CS-GY 6513 Big Data Project Proposal

Yamini L, yl9822
Sakthi U, sm10539
Geetika B, gb2642
Ramanarayanan S, rs8117
Anudeep T, at5373

Idea #1
Reverse Image Search

Objective

While training Machine Learning models, often times when we need to fine tune the model ‘s prediction accuracy, one of the first steps is to increase the amount of relevant data. For most ML models, the more practice it gets, the better it learns. While it is possible to give the entire dataset to the model and train again, it is more efficient to fine tune the last few layers of the model with the additional relevant data that is needed. For example, self-driving cars use frames from videos as images to make decisions about the trajectory path to be taken. To get the images that are similar to a new condition (different weathers, similar traffic signs), it is efficient to address this as a big data problem.

Problem Statement

Reverse image search employs image similarity algorithms to discover groups of images that could be used for fine-tuning the Machine Learning algorithms. The model would take as input an image and output a group of similar images, here the similarity could be content-based, pixel based, etc. To get similar images, it is necessary to dig a corpus of huge amounts of information and that's where big data comes into play.

There are various approaches to perform similarity matching not limited to:

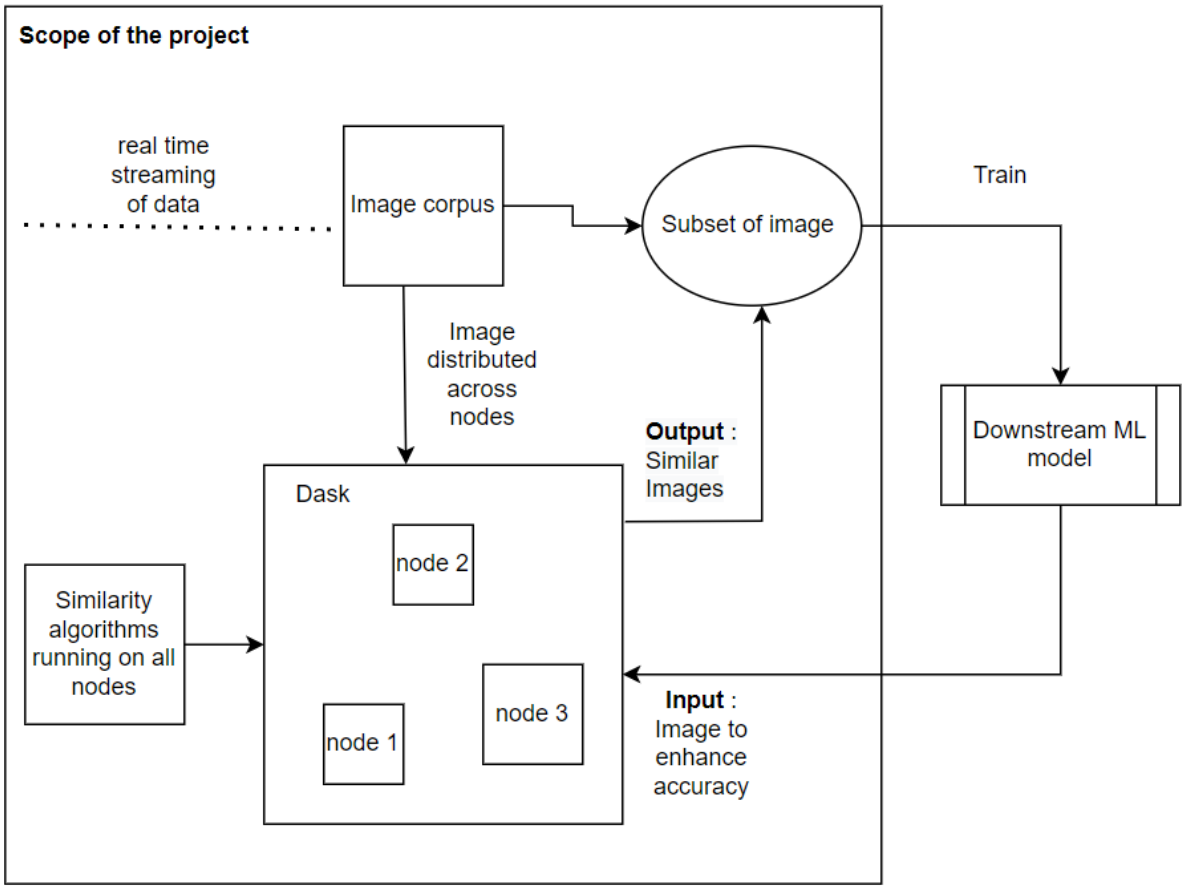
1. Non Model Approaches
    a. Cosine similarity
2. Model approach
    a. Nearest neighbor algorithm
    b. Embedding or vectors from one of the inner layers in the model and perform similarity matching.

