Chandrashish Prasad (MDS202015)
Prasun Agarwal (MDS202028)
Yash Jain (MDS202048)

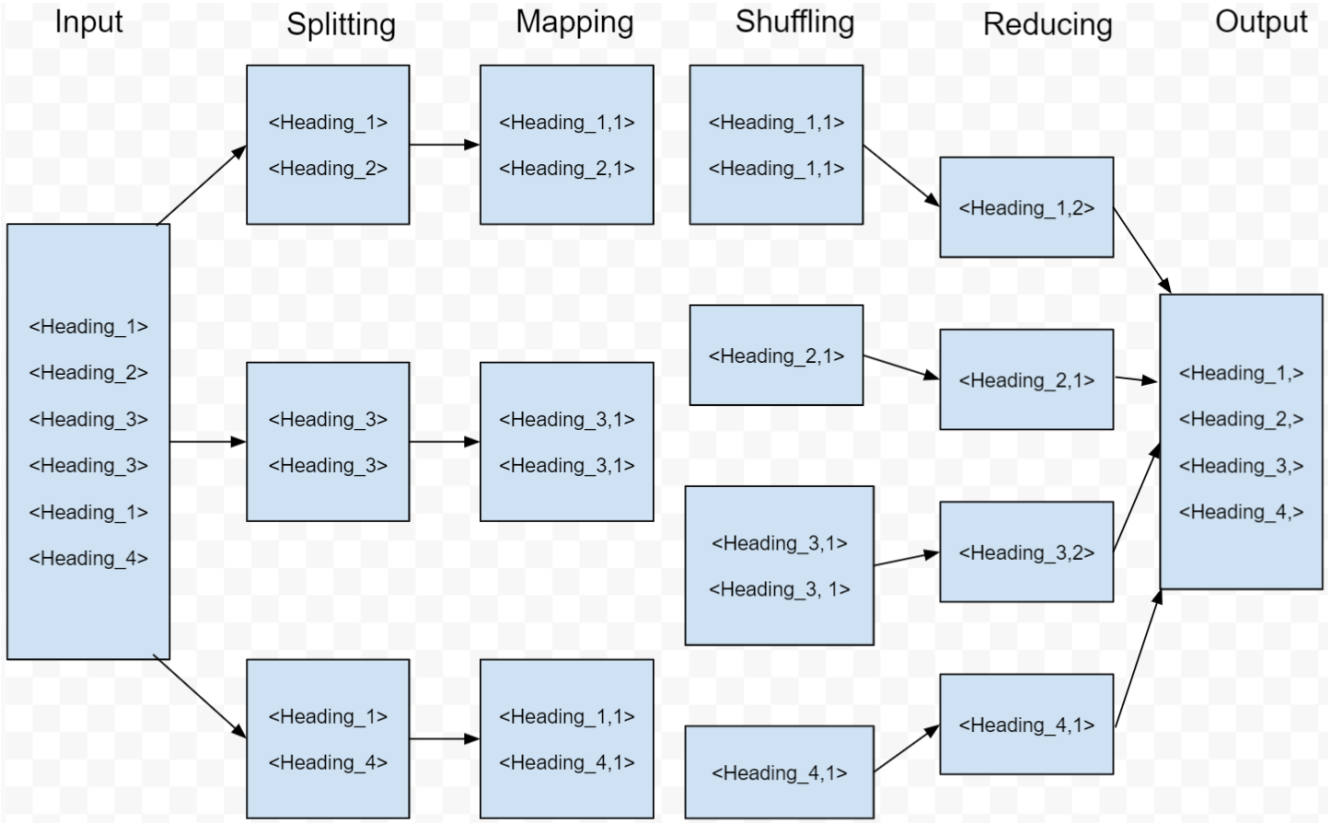
BIG DATA PROJECT

30th June 2021

GOALS

1. Ingesting articles into hadoop HDFS
2. Write a map-reduce code to remove news articles with duplicate headlines.

OVERVIEW



REQUIREMENTS

- Google Cloud Platform(GCP)
- The headlines data was extracted from the Kaggle in json Format.

PREPROCESSING

- For the given Data, we extracted all headlines in a txt file using python and imported the txt file into GCP.
- Making a mapper code: Take input from STDIN and print input with it's count as a key-value pair.
- Making a reducer code: Take input from STDIN (sorted output of the mapper) and output the non-duplicate headlines.

PROCEDURE

Getting Started

1) docker pull cloudera/quickstart:latest (Install cloudera on docker)
2) docker images (Note down the relevant image_id)
3) docker run --hostname=quickstart.cloudera --privileged=true -t -i -p 8777:8888 -p 7190:7180 -p 90:80 <image_id> /usr/bin/docker-quickstart (Run the cloudera image to start working with hadoop)
4) docker ps (for viewing container ID in a new console)

Ingesting data into hadoop HDFS

1) docker cp headlines.txt <container id>:/root (This command will copy the file from Local to the Hadoop container)
2) docker cp mapper.py <container id>:/root
3) docker cp reducer.py <container id>:/root

Check all files in hadoop directory

1) hadoop fs -mkdir MRDemo (Make the directory Project in Root folder)
2) hdfs dfs -ls MRDemo(To view files in Project Directory in Hadoop)

