VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

TEXT EXTRACTION FROM VIDEO USING HADOOP MAPREDUCE

LARGE SCALE DATA PROCESSING (CSE3025)

PROJECT

By,

Shantanu	17BCE1161
Vaibhav Aggarwal	17BCE1245
Tahim Khan	17BCE1203

Submitted To,

Prof. Ramesh Ragala

November 2019

CERTIFICATE

This is to certify that the project entitled "**Text Extraction from Video using Hadoop MapReduce**" that is being submitted for CAL in B. Tech **Large Scale Data Processing** (CSE3025) is a record of bonafide work done under my supervision. The contents of this project, in full or in parts, have neither been taken from any source nor have been submitted for any other CAL course.

Place: Chennai

Date: 23rd November 2019

Signature of Students:

Shantanu	17BCE1161
Vaibhav	17BCE1245
Tahim	17BCE1203

Signature of Faculty:

Prof. Ramesh Ragala

School of Computer Science and Engineering (SCSE)

ABTRACT

The aim of the project is to extract text from video using Hadoop MapReduce framework. A video is a collection of images rolled on one after another. These images are often called frames. So, video processing is basically dividing the video into frames and then processing collectively the latter, resulting in the processing of the former.

Image processing is the method which is utilized to handle different pictures. In this work, text is extracted from the videos. In the recent times different procedures has been proposed and among these characters location and make apply grouping on these recognized characters for player recognizing. The videos that are converted to the images can situated on many angles due to which precision of content extraction can be diminished. In this work, enhancement will be proposed using SIFT algorithm for player reorganization.

Common urban scene pictures contain numerous issues for character acknowledgment such as luminance noise, varying font styles or cluttered foundations. Recognizing and recognizing content in a characteristic scene could be a troublesome problem. Several procedures have been proposed to overcome these issues. These are, in any case, ordinarily based on a bottom-up plot, which gives a parcel of wrong positives, untrue negatives and seriously computation. Subsequently, an elective, productive, character-based expectancy-driven strategy is required This paper presents a modelling approach that's usable for expectancy-driven methods based on the well-known Sift algorithm. The created models (Question Consideration Patches) are assessed in terms of their person provisory character recognition execution. Hence, the prepared fix models are utilized in preparatory tests on text detection in scene pictures. The comes about appear that our proposed model-based approach can be connected for a coherent SIFT-based content discovery and acknowledgment prepare.

KEYWORDS: Character Recognition, SIFT

INTRODUCTION

Conventional strategies that include the utilization of paper and write are still utilized by individuals to store important information indeed in spite of the fact that a few mechanical composing apparatuses have been outlined. Be that as it may, putting away and accessing these physical reports is exceptionally troublesome

Optical Character Recognition (OCR) is an important application of computer vision and is broadly applied for an assortment of elective purposes such as the recognition of road signs or buildings in common scenes. To recognize a text from photos, the characters first need to be distinguished, but the scene pictures contain many deterrents that influence the character identification performance Visual recognition issues, such as luminance clamor, shifting 2D and 3D text style styles or a cluttered foundation, cause troubles within the OCR process. In differentiate, scanned documents as a rule incorporate level, machine-printed characters, which are in conventional text style styles, have stable lighting, and are clear against a plain foundation. For these reasons, the OCR of the video graphic images is still a challenge.

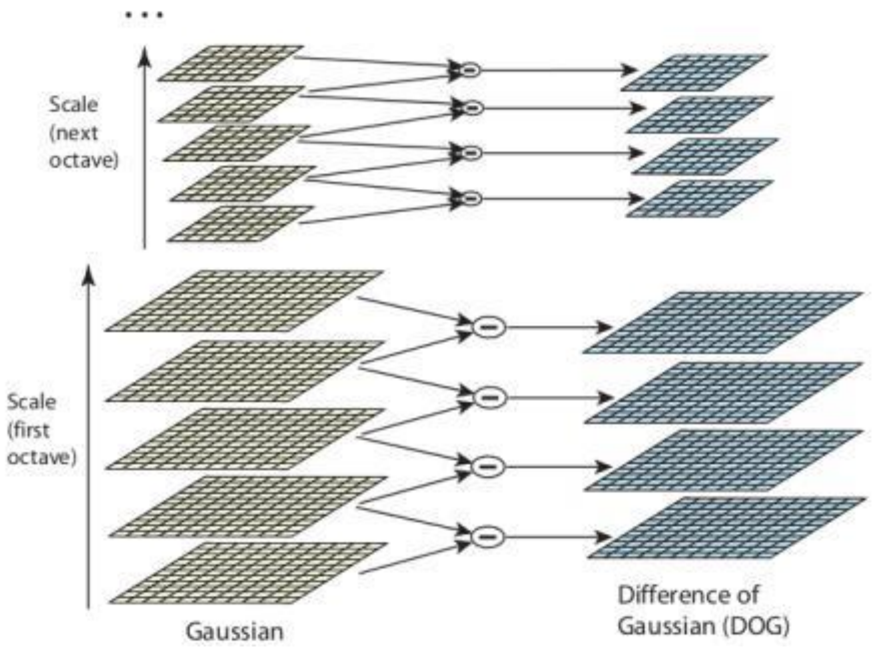
Numerous procedures to dispense with the specified OCR obstacles in scene pictures have been considered. For case, recognizing of and extracting objects from an assortment of background colors may be somewhat illuminated by color-based component analysis (Stop et al., 2007; Li et al., 2001). The trouble of text detection in a complex foundation can be overcome by utilizing the Stroke-Width Transform (Epshtein et al., 2010) and Stroke Gabor Words (Yi and Tian, 2011) methods.

A basic modeling approach would comprise of a full convolution of character demonstrate shapes along an image. Such an approach is restrictive: it would require to check for all the characters in a letter set, employing a number of layout sizes and of introduction variations. All of these forms would make the computation as well costly. In this manner, a quick invariant text detector would be alluring. A well-known technique for recognizing an object in a video graphic image is the Scale Invariant Feature Transform (Sift) (Lowe, 2004). It is computationally worthy, invariant and more advanced than a basic text-salience heuristic. In this manner, we address the address of whether Sift is usable for both content location and character acknowledgment.

This paper presents the character models' development from video graphic images and execution markers for discovery purposes, which can be utilized further to recognize a content in English.

Scale Invariant Feature Transform (SIFT)

The Sift procedure is created to fathom the issue of identifying images that are diverse in scale, rotation, perspective and brightening. The principle of Sift is that the image will be changed to scale-invariant coordinates relative to nearby features (Lowe, 2004). In arrange to get the Sift features, it begins with finding scale spaces of the initial image, the Difference-of-Gaussian (Burt and Adelson, 1983) work is computed to discover curiously key points, scales and introduction invariances.



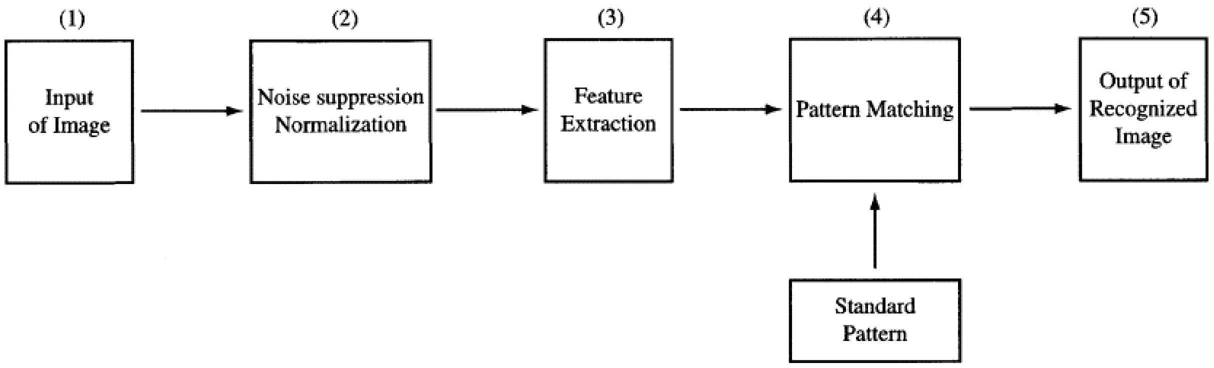
Another indicating the area where the exact key point is, a curiously point will be compared to its neighbors that then generally presents maxima and minima pixels in the image. These pixels can be utilized to create subpixel values in arrange to move forward the quality of the key point localization utilizing the Taylor expansion algorithm. The made strides key points are superior in coordinating and solidness due to this method. In any case, some low-contrast key points found along the edge, which are considered to be destitute highlights will be dispensed with.

