An evaluation of clustering algorithms for IVF vector indexes

Yash Chaudhari, Varshita Rodda, Zhiwei Song, Shahab Ahmad Khan from *Byte Brigade*

2024.05.04

1 Introduction

The proliferation of AI applications, particularly those employing large language models (LLMs), generative AI, and semantic search, hinges on the efficient processing of massive datasets. These applications rely heavily on vector embeddings, a sophisticated data representation that encodes semantic information crucial for AI understanding. However, the high dimensionality of embeddings, with numerous attributes or features, presents a challenge in managing and searching this data. Therefore, scalable and efficient algorithms are needed to help query the large database containing millions of records within a short time frame [5].

Inverted File Indexing (IVF) [16] emerges as a powerful solution for accelerating similarity search in high-dimensional spaces. IVF drastically reduces the search scope by leveraging clustering algorithms, leading to significant performance improvements. IVF operates by grouping similar vectors into clusters. During the indexing phase, a clustering algorithm like k-means partitions the vector space. Each cluster is then assigned a centroid, a representative vector at its center. When a query vector arrives, IVF first identifies the closest centroid. Subsequently, the search is confined to the specific cluster containing this closest centroid, significantly reducing the number of vectors that need to be compared. This targeted search approach translates to faster retrieval times.

IVF operates by grouping similar vectors into clusters. During the indexing phase, a clustering algorithm like k-means partitions the vector space. Each cluster is then assigned a centroid, a representative vector at its center. When a query vector arrives, IVF first identifies the closest centroid. Quality of Clustering becomes crucial and has an impact on search efficiency as follows: (1) Tightly Packed Clusters: Within a cluster, vectors should be highly similar to minimize the number of comparisons needed during the search within that cluster. Poor clustering algorithms may create ”loose” clusters with dissimilar vectors, leading to unnecessary comparisons and hampering search efficiency. (2) Balanced Clusters: Clusters should ideally have a roughly equal number of vectors. Uneven distributions can create bottlenecks. Imagine a scenario with one cluster containing a thousand vectors and another with only ten. A query landing in the smaller cluster might be retrieved quickly, but queries directed to the larger cluster would face a significant slowdown due to the sheer number of vectors present.

Therefore, selecting the optimal clustering algorithm for IVF is critical. A good algorithm will create tightly packed, balanced clusters, significantly reducing the search space and leading to faster retrieval times while maintaining accuracy [12]. We are currently considering multiple clustering algorithms such as Variants of k-means, Density-Based Clustering Algorithms, Streaming and many more.

To comprehensively evaluate the performance of different clustering algorithms in the context of IVF, we will consider the following key metrics: (1) Build Time: Measures the time taken for the algorithm to construct the clusters from the vector data. Ideally, the build process should be efficient, especially for large datasets. (2) Search Time to Recall: Evaluates the time required to identify the desired number of relevant vectors (recall) during a search query. Faster search times are desirable. (3) Quantization Error: Quantifies the loss of information incurred when representing the original high-dimensional vectors using lower-dimensional cluster centroids. Lower quantization error signifies a more faithful representation [11]. (4) Cluster Imbalance: Measures the degree of unevenness in the distribution of vectors across clusters. Balanced clusters lead to more efficient search within the IVF framework.

By analyzing these metrics, we can identify the clustering algorithm that delivers the best trade-off between efficiency, accuracy, and scalability for IVF indexing in vector embedding applications. We will focus on algorithms that demonstrate superior or equivalent scalability compared to k-means, a commonly used but potentially sub-optimal solution for large datasets [4]. Our goal is to benchmark the effect of various clustering algorithms on IVF index performance.

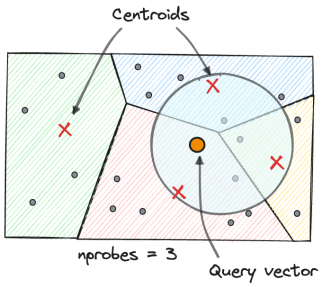

Figure 1: Voronoi Cell Structure with nprobe = 3

2 Background

Central to many AI applications lies the concept of vector embeddings. These embeddings offer a method for representing textual data in a continuous, high-dimensional space, where similar semantic entities are mapped closer together [10]. This encoding of semantic information is pivotal for enabling AI systems to effectively comprehend and process natural language.

