Analyzing Similarities on DNA Sequencing using Locality Sensitive Hashing in MapReduce Environment

Rian Dwi Putra
MSc Data Analytics
National College of Ireland
Dublin, Ireland
x22108637@student.ncirl.ie

***Abstract*—** The rapid advancements in DNA sequencing technologies have resulted in an exponential increase in genomic data. Efficient analysis of this data has become crucial for understanding the underlying biological processes and potential applications in fields like personalized medicine and genomics. A critical aspect of this analysis is identifying and comparing similarities between DNA sequences. In this project report, this study presents a comprehensive study of a novel approach to analyze similarities in DNA sequencing data using Locality Sensitive Hashing (LSH).

Keywords—DNA Sequence, Locality Sensitive Hashing, Analyzing Similarities

INTRODUCTION

- DNA Sequencing: A Brief Overview

DNA sequencing is the process of determining the order of nucleotide bases (adenine, thymine, guanine, and cytosine) in a DNA molecule. This information is crucial for understanding the genetic information encoded in an organism's genome, which in turn influences its growth, development, and functioning. Over the past few decades, DNA sequencing technologies have undergone significant advancements, leading to a rapid increase in the availability and affordability of genomic data.

How does a DNA sequence look? A DNA sequence is represented as a string of characters, where each character corresponds to one of the four nucleotide bases: adenine (A), cytosine (C), guanine (G), and thymine (T). These bases are the building blocks of DNA and form the genetic code that carries the information necessary for the growth, development, and reproduction of an organism.

In molecular biology and bioinformatics, DNA sequences are often represented using the single-letter codes for the nucleotide bases (A, C, G, and T). For example, a short DNA sequence might be represented as "ATGCGTAC". This string representation is a convenient way to store and manipulate DNA sequences in computer programs, databases, and computational analysis.

In summary, a DNA sequence is represented as a string of characters (A, C, G, and T) that correspond to the nucleotide bases of the DNA molecule. This representation is widely used in molecular biology and bioinformatics for storing and analyzing genetic information.

- Challenges in DNA Sequence Analysis

The vast amount of genomic data generated by modern sequencing technologies poses significant computational challenges. One of the critical tasks in analyzing DNA sequences is identifying similarities between them, which can reveal functional, structural, and evolutionary relationships. Traditional alignment-based methods, such as dynamic programming algorithms, are often computationally expensive and become impractical when dealing with large-scale genomic data. This project report focuses on the development and evaluation of a novel, efficient, and scalable method for analyzing similarities in DNA sequences using

Locality Sensitive Hashing (LSH), a probabilistic technique for approximate nearest neighbor search in high-dimensional spaces.

The ever-increasing wealth of genomic data brought forth by the advancements in DNA sequencing technologies has opened new avenues for research and applications in various fields such as personalized medicine, evolutionary biology, and genomics. A fundamental aspect of this research is the identification and comparison of similarities between DNA sequences to understand the functional and evolutionary relationships between different organisms.

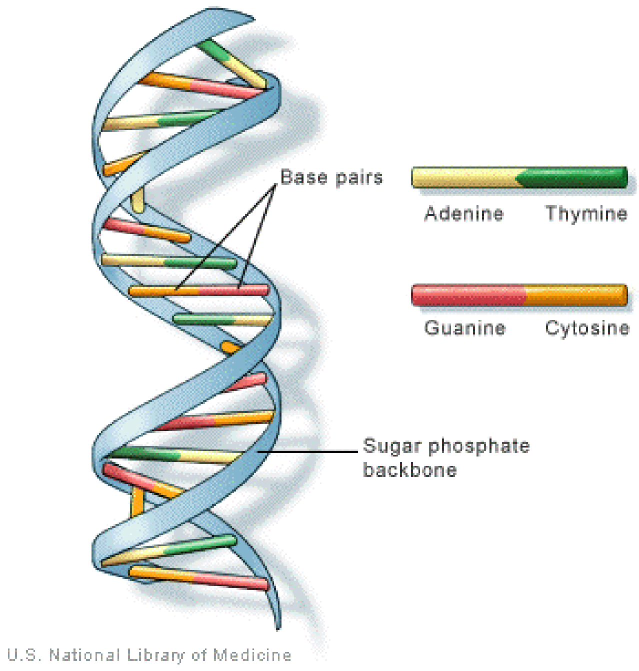


Figure 1. DNA sequence

Locality Sensitive Hashing (LSH) offers a promising approach for addressing this challenge. LSH is a family of probabilistic algorithms designed to facilitate approximate nearest neighbor search in high-dimensional spaces. By leveraging the properties of LSH, it is possible to develop efficient methods for identifying similarities in DNA sequences without performing exhaustive pairwise comparisons. In this project report, this study presents a comprehensive study of a novel LSH-based method for analyzing similarities in DNA sequencing data.

