A Comparative Study of Locality-Sensitive Hashing Techniques for Text Similarity Search

Description of the Study

Locality-sensitive hashing (LSH) is a widely popular technique used for the efficient approximation of nearest neighbor searches in high-dimensional data spaces, including text documents represented by vectors like **Term Frequency-Inverse Document Frequency (TF-IDF)**. TF-IDF is a statistical measure that evaluates word importance within a document relative to a corpus, creating high-dimensional vectors that support similarity searches. LSH allows for fast approximate searches in these high dimensional spaces by hashing similar data points into the same buckets, facilitating applications such as search engines, document clustering and recommendation systems.

Various LSH methods present trade-offs in performance metrics such as precision, query time and memory usage. Signed Random Projections (SRP) is a traditional LSH method that utilizes random hyperplanes to convert data into lower-dimensional binary hash codes, preserving the cosine similarity between data points. This enables efficient approximate similarity search, as documents with higher cosine similarity are more likely to be hashed into the same bucket. While SRP-LSH is computationally efficient and appropriate for uniformly distributed data, its random nature may struggle in capturing semantic groupings in unevenly distributed data. Conversely, K-means LSH is another strategy that integrates the K-means clustering algorithm into the LSH framework. By clustering text documents based on their TF-IDF feature vectors and using cluster assignments as hash codes, K-means LSH adapts to the distribution of the data, potentially improving retrieval accuracy by hashing semantically similar documents into the same buckets. However, K-means LSH can incur a higher memory overhead and computational complexity due to the clustering process and the need to maintain centroids. **Therefore**, **in this study, we aim to compare both K-means LSH and SRP-LSH on a dataset of 18000 documents, commonly encountered in tasks like large-scale news article clustering. Using three key metrics: precision of groupings, query times and memory usage, we evaluate how each method performs in balancing speed and accuracy. This will provide actionable insights into the applicability of each LSH method to large scale text similarity tasks.**

Literature Survey

The various forms and methods surrounding Locality-sensitive hashing (LSH) have been a hot topic for researchers in conducting efficiency tests for similarity search on large datasets. Through our research, we have consulted different viewpoints for their feedback on K-mean LSH and Signed Random Projections LSH. Through these, we aimed to understand their tradeoffs and determine which is optimal when finding similarities in text documents.

In "A Survey on Locality Sensitive Hashing Algorithms and their Applications" by Jafari, the authors analyze various LSH algorithms and their performance. They discuss how we can utilize the Signed Random Projections LSH algorithm to reduce dimensionality while preserving similarity between data points. The paper states that for this implementation, the strategy is to use random hyperplanes to hash high-dimensional vectors into binary codes. This ensures that the method is effective for highly sparse and high dimensional data, making it effective for similarity search in text documents. "Locality sensitive hashing: A comparison of hash function types and querying mechanisms" by Pauleve is another paper that emphasizes exploring the efficiency of different hash functions and querying mechanisms concerning LSH. The paper places a more primary focus on investigating K-means LSH, describing the process as using K-means clustering as an unstructured quantizer within the LSH framework. It is highlighted that the use of K-means adapts the actual data distribution and thus better captures the underlying data. This points towards a more successful grouping of similar items into common bins. In applications such as text documents with semantic similarities, this approach would be highly beneficial.

By drawing from the knowledge of these various studies, we will aim to directly compare K-means LSH and Signed Random Projections LSH to evaluate their performance and draw conclusions on the applicability of the respective strategies.

Thesis:

We hypothesize that Signed Random Projections LSH will outperform K-means LSH in accurately identifying similar text documents while maintaining computational efficiency, making it well-suited for our TF-IDF matrix.