After getting the key points, the local orientation of each key point will be doled out by collecting gradient directions and after that computing magnitude and orientation of the pixels around that key point. The result will be put into an introduction histogram, which has 360 degrees of orientation, and after that partitioned into 36 canisters. Any container rate that's higher than 80% (Lowe, 2004) will be assigned to the key point. At the end, image descriptors are made. The descriptors are computed utilizing the angle size and introduction around the key point. This calculation is executed from 16x16 pixels and gathered into 4x4 cells. Each cell will be utilized to create the 8-bin histogram. Finally, histogram values for all the cells will be combined into 128 descriptors and allotted as the key point descriptor.

An imperative parameter in Sift is the distance ratio threshold. Within the unique paper (Lowe, 2004), an ideal esteem of 0.8 is proposed. Be that as it may, the optimality of this threshold depends on the application. In preparing mode, wrong positive key points are the issue while in 'classification testing' mode, false negatives may be undesirable. Subsequently, we will utilize diverse values for this parameter within the diverse handling stages.

Feature Extraction

Each grayscale image within the dataset is calculated for its bounding box of the character and is at that point cropped based on its recognized box. The picture is extracted features by Sift that called key points (KP), which comprise of the coordinate (x, y), scale, introduction and 128 key point descriptors, and collected into a database. In order to extend the number of KPs they are too extracted from a binarized (B/W) duplicate of the character image. After getting all the key points, the original source images are now not required within the process. Only the key point vectors will be utilized.

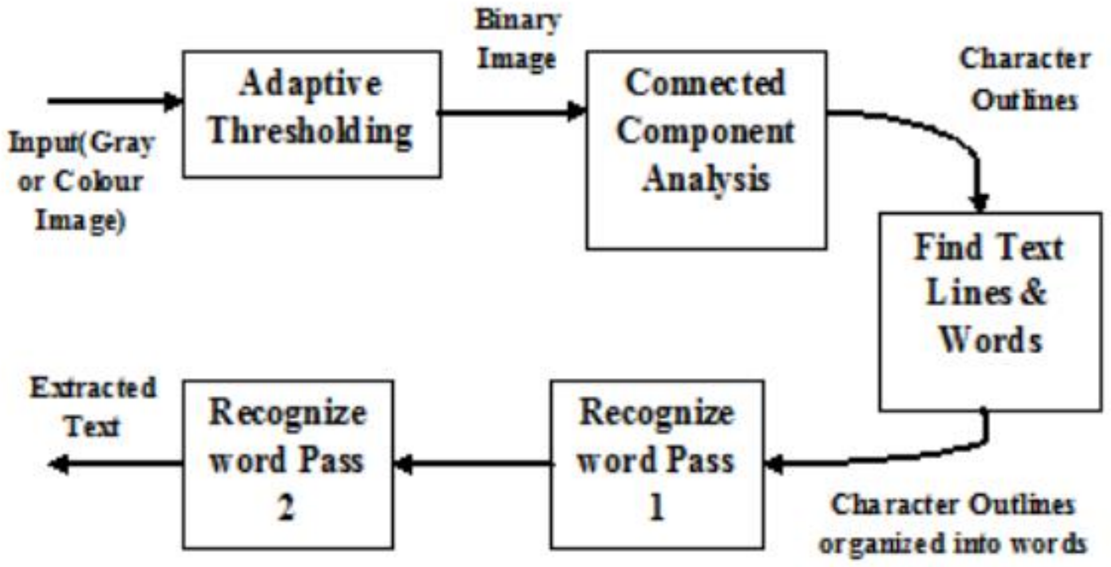


Tesseract - OCR

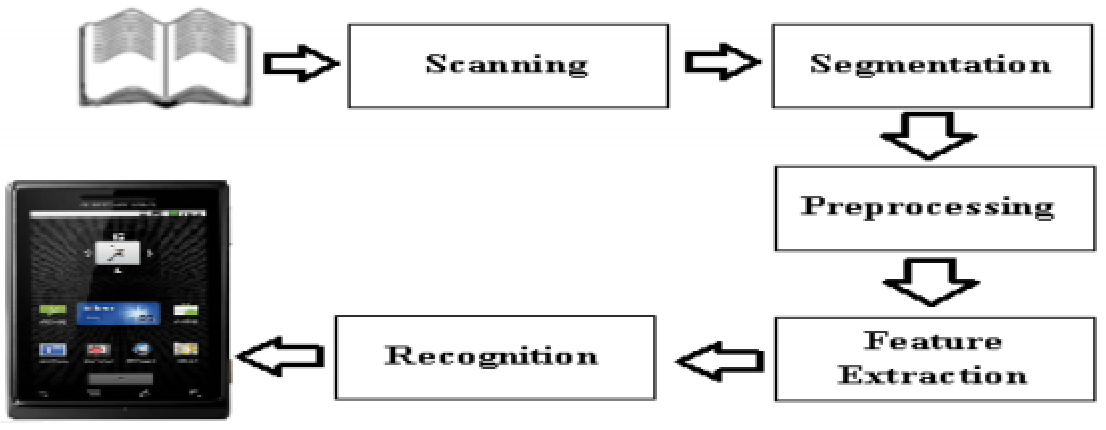
Tesseract is an open-source OCR engine that was developed at HP between 1984 and 1994. Like a supernova, it appeared from nowhere for the 1995 UNLV Annual Test of OCR Accuracy, shone brightly with its results, and then vanished back under the same cloak of secrecy under which it had been developed. Now for the first time, details of the architecture and algorithms can be revealed. Tesseract began as a PhD research project in HP Labs, Bristol, and gained momentum as a possible software and/or hardware add-on for HP's line of flatbed scanners. Motivation was provided by the fact that the commercial OCR engines of the day were in their infancy, and failed miserably on anything but the best quality print.

After a joint project between HP Labs Bristol, and HP's scanner division in Colorado, Tesseract had a significant lead in accuracy over the commercial engines, but did not become a product. The next stage of its development was back in HP Labs Bristol as an investigation of OCR for compression. Work concentrated more on improving rejection efficiency than on base-level accuracy. At the end of this project, at the end of 1994, development ceased entirely. The engine was sent to UNLV for the 1995

Annual Test of OCR Accuracy, where it proved its worth against the commercial engines of the time. In late 2005, HP released Tesseract for open source. It is now available at https://code.google.com/p/tesseract-ocr.



Architecture of Tesseract



OCR (Optical Character Recognition)

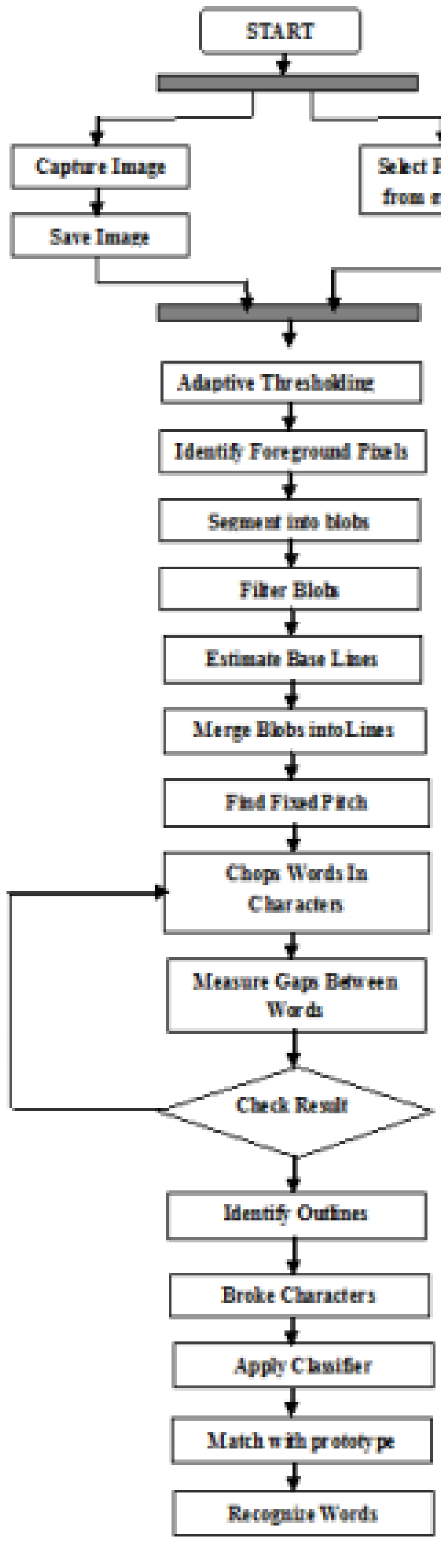


Image to text

IMPLEMENTATION

The aim of the project is fulfilled in two ways:

1. Hadoop MapReduce + OpenCV + Tesseract – OCR on a lyrical song video
2. SIFT performed on single Image for Feature Extraction

Hadoop MapReduce + OpenCV + Tesseract – OCR

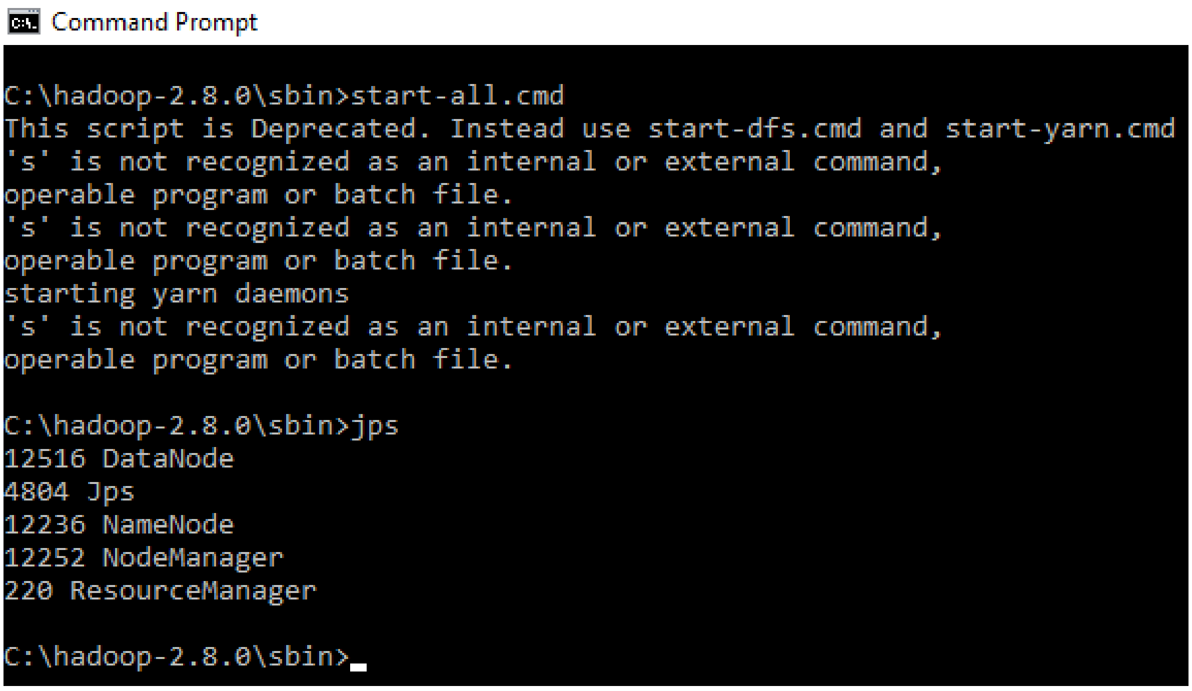
mapper.py

```
import cv2
import os
import shutil
import numpy as np
import matplotlib.pyplot as plt
import pytesseract
from PIL import Image
import os
import sys


for line in sys.stdin:
    line = line.strip()

    cam = cv2.VideoCapture(line)

    list_string = line.split('/')
    list_string_name = list_string[-1].split('.')
    y = '/'.join(list_string[:-1])
    y = y+'/'

    if os.path.exists(y+list_string_name[0]):
        shutil.rmtree(y+list_string_name[0])

    os.makedirs(y+list_string_name[0])

    currentframe = -1
    while(1):
        ret,frame = cam.read()
        if not ret:
            break
        else:
            currentframe += 1
            if currentframe%30 == 0:
                name    =    y+list_string_name[0]+'/frame'    +
str(currentframe) + '.jpg'
                cv2.imwrite(name, frame)
                print(name)


    # Release all space and windows once done

    cam.release()
    cv2.destroyAllWindows()
```

reducer.py

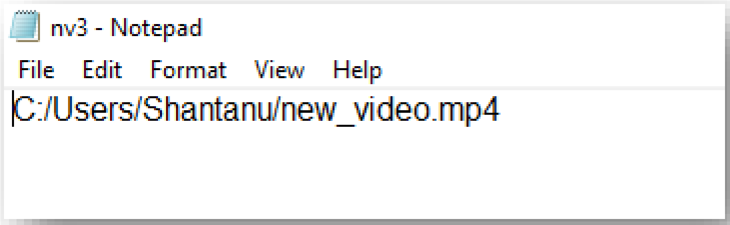
```
#!/usr/bin/env python
"""reducer.py"""

import cv2
import os
import shutil
import numpy as np
import matplotlib.pyplot as plt
import pytesseract
from PIL import Image
import os
import sys



# input comes from STDIN
for line in sys.stdin:
    line = line.strip()
    #print(line)
    #new_loc_hdfs =
'C:/Users/Shantanu/'+list_string_name[0]+'/frame60.jpg'
    img = cv2.imread(line, cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (1268,720))
    ret,thresh1 = cv2.threshold(img,170,255,cv2.THRESH_BINARY)
    kernel = np.ones((5,5),np.float32)/25
    dst = cv2.filter2D(thresh1,-1,kernel)
    pytesseract.pytesseract.tesseract_cmd = r"C:/Program Files
(x86)/Tesseract-OCR/tesseract.exe"
    img1 = Image.fromarray(dst)
    text = pytesseract.image_to_string(img1)
    if(text != ""):
        text = text.strip()
        text = text.split("\n")
        for t in text:
            if(t != "" and len(t)>5):
                print(t.encode("utf-8"))
        print('\n')
```

Start Hadoop in the system



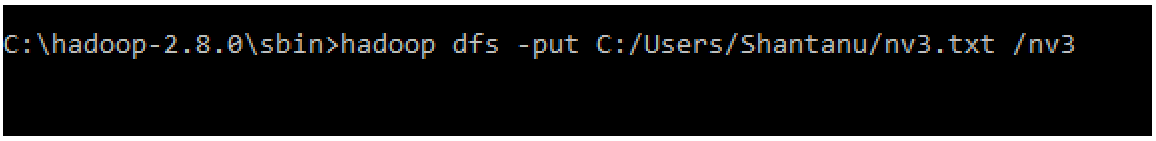
```
C:\hadoop-2.8.0\sbin>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
's' is not recognized as an internal or external command,
operable program or batch file.
's' is not recognized as an internal or external command,
operable program or batch file.
starting yarn daemons
's' is not recognized as an internal or external command,
operable program or batch file.

C:\hadoop-2.8.0\sbin>jps
12516 DataNode
4804 Jps
12236 NameNode
12252 NodeManager
220 ResourceManager

C:\hadoop-2.8.0\sbin>
```

Create a text file to store the location of the video:

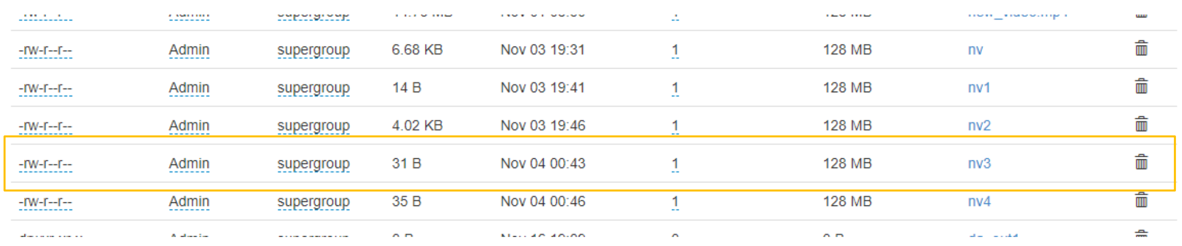


```
C:/Users/Shantanu/new_video.mp4
```

Upload this text file in HDFS:



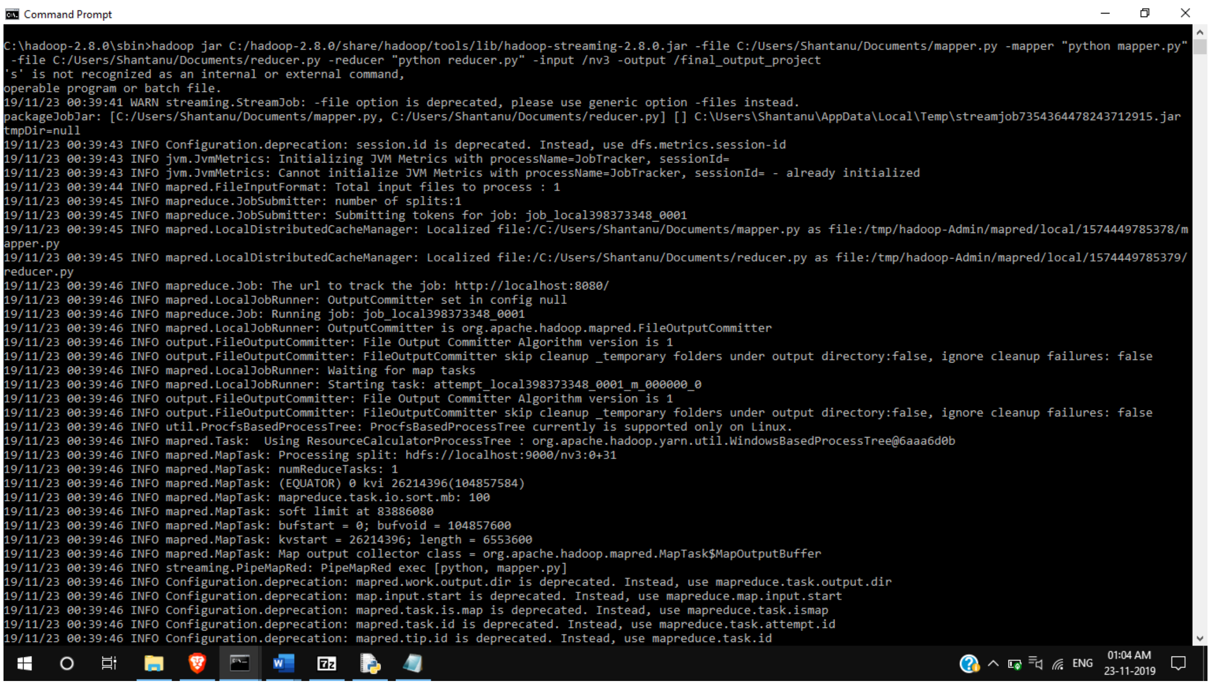
```
C:\hadoop-2.8.0\sbin>hadoop dfs -put C:/Users/Shantanu/nv3.txt /nv3
```



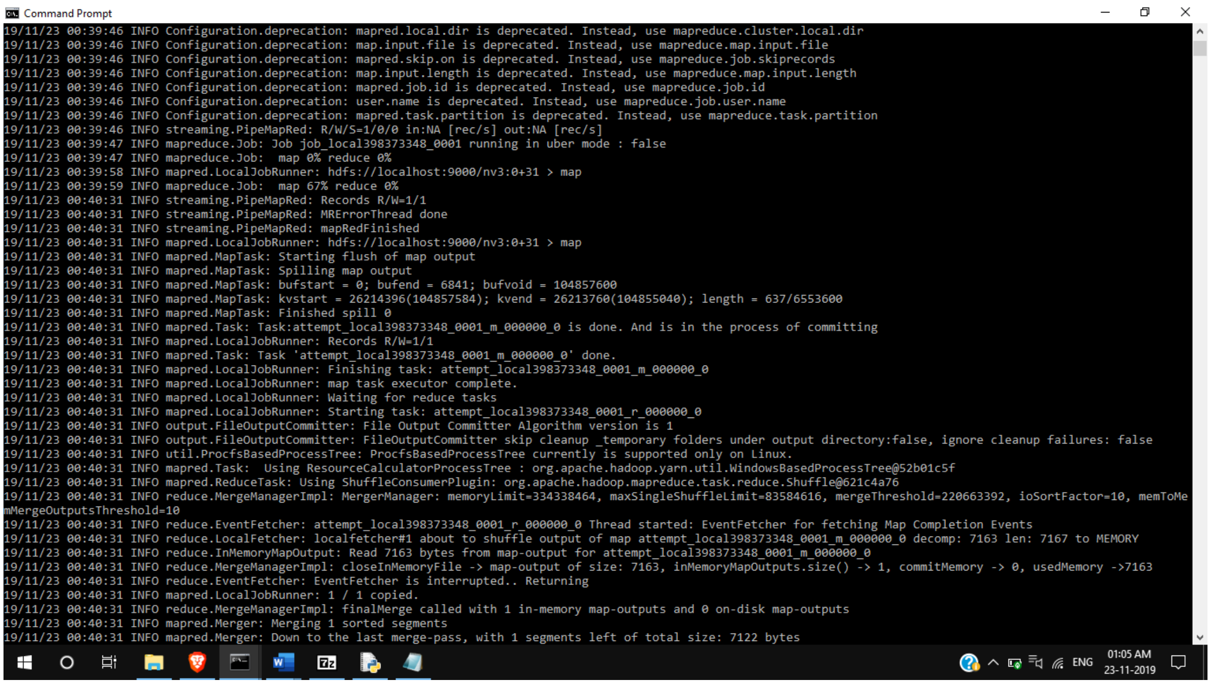
-rw-r--r--	Admin	supergroup	6.68 KB	Nov 03 19:31	1	128 MB	nv	🗑
-rw-r--r--	Admin	supergroup	14 B	Nov 03 19:41	1	128 MB	nv1	🗑
-rw-r--r--	Admin	supergroup	4.02 KB	Nov 03 19:46	1	128 MB	nv2	🗑
-rw-r--r--	Admin	supergroup	31 B	Nov 04 00:43	1	128 MB	nv3	🗑
-rw-r--r--	Admin	supergroup	35 B	Nov 04 00:46	1	128 MB	nv4	🗑

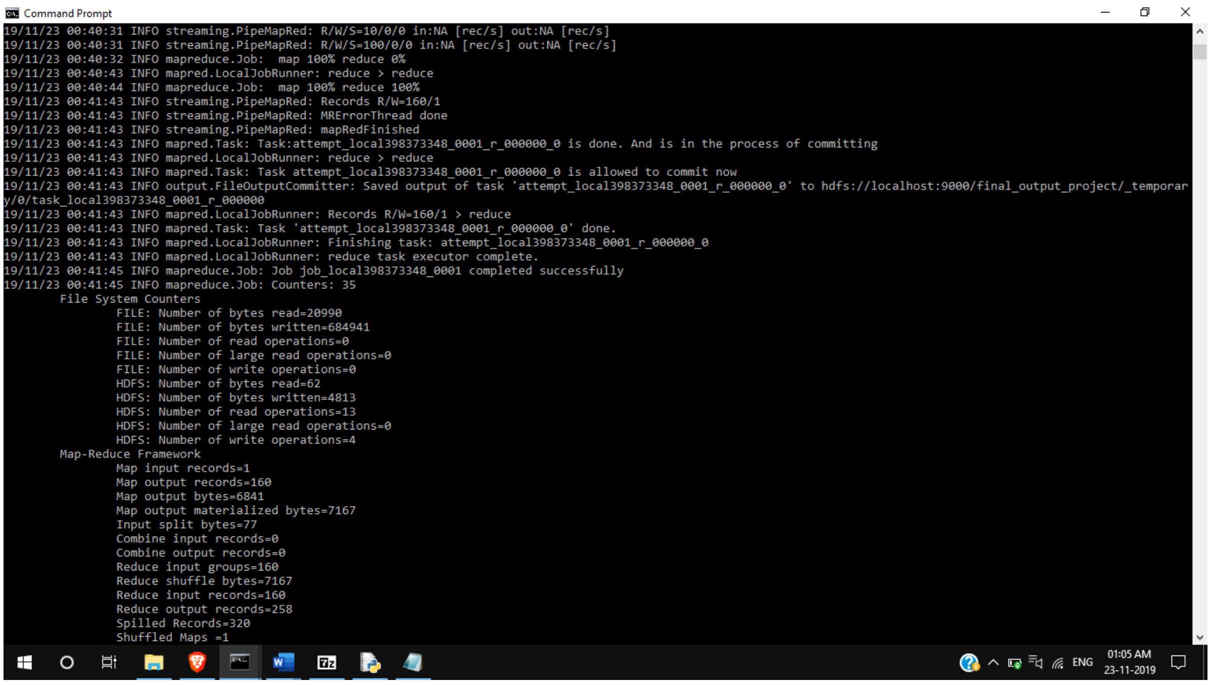
Run the mapper.py and reducer.py on the Hadoop Framework using the Hadoop streaming jar file in the following directory:

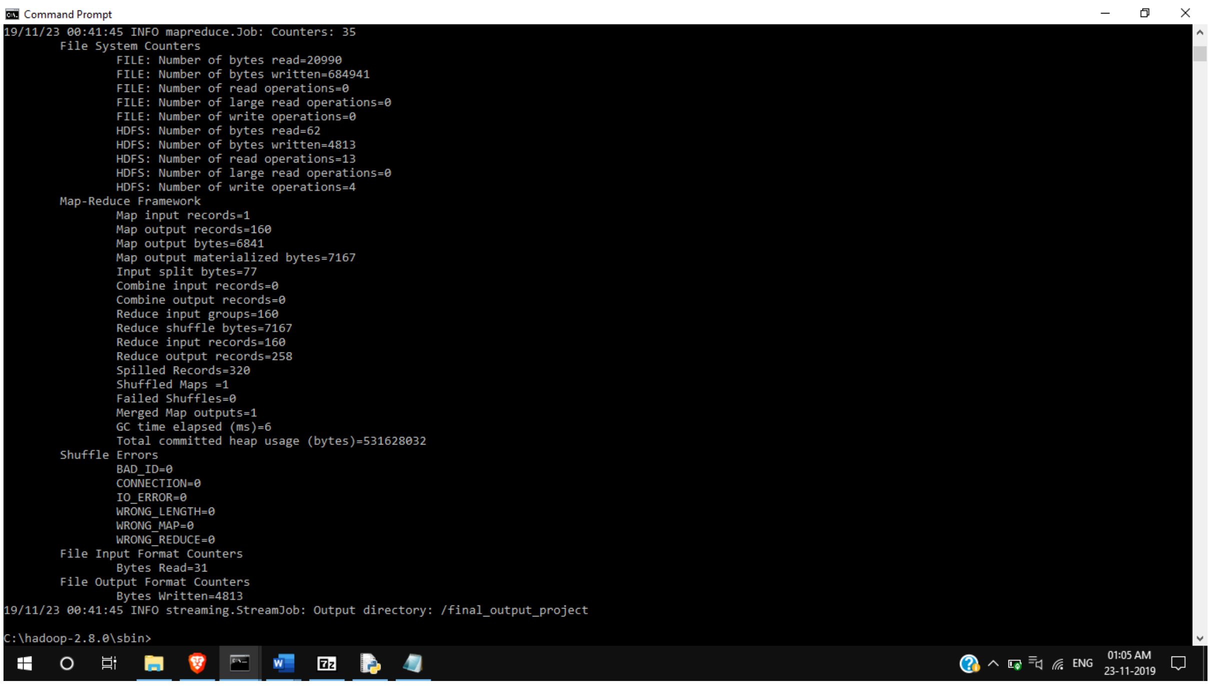
C:/Hadoop-2.8.0/share/hadoop/tools/lib/hadoop-streaming-2.8.0.jar



```
C:\hadoop-2.8.0\sbin>hadoop jar C:/hadoop-2.8.0/share/hadoop/tools/lib/hadoop-streaming-2.8.0.jar -file C:/Users/Shantanu/Documents/mapper.py -mapper "python mapper.py"
 -file C:/Users/Shantanu/Documents/reducer.py -reducer "python reducer.py" -input /nv3 -output /final_output_project
's' is not recognized as an internal or external command,
operable program or batch file.
19/11/23 00:39:41 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [C:/Users/Shantanu/Documents/mapper.py, C:/Users/Shantanu/Documents/reducer.py] [] C:\Users\Shantanu\AppData\Local\Temp\streamjob7354364478243712915.jar
tmpDir=null
19/11/23 00:39:43 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
19/11/23 00:39:43 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
19/11/23 00:39:43 INFO jvm.JvmMetrics: Cannot initialize JVM Metrics with processName=JobTracker, sessionId= - already initialized
19/11/23 00:39:44 INFO mapred.FileInputFormat: Total input files to process : 1
19/11/23 00:39:45 INFO mapreduce.JobSubmitter: number of splits:1
19/11/23 00:39:45 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local398373348_0001
19/11/23 00:39:45 INFO mapred.LocalDistributedCacheManager: Localized file:/C:/Users/Shantanu/Documents/mapper.py as file:/tmp/hadoop-Admin/mapred/local/1574449785378/m
apper.py
19/11/23 00:39:45 INFO mapred.LocalDistributedCacheManager: Localized file:/C:/Users/Shantanu/Documents/reducer.py as file:/tmp/hadoop-Admin/mapred/local/1574449785379/
reducer.py
19/11/23 00:39:46 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
19/11/23 00:39:46 INFO mapred.LocalJobRunner: OutputCommitter set in config null
19/11/23 00:39:46 INFO mapreduce.Job: Running job: job_local398373348_0001
19/11/23 00:39:46 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
19/11/23 00:39:46 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
19/11/23 00:39:46 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
19/11/23 00:39:46 INFO mapred.LocalJobRunner: Waiting for map tasks
19/11/23 00:39:46 INFO mapred.LocalJobRunner: Starting task: attempt_local398373348_0001_m_000000_0
19/11/23 00:39:46 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
19/11/23 00:39:46 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
19/11/23 00:39:46 INFO util.ProcfsBasedProcessTree: ProcfsBasedProcessTree currently is supported only on Linux.
19/11/23 00:39:46 INFO mapred.Task:  Using ResourceCalculatorProcessTree : org.apache.hadoop.yarn.util.WindowsBasedProcessTree@6aaa6d0b
19/11/23 00:39:46 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/nv3:0+31
19/11/23 00:39:46 INFO mapred.MapTask: numReduceTasks: 1
19/11/23 00:39:46 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
19/11/23 00:39:46 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
19/11/23 00:39:46 INFO mapred.MapTask: soft limit at 83886080
19/11/23 00:39:46 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
19/11/23 00:39:46 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
19/11/23 00:39:46 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
19/11/23 00:39:46 INFO streaming.PipeMapRed: PipeMapRed exec [python, mapper.py]
19/11/23 00:39:46 INFO Configuration.deprecation: mapred.work.output.dir is deprecated. Instead, use mapreduce.task.output.dir
19/11/23 00:39:46 INFO Configuration.deprecation: map.input.start is deprecated. Instead, use mapreduce.map.input.start
19/11/23 00:39:46 INFO Configuration.deprecation: mapred.task.is.map is deprecated. Instead, use mapreduce.task.ismap
19/11/23 00:39:46 INFO Configuration.deprecation: mapred.task.id is deprecated. Instead, use mapreduce.task.attempt.id
19/11/23 00:39:46 INFO Configuration.deprecation: mapred.tip.id is deprecated. Instead, use mapreduce.task.id
```


```
19/11/23 00:39:46 INFO Configuration.deprecation: mapred.local.dir is deprecated. Instead, use mapreduce.cluster.local.dir
19/11/23 00:39:46 INFO Configuration.deprecation: map.input.file is deprecated. Instead, use mapreduce.map.input.file
19/11/23 00:39:46 INFO Configuration.deprecation: mapred.skip.on is deprecated. Instead, use mapreduce.job.skiprecords
19/11/23 00:39:46 INFO Configuration.deprecation: map.input.length is deprecated. Instead, use mapreduce.map.input.length
19/11/23 00:39:46 INFO Configuration.deprecation: mapred.job.id is deprecated. Instead, use mapreduce.job.id
19/11/23 00:39:46 INFO Configuration.deprecation: user.name is deprecated. Instead, use mapreduce.job.user.name
19/11/23 00:39:46 INFO Configuration.deprecation: mapred.task.partition is deprecated. Instead, use mapreduce.task.partition
19/11/23 00:39:46 INFO streaming.PipeMapRed: R/W/S=1/0/0 in:NA [rec/s] out:NA [rec/s]
19/11/23 00:39:47 INFO mapreduce.Job: Job job_local398373348_0001 running in uber mode : false
19/11/23 00:39:47 INFO mapreduce.Job:  map 0% reduce 0%
19/11/23 00:39:58 INFO mapred.LocalJobRunner: hdfs://localhost:9000/nv3:0+31 > map
19/11/23 00:39:59 INFO mapreduce.Job:  map 67% reduce 0%
19/11/23 00:40:31 INFO streaming.PipeMapRed: Records R/W=1/1
19/11/23 00:40:31 INFO streaming.PipeMapRed: MRErrorThread done
19/11/23 00:40:31 INFO streaming.PipeMapRed: mapRedFinished
19/11/23 00:40:31 INFO mapred.LocalJobRunner: hdfs://localhost:9000/nv3:0+31 > map
19/11/23 00:40:31 INFO mapred.MapTask: Starting flush of map output
19/11/23 00:40:31 INFO mapred.MapTask: Spilling map output
19/11/23 00:40:31 INFO mapred.MapTask: bufstart = 0; bufend = 6841; bufvoid = 104857600
19/11/23 00:40:31 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend = 26213760(104855040); length = 637/6553600
19/11/23 00:40:31 INFO mapred.MapTask: Finished spill 0
19/11/23 00:40:31 INFO mapred.Task: Task:attempt_local398373348_0001_m_000000_0 is done. And is in the process of committing
19/11/23 00:40:31 INFO mapred.LocalJobRunner: Records R/W=1/1
19/11/23 00:40:31 INFO mapred.Task: Task 'attempt_local398373348_0001_m_000000_0' done.
19/11/23 00:40:31 INFO mapred.LocalJobRunner: Finishing task: attempt_local398373348_0001_m_000000_0
19/11/23 00:40:31 INFO mapred.LocalJobRunner: map task executor complete.
19/11/23 00:40:31 INFO mapred.LocalJobRunner: Waiting for reduce tasks
19/11/23 00:40:31 INFO mapred.LocalJobRunner: Starting task: attempt_local398373348_0001_r_000000_0
19/11/23 00:40:31 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
19/11/23 00:40:31 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
19/11/23 00:40:31 INFO util.ProcfsBasedProcessTree: ProcfsBasedProcessTree currently is supported only on Linux.
19/11/23 00:40:31 INFO mapred.Task:  Using ResourceCalculatorProcessTree : org.apache.hadoop.yarn.util.WindowsBasedProcessTree@52b01c5f
19/11/23 00:40:31 INFO mapred.ReduceTask: Using ShuffleConsumerPlugin: org.apache.hadoop.mapreduce.task.reduce.Shuffle@621c4a76
19/11/23 00:40:31 INFO reduce.MergeManagerImpl: MergerManager: memoryLimit=334338464, maxSingleShuffleLimit=83584616, mergeThreshold=220663392, ioSortFactor=10, memToMe
mMergeOutputsThreshold=10
19/11/23 00:40:31 INFO reduce.EventFetcher: attempt_local398373348_0001_r_000000_0 Thread started: EventFetcher for fetching Map Completion Events
19/11/23 00:40:31 INFO reduce.LocalFetcher: localfetcher#1 about to shuffle output of map attempt_local398373348_0001_m_000000_0 decomp: 7163 len: 7167 to MEMORY
19/11/23 00:40:31 INFO reduce.InMemoryMapOutput: Read 7163 bytes from map-output for attempt_local398373348_0001_m_000000_0
19/11/23 00:40:31 INFO reduce.MergeManagerImpl: closeInMemoryFile -> map-output of size: 7163, inMemoryMapOutputs.size() -> 1, commitMemory -> 0, usedMemory ->7163
19/11/23 00:40:31 INFO reduce.EventFetcher: EventFetcher is interrupted.. Returning
19/11/23 00:40:31 INFO mapred.LocalJobRunner: 1 / 1 copied.
19/11/23 00:40:31 INFO reduce.MergeManagerImpl: finalMerge called with 1 in-memory map-outputs and 0 on-disk map-outputs
19/11/23 00:40:31 INFO mapred.Merger: Merging 1 sorted segments
19/11/23 00:40:31 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 7122 bytes
```