Putting files in directory

1) hdfs dfs -put headlines.txt MRDemo

Note that mapper.py and reducer.py are copied in Root Node folder and not in the Project Directory

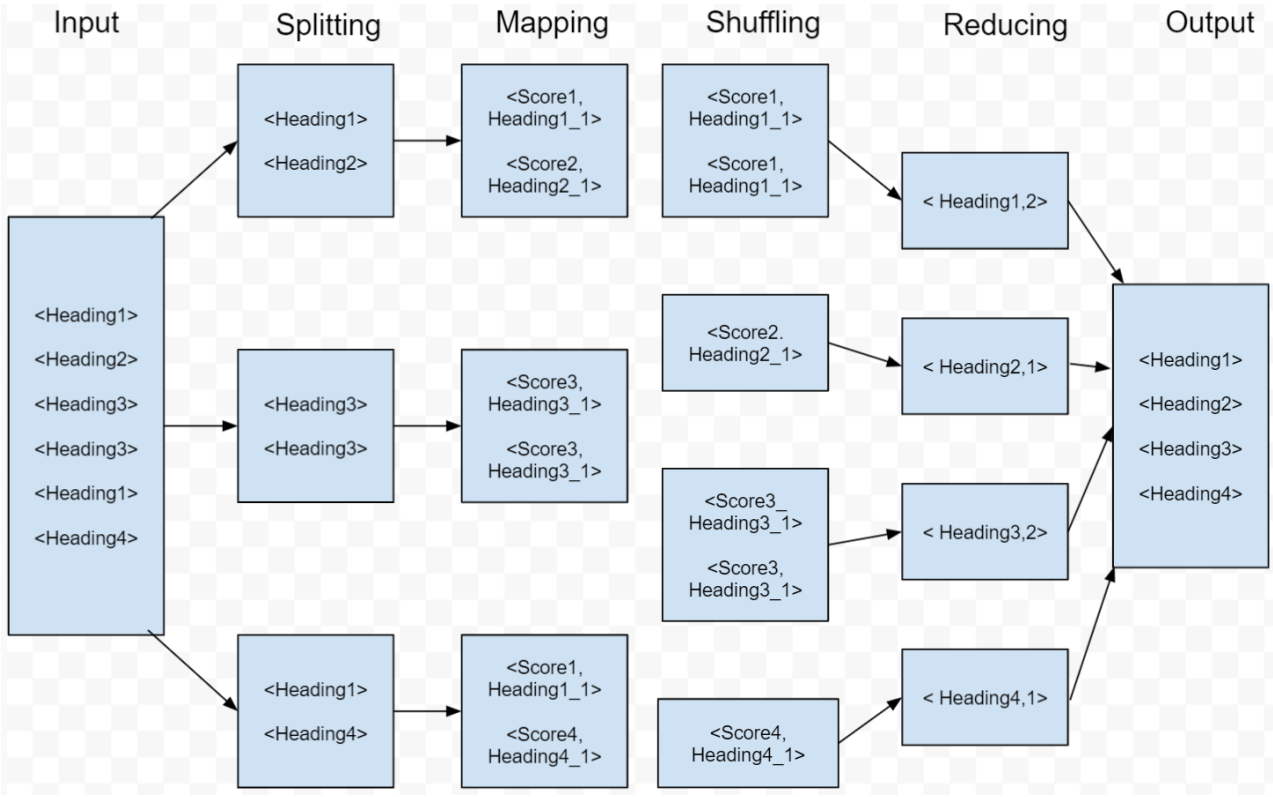
Running Map-reduce code and saving output

1) cat headlines.txt | python mapper.py | sort -k1,1 | python reducer.py (to run the MR job)
2) cat headlines.txt | python mapper.py | sort -k1,1 | python reducer.py &> output.txt (to run the MR job as well as save the final output)

FUTURE WORK

Here, we have considered duplicate headlines in a Syntactic sense. There can be some headlines which are syntactically different but convey the same (similar) information semantically.

- We firstly define a **semantic score** which captures the semantic essence of the headline using the number of words, nouns, grammar, etc. present in the headline. To achieve this we can leverage the pre-trained models like **BERT** (Bidirectional Encoder Representations from Transformers) or use some modules available like **Spacy** to associate a score to an input sentence (here, headlines).
- **Mapper**: The Mapper code will be designed in a way so that it assigns this score to each headline. It outputs the <key, value> pair where key is the **semantic score** and value is **"headline_1"** where headline in the value is the input headline sentence.
- **Shuffling**: It will shuffle/sort the data using the scores (key) obtained above. Now we have a **sorted list** as per the key values.
- **Reducing**: This is the most important phase. The reducer code will assign a range and all documents within that particular **range** will be considered **semantically similar**.
- **Output**: Of all the semantically similar documents produced by the reducer from each range, any one of the headlines is produced as **output** (as all other headlines of this range are Semantically the same).
- In this way, most of the unique headlines (**semantically and syntactically unique**) are identified and produced as the output. However, the output will have a certain **accuracy** associated with it, where accuracy will be the fraction of number of unique headlines identified out of the actual number of unique headlines present in the input.
- The **diagram** below can be referred to understand the above procedure.



Note: Multiple instances of **Heading1** used in the diagram above are **not necessarily** same syntactically, but are similar semantically.

Github link: https://github.com/Chandrashish/Map-Reduce