1. Introduction

Project Name: Implementing locality sensitive hashing to compare large scale, out of core machine learning datasets
Organization: Python Software Foundation
Suborganization: Activeloop

2. Student Information

Name: Rahul Babu Shrestha
Time Zone: Nepal (GMT +5:45)
Github: https://github.com/rahulbshrestha
Twitter: https://twitter.com/rahulbshrestha
LinkedIn: https://www.linkedin.com/in/rahulbshrestha/
GSoC blog: Will be hosted in my personal webpage: https://rahulbshrestha.github.io/

3. Project Info

Proposal title:

Implementing locality sensitive hashing to compare large scale, out of core machine learning text and image datasets

Problem Description:

Machine learning datasets are often continuously modified yet there is no efficient way to determine if two datasets are the same. This challenge becomes problematic when the datasets are large (20 GB+) and cannot be easily loaded into memory.

Problem Solution

This project intends to use Locality Sensitive Hashing (LSH) with the MapReduce paradigm to compare large machine learning datasets.

LSH is an algorithm which hashes similar inputs into the same "buckets" with high probability. LSH is different from conventional cryptographic hashing functions as it tries to ensure hash collisions rather than avoid it. Input data is hashed and put into "buckets". The more similar the objects are, higher the probability that they are in the same bucket. Hash collisions are helpful as the goal is to reduce the number of comparisons required to find similar items and similar datasets are highly likely to have similar hash values. LSH is used for solving probabilistic dimension reduction of high dimensional spaces. For example, it is used in nearest neighbour search on large scale data [2].

LSH helps find similar pairs in a large dataset in optimal time. The time complexity for brute forcing i.e comparing every possible pair has a time complexity of $N!/(2!(N-2)!) \sim N^2/2 = O(N^2)$. LSH improves this with a time complexity of $O(N)$.

I intend to implement LSH with the MapReduce paradigm so large datasets can easily be hashed and compared for similarities. LSH outperforms other traditional hashing algorithms, and the MapReduce paradigm running on Apache Spark/Hadoop makes it scalable to large datasets. If my project is complete, it will be useful for Hub to compare large image and text datasets for duplicates.

Timeline and Deliverables

<u>Before June 7</u>

- Decide on using Apache Spark vs Hadoop for the distributed architecture. This will depend upon the possibility of integration with my Python package. As of now, I am leaning towards Hadoop because of a paper I read that already implemented something similar [2].
- Get comfortable with the concept of MapReduce paradigm on Spark/ Hadoop and Locality Sensitive Hashing.
- Familiarize myself with the Hub community, coding practices, documentation, testing system.
- Setup environment and collect sample image and text datasets.

<u>Week 1 (June 7 - June 13)</u>

- Create skeleton for Python package.
- Implement operations to collect and split data from datasets.
- Write code to vectorize datasets.
- This can be done by enumerating its “k-shingles”. K-shingles are k-consecutive characters occurring in the dataset. Groups of 5-7 characters are made into shingles and put in a big set.

Week 2 (June 14 - June 20)

- Implement minhash algorithm, which is used to calculate the probability of hash collisions.
- The probability of collision in LSH using minhash is the same as the Jaccard Similarity metric [3]. Jaccard Similarity is used to compare the similarity between any two set
- **Milestone:** A Python package which can split a text dataset into shingles and apply the minhash algorithm

Week 3 (June 21 - June 27)

- Start working on a distributed system architecture based on this research paper [2] which uses the MapReduce paradigm with Hadoop.
- Integrate my existing Python code with this distributed system architecture, so a basic implementation of dividing a large dataset into shingles and applying the minhash algorithm is possible.

Week 4 (June 28 - July 4)

- At this point, I will have decided if I want to go forward with Hadoop or Apache Spark.
- Implement "candidate generation" phase, where pairs of objects that are very likely similar are selected for comparison.
- The MinHash algorithm is adapted to the MapReduce paradigm in Hadoop/Spark. To implement MinHash on Hadoop, the library Likelike is available.

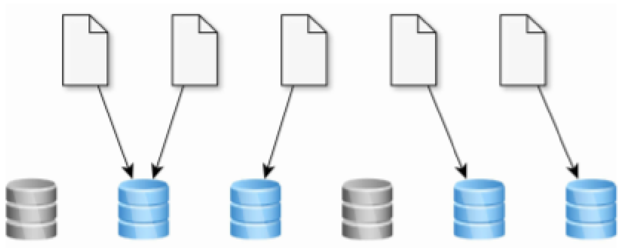

Fig: Candidate generation phase [2]

Week 5 (July 5 - July 11)

- Implement "candidate verification" phase, where the similarity of all pairs of objects received from a single bucket is verified. This will be done by calculating the similarity of two documents based on the Jaccard similarity coefficient [3].

- **Milestone:** A Python package with Spark/Hadoop using MapReduce that can apply Locality Sensitive Hashing to a large text dataset. This package should be able to state if both datasets are similar or not.

