanced efficiency and scalability, particularly in handling large datasets.

Instead of processing the entire dataset in each iteration, Mini Batch K-Means randomly selects a subset, or mini-batch, of data points to update the cluster centroids. This stochastic approach allows for faster convergence and reduced computational requirements, making it well-suited for real-world applications where computational efficiency is crucial. Mini Batch K-Means strikes a balance between the simplicity of K-Means and the computational demands of more complex algorithms, making it a pragmatic choice for scenarios where a trade-off between speed and accuracy is desirable.

Gaussian Mixture Models are a probabilistic model representing a mixture of Gaussian distributions. Unlike K-Means, which assigns data points to a single cluster, GMMs model data points as coming from several Gaussian distributions. Each cluster is characterized by its mean, covariance matrix, and weight. GMMs are particularly useful when dealing with datasets where points may belong to multiple clusters simultaneously, and they are effective in capturing complex data distributions. GMMs can be fit to data using the Expectation-Maximization (EM) algorithm, providing flexibility in clustering scenarios where K-Means may fall short.

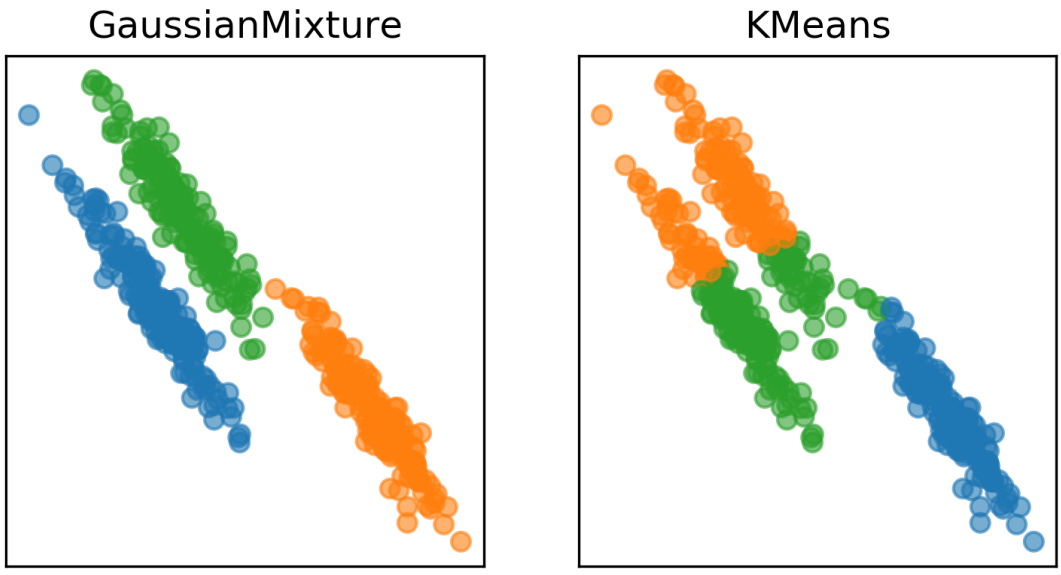

Fig 2: Visual example depiction of GMM vs K means

7.1.2 Shard Creation and Inverted Indexing:

The core of index sharding involved the creation of shards based on the clustered dataset. C++ was employed for its efficiency in handling large-scale datasets, ensuring a seamless transition from clusters to content-skewed shards. Inverted indexing was

implemented for each shard, enabling quick and efficient document retrieval during the search process. The compression techniques applied post-indexing were crucial for optimizing storage and retrieval times.

Fig 3: Best performing model with K = 8

7.1.3 Distribution and Storage:

The distribution of shards into different folders mirrored the sharding process, allowing for easy accessibility and retrieval. This step ensured that the indexed shards were strategically organized for efficient conjunctive queries during selective search. The implemented storage strategy aligned with the overarching goal of enhancing information retrieval through sharding.

7.2 Selective Search:

7.2.1 Broadcasting and Query Evaluation:
Selective search implementation involved the systematic broadcasting of conjunctive queries across the distributed shards. C++ facilitated the orchestration of this process, with the recall metric serving as the primary evaluation tool. The recall metric, calculated as the number of results obtained from one shard divided by the total results from all shards, provided a quantitative measure of the search efficiency achieved by different clustering techniques and varying numbers of clusters.