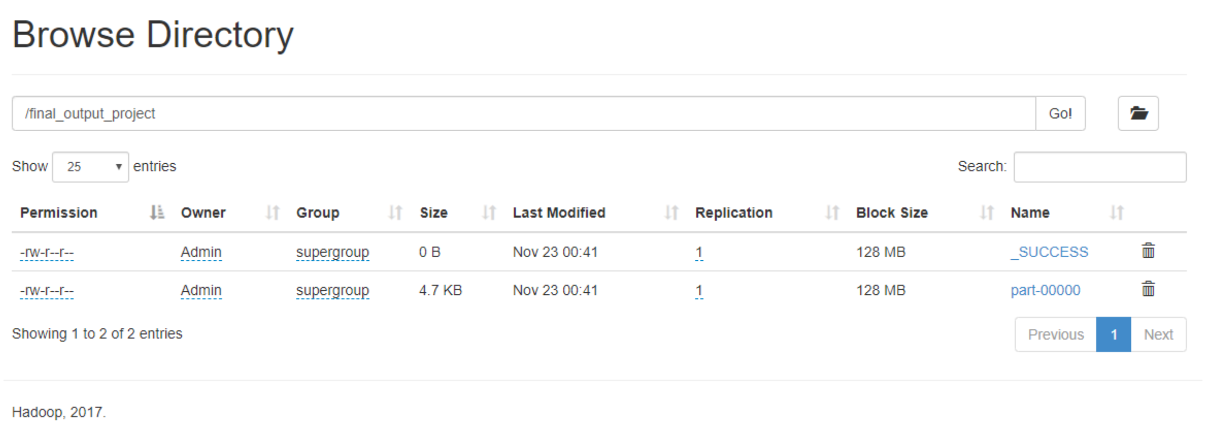

```
19/11/23 00:40:31 INFO streaming.PipeMapRed: R/W/S=10/0/0 in:NA [rec/s] out:NA [rec/s]
19/11/23 00:40:31 INFO streaming.PipeMapRed: R/W/S=100/0/0 in:NA [rec/s] out:NA [rec/s]
19/11/23 00:40:32 INFO mapreduce.Job:  map 100% reduce 0%
19/11/23 00:40:43 INFO mapred.LocalJobRunner: reduce > reduce
19/11/23 00:40:44 INFO mapreduce.Job:  map 100% reduce 100%
19/11/23 00:41:43 INFO streaming.PipeMapRed: Records R/W=160/1
19/11/23 00:41:43 INFO streaming.PipeMapRed: MRErrorThread done
19/11/23 00:41:43 INFO streaming.PipeMapRed: mapRedFinished
19/11/23 00:41:43 INFO mapred.Task: Task:attempt_local398373348_0001_r_000000_0 is done. And is in the process of committing
19/11/23 00:41:43 INFO mapred.LocalJobRunner: reduce > reduce
19/11/23 00:41:43 INFO mapred.Task: Task attempt_local398373348_0001_r_000000_0 is allowed to commit now
19/11/23 00:41:43 INFO output.FileOutputCommitter: Saved output of task 'attempt_local398373348_0001_r_000000_0' to hdfs://localhost:9000/final_output_project/_temporar
y/0/task_local398373348_0001_r_000000
19/11/23 00:41:43 INFO mapred.LocalJobRunner: Records R/W=160/1 > reduce
19/11/23 00:41:43 INFO mapred.Task: Task 'attempt_local398373348_0001_r_000000_0' done.
19/11/23 00:41:43 INFO mapred.LocalJobRunner: Finishing task: attempt_local398373348_0001_r_000000_0
19/11/23 00:41:43 INFO mapred.LocalJobRunner: reduce task executor complete.
19/11/23 00:41:45 INFO mapreduce.Job: Job job_local398373348_0001 completed successfully
19/11/23 00:41:45 INFO mapreduce.Job: Counters: 35
        File System Counters
                FILE: Number of bytes read=20990
                FILE: Number of bytes written=684941
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=62
                HDFS: Number of bytes written=4813
                HDFS: Number of read operations=13
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=4
        Map-Reduce Framework
                Map input records=1
                Map output records=160
                Map output bytes=6841
                Map output materialized bytes=7167
                Input split bytes=77
                Combine input records=0
                Combine output records=0
                Reduce input groups=160
                Reduce shuffle bytes=7167
                Reduce input records=160
                Reduce output records=258
                Spilled Records=320
                Shuffled Maps =1
```



```
19/11/23 00:41:45 INFO mapreduce.Job: Counters: 35
        File System Counters
                FILE: Number of bytes read=20990
                FILE: Number of bytes written=684941
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=62
                HDFS: Number of bytes written=4813
                HDFS: Number of read operations=13
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=4
        Map-Reduce Framework
                Map input records=1
                Map output records=160
                Map output bytes=6841
                Map output materialized bytes=7167
                Input split bytes=77
                Combine input records=0
                Combine output records=0
                Reduce input groups=160
                Reduce shuffle bytes=7167
                Reduce input records=160
                Reduce output records=258
                Spilled Records=320
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=6
                Total committed heap usage (bytes)=531628032
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=31
        File Output Format Counters
                Bytes Written=4813
19/11/23 00:41:45 INFO streaming.StreamJob: Output directory: /final_output_project

C:\hadoop-2.8.0\sbin>
```

Output:

Checking the HDFS:



Browse Directory

/final_output_project Go!