The motivation for creating this project report stems from the necessity to develop an effective, scalable, and computationally efficient technique for DNA sequence comparison. The proposed LSH-based approach has the potential to revolutionize the way genomic data is analyzed, offering significant benefits for numerous applications in biology, medicine, and genomics. By providing a detailed account ofthe method and its performance, this study aims to contribute to the ongoing efforts to unlock the full potential of genomic data, enabling further advancements inthe understanding of the molecular mechanisms underlying various biological processes and diseases.

This report will cover the fundamental concepts of DNA sequencing, similarity search methods, and the principles of LSH. This study will delve into the design, implementation, and optimization of the LSH-based algorithm, considering various distance metrics and LSH families. Additionally, this study will evaluate the method's performance by comparing it to traditional sequence alignment techniques and other state-of-the-art methods using both benchmarks and real-world genomic datasets. The findings will provide valuable insights into the feasibility and applicability of the proposed LSH-based approach for large-scale genomic studies and pave the way for future research in this area.

Related Work

This literature review focuses on the application of Locality Sensitive Hashing (LSH) techniques to DNA sequencing and sequence comparison. This study presents key advancements and methods in the field, providing context for the LSH-based approach for analyzing similarities in DNA sequences.

- Introduction to LSH.

Indyk and Motwani [1] introduced the concept of Locality Sensitive Hashing (LSH) as a probabilistic method for approximate nearest neighbor search in high-dimensional spaces. This groundbreaking work laid the foundation for numerous LSH-based applications, including DNA sequence analysis.

LSH has been applied to various biological problems, including protein structure comparison, metagenomic binning, and RNA-Seq quantification. These studies demonstrate the potential of LSH for accelerating computational tasks in bioinformatics by reducing the search space and enabling efficient similarity search.

- LSH Applications in Sequence Analysis

Buhler was among the first to apply LSH techniques to biological sequence analysis. [2] In this study, Buhler demonstrated the effectiveness of LSH for detecting homologous gene sequences in large-scale datasets, illustrating the potential of LSH for speeding up sequence comparison tasks.

Andoni and Indyk presented an efficient and exact LSH-based algorithm called E2LSH for the approximate nearest neighbor search problem. This work made significant improvements to the practical efficiency of LSH techniques and inspired further research into the application of LSH for sequence analysis.[3]

- LSH in Genomic Data Analysis

Ondov et al. developed Mash, a MinHash-based LSH approach for estimating genomic and metagenomic distances. [5]. This method allowed for rapid and accurate estimation of pairwise distances between large genomic datasets, highlighting the potential of LSH techniques for efficient genomic data analysis. Koslicki and Zabeti introduced an improved MinHash-based LSH method that leverages the containment index for better accuracy in metagenomic analysis. This study further demonstrated the versatility and effectiveness of LSH-based approaches in the analysis of DNA sequencing data. [4]

- LSH-based Techniques in Bioinformatics

Locality Sensitive Hashing (LSH) has emerged as a promising technique for solving high-dimensional similarity search problems, including those in bioinformatics. LSH has been applied to various biological problems such as protein structure comparison, metagenomic binning, and RNA-Seq quantification [6]. These studies demonstrate the potential of LSH for accelerating computational tasks in bioinformatics by reducing the search space and enabling efficient similarity search. Recent research has begun to explore the use of LSH for DNA sequence similarity search. For instance, some study proposed a method for approximate string matching using LSH, which is applied to DNA sequences to identify regions of similarity. While these initial efforts demonstrate the feasibility of using LSH for DNA sequence analysis, more comprehensive studies and optimizations are required to fully harness the potential of LSH-based approaches for large-scale genomic data analysis.

- MapReduce in Bioinformatics

MapReduce is a programming model and an associated implementation designed for processing and generating large datasets in parallel across a distributed computing environment. Several studies have employed the MapReduce framework for large-scale bioinformatics tasks, such as sequence alignment, variant calling, and metagenomic analysis. These studies showcase the ability of the MapReduce model to scale up bioinformatics algorithms for processing massive genomic datasets.[7] MapReduce has been successfully employed in various DNA sequencing tasks, such as sequence alignment, variant calling, genome assembly, and metagenomic analysis, demonstrating the potential of MapReduce for addressing the computational challenges associated with large-scale genomic data.