However, dealing with high-dimensional data, such as vector embeddings, presents substantial computational hurdles. The sheer size and dimensionality of the data necessitate scalable and efficient algorithms for effective management and retrieval. In similarity search tasks, where the objective is to retrieve items most akin to a given query, traditional search methods can become impractical due to computational constraints.

One strategy to tackle this challenge involves leveraging IVF. IVF is a technique tailored to expedite similarity search in high-dimensional spaces by harnessing clustering principles [14]. In IVF, akin vectors are grouped into clusters, with each cluster associated with a centroid—a representative vector at its core. During the search process, IVF first identifies the closest centroid to the query vector and then confines the search to the cluster housing this centroid. This targeted search approach significantly reduces the number of comparisons needed and, consequently, enhances retrieval times.

The effectiveness of IVF heavily hinges on the quality of the clustering algorithm employed to partition the vector space. Clustering algorithms play a pivotal role in determining how well vectors are grouped into clusters and the distribution of vectors within each cluster. Tightly packed clusters with balanced distributions facilitate faster and more efficient search operations within the IVF framework.

Moreover, Voronoi cells, in the context of clustering algorithms, delineate regions in the vector space around each centroid. Each vector is assigned to the Voronoi cell as shown in **Figure 1** corresponding to the centroid it is closest to, effectively partitioning the space into regions

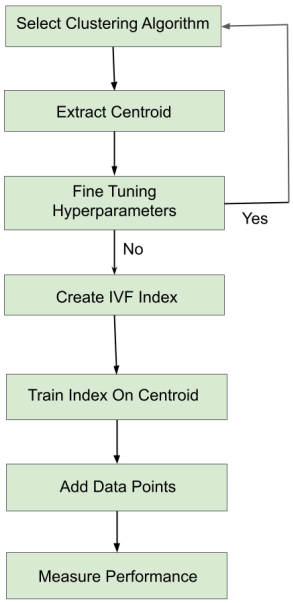

Figure 2: Approach used to build index and measure performance

associated with different clusters. This spatial partitioning aids in the organization and optimization of search operations within the IVF framework. The parameter *nprobe* determines the number of clusters (centroids) to be examined during the search process. A higher value of *nprobe* increases the number of centroids considered, potentially leading to more accurate search results but also increasing computational overhead. Finding an optimal value for *nprobe* is essential for balancing search accuracy and efficiency.

3 Design

Figure 2 outlines the procedural steps involved in constructing an IVF and evaluating various clustering algorithms. This section details the systematic approach adopted to generate the IVF index and assess the performance of clustering methods. The benchmarking process is conducted using the Scale-Invariant Feature Transform (SIFT) dataset [9], which comprises one million vectors, each consisting of 128 dimensions. Furthermore, the SIFT query dataset, composed of 10,000 128-dimensional vectors, is utilized to evaluate the search performance and quality. The clustering phase leverages established modules such as Scikit-learn [23], HDBSCAN [19], and River [20], while the indexing phase employs the Faiss module [8].

3.1 Select Clustering Algorithm

During this stage, a crucial decision is made regarding the clustering algorithm to be utilized for benchmarking purposes. The chosen algorithms, including K-Means [15][6][21], Bisecting K Means [25], Mini Batch K-Means [22], HDBScan [18], CluStream [2], and DenStream [7], represent a diverse array tailored to accommodate the nuances inherent in dataset characteristics.

Implementation of these algorithms relies on established libraries and frameworks. Specifically, the Scikit-learn library serves as the foundation for implementing K-Means and its variants, ensuring a robust and standardized approach. Additionally, HDBScan is implemented using the dedicated HDBScan library, while River, a dynamic data stream processing library, is utilized for executing stream clustering algorithms.

This deliberate selection process sets the stage for a comprehensive assessment of clustering performance, ensuring a nuanced understanding of each algorithm's strengths and limitations across varied datasets.

3.2 Centroid Extraction

In this phase, following the training of the clustering algorithm on the benchmarking dataset, we proceed to extract or estimate centroids, depending on the nature of the algorithm employed.

For centroid-based algorithms, such as K-Means and its variants, centroids are readily available post-training and can be directly stored for subsequent use. However, for density-based algorithms like HDBScan, which do not inherently produce centroids, the centroid for each cluster must be estimated. This estimation is accomplished by computing the mean of the data points belonging to each cluster.