Experimental Settings

Our study evaluates the performance of the K-means LSH and Signed Random Projections LSH (SRP-LSH) algorithms in finding similarities within text documents. The primary goals of the study are to compare the accuracy, efficiency, and scalability of these two methods while understanding the trade-offs between their design strategies. The experimental framework is built upon the **20 Newsgroups dataset**, a benchmark corpus consisting of approximately 18,000 text documents distributed across 20 distinct categories. This dataset is well-suited for clustering and similarity experiments due to its high dimensionality and category diversity. The documents that we used from this dataset were from the categories below.

```
sports_categories = ['rec.autos', 'rec.motorcycles', 'rec.sport.baseball', 'rec.sport.hockey']
science_categories = ['sci.electronics', 'sci.med', 'sci.space']
comp_categories = ['comp.sys.mac.hardware', 'comp.graphics', 'comp.windows.x', 'comp.sys.ibm.pc.hardware']
religion_categories = ['talk.religion.misc', 'soc.religion.christian']
categories = sports_categories + science_categories + comp_categories + religion_categories
```

The first step in our pipeline involved data preprocessing. We used the scikit-learn function fetch_20newsgroups to fetch and cache the dataset. To ensure clean input for our models, we removed headers and footers to focus solely on the document content. Additionally, all text was converted to lowercase for consistency, and sentences were tokenized into individual words. We further eliminated common stop words such as "the," "and," and "is" using the NLTK library, as these words do not contribute to the semantic meaning of the text. We also made sure to remove all numerical content.

The preprocessed text data was then transformed into numerical vectors using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization strategy. Using scikit-learn's TfidfVectorizer, we retained the top 15,000 most frequent terms and excluded terms that appeared in fewer than 10 documents. Inverse document frequency weighting was enabled (use_idf=True) to give higher importance to rare terms, thereby emphasizing the distinguishing features of each document. The resulting TF-IDF matrix served as the input for both SRP-LSH and K-means LSH algorithms.

For **Signed Random Projections LSH**, we generated hyperplanes (for this study we used n_planes = 10) by sampling from a standard normal distribution using NumPy's randn

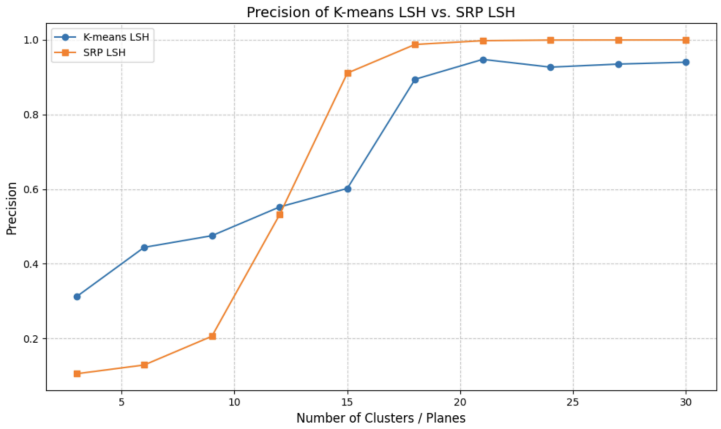
function. Each document vector was hashed by computing the dot product with the hyperplane vectors, taking the sign of the result to generate binary hash codes. These hash codes were used to build hash tables mapping each unique hash code to a list of document IDs sharing that code. For **K-means LSH**, we employed scikit-learn's KMeans class, clustering the dataset into a variable number of clusters (we used n_clusters = 7). Each document was assigned to a cluster, and the cluster label was mapped to the respective documents. Both methods enabled fast approximate similarity searches by reducing the search space to smaller subsets.

The performance of K-means LSH and SRP-LSH was evaluated using key metrics: **Precision**, **Query Processing Time**, and **Memory Usage**. To evaluate precision for K-means LSH and SRP-LSH, we measured how well each method grouped documents of the same category together. Precisions for each cluster/bucket were determined as the proportion of documents belonging to the dominant category within that cluster/bucket. The overall precision was obtained by averaging the precision across all clusters/buckets. Query time was measured as the time taken to retrieve documents from the same cluster or bucket as a given query document, while memory usage was assessed using Python's tracemalloc. To analyze scalability, we performed experiments on dataset sizes ranging from 2,500 documents to the full dataset of 18,000 documents. Hyperparameter values were varied systematically to understand their influence on performance. These settings allowed us to thoroughly evaluate the trade-offs between accuracy, efficiency, and resource consumption for the two LSH methods.