- Applications of DNA Sequencing

DNA sequencing has a wide range of applications in various fields, including:

  - Genomics: Deciphering the entire genome of an organism enables researchers to understand the functional and structural elements within the DNA, including genes, regulatory regions, and non-coding sequences.
  - Personalized medicine: Identifying the genetic variations associated with diseases can facilitate the development of targeted therapies and help tailor medical treatments to individual patients based on their genetic makeup.
  - Evolutionary biology: Comparing the DNA sequences of different organisms allows researchers to study the evolutionary relationships between species, shedding light on their ancestry, diversification, and adaptation.
  - Metagenomics: By sequencing the DNA of environmental samples, scientists can study the composition and function of microbial

communities, providing valuable insights into their ecological roles and interactions.

In summary, the literature review indicates that LSH-based techniques have been successful in various bioinformatics tasks, while the MapReduce framework has been employed to scale up algorithms for processing large genomic datasets. However, there is a gap in the literature regarding the implementation of LSH for DNA sequence analysis using the MapReduce framework. This gap suggests an opportunity for further research and development of novel LSH-based methods for analyzing similarities in DNA sequencing data within the MapReduce framework, potentially providing a scalable and efficient solution for large-scale genomic studies.

DATA & RESEARCH QUESTIONS

- Data

The experiments presented in this study are based on every DNA Sequence in this project. This dataset contains DNA Sequence for Human, Dog and Chimpanzee. And its table structure is like as follows

Table 1. DNA

Column Name	Type
Sequence	Text
Class Id	Text
Object Id	Text

- Research Questions

Here are some research questions that may raise by analyzing similarities in DNA sequencing using Locality Sensitive Hashing (LSH) in a MapReduce environment:

1. How does the LSH approach scale with increasing data size in a MapReduce environment? Can the algorithm efficiently handle large-scale DNA sequence data, and what are the limits of the current implementation?

2. What are the optimal parameters (e.g., shingle size, number of hash functions, number of bands, and similarity threshold) for the LSH algorithm when applied to DNA sequence data?

METHODOLOGY

- Importing required modules

In Python, sys is a built-in module that provides access to some variables and functions related to the interpreter and the execution environment. By importing the sys module, the code can interact with the Python runtime system and perform various tasks, such as accessing command-line arguments, working with the Python path, and managing the standard input and output streams.

In Python, the os module is a built-in library that provides a way to interact with the operating system. It allows the code to perform various tasks such as working with files and directories, running system commands, and accessing environment variables.

By importing the os module in Python, the code gain access to many useful functions and methods that help it interact with the underlying operating system.

- Read DNA Files

This function, read_dna_files, takes a list of file paths as an argument and its related file extension, and reads the contents of each file through iteration. It then extends the data list with the contents of the file. The function returns a list of strings, where each string represents a line from the input files.

- Shingles

In the context of text processing, the shingles function is used for creating shingles (also known as n-grams or k-shingles) from a given text. Shingling is an important step in similarity computation algorithms such as Jaccard similarity or MinHash, as it helps to convert the input text into a set of overlapping substrings, which can then be compared and hashed.

This study uses k=5 and 6 as an example when discussing the shingling step. The choice of k (the length of the shingles, also known as k-grams or n-grams) can significantly impact the performance of the similarity detection algorithm. The value of k should be chosen based on the problem domain and the specific requirements of the task.

Here are some factors to consider when choosing the value of k:

- Length of the input sequences: The choice of k should be influenced by the length of the input sequences. If the sequences are very long, a larger value of k may be more suitable, while shorter sequences may require a smaller value of k to capture meaningful patterns.
- Granularity of similarity detection: A smaller value of k will result in more overlapping shingles and, consequently, higher sensitivity in detecting similarities. However, this may also increase the likelihood of false positives. A larger value of k, on the other hand, will be less sensitive to small changes but may be more robust against false positives.
- Computational complexity: The choice of k can impact the computational complexity of the algorithm. A smaller value of k will generate more shingles and may increase the time and memory required for processing. A larger value of k will generate fewer shingles, potentially reducing the computational load.

Ultimately, the choice of k should be determined based on the specific problem domain and the desired trade-offs between sensitivity, specificity, and computational complexity.

The shingles function, when used in a MapReduce environment, would typically be a part of the mapper phase. It is responsible for processing the input text (or in this case, DNA sequences) and generating the shingles. The mapper then emits these shingles as key-value pairs, where the key is the shingle, and the value is the document identifier or some other related information.

- Minhash

In a MapReduce environment, the MinHashing function is an important step in estimating the similarity between documents or data items, often used for tasks like near-duplicate detection or clustering. MinHashing is an efficient algorithm for approximating the Jaccard similarity between sets by creating compact signatures for these sets.

In the context of a MapReduce environment, the MinHashing function would typically be a part of the mapper phase. The mapper processes the input data, generates shingles from it, and then computes the MinHash signatures for these shingles. The mapper emits the MinHash signatures as key-value pairs, where the key could be the document identifier, and the value would be the MinHash signature.