This differentiation in centroid extraction underscores the adaptability of the process to accommodate various clustering methodologies, ensuring that the necessary centroid information is obtained regardless of the algorithm's underlying principles.

3.3 Fine Tune Hyperparameters

Upon completion of the initial clustering process, an evaluation of the results is conducted to assess the adequacy of the clustering solution. This evaluation encompasses aspects such as the optimal number of clusters and cluster balance.

Should the initial results prove satisfactory in terms of these criteria, indicating a robust clustering solution, the fine-tuning of hyperparameters may be deemed unnecessary. In such cases, the process proceeds to subsequent stages without additional hyperparameter adjustment.

However, if the evaluation reveals shortcomings in the clustering solution, indicating the need for refinement, the hyperparameters of the clustering algorithms are adjusted accordingly. This involves a meticulous process of retraining the clustering algorithms while systematically tuning the hyperparameters to optimize the clustering performance.

This adaptive approach ensures that the clustering process remains dynamic and responsive to the specific characteristics of the dataset, thereby striving for optimal clustering outcomes.

3.4 Create IVF index and Train the index on centroids

To initialize the index, centroids derived from the clustering algorithms employed in prior steps serve as pivotal elements. Using the Faiss library, we instantiate the IVF index, configuring it to accommodate the characteristics of the centroids. Subsequently, the index is trained using the centroid vectors obtained from the clustering process.

By training the IVF index on centroids, we establish Voronoi Cells within the index structure. These cells delineate regions in the vector space associated with each centroid, enabling expedited search operations by directing queries to their nearest centroid-based Voronoi Cell

3.5 Add Datapoints

In this step, we augment the dataset by incorporating additional data points sourced from the SIFT dataset. Each data point is strategically assigned to its respective Voronoi Cell within the IVF index based on the calculated L2 distance from the centroid.

3.6 Measure the performance of the index

In this phase, we conduct a comprehensive evaluation of the clustering algorithms' performance utilizing the Voronoi Structure IVF. The assessment is based on four key metrics: 1) Build Time, 2) Recall/Search Time, 3) Quantization Error and 4) Cluster Imbalance

Additionally, we analyze the performance and tradeoff between search quality and speed. This entails evaluating the efficiency of the clustering algorithms in balancing the need for accurate search results with the imperative of swift query processing

4 Evaluation

In this section, we present the experiments conducted to evaluate various clustering algorithms for IVF indexing in large-scale AI applications. We provide details on the

experiments performed, the setup used, and the key takeaways from each experiment. Additionally, we explain the experiment setup in detail to facilitate reproducibility by others.

4.1 Experiment Setup

Dataset We utilized the Scale-Invariant Feature Transform (SIFT) dataset, comprising one million vectors, each consisting of 128 dimensions. Additionally, we employed the SIFT query dataset, composed of 10,000 128-dimensional vectors, to evaluate search performance and quality.

Clustering Algorithms We evaluated several clustering algorithms, including K-Means, Bisecting K Means, Mini Batch K-Means, K-means Streaming, DBSCAN, CluStream, and Denstream. These algorithms were chosen to represent a diverse array tailored to accommodate the nuances inherent in dataset characteristics.

Implementation The clustering algorithms were implemented using established libraries and frameworks. K-Means and its variants were implemented using the Scikit-learn library, while DBSCAN was implemented using the same library. Additionally, River, a dynamic data stream processing library, was utilized to execute stream clustering algorithms(CluStream and DenStream) on CloudLab.

4.2 Experiments Conducted

Search Time We measured the time taken by each clustering algorithm to perform a similarity search using the IVF index. The goal was to assess the efficiency of each algorithm in retrieving similar vectors.

Build Time The build time for each clustering algorithm was recorded to evaluate the efficiency of constructing the IVF index. This metric provides insights into the scalability of the algorithms.

Quantization Error We calculated the quantization error for each clustering algorithm to assess the fidelity of the centroid representation. Higher quantization errors indicate potential loss of information during clustering.

Cluster Imbalance The balance of clusters formed by each clustering algorithm was analyzed to understand the distribution of vectors within clusters. We calculated the average imbalance by calculating the standard deviation on cluster sizes. Imbalanced clusters may impact search efficiency and accuracy.