We are planning to use DASK where each node will have a subset of the entire corpus and employ one of the above algorithms to determine the similarity. The output from all the nodes would be the list of images similar to the input image.

DataSet

1. https://image-net.org/
2. https://public.roboflow.com/object-detection/self-driving-car
3. https://www.nuscenes.org/nuimages

High level architecture:



Idea #2
E-commerce Recommender System

Problem Statement

Recommending relevant products to consumers on e-commerce websites leads to a win-win for the consumer and the retailer. Generally, it is seen that good recommendations emerge from models trained on massive amounts of data. However, owing to the computational complexity of ML algorithms, it is quite expensive to train and update such models with the time constraints of the real world.

Objective

We intend to implement a pipeline to efficiently train and update recommendation models based on frameworks like Spark and Dask. We are competing in the OTTO Multi-objective RecSys Competition on Kaggle and building upon their dataset. Though the dataset has a fixed number of samples to use for training, we may, if time permits, tweak our pipeline to train on-the-go by simulating a stream of data.
A small part of our motivation also stems from the cash prize we may win from this competition (up to $15,000).

Dataset Description

A massive real-world dataset released by OTTO, a German e-commerce company.

Attribute	Value
Total User Sessions	12 million
Total events within all sessions	220 million
Total products	1.2 million

Each session contains a number of events with each event bearing three attributes. Each event comprises the following

event_type	timestamp	id
click, *add to cart*, or *order*	The UNIX timestamp of the event occurrence	ID of the product with which the user interacted

Metrics

For testing, we are to predict the next clicks, add-to-carts, and orders of a truncated session. We may only predict up to 20 items for each event type.
Recommendation systems are often measured in terms of top-20 recall. That is,

$$recall@20 = \frac{Number\ of\ relevant\ predictions\ in\ top{-}20\ predictions}{Number\ of\ total\ relevant\ items}$$

This gives us an estimate of the goodness of ranking of recommendations.

Proposed Outline

Please note that we intend to formulate this as a classification problem initially.
We propose a rough roadmap consisting of the following steps

1. Thoroughly explore the data using analytics from PySpark.
2. Select a much smaller subset of the data at random for quick prototyping.
3. Model the data **without timestamps** to use simple ML algorithms like Naive Bayes, Logistic Regression and Decision Trees.
4. Model the problem as a time-series one to use more complex algorithms like LightGBM and LSTM networks.

(*) In parallel to 3 and 4, we will scale such algorithms to run on the whole dataset with the help of Spark MLlib and Dask-ML.

Potential Learning

We believe we would learn the following from this project

- ➔ Better understanding of PySpark analytics
- ➔ Selecting subsets of massive data that closely represent the whole data
- ➔ Better understanding of PySpark and Dask architectures and usage
- ➔ To develop our own variations of algorithms in PySpark and Dask ML libraries
- ➔ A general sense of problems involving big data and recommendation systems in the real world

Such learning is highly applicable in today's world, and therefore we choose this as one of our ideas.

References

Self Driving/ Image Search

1. https://blog.cambridgespark.com/50-free-machine-learning-datasets-self-driving-cars-d37be5a96b28
2. https://scale.com/self-driving-cars
3. https://www.instructables.com/Self-Driving-Car-With-Udacity-Simulator/
4. https://level-5.global/data/
5. https://archive.org/details/comma-dataset

6. https://www.oreilly.com/library/view/practical-deep-learning/9781492034858/ch04.html
7. https://www.sciencedirect.com/science/article/pii/S0386111219301566

Multi-objective E-commerce Recommendation:

Overview, data, and evaluation procedure - https://www.kaggle.com/competitions/otto-recommender-system/
Metrics (Recall@k) - https://medium.com/@m_n_malaeb/recall-and-precision-at-k-for-recommender-systems-618483226c54