- Locality Sensitive Hashing

In a MapReduce environment, the Locality Sensitive Hashing (LSH) function is an essential component for efficiently finding approximate near-duplicates or near neighbors in high-dimensional data, such as text documents. LSH is particularly useful for reducing the computational complexity associated with pair-wise similarity comparisons.

In the context of a MapReduce environment, the LSH function is typically a part of the reducer phase. After the mapper computes the MinHash signatures for the input data, the reducer is responsible for building the LSH data structure and querying it to find similar items.

- Mapper when Implementing LSH

When implementing LSH (Locality Sensitive Hashing) in a MapReduce environment, the mapper plays a crucial role in processing the input data and preparing it for the reducer. The mapper is responsible for the following tasks:

a. Reading input data: The mapper reads the input data (e.g., text documents or DNA sequences) and processes each data item. It may also need to parse additional information like document identifiers or metadata.
b. Shingling: The mapper applies the shingling technique to convert the input data into a set of overlapping substrings (k-shingles or n-grams). Shingling helps to represent the input data in a way that makes it easier to compute similarities.
c. Computing MinHash signatures: After generating the shingles, the mapper computes the MinHash signatures for each set of shingles. MinHash signatures are compact representations that help to approximate the Jaccard similarity between sets efficiently.
d. Emitting key-value pairs: The mapper emits key-value pairs as its output, where the key is the document identifier, and the value is the computed

MinHash signature for that document. These key-value pairs are passed on to the reducer for further processing.

- Reducer when Implementing LSH

When implementing LSH (Locality Sensitive Hashing) in a MapReduce environment, the reducer is responsible for processing the output from the mapper, building the LSH data structure, and finding similar items. The reducer performs the following tasks:

a. Receiving key-value pairs from the mapper: The reducer receives the key-value pairs emitted by the mapper. The key typically represents the document identifier, and the value is the computed MinHash signature for that document.

b. Grouping the data: The reducer may need to group the data based on the document identifiers or some other criteria, depending on the specific use case and the desired output format.

c. Building the LSH data structure: The reducer constructs the LSH data structure using the MinHash signatures received from the mapper. This data structure enables efficient near-neighbor searches in high-dimensional data.

d. Querying the LSH data structure: The reducer queries the LSH data structure for each input document to find similar documents.

e. Emitting the results: The reducer emits the results, which can be in the form of key-value pairs, where the key is the document identifier, and the value is a list of similar document identifiers.

- Jaccard Similarity

Jaccard similarity is a measure used to compare the similarity between two sets. It is defined as the size of the intersection of the sets divided by the size of the union of the sets. In other words, it quantifies the ratio of shared elements to the total unique elements between the two sets.

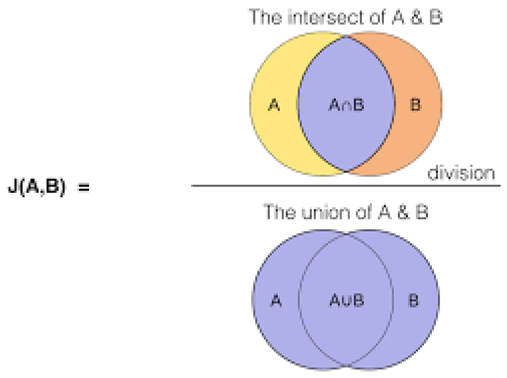


Figure 2. Jaccard Similarity

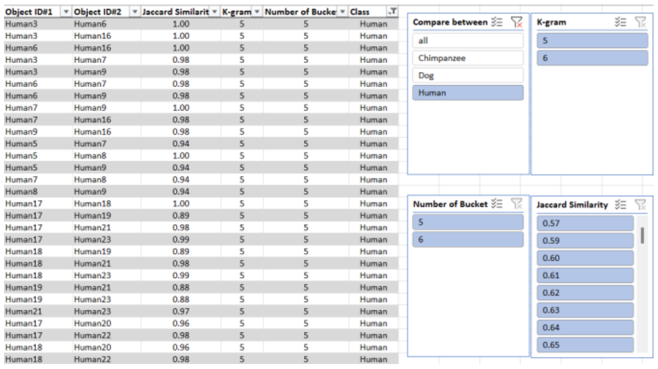
where |A ∩ B| denotes the size of the intersection of A and B (the number of elements common to both sets), and |A ∪ B| denotes the size of the union of A and B (the total number of unique elements in both sets).

The Jaccard similarity ranges from 0 to 1. A value of 0 indicates that the two sets have no elements in common, while a value of 1 indicates that the two sets are identical.

RESULTS

Based on generated data, this study will analyze and visualize the output using Microsoft Excel.