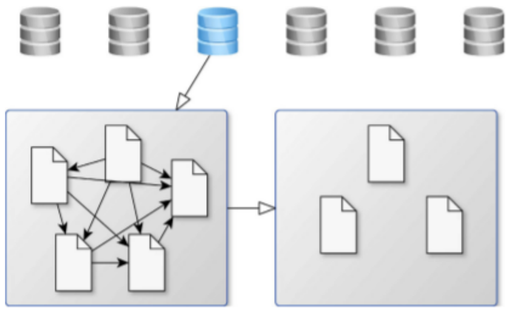

Fig: Candidate verification phase [2]

<u>Week 6 (July 12 - July 18)</u>

- I intend to start off this project by working primarily with text datasets as those are easier to work with. At this point, I will move to image datasets. This shouldn't be too different as the only step that will be different is during the first part when the image is split into shingles. The architecture and the algorithms (Min hash, Jaccard similarity) used will be the same.

<u>Week 7 (July 19 - July 25)</u>

- By this point, I hope to have a distributed system architecture with LSH working on comparing datasets. I intend to use test datasets to test the accuracy of my algorithm. Adjusting the parameters used in LSH changes its accuracy and speed. I would like to better understand the optimal conditions by testing with several datasets.
- This will also be a great time to fix any remaining issues with the distributed system architecture.
- Prepare test datasets (text and image) to test the efficacy of the program I've implemented.

<u>Week 8 (July 26 - August 1)</u>

- Generate documentation and add robust test cases for my Python package. I will be adding comments as I code, this step only formalizes the whole process by generating a documentation.

Week 9 (August 2 - August 8) and Week 10 (August 9 - August 15)

- Final review. This period will be used to complete leftover tasks and polish documentation and test cases. Integration/QA will also have been completed.
- **Milestone:** The final deliverable will be a Python package (with PySpark/Hadoop dependency and MapReduce implemented) which can compare large text and image datasets for similarities.

Future Improvements and Contributions

- I hope to continue working on this project even after the end of GSoC. I would like to extend to other types of datasets (non image and text) and would like to analyze the performance of altering the parameters of the hashing algorithm. This could also be an interesting research topic to explore during my graduate studies.

4. Personal Information

I graduated in 2020 with a Bachelor's degree in Computer Science from Jacobs University Bremen. I will be starting as a Master's student in Informatics at TU Munich from October 2021.

The programming language I am most comfortable with is C++ and Python. I've used Python for:
- Programming assignments in courses such as, "Machine Learning", "Programming in Python", "Introduction to Deep Learning"
- Writing scripts for visualising information (https://github.com/rahulbshrestha/thesis/tree/master/analyse)
- Other smaller scripts for processing text files and debugging logs.

Résumé

https://drive.google.com/file/d/1PCrTMArfPKzXyStY0MgX0URD28h8aZ4t/view?usp=sharing

Why did you choose Hub?

Hub is one of the few organizations working on machine learning infrastructure. Data is an important (and underrated) part of machine learning. Working at Hub aligns with my personal career goals of working as an ML engineer. This experience would be invaluable for me when I am dealing with similar machine learning infrastructure projects in the future.

This project perfectly aligns with my goals. I will be starting my Master's degree from October onwards and I intend to specialize in machine learning and distributed systems. The project I've picked is challenging, therefore, I've done quite a bit of research for this proposal. I didn't want to work on an easy project as I won't learn much on a shallow learning curve. Working on this project will help strengthen my skills on data engineering and distributed computing. I enjoyed the research I had to do to work on this proposal. There aren't lots of papers online explaining ways to implement hashing techniques for large datasets so I asked a question on r/MachineLearning. That led me down a rabbit hole of tons of hashing techniques. I finally settle on Locality Sensitive Hashing as it seems to be the most promising of them all.

Overall, I am looking forward to working on this project which will definitely be a great learning experience!

Are you part of an underrepresented group in STEM?

No.

Other commitments

No other commitments during this time period. I'm currently working as a data science intern which started in March and ends in May. I will be starting graduate school from October onwards so I will be free from June to September.

Code I've worked on

Some projects I've worked on that are on my Github:

https://github.com/rahulbshrestha/pointcloudbuilder (C++)
https://github.com/rahulbshrestha/email-service (Python)
https://github.com/rahulbshrestha/spi (C)

The work I'm most proud of is the research I did for my Bachelor's thesis. Since the thesis was done at the German Research Center for AI, the code is on a private repo. I've explained the algorithm I used in my thesis. I'm allowed to demo the code so it is available upon request!

Python test scripts and files I used for my thesis: https://github.com/rahulbshrestha/thesis
Thesis: https://github.com/rahulbshrestha/thesis/blob/master/thesis.pdf

5. References

[1]
https://medium.com/kalibrr-research/locality-sensitive-hashing-lsh-a-scalable-solution-for-deduplicating-jobs-from-multiple-sources-dd23460432de#:~:text=Locality%20Sensitive%20Hashing%20(LSH)%20%E2%80%94%20the%20scalable%20technique&text=The%20LSH%20method%20can%20perform,an%20equivalent%20tf%2Didf%20transformation.

[2] Szmit, R. (2013). *Locality Sensitive Hashing for Similarity Search Using MapReduce on Large Scale Data. Lecture Notes in Computer Science, 171–178.* doi:10.1007/978-3-642-38634-3_19
https://link.springer.com/chapter/10.1007/978-3-642-38634-3_19

[3] https://en.wikipedia.org/wiki/Jaccard_index