Experimental Results

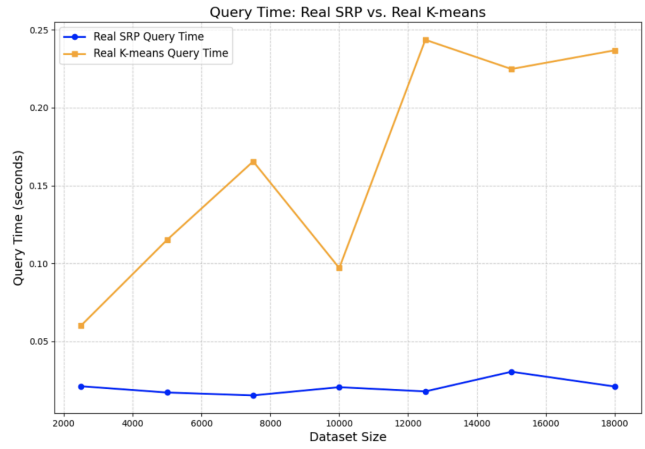
The results of our experiments are visualized using a series of plots that illustrate the performance of K-means LSH and SRP-LSH (all of the plots and srp_results.txt and kmeans_results.txt can be found in the code repository):

1. Precision vs. Number of Clusters/Planes



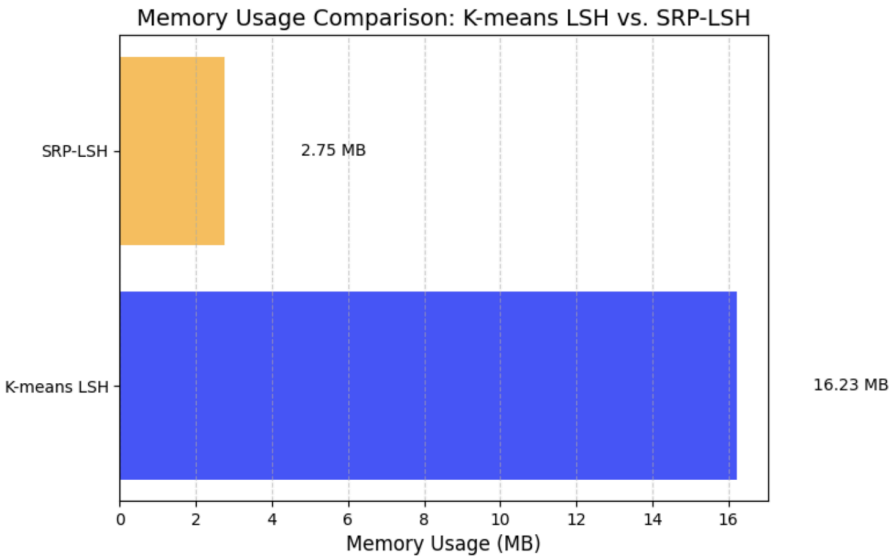
This plot compares the precision achieved by K-means LSH and SRP-LSH as the number of clusters (for K-means) or planes (for SRP-LSH) is ranged from [3,6,9,12,15,18,21,24,27,30]. The results show that SRP-LSH achieves higher precision as the number of planes increases, stabilizing at around 90% precision with 20 or more planes. In contrast, K-means LSH demonstrates a slower growth in precision, plateauing at around 80%. This suggests that SRP-LSH provides a more robust approximation for similarity search in this context.