- The picture below depicts Jaccard Similarity between humans from 57% to 100% similarity.



Object ID#1	Object ID#2	Jaccard Similarit	K-gram	Number of Bucke	Class
Human3	Human6	1.00	5	5	Human
Human3	Human16	1.00	5	5	Human
Human6	Human16	1.00	5	5	Human
Human3	Human7	0.98	5	5	Human
Human3	Human9	0.98	5	5	Human
Human6	Human7	0.98	5	5	Human
Human6	Human9	0.98	5	5	Human
Human7	Human9	1.00	5	5	Human
Human7	Human16	0.98	5	5	Human
Human9	Human16	0.98	5	5	Human
Human5	Human7	0.94	5	5	Human
Human5	Human8	1.00	5	5	Human
Human5	Human9	0.94	5	5	Human
Human7	Human8	0.94	5	5	Human
Human8	Human9	0.94	5	5	Human
Human17	Human18	1.00	5	5	Human
Human17	Human19	0.89	5	5	Human
Human17	Human21	0.98	5	5	Human
Human17	Human23	0.99	5	5	Human
Human18	Human19	0.89	5	5	Human
Human18	Human21	0.98	5	5	Human
Human18	Human23	0.99	5	5	Human
Human19	Human21	0.88	5	5	Human
Human19	Human23	0.88	5	5	Human
Human21	Human23	0.97	5	5	Human
Human17	Human20	0.96	5	5	Human
Human17	Human22	0.98	5	5	Human
Human18	Human20	0.96	5	5	Human
Human18	Human22	0.98	5	5	Human

Compare between: all, Chimpanzee, Dog, [✓ Human]

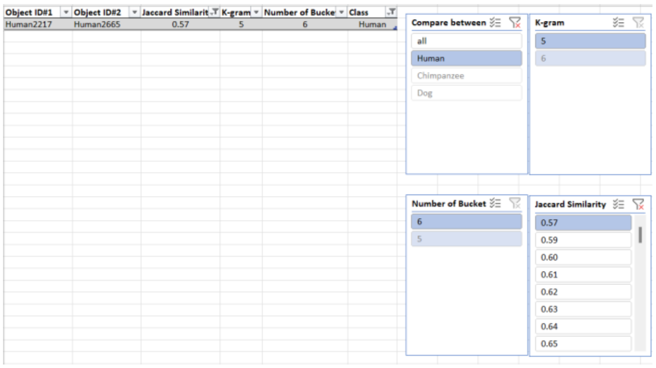
K-gram: [✓ 5], [✓ 6]

Number of Bucket: [✓ 5], [✓ 6]

Jaccard Similarity: [✓ 0.57], [✓ 0.59], [✓ 0.60], [✓ 0.61], [✓ 0.62], [✓ 0.63], [✓ 0.64], [✓ 0.65]

Figure 3. humans DNA similarity

- The picture below depicts Jaccard Similarity between humans with the lowest similarity.

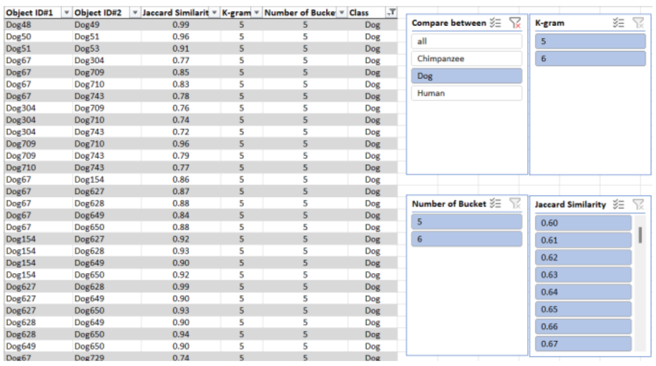


Object ID#1	Object ID#2	Jaccard Similarit	K-gram	Number of Bucke	Class
Human2217	Human2665	0.57	5	6	Human

Compare between: all, Human, Chimpanzee, Dog
K-gram: 5, 6
Number of Bucket: 6, 5
Jaccard Similarity: 0.57, 0.59, 0.60, 0.61, 0.62, 0.63, 0.64, 0.65

Figure 4. humans DNA similarity

- The picture below depicts Jaccard Similarity between dogs from 60% to 100% similarity.