4.3 Experiment Results and Takeaways

Search Time K-Means and its variants exhibited superior search times compared to density-based algorithms such as DBSCAN **(Figure 3)** and Stream-Based Algorithms such as DenStream and Stream K-Means. The ability to specify the number of clusters and return centroids contributed to their efficiency in creating Voronoi Cells in IVF. Additionally, there was a linear relationship between search time and the *nprobe* value. We also observe that as we increase the nprobe value, accuracy increases as seen in **Table 1** but search time becomes worse as seen in **Figure 3**, indicating a trade-off between accuracy and search time across nprobe. However, we can see in **Figure 3** that all the IVF indexing, irrespective of algorithms, show lower search time than traditional search methodology, showcasing the significance of IVF.

For density-based algorithms, fewer clusters are detected, and there is a higher cluster imbalance value. This implies that with a fixed value of *nprobe*, we need to search for more data points within the cluster and compare them with the query vector. This leads to a higher search time compared to the K-means algorithm and other K-mean variants as seen in **Figure 3** (For Denstream, which is the Density Streaming Algorithm, it is approximately 6-7 times slower than vanilla K-means). However, there is not much gain in accuracy when *nprobe* is 1 as seen in **Table 1**.

Model	**Accuracy Details (%)**	
	nprobe = 1	nprobe = 10
K-Means	60.80	98.30
Bisecting K-Means	53.60	97.20
MiniBatch K-Means	55.23	98.20
HDBScan	95.60	
DenStream	60.90	96.80
K-Means Streaming	59.43	98.00

Table 1: Accuracy of various algorithms under *nprobe* = 1 and *nprobe* = 10

Build Time We observed notable differences in build times among the evaluated clustering algorithms. K-means variants, including bisecting K-means, minibatch K-means, and K-means streaming, exhibited significantly lower build times compared to Vanilla K-means **(Figure 4)**. This improvement can be attributed to optimization techniques implemented in these variants. In contrast, density-based clustering algorithms, such as HDBSCAN and DenStream, generally exhibited around 1.5 to 2 times of the vanilla K-means build time. This is due to the computational overhead associated with extensive pairwise distance computations and dense region checks. Moreover, for the DenStream algorithm, the training data is in a stream fashion and processed incre-