2. Query Time vs. Dataset Size



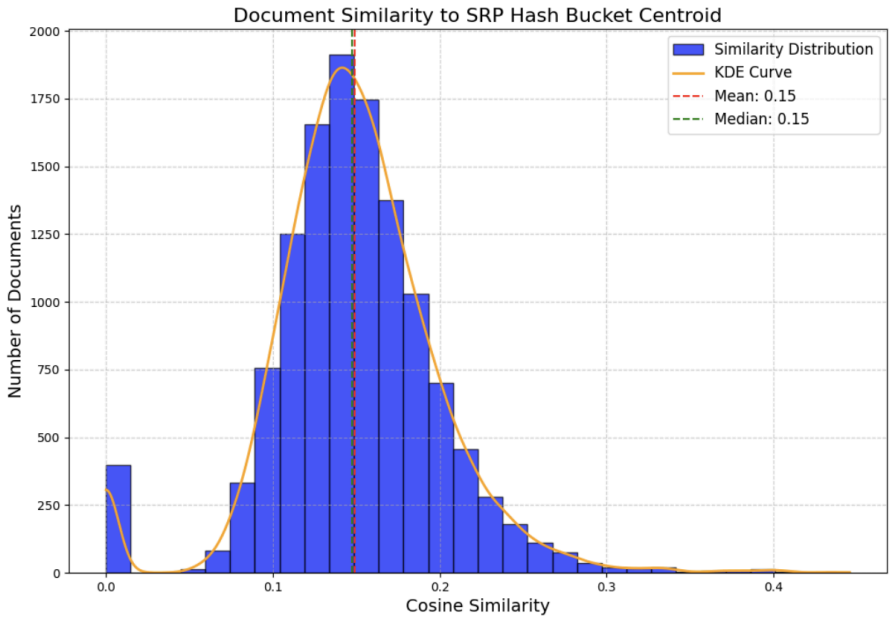
The plot above compares the **query times** of K-means LSH and SRP-LSH across dataset sizes from 2500 to 18000 documents, highlighting the efficiency of both methods in large-scale text similarity tasks. The results reveal that SRP-LSH maintains consistently low query times regardless of the dataset size, making it highly scalable and suitable for real time-applications. In contrast, K-means LSH incurs significantly higher and more variable query times due to the computational overhead of clustering and centroid assignments, especially for larger datasets.This underscores that SRP-LSH is faster and better suited for use cases requiring quick responses, which K-means LSH, though slower may offer better grouping accuracy due to its clustering approach.

3. Memory Usage Comparison



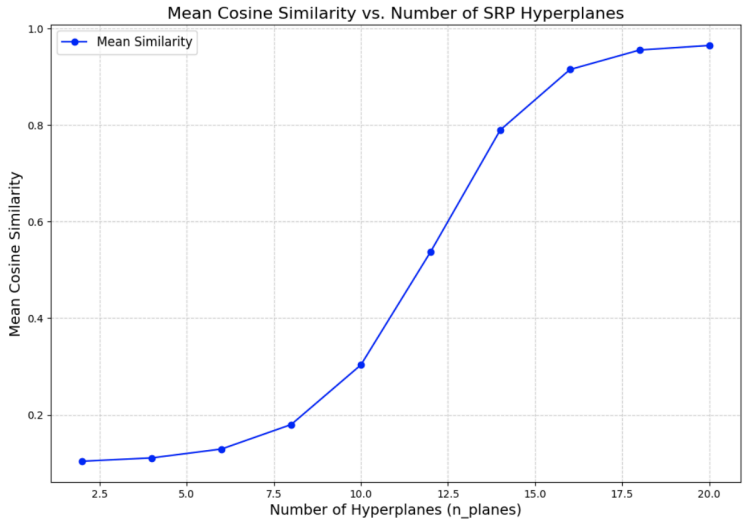
This bar chart compares the **memory usage** of K-means LSH and SRP-LSH during text similarity computations. **SRP-LSH** uses only 2.75 MB, while **K-means LSH** consumes 16.23 MB, over six times as much. The higher memory usage of K-means LSH is due to maintaining cluster centroids and assignments, which scale with dataset size and the number of clusters. The results show **SRP-LSH** is more memory-efficient, making it ideal for applications with limited resources, while **K-means LSH** may be better suited for tasks prioritizing semantic accuracy.