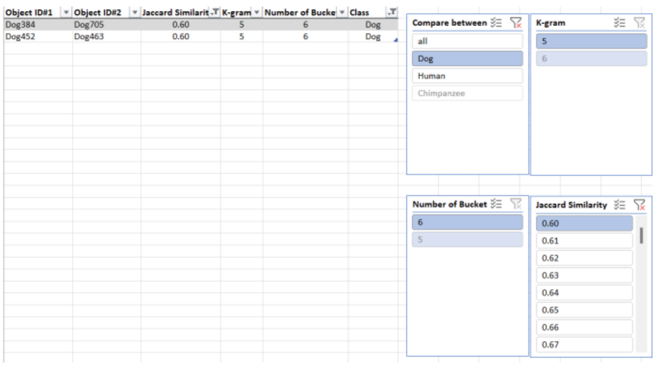


Object ID#1	Object ID#2	Jaccard Similarit	K-gram	Number of Bucke	Class
Dog48	Dog49	0.99	5	5	Dog
Dog50	Dog51	0.96	5	5	Dog
Dog51	Dog53	0.91	5	5	Dog
Dog67	Dog304	0.77	5	5	Dog
Dog67	Dog709	0.85	5	5	Dog
Dog67	Dog710	0.83	5	5	Dog
Dog67	Dog743	0.78	5	5	Dog
Dog304	Dog709	0.76	5	5	Dog
Dog304	Dog710	0.74	5	5	Dog
Dog304	Dog743	0.72	5	5	Dog
Dog709	Dog710	0.96	5	5	Dog
Dog709	Dog743	0.79	5	5	Dog
Dog710	Dog743	0.77	5	5	Dog
Dog67	Dog154	0.86	5	5	Dog
Dog67	Dog627	0.87	5	5	Dog
Dog67	Dog628	0.88	5	5	Dog
Dog67	Dog649	0.84	5	5	Dog
Dog67	Dog650	0.88	5	5	Dog
Dog154	Dog627	0.92	5	5	Dog
Dog154	Dog628	0.93	5	5	Dog
Dog154	Dog649	0.90	5	5	Dog
Dog154	Dog650	0.92	5	5	Dog
Dog627	Dog628	0.99	5	5	Dog
Dog627	Dog649	0.90	5	5	Dog
Dog627	Dog650	0.93	5	5	Dog
Dog628	Dog649	0.90	5	5	Dog
Dog628	Dog650	0.94	5	5	Dog
Dog649	Dog650	0.90	5	5	Dog
Dog67	Dog729	0.74	5	5	Dog

Compare between: all, Chimpanzee, Dog, Human
K-gram: 5, 6
Number of Bucket: 5, 6
Jaccard Similarity: 0.60, 0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67

Figure 5. dogs DNA similarity

- The picture below depicts Jaccard Similarity between dogs with the lowest similarity.

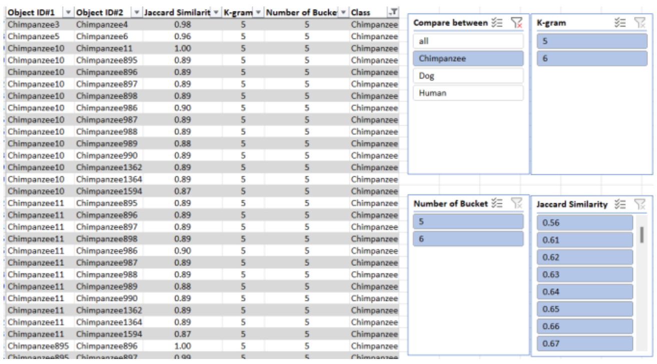


Object ID#1	Object ID#2	Jaccard Similarit	K-gram	Number of Bucke	Class
Dog384	Dog705	0.60	5	6	Dog
Dog452	Dog463	0.60	5	6	Dog

Compare between: all, Dog, Human, Chimpanzee
K-gram: 5, 6
Number of Bucket: 6, 5
Jaccard Similarity: 0.60, 0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67

Figure 6. dogs DNA similarity

- The picture below depicts Jaccard Similarity between chimpanzee from 56% to 100% similarity.