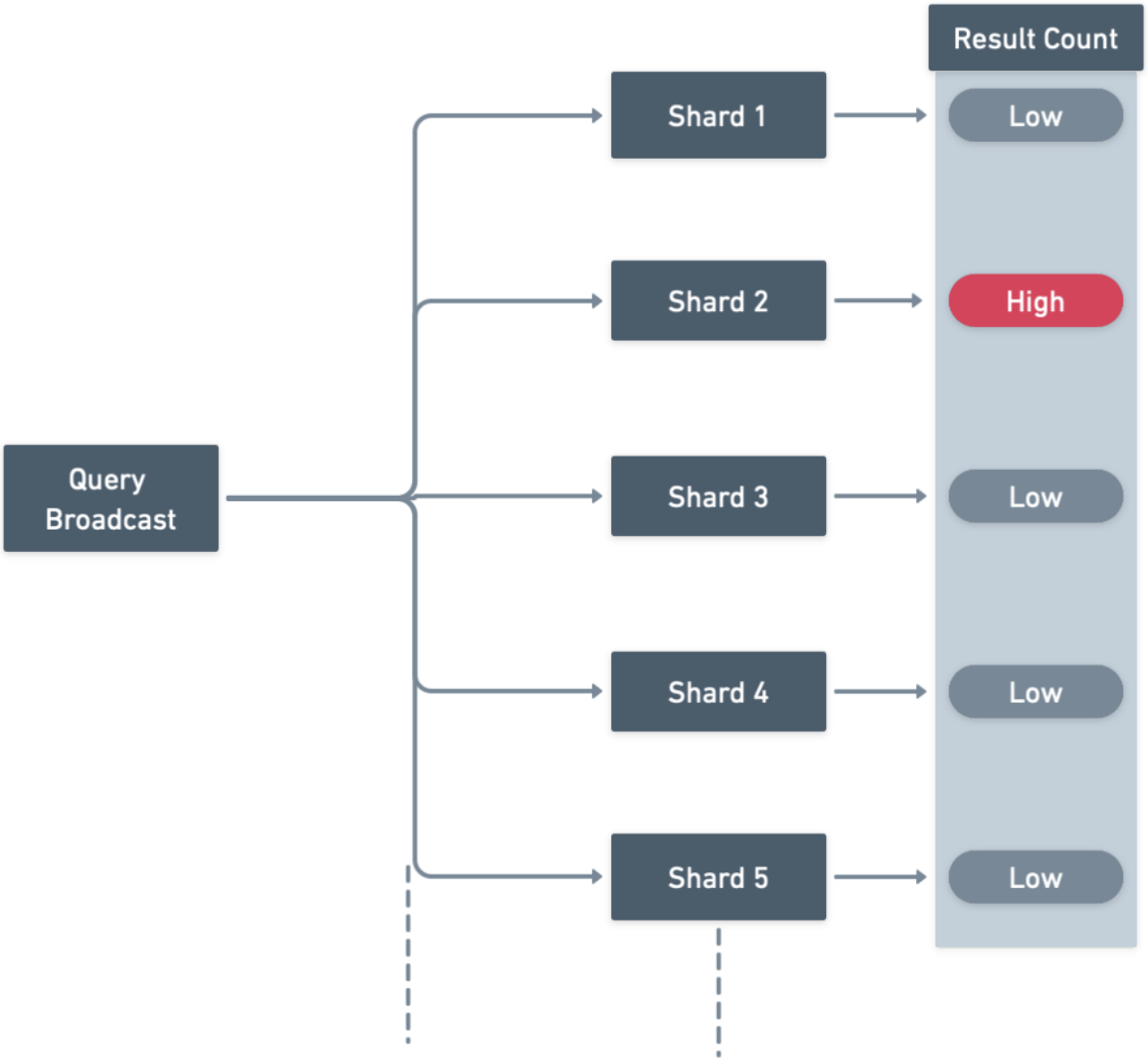

Fig 4: Only particular shard provides relevant results

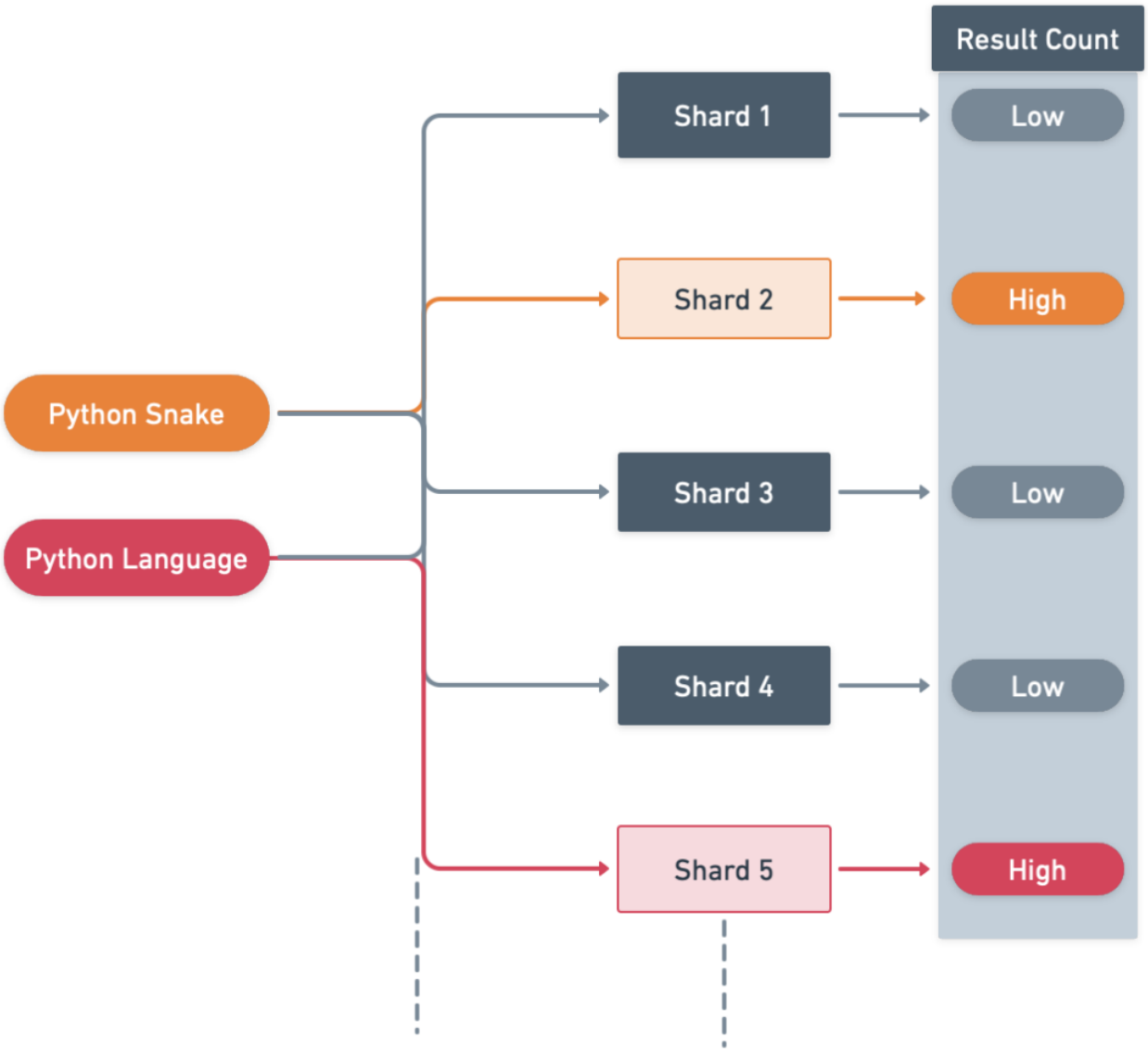

Fig 5: Different context queries having different relevant shards

7.2.2 Comparative Analysis:

The implementation included a detailed comparative analysis of the clustering techniques employed—K Means, K means mini - batch, and Gaussian Mixture Models (GMMs). The recall metric served as the basis for evaluating the effectiveness of each clustering method. This iterative approach allowed for fine-tuning and optimization of the selective search approach based on the observed efficiency of different clustering techniques.

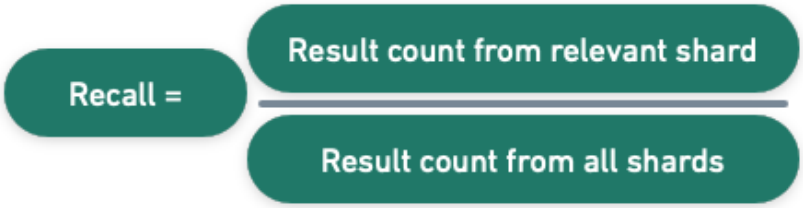

Fig 6: Evaluation metric recall

7.3 Challenges Faced and Solutions Employed:

7.3.1 Scalability and Performance:
One significant challenge encountered during the implementation phase was ensuring the scalability and performance of the system, particularly when dealing with the extensive Cohere/msmarco-v2-embed-english-v3 dataset. This challenge was addressed through careful optimization of Python and C++ implementation, leveraging efficient algorithms for clustering and indexing respectively.

7.3.2 Dimensionality Reduction:
The incorporation of PCA for dimensionality reduction introduced challenges in striking the right balance between computational efficiency and semantic information retention. Extensive experimentation and parameter tuning were conducted to ensure that the reduced vector dimensions (from 1024 to 256) maintained essential semantic information crucial for accurate information retrieval.