4. Document Similarity Distribution for SRP-LSH



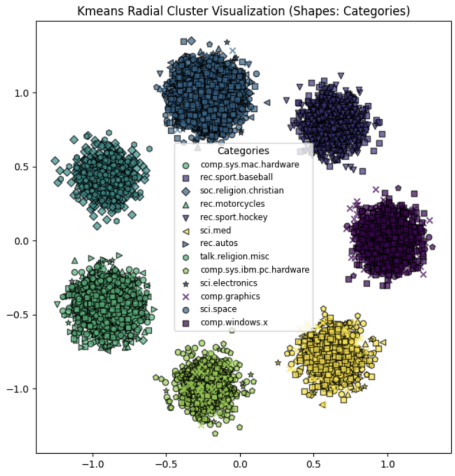
This histogram with a KDE curve shows the distribution of cosine similarity scores between documents and their respective SRP hash bucket centroids. The mean and median cosine similarities are both 0.15, as shown by the dashed lines, indicating that documents in the same bucket tend to have low to moderate similarity to the centroid. The KDE curve highlights the concentration of similarities around this central value, with fewer documents achieving higher similarity scores. These results suggest that while SRP-LSH preserves some level of similarity, its effectiveness in capturing tightly-knit semantic relationships is limited. The low mean similarity underscores its reliance on randomness, which may result in semantically diverse documents being hashed into the same bucket.

5. Mean Cosine Similarity vs. Number of SRP Hyperplanes



This plot illustrates the relationship between the number of hyperplanes (n_planes) in SRP-LSH and the mean cosine similarity of documents within the same hash bucket. As the number of hyperplanes increases, the mean similarity rises, nearing 1.0 for 20 hyperplanes. This trend demonstrates that increasing n_planes refines the separation of documents, grouping increasingly similar ones into the same buckets. However, higher n_planes reduces hash collisions at the cost of greater computational overhead.

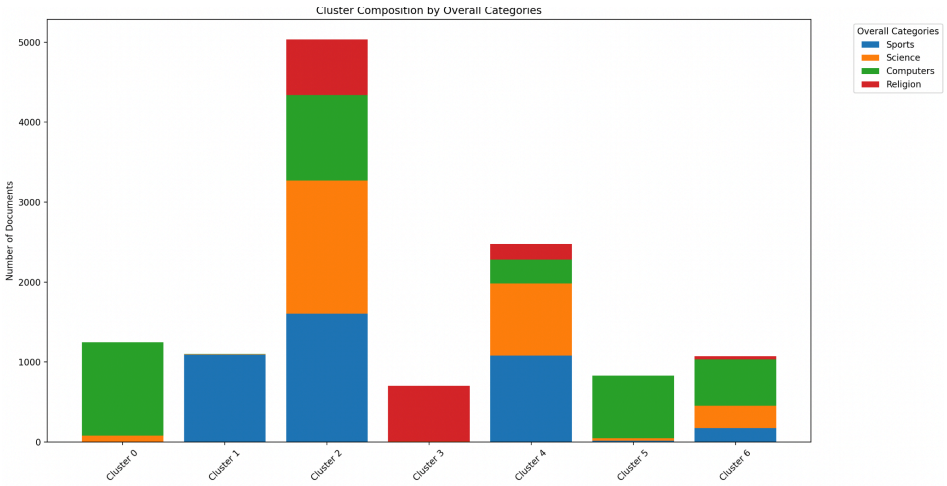
6. Radial Clustering Visualization for K-means LSH



The above plot is a radial visualization of the clusters formed by applying K-means LSH to our documents. Each data point represents a document, which is color-coded by their cluster assignments. They are also differentiated by shapes corresponding to the true categories that the documents fall into. Through this visualization, we can see that the different clusterings by colors