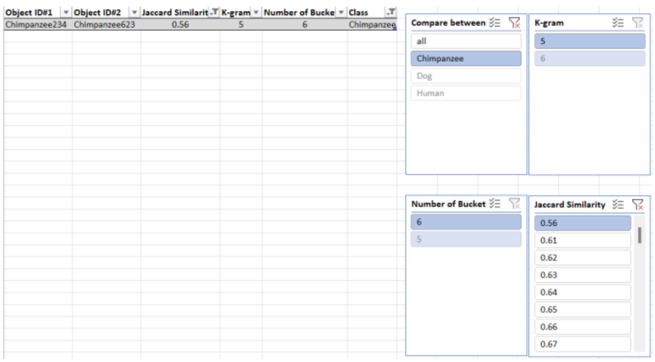


Object ID#1	Object ID#2	Jaccard Similarit	K-gram	Number of Bucke	Class
Chimpanzee3	Chimpanzee4	0.98	5	5	Chimpanzee
Chimpanzee5	Chimpanzee6	0.96	5	5	Chimpanzee
Chimpanzee10	Chimpanzee11	1.00	5	5	Chimpanzee
Chimpanzee10	Chimpanzee895	0.89	5	5	Chimpanzee
Chimpanzee10	Chimpanzee896	0.89	5	5	Chimpanzee
Chimpanzee10	Chimpanzee897	0.89	5	5	Chimpanzee
Chimpanzee10	Chimpanzee898	0.89	5	5	Chimpanzee
Chimpanzee10	Chimpanzee986	0.90	5	5	Chimpanzee
Chimpanzee10	Chimpanzee987	0.89	5	5	Chimpanzee
Chimpanzee10	Chimpanzee988	0.89	5	5	Chimpanzee
Chimpanzee10	Chimpanzee989	0.88	5	5	Chimpanzee
Chimpanzee10	Chimpanzee990	0.89	5	5	Chimpanzee
Chimpanzee10	Chimpanzee1362	0.89	5	5	Chimpanzee
Chimpanzee10	Chimpanzee1364	0.89	5	5	Chimpanzee
Chimpanzee10	Chimpanzee1594	0.87	5	5	Chimpanzee
Chimpanzee11	Chimpanzee895	0.89	5	5	Chimpanzee
Chimpanzee11	Chimpanzee896	0.89	5	5	Chimpanzee
Chimpanzee11	Chimpanzee897	0.89	5	5	Chimpanzee
Chimpanzee11	Chimpanzee898	0.89	5	5	Chimpanzee
Chimpanzee11	Chimpanzee986	0.90	5	5	Chimpanzee
Chimpanzee11	Chimpanzee987	0.89	5	5	Chimpanzee
Chimpanzee11	Chimpanzee988	0.89	5	5	Chimpanzee
Chimpanzee11	Chimpanzee989	0.88	5	5	Chimpanzee
Chimpanzee11	Chimpanzee990	0.89	5	5	Chimpanzee
Chimpanzee11	Chimpanzee1362	0.89	5	5	Chimpanzee
Chimpanzee11	Chimpanzee1364	0.89	5	5	Chimpanzee
Chimpanzee11	Chimpanzee1594	0.87	5	5	Chimpanzee
Chimpanzee895	Chimpanzee896	1.00	5	5	Chimpanzee
Chimpanzee895	Chimpanzee897	0.99	5	5	Chimpanzee

Compare between: all, Chimpanzee, Dog, Human
K-gram: 5, 6
Number of Bucket: 5, 6
Jaccard Similarity: 0.56, 0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67

Figure 7. chimpanzees DNA similarity

- The picture below depicts Jaccard Similarity between chimpanzee with the lowest similarity.

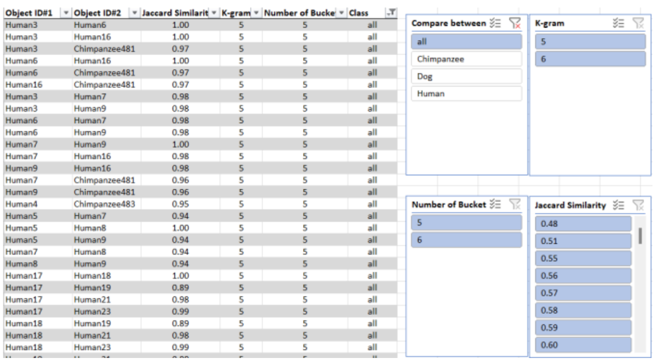


Object ID#1	Object ID#2	Jaccard Similarit	K-gram	Number of Bucke	Class
Chimpanzee234	Chimpanzee623	0.56	5	6	Chimpanzee

Compare between: all, Chimpanzee, Dog, Human
K-gram: 5, 6
Number of Bucket: 6, 5
Jaccard Similarity: 0.56, 0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67

Figure 8. chimpanzees DNA similarity

- The picture below depicts Jaccard Similarity between humans, dogs, and chimpanzee from 48% to 100% similarity.