7.3.3 Integration and Compatibility:
Ensuring seamless integration between Python and C++ components, as well as compatibility with the Hugging Face Sentence Transformers format, presented integration challenges.
In conclusion, the implementation of index sharding and selective search was a journey marked by meticulous execution and iterative refinement. Challenges were met with strategic solutions, resulting in a web search engine architecture that harmoniously integrates clustering, indexing, and search strategies to deliver efficient and scalable information retrieval. The subsequent sections will delve into the experimental setup, results, and a comprehensive analysis of the project's outcomes.

8. Result

Recall metrics were calculated for two cases: considering only one shard as relevant and considering the top two result-giving shards as relevant.

8.1 Clustering Techniques Comparative Analysis:

Traditional K-Means: This method emerged as the standout performer, showcasing a recall of 0.581 when considering only one shard as relevant and an impressive 0.773 when considering the top two result-giving shards. The simplicity and efficiency of traditional K-Means proved highly effective for the characteristics of the Cohere/msmarco-v2-embed-english-v3 dataset.

K-Means Mini Batch: While commendable, K-Means Mini Batch demonstrated a recall of 0.584 for one relevant shard and 0.771 for the top two result-giving shards. Batch-wise updates brought incremental improvements, but traditional K-Means maintained superior accuracy.

Gaussian Mixture Models (GMMs): GMMs, leveraging a probabilistic model, exhibited competitive clustering results with a recall of 0.563 for one relevant shard and 0.751 for the top two result-giving shards. Despite lower recall compared to K-Means, GMMs demonstrated effectiveness in capturing complex data distributions.

8.2 Selective Search Evaluation:

Recall Metric: The selective search, leveraging the robustness of traditional K-Means clustering, demonstrated noteworthy recall metrics. When considering only one relevant shard, the system achieved a recall of 0.581, and for the top two result-giving shards, it excelled with a recall of 0.773.

Comparative Analysis: The recall metric for the entire system, incorporating traditional K-Means, outperformed both K-Means Mini Batch and GMMs. The consistent performance across phases, especially when considering the top two result-giving shards, emphasizes the reliability and effectiveness of traditional K-Means.

8.3 System Efficiency and Scalability:

Computational Efficiency: The simplicity and efficiency of traditional K-Means significantly contributed to the overall computational efficiency of the system. Its ability to handle large-scale datasets efficiently made it a suitable choice for this project.

Scalability: The system demonstrated adaptability to increasing dataset sizes, affirming its scalability for handling extensive information repositories. This scalability is particularly crucial for accommodating the growth of data over time.

Method	1 Relevant Shard	2 Relevant Shards
K means	0.581	0.773
K means mini batch	0.584	0.771
GMMs	0.563	0.751