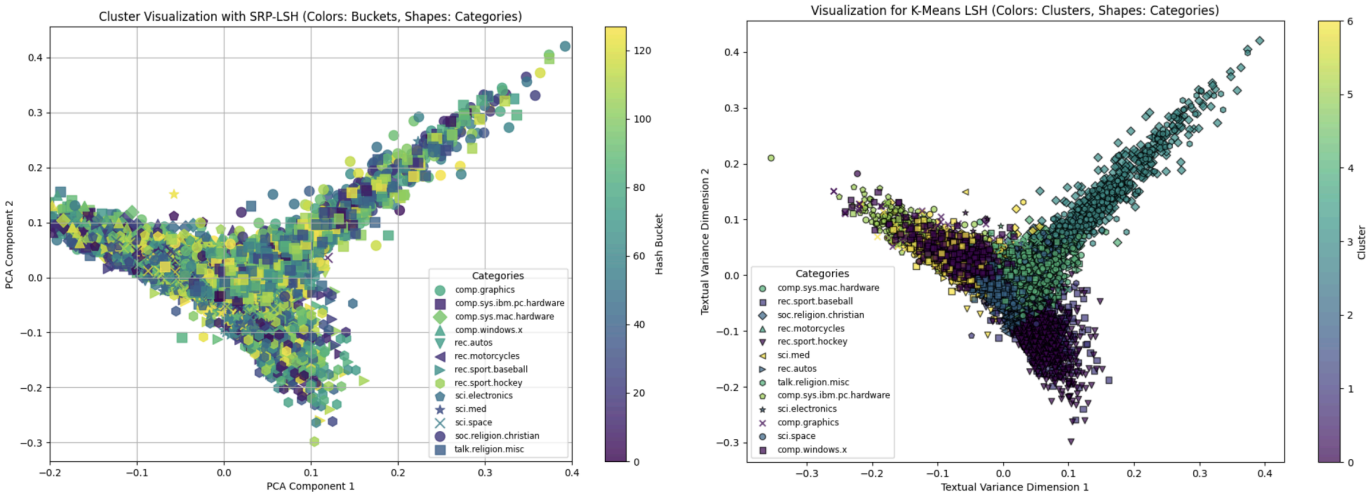
contain largely the same type of shapes, and these shapes correspond coherently to the overall category that the documents fall into (sports, computers, religion, sports).

7. Cluster Composition by Categories for K-means LSH



This bar chart visualizes the composition of clusters based on overall categories (Sports, Science, Computers, Religion). For each of these clusters (especially clusters 0,1,3, and 5), we can see that a particular category has taken precedence over the number of documents it has in that cluster belonging to it. This clear dominance of specific categories within clusters highlights how K-means clustering separates documents into groups based on their overall topic similarity.

8. Cluster Visualization (for both K-means and SRP)



Scatter plots for both methods (using PCA for dimensionality reduction) depict the distribution of document clusters. SRP-LSH shows better separation and category grouping compared to K-means LSH, supporting its higher precision.