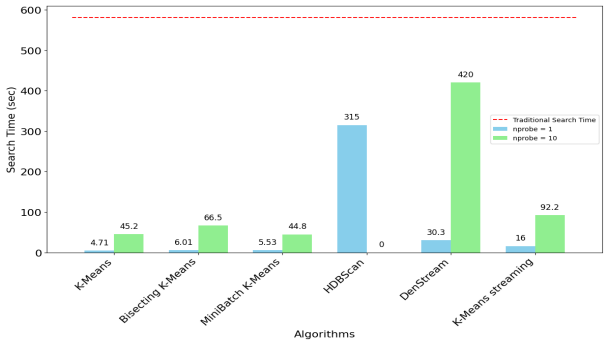

Figure 3: Search Time

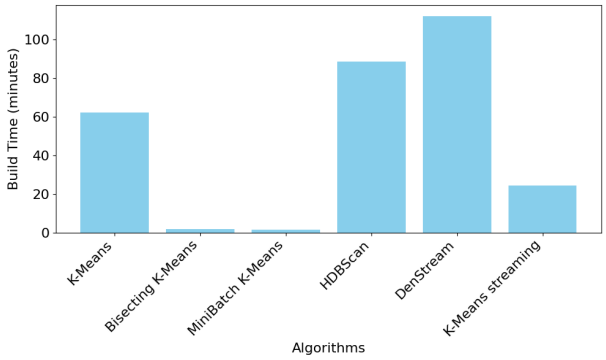

Figure 4: Build Time

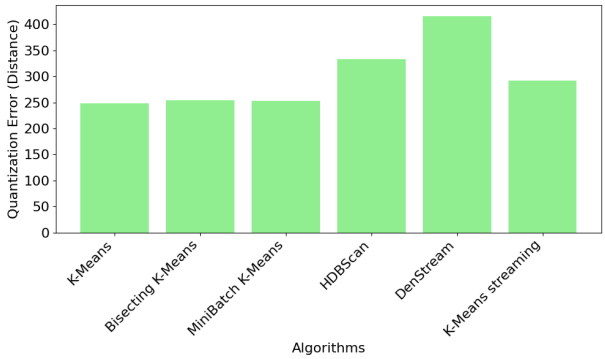

Figure 5: Quantization error

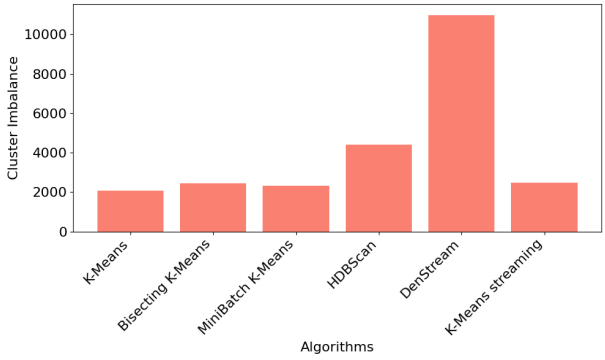

Figure 6: Cluster Imbalance

mentally, one entry at a time. This means the density area is recomputed and updated in each iterative step and becomes the bottleneck in building initial clusters.

Quantization Error Density-based clustering algorithms tend to exhibit higher quantization errors compared to K-Means (**Figure 5**). This suggests that the centroid representation in density-based clustering may not be as effective. Unlike K-Means, where the number of clusters can be specified, density-based algorithms lack this flexibility, potentially resulting in fewer clusters being formed.

Moreover, density-based algorithms do not derive centroids for clustering, necessitating their estimation. This estimation, coupled with situations where a high number of data points are grouped within a single cluster due to the density-based criteria, contributes to the higher quantization error observed. The result for the streaming algorithm is similar because DenStream uses the density-based clustering method as the underlying cluster algorithm. Similarly, K-Means Streaming uses K-Means as the underlying cluster algorithm

The higher quantization error in density-based clustering highlights the limitations of these algorithms in accurately representing the underlying data distribution. Thus, it's crucial to consider the clustering algorithm's characteristics and suitability for the dataset when performing IVF indexing for large-scale AI applications.

Cluster Imbalance We observed varying degrees of cluster imbalance across different clustering algorithms (**Figure 6**). HDBSCAN exhibited higher cluster imbalance compared to K-Means, implying potentially longer searching times, especially in scenarios with imbalanced clusters. While K-Means and its variants displayed similar cluster imbalances, slight differences may still impact recall time, particularly as the *nprobe* value increases. In density-based algorithms, we don't have a parameter to initialize the number of clusters, resulting in fewer clusters formed; for example, in the Denstream algorithm, we only have 15 clusters and HDBScan produces only 4 clusters after we compute the density area of the training data and cluster assignments for each data point. This leads to a larger cluster size distribution deviation than other methods. A high cluster imbalance will result in different search times per cluster. Therefore, it is ideal to have a lower cluster imbalance.

Note The evaluation revealed that configuring CluStream with 128 clusters resulted in a build time exceeding 7 hours. This prolonged build time underscores the computational complexity associated with maintaining micro-clusters and summarizing evolving data streams, especially with a higher number of clusters.

Upon experimentation with smaller cluster sizes and alternative hyperparameters, notable improvements were observed. For instance, reducing the number of clusters to 15 and adjusting the hyperparameters yielded a reduced build time of 153 minutes. However, this configuration still exhibited a relatively high cluster imbalance of 124117.88 and quantization error of 438.11.

Further optimization of CluStream's parameter settings, such as fine-tuning the decay factor and time gap, may be necessary to achieve a balance between build time, clustering accuracy, and cluster balance. Additionally, exploring alternative clustering algorithms or hybrid approaches could offer insights into achieving more expedited processing times while maintaining clustering accuracy.

5 Related Work

We have collected and analyzed several clustering algorithms that can achieve better time complexity equally or better than the k-means algorithm. In this section, we discussed the related research efforts on clustering algorithms and categorized them based on the vital algorithm characteristics.

5.1 K-means algorithm and its variants

The k-means algorithm [15][6][21] utilizes the expectation-maximization objective to partition the unlabeled datasets into a fixed number of clusters (usually denoted by k clusters) through an iterative process. Each step involves the calculation of clusters belonging to each node and the reassignment of the centroids based on the center location of each cluster. The algorithm will stop until the clusters converge. This method has been considered a benchmark for evaluating querying IVF indexing [4]. However, according to Wang and Su [24], through a density-based noise data detection technique, noise data are filtered from the datasets. This might improve the time performance of iterative steps in the standard k-means algorithm. Another variant is the mini-batch [22] and balanced [17] k-means algorithm. The former only uses a subset of datasets to update assigned clusters in each iteration and achieves the same objective as vanilla k-means; the latter balances the cluster size after several steps in the iterative process.