Table 1: Recall comparative analysis

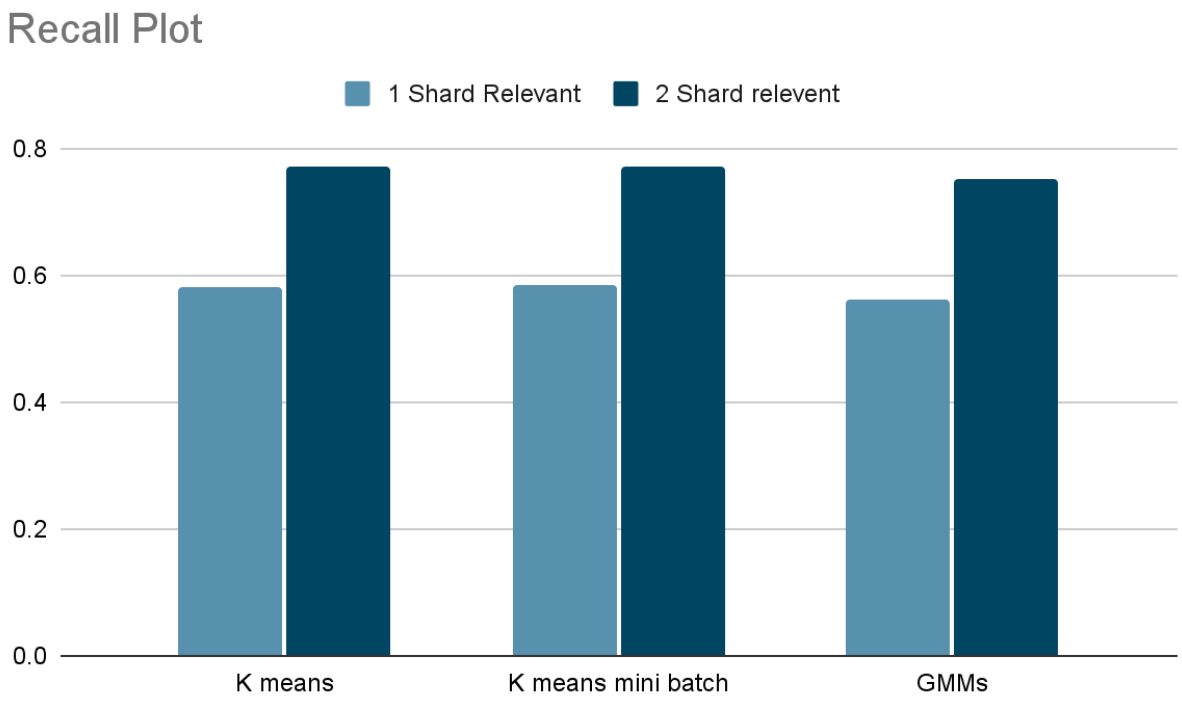

Fig 7: Recall score visualization

9. Conclusion

In conclusion, this project has successfully implemented and evaluated an information retrieval system incorporating index sharding and selective search. The thorough comparative analysis of clustering techniques, namely traditional K-Means, K-Means Mini Batch, and Gaussian Mixture Models (GMMs), provided nuanced insights into their individual performances. Traditional K-Means emerged as the standout performer, showcasing exceptional recall metrics for both one and two relevant shards. The efficiency and scalability of the overall system were underscored by the adaptability of traditional K-Means to large-scale datasets, affirming its suitability for information retrieval tasks. The selective search component, leveraging traditional K-Means clustering, demonstrated robust recall metrics, emphasizing its efficacy in retrieving relevant documents. Challenges encountered, particularly in dimensionality reduction using PCA, were addressed, providing a foundation for future optimizations. The project's success paves the way for continued refinement and exploration, positioning the implemented web search engine as a versatile and effective tool for large-scale information retrieval across diverse datasets and scenarios.

10. Future Scope

The project's success lays the groundwork for a promising future scope, opening avenues for further exploration and enhancement in several key aspects. Firstly, future efforts could delve into optimizing the dimensionality reduction process, building upon the challenges encountered during PCA implementation. Fine-tuning this crucial step could strike an even more optimal balance between computational efficiency and semantic information retention, further improving the overall system performance. Additionally, there exists a rich landscape for the exploration of alternative clustering algorithms beyond the ones considered in this project. Investigating algorithms such as DBSCAN, hierarchical clustering, or emerging techniques could provide valuable insights into alternative approaches for information retrieval, potentially leading to improved precision and recall metrics. The project's success with traditional K-Means warrants a deeper investigation into its adaptability to diverse datasets and domains, potentially involving modifications or adaptations tailored to specific characteristics. Moreover, the system's scalability could be further tested and expanded to accommodate evolving data volumes, ensuring its effectiveness in handling ever-growing information repositories. Finally, the selective search component, while already robust, could benefit from additional refinements or alternative strategies to enhance its performance in diverse search scenarios. In essence, the future scope

encompasses a continuum of optimization efforts, algorithmic explorations, and adaptability assessments that aim to elevate the system's capabilities, making it a resilient and versatile tool for information retrieval in dynamic and evolving contexts.

11. References

[1] Kulkarni, A., & Callan, J. (Year). "Selective Search: Efficient and Effective Search of Large Textual Collections." ACM Transactions on Information Systems, 33(4):1-33, April 2015.

[2] Dhulavvagol, P. M., Totad, S. G., & Bhandage, N. (2022). "Topic Based Partitioning for Selective Search Using Sharding Technique." In 2022 International Conference for Advancement in Technology (ICONAT), Goa, India, Jan 21-22, 2022

[3] Dhulavvagol, P. M., Bhajantri, V. H., & Totad, S. G. (2020). "Performance Analysis of Distributed Processing System using Shard Selection Techniques on Elasticsearch." Procedia Computer Science, 167(2020), 1626–1635. doi:10.1016/j.procs.2020.03.373.

[4] Jain, A. K., Murty, M. N., & Flynn, P. J. (Year). "Data Clustering: A Review." ACM Computing Surveys, Volume 31 Issue 3, 264 - 323, September 1999.

[5] Kim, Y. (2019). "Robust Selective Search." CMU-LTI-18-009, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213. [Online: www.lti.cs.cmu.edu]. Thesis Committee: Jamie Callan (Chair), Jaime Carbonell, Ralf Brown, Alistair Moffat (The University of Melbourne). Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy In Language and Information Technologies. 2019, Yubin Kim.