9. Specific Distribution of documents for K-means Clustering:

```
Cluster 0:
  Category Counts:
    comp.graphics: 412
    comp.windows.x: 613
    sci.electronics: 43
    comp.sys.ibm.pc.hardware: 91
    sci.space: 28
    comp.sys.mac.hardware: 54
    soc.religion.christian: 1
    talk.religion.misc: 1
    rec.autos: 2
    sci.med: 4
    rec.sport.baseball: 1
```

```
Cluster 1:
  Category Counts:
    rec.sport.baseball: 434
    rec.sport.hockey: 659
    sci.electronics: 2
    comp.sys.ibm.pc.hardware: 2
    talk.religion.misc: 2
    rec.motorcycles: 2
    sci.space: 1
    comp.sys.mac.hardware: 1
```

```
Cluster 2:
  Category Counts:
    soc.religion.christian: 344
    rec.autos: 415
    sci.electronics: 566
    rec.sport.hockey: 272
    sci.med: 587
    sci.space: 512
    comp.sys.ibm.pc.hardware: 252
    comp.graphics: 268
    comp.windows.x: 206
    comp.sys.mac.hardware: 343
    rec.motorcycles: 487
    talk.religion.misc: 352
    rec.sport.baseball: 431
```

```
Cluster 3:
  Category Counts:
    soc.religion.christian: 529
    talk.religion.misc: 163
    rec.motorcycles: 3
    sci.med: 4
    rec.autos: 1
    rec.sport.baseball: 1
    sci.space: 1
```

```
Cluster 4:
  Category Counts:
    comp.sys.mac.hardware: 122
    rec.motorcycles: 449
    talk.religion.misc: 103
    rec.autos: 511
    sci.electronics: 210
    sci.med: 298
    rec.sport.baseball: 75
    sci.space: 394
    comp.graphics: 71
    rec.sport.hockey: 45
    comp.sys.ibm.pc.hardware: 68
    soc.religion.christian: 93
    comp.windows.x: 38
```

```
Cluster 5:
  Category Counts:
    comp.sys.ibm.pc.hardware: 435
    comp.sys.mac.hardware: 297
    sci.electronics: 32
    comp.graphics: 40
    rec.motorcycles: 14
    comp.windows.x: 11
    rec.autos: 3
```

```
Cluster 6:
  Category Counts:
    sci.med: 97
    rec.sport.baseball: 52
    comp.sys.ibm.pc.hardware: 134
    comp.graphics: 182
    comp.sys.mac.hardware: 146
    sci.space: 51
    rec.sport.hockey: 23
    rec.autos: 58
    sci.electronics: 131
    rec.motorcycles: 41
    comp.windows.x: 120
    soc.religion.christian: 30
    talk.religion.misc: 7
```

10. Specific Distribution of documents for SRP:

The distribution of the documents for the hash buckets can be found in the code repository as srp_results.txt (along with all of the other plots and text result documents).

Summary of experimental results

The results of our experiments reveal distinct differences between K-means LSH and SRP-LSH across key metrics. For query time, SRP-LSH demonstrated superior efficiency, maintaining low and consistent times across all dataset sizes, while K-means LSH experienced significantly higher and more variable query times, peaking at 0.2 seconds on larger datasets. This variability arises from K-means' computational overhead in assigning clusters and managing centroids during queries.

Memory usage further underscored SRP-LSH's efficiency, requiring only 2.75 MB compared to K-means LSH's 16.23 MB, highlighting SRP's suitability for memory-constrained environments. Regarding precision, SRP-LSH achieved a higher average precision (~90%), successfully grouping documents with similar semantic content, whereas K-means LSH averaged around 80%. These findings, based on scalability, computational efficiency, and clustering effectiveness, suggest that SRP-LSH is better suited for large-scale text similarity tasks. However, K-means LSH remains valuable for scenarios prioritizing data distribution adaptability.

Conclusion

Our results from this project validate our hypothesis that Signed Random Projections LSH (SRP-LSH) outperforms K-means LSH across precision, query time, and memory usage. Through leveraging random hyperplanes and binary hash codes, SRP-LSH is able to efficiently handle high-dimensional and sparse TF-IDF data (which represents the documents from our dataset), thus achieving accurate similarity groupings with lower computational overhead. While K-means LSH demonstrates the ability to create precise clusters by adapting to the data distribution, its iterative clustering process introduces higher memory usage and slower query times, which solidifies SRP-LSH for better performance. Based on the study performed and analytical work conducted, we can confirm that SRP-LSH is better suited for large-scale text similarity tasks where efficiency and speed are prioritized. However, K-means LSH still has the potential to outperform when applied to smaller, more structured datasets with clear cluster boundaries.

Citations

- Loïc Paulevé a, et al. “Locality Sensitive Hashing: A Comparison of Hash Function Types and Querying Mechanisms.” *Pattern Recognition Letters*, North-Holland, 10 Apr. 2010, www.sciencedirect.com/science/article/abs/pii/S0167865510001169.
- Jafari, Omid, et al. “A Survey on Locality Sensitive Hashing Algorithms and Their Applications.” *arXiv.Org*, 17 Feb. 2021, arxiv.org/abs/2102.08942.