5.2 Hierarchical clustering algorithms

The most well-known algorithm is the agglomerative hierarchical clustering algorithm [1]. Each data entry is initially viewed as a single cluster, and each step merges the two closed clusters based on distance metrics. This procedure is carried out in an iterative way until only k clusters are left. Although the method seems very standard, there are many variants of the distance metrics. The distance between each pair of clusters can be the distance between the nearest points, or the distance between the centroids.

5.3 Streaming-based clustering algorithms

Stream-based clustering algorithms are designed to handle continuously arriving data streams. Unlike traditional clustering algorithms that process static datasets, stream-based algorithms incrementally update cluster representations as new data points arrive. The three discussed papers tackle data stream clustering from different angles. Zubaroğlu and Atalay [26] methodically compare existing algorithms, providing a broad perspective without introducing a new implementation. In contrast, SKDStream [13], a novel algorithm designed for time-decaying data, optimizing for real-time relevance without the weight of historical data. AI-amri et al. [3] propose a clustering method using a temporal-spatial hypercube to adapt dynamically to evolving data streams, emphasizing data point density. While the first offers an analytical overview, the latter two introduce practical solutions targeting specific challenges of real-time data analysis.

6 Future work

The insights derived from this research provide a foundation for future investigations aimed at refining our understanding of the optimal clustering algorithms for IVF indexing across various use cases. Expanding upon the findings elucidated in this report, several promising avenues emerge for further exploration and inquiry. This section outlines potential future directions, encompassing both theoretical and practical implications, which can guide subsequent research endeavors in this domain.

Multi-Modal and Heterogeneous Data: Future research endeavors should prioritize extending clustering and indexing techniques to accommodate multi-modal and heterogeneous data types, including text, images, and sensor data. This necessitates the development of integrative approaches capable of synthesizing information from diverse sources to enable comprehensive analysis and clustering of heterogeneous datasets.

Scalability and Parallelism: Addressing the scalability challenges inherent in handling large-scale datasets warrants exploration into parallel and distributed computing techniques. Investigating methods for partitioning and parallelizing index operations across multiple computing nodes is essential for achieving scalability and efficiently managing the burgeoning volumes of data. Current implementations are not leveraging parallelism effectively, presenting an opportunity for optimization.

Fine-Tuning Density-Based Clustering: Given the inherent limitations of density-based clustering algorithms, such as HDBScan, in forming a limited number of clusters compared to centroid-based methods like K-Means, further fine-tuning of parameters is imperative. Particularly on datasets of significant size, such as the SIFT dataset with one million vectors, efforts should be directed toward optimizing parameters to foster the formation of more clusters, thereby enhancing search efficiency.

Development of Efficient Clustering Algorithms: The need for more efficient clustering algorithms tailored for high-dimensional data is paramount. Current implementations in widely used libraries like Scikit-learn may not be optimized for such datasets, resulting in sluggish convergence and suboptimal performance. Exploring the creation of custom clustering algorithms optimized for high-dimensional data could lead to faster and more effective clustering solutions.

Benchmarking on Higher Dimensional Datasets: Extending the benchmarking analysis to higher dimensional datasets presents an intriguing avenue for future research. While the current evaluation focuses on a 128-dimensional dataset, investigating whether the observed results hold true in higher dimensional spaces could provide valuable insights into the robustness and scalability of the clustering algorithms under consideration. This exploration would enhance the applicability of the findings across a broader range of real-world scenarios.

7 Conclusion

In this study, we conducted a comprehensive evaluation of various clustering algorithms for IVF in the context of large-scale AI applications. The evaluation revealed the superiority of K-Means and its variants in terms of search time efficiency compared to density-based algorithms like DBSCAN and stream-based algorithms like DenStream. This can be attributed to the ability of K-Means to specify the desired number of clusters and return centroids, facilitating the creation of efficient Voronoi cells within the IVF index. However, density-based algorithms exhibited higher quantization errors and cluster imbalances, indicating potential limitations in accurately representing the underlying data distribution.

In conclusion, this study underscores the importance of selecting appropriate clustering algorithms for IVF indexing in large-scale AI applications. The findings highlight the trade-offs between efficiency, accuracy, and scalability, emphasizing the need for careful consideration of dataset characteristics and application requirements when choosing a clustering algorithm for optimal performance.