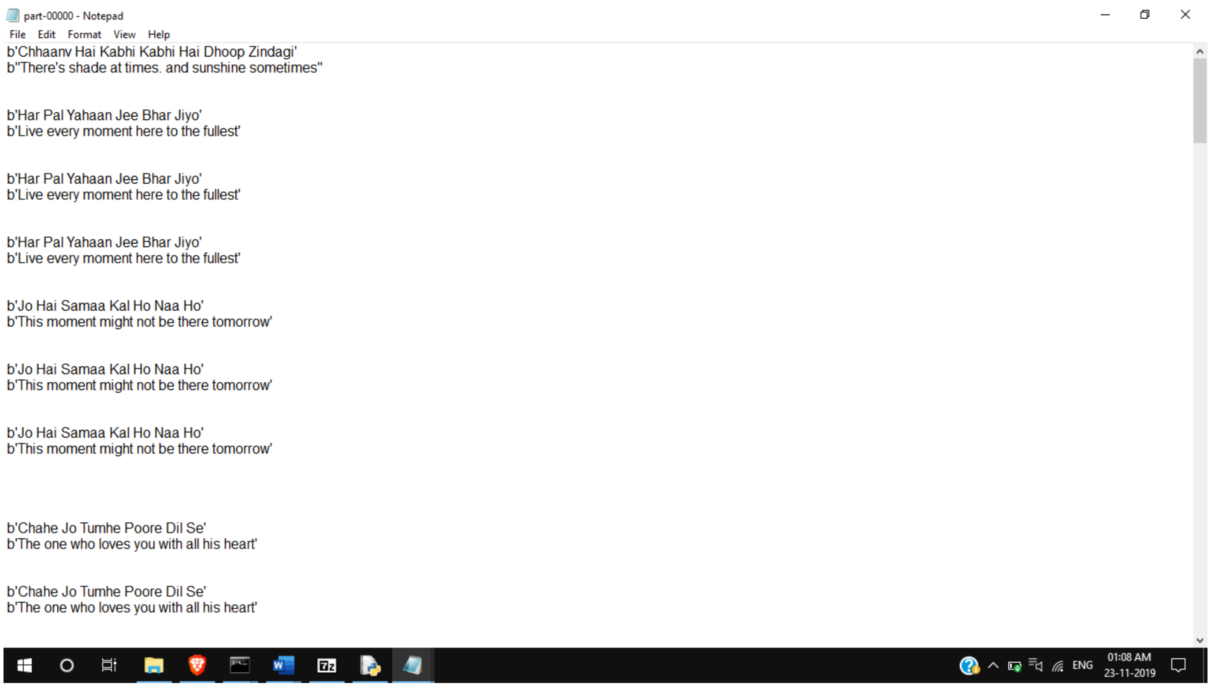
Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	Admin	supergroup	0 B	Nov 23 00:41	1	128 MB	_SUCCESS
-rw-r--r--	Admin	supergroup	4.7 KB	Nov 23 00:41	1	128 MB	part-00000

Showing 1 to 2 of 2 entries

Hadoop, 2017.

Opening the part-r-00000 file using a text editor.



```
b'Chhaanv Hai Kabhi Kabhi Hai Dhoop Zindagi'
b"There's shade at times, and sunshine sometimes"

b'Har Pal Yahaan Jee Bhar Jiyo'
b'Live every moment here to the fullest'

b'Har Pal Yahaan Jee Bhar Jiyo'
b'Live every moment here to the fullest'

b'Har Pal Yahaan Jee Bhar Jiyo'
b'Live every moment here to the fullest'

b'Jo Hai Samaa Kal Ho Naa Ho'
b'This moment might not be there tomorrow'

b'Jo Hai Samaa Kal Ho Naa Ho'
b'This moment might not be there tomorrow'

b'Jo Hai Samaa Kal Ho Naa Ho'
b'This moment might not be there tomorrow'


b'Chahe Jo Tumhe Poore Dil Se'
b'The one who loves you with all his heart'

b'Chahe Jo Tumhe Poore Dil Se'
b'The one who loves you with all his heart'
```

The link for the video used in this execution is as follows:

https://youtu.be/sjQZkE8o_oY

SIFT on a sample Image for Feature Extraction

main.py

```
from skimage.io import imread
from sift import SIFT
import cv2
import pickle
from os.path import isfile

if __name__ == '__main__':
     num_img = 3

     kp_pyrs = []
     ims = []

     im = imread('img2.jpg')
     im = cv2.resize(im, (1268,720))
     ims.append(im)

     if isfile('kp_pyr.pkl'):
          kp_pyrs.append(pickle.load(open('kp_pyr.pkl',
'rb')))


     print('Performing SIFT on image')

     sift_detector = SIFT(im)
     _ = sift_detector.get_features()
     kp_pyrs.append(sift_detector.kp_pyr)

     pickle.dump(sift_detector.kp_pyr, open('kp_pyr.pkl',
'wb'))
     pickle.dump(sift_detector.feats, open('feat_pyr.pkl',
'wb'))

     import matplotlib.pyplot as plt


     _, ax = plt.subplots(1, 1)
     ax.imshow(ims[0])

     kps = kp_pyrs[0][0]*(2**0)
     ax.scatter(kps[:,0], kps[:,1], c='b', s=2)
     #plt.show()
     plt.savefig('plot.png')
```

sift.py

```
from skimage.color import rgb2gray
from scipy.ndimage.filters import convolve

from gaussian_filter import gaussian_filter
from gaussian_pyramid import generate_gaussian_pyramid
from DoG_pyramid import generate_DoG_pyramid
from keypoints import get_keypoints
from orientation import assign_orientation
from descriptors import get_local_descriptors

class SIFT(object):
    def __init__(self, im, s=3, num_octave=4, s0=1.3, 
sigma=1.6, r_th=10, t_c=0.03, w=16):
        self.im = convolve(rgb2gray(im), gaussian_filter(s0))
        self.s = s
        self.sigma = sigma
        self.num_octave = num_octave
        self.t_c = t_c
        self.R_th = (r_th+1)**2 / r_th
        self.w = w

    def get_features(self):
        gaussian_pyr = generate_gaussian_pyramid(self.im, 
self.num_octave, self.s, self.sigma)
        DoG_pyr = generate_DoG_pyramid(gaussian_pyr)
        kp_pyr = get_keypoints(DoG_pyr, self.R_th, self.t_c, 
self.w)
        feats = []

        for i, DoG_octave in enumerate(DoG_pyr):
            kp_pyr[i] = assign_orientation(kp_pyr[i], 
DoG_octave)
            feats.append(get_local_descriptors(kp_pyr[i], 
DoG_octave))

        self.kp_pyr = kp_pyr
        self.feats = feats

        return feats
```

gaussian_filter.py

```
import numpy as np

def gaussian_filter(sigma):
     size = 2*np.ceil(3*sigma)+1
     x, y = np.mgrid[-size//2 + 1:size//2 + 1, -size//2 +
1:size//2 + 1]
     g = np.exp(-((x**2 + y**2)/(2.0*sigma**2))) /
(2*np.pi*sigma**2)
     return g/g.sum()
```

gaussian_pyramid.py

```
import numpy as np
from scipy.ndimage.filters import convolve
from gaussian_filter import gaussian_filter
def generate_octave(init_level, s, sigma):
     octave = [init_level]
     k = 2**(1/s)
     kernel = gaussian_filter(k * sigma)
     for i in range(s+2):
          next_level = convolve(octave[-1], kernel)
          octave.append(next_level)
     return octave

def generate_gaussian_pyramid(im, num_octave, s, sigma):
    pyr = []
    for _ in range(num_octave):
        octave = generate_octave(im, s, sigma)
        pyr.append(octave)
        im = octave[-3][::2, ::2]
    return pyr
```