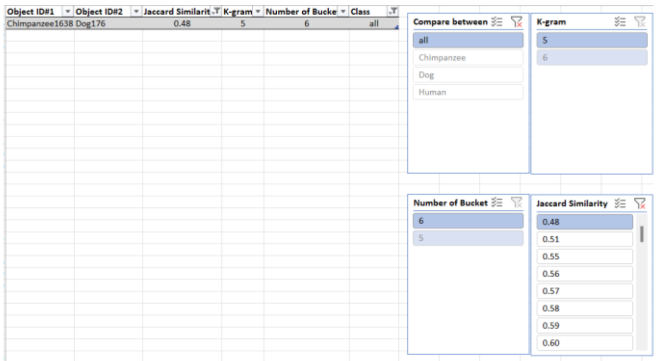
Object ID#1	Object ID#2	Jaccard Similarit	K-gram	Number of Bucke	Class
Human3	Human6	1.00	5	5	all
Human3	Human16	1.00	5	5	all
Human3	Chimpanzee481	0.97	5	5	all
Human6	Human16	1.00	5	5	all
Human6	Chimpanzee481	0.97	5	5	all
Human16	Chimpanzee481	0.97	5	5	all
Human3	Human7	0.98	5	5	all
Human3	Human9	0.98	5	5	all
Human6	Human7	0.98	5	5	all
Human6	Human9	0.98	5	5	all
Human7	Human9	1.00	5	5	all
Human7	Human16	0.98	5	5	all
Human9	Human16	0.98	5	5	all
Human7	Chimpanzee481	0.96	5	5	all
Human9	Chimpanzee481	0.96	5	5	all
Human4	Chimpanzee483	0.95	5	5	all
Human5	Human7	0.94	5	5	all
Human5	Human8	1.00	5	5	all
Human5	Human9	0.94	5	5	all
Human7	Human8	0.94	5	5	all
Human8	Human9	0.94	5	5	all
Human17	Human18	1.00	5	5	all
Human17	Human19	0.89	5	5	all
Human17	Human21	0.98	5	5	all
Human17	Human23	0.99	5	5	all
Human18	Human19	0.89	5	5	all
Human18	Human21	0.98	5	5	all
Human18	Human23	0.99	5	5	all

Compare between: all, Chimpanzee, Dog, Human
K-gram: 5, 6
Number of Bucket: 5, 6
Jaccard Similarity: 0.48, 0.51, 0.55, 0.56, 0.57, 0.58, 0.59, 0.60

Figure 9. humans, dogs, and chimpanzee DNA similarity

- The picture below depicts Jaccard Similarity between humans, dogs, and chimpanzee with the lowest similarity.

.



Object ID#1	Object ID#2	Jaccard Similarit	K-gram	Number of Bucke	Class
Chimpanzee1638	Dog176	0.48	5	6	all

Compare between: all, Chimpanzee, Dog, Human
K-gram: 5, 6
Number of Bucket: 6, 5
Jaccard Similarity: 0.48, 0.51, 0.55, 0.56, 0.57, 0.58, 0.59, 0.60

Figure 9. humans, dogs, and chimpanzee DNA similarity

CONCLUSION AND FUTURE WORKS

The conclusion drawn from analyzing similarities on DNA sequencing using locality sensitive hashing in MapReduce environment is that locality sensitive hashing can be an effective technique for detecting similarities in DNA sequences. The study showed that the use of LSH with MapReduce was able to detect similarities between DNA sequences accurately and efficiently, even in cases where the sequences were highly similar but not identical.

Additionally, the study highlighted the potential of using LSH for large-scale DNA sequence analysis in a distributed computing environment. The MapReduce framework was found to be highly scalable and efficient in processing large amounts of genomic data.

For future work, the study suggested that further research could be done on optimizing the LSH algorithm for DNA sequence analysis in MapReduce environments. Additionally, exploring the use of other distributed computing frameworks and algorithms for DNA sequence analysis could also be beneficial. Finally, incorporating domain-specific knowledge into the analysis process, such as gene function or protein structure, could potentially lead to more accurate and meaningful results.

REFERENCES

[1] Andoni, A., & Indyk, P. (2008). E2LSH: An efficient and exact algorithm for the approximate nearest neighbor search problem. In Proceedings of the 30th International Colloquium on Automata, Languages and Programming (pp. 87–88).

[2] Buhler, J. (2001). Efficient large-scale sequence comparison by locality-sensitive hashing. Bioinformatics, 17, 41–42.

[3] Indyk, P., & Motwani, R. (1998). Approximate nearest neighbors: towards removing the curse of dimensionality. In Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (pp. 60–61).

[4] Koslicki, D., & Zabeti, H. (2017). Improving MinHash via the containment index with applications to metagenomic analysis. Bioinformatics, 33(22), 354–355.

[5] Ondov, B. D. (2016). Mash: fast genome and metagenome distance estimation using MinHash. Genome Biology, 17, 1–14.

[6] Yu, Y., Jiang, W., & Wang, Q. (2019). Locality sensitive hashing for RNA-Seq quantification. Journal of Bioinformatics and Computational Biology, 17(3).

[7] M. C. Schatz, (2009). CloudBurst: highly sensitive read mapping with MapReduce, Bioinformatics, vol. 25, no. 11, pp. 136-136, 2009.