8 Contribution

Yash Chaudhari - KMeans and Variants, Metric Design
Varshita Rodda - Streaming(K-Means and CluStream)
Shahab Ahmad Khan - Density Based (HDBSCAN)
Zhiwei Song - Streaming Based (DenStream)
Everybody has contributed to making poster and report

References

[1] Algorithms for hierarchical clustering: an overview - murtagh - 2012 - WIREs data mining and knowledge discovery - wiley online library.

[2] C. C. Aggarwal, P. S. Yu, J. Han, and J. Wang. - a framework for clustering evolving data streams. In J.-C. Freytag, P. Lockemann, S. Abiteboul, M. Carey, P. Selinger, and A. Heuer, editors, *Proceedings 2003 VLDB Conference*, pages 81–92. Morgan Kaufmann.

[3] R. Al-amri, R. K. Murugesan, M. Almutairi, K. Munir, G. Alkawsi, and Y. Baashar. A clustering algorithm for evolving data streams using temporal spatial hyper cube. 12(13):6523. Number: 13 Publisher: Multidisciplinary Digital Publishing Institute.

[4] K. Aoyama, K. Saito, and T. Ikeda. Inverted-file k-means clustering: Performance analysis. *ArXiv*.

[5] D. Baranchuk, A. Babenko, and Y. Malkov. Revisiting the inverted indices for billion-scale approximate nearest neighbors. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, pages 209–224. Springer International Publishing.

[6] L. M. L. Cam and J. Neyman. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press. Google-Books-ID: IC4Ku_7dBFUC.

[7] F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *SDM*, 2006.

[8] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou. The faiss library. 2024.

[9] M. B. M. S. A. M. Fu, Xiping and L. Szymanski. SIFT10M. UCI Machine Learning Repository, 2016. DOI: https://doi.org/10.24432/C5S603.

[10] Z. Jing, Y. Su, Y. Han, B. Yuan, H. Xu, C. Liu, K. Chen, and M. Zhang. When large language models meet vector databases: A survey. version: 2.

[11] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. 33(1):117–128. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[12] C. Li, M. Zhang, D. G. Andersen, and Y. He. Improving approximate nearest neighbor search through learned adaptive early termination. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, pages 2539–2554. Association for Computing Machinery.

[13] H. Liu, A. Wu, M. Wei, and C.-C. Chang. SKD-Stream: a dynamic clustering algorithm on time-decaying data stream. 2022(1):102.

[14] Y. Liu, Z. Pan, L. Wang, and Y. Wang. A new fast inverted file-based algorithm for approximate nearest neighbor search without accuracy reduction. 608:613–629.

[15] S. Lloyd. Least squares quantization in PCM. 28(2):129–137. Conference Name: IEEE Transactions on Information Theory.

[16] A. K. Mahapatra and S. Biswas. Inverted indexes: Types and techniques. *International Journal of Computer Science Issues*, 8.

[17] M. I. Malinen and P. Fränti. Balanced k-means for clustering. In P. Fränti, G. Brown, M. Loog, F. Escolano, and M. Pelillo, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, Lecture Notes in Computer Science, pages 32–41. Springer.

[18] C. Malzer and M. Baum. A hybrid approach to hierarchical density-based cluster selection. In *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 223–228.

[19] L. McInnes, J. Healy, and S. Astels. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11):205, 2017.

[20] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem, et al. River: machine learning for streaming data in python. 2021.

[21] S. Na, L. Xumin, and G. Yong. Research on k-means clustering algorithm: An improved k-means clustering algorithm. In *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, pages 63–67.

[22] J. Newling and F. Fleuret. Nested mini-batch k-means. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[24] L. Wang, L. Bo, and L. Jiao. A modified k-means clustering with a density-sensitive distance metric. In G.-Y. Wang, J. F. Peters, A. Skowron, and Y. Yao, editors, *Rough Sets and Knowledge Technology*, Lecture Notes in Computer Science, pages 544–551. Springer.

[25] Y. Zhuang, Y. Mao, and X. Chen. A limited-iteration bisecting k-means for fast clustering large datasets. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 2257–2262. ISSN: 2324-9013.

[26] A. Zubaroğlu and V. Atalay. Data stream clustering: a review. *Artificial Intelligence Review*, 54(2):1201–1236.