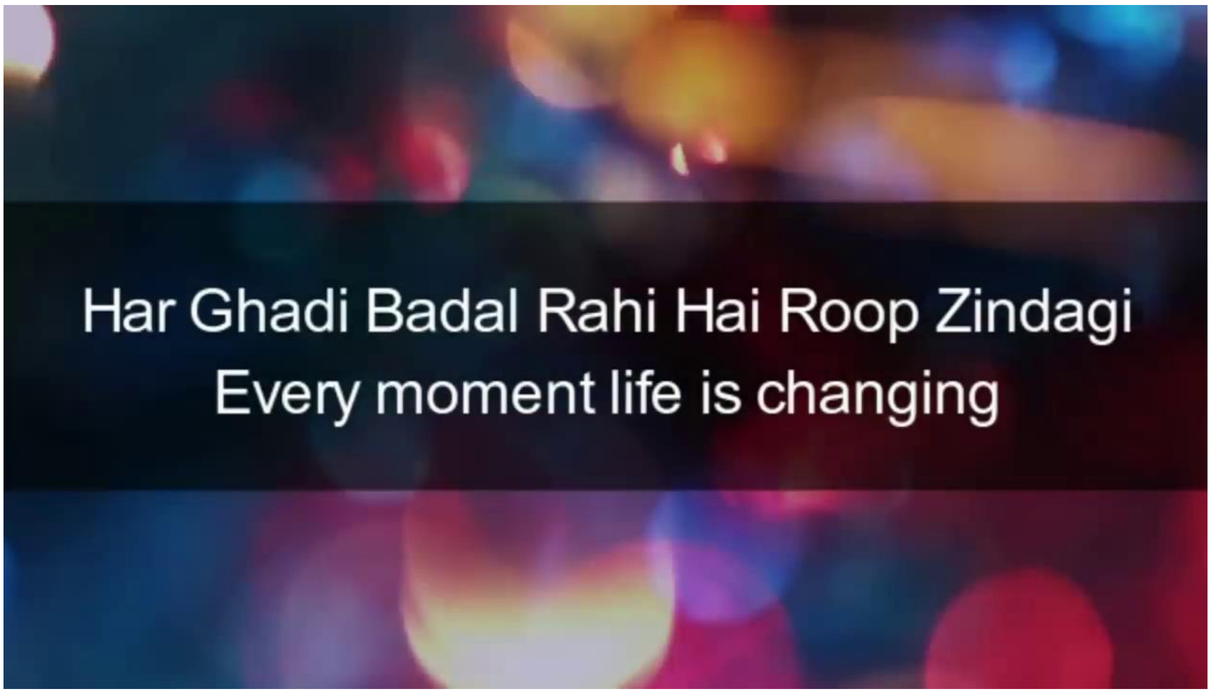
DoG_pyramid.py

```
import numpy as np

def generate_DoG_octave(gaussian_octave):
    octave = []
    for i in range(1, len(gaussian_octave)):
        octave.append(gaussian_octave[i] - gaussian_octave[i-
1])
```

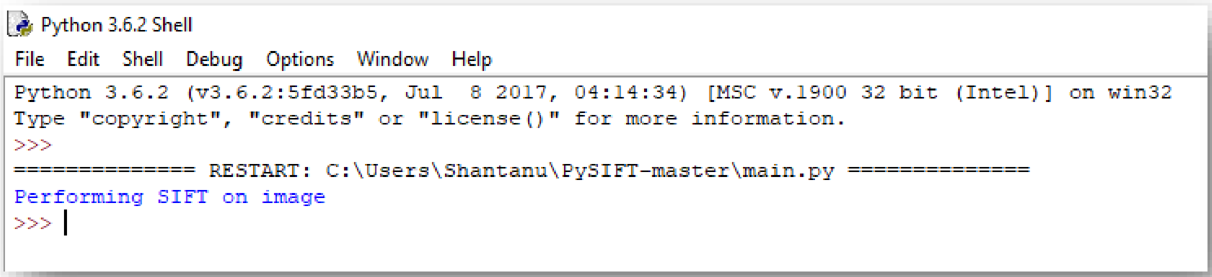
```
    return np.concatenate([o[:,:,np.newaxis] for o in octave],
axis=2)

def generate_DoG_pyramid(gaussian_pyramid):
    pyr = []
    for gaussian_octave in gaussian_pyramid:
        pyr.append(generate_DoG_octave(gaussian_octave))
    return pyr
```

Output:

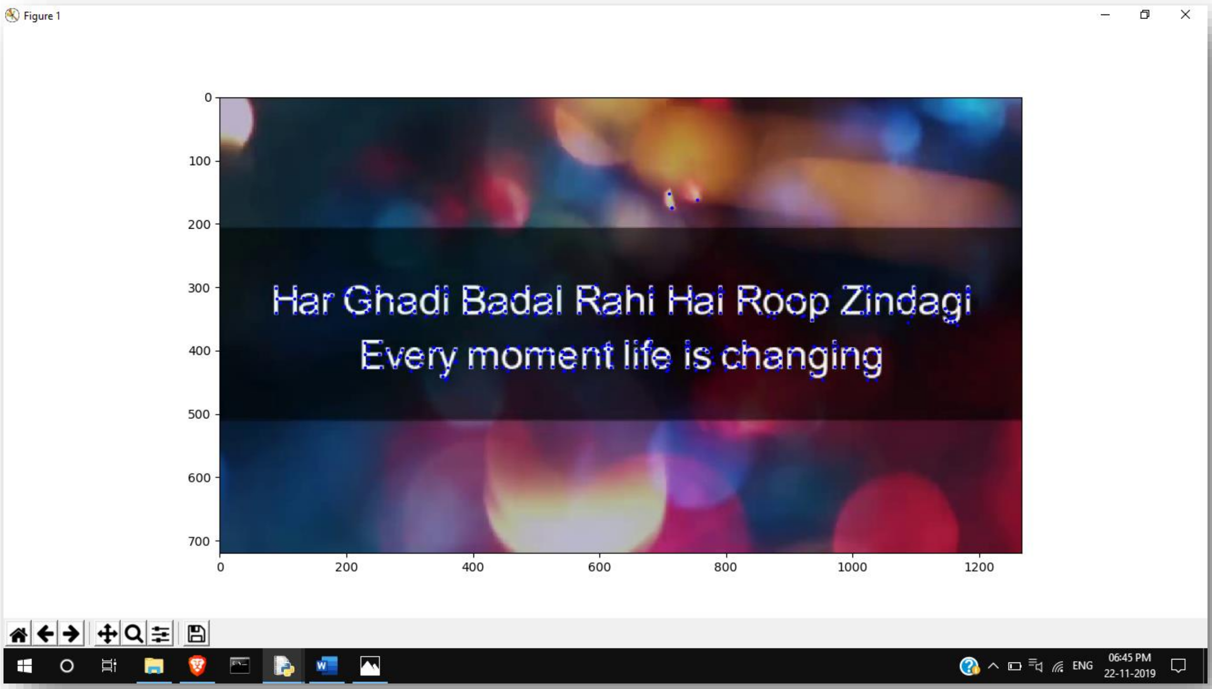
For this particular image:


Har Ghadi Badal Rahi Hai Roop Zindagi
Every moment life is changing

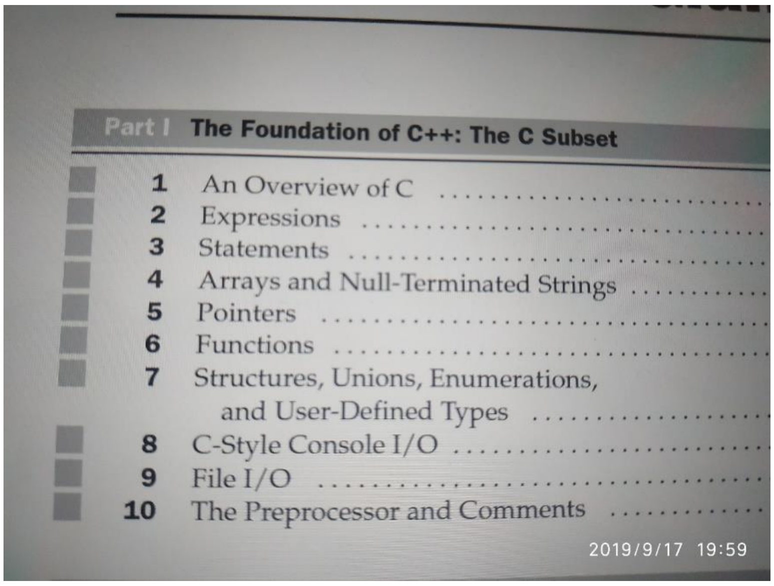
The output is:



```
Python 3.6.2 (v3.6.2:5fd33b5, Jul  8 2017, 04:14:34) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
============== RESTART: C:\Users\Shantanu\PySIFT-master\main.py ==============
Performing SIFT on image
>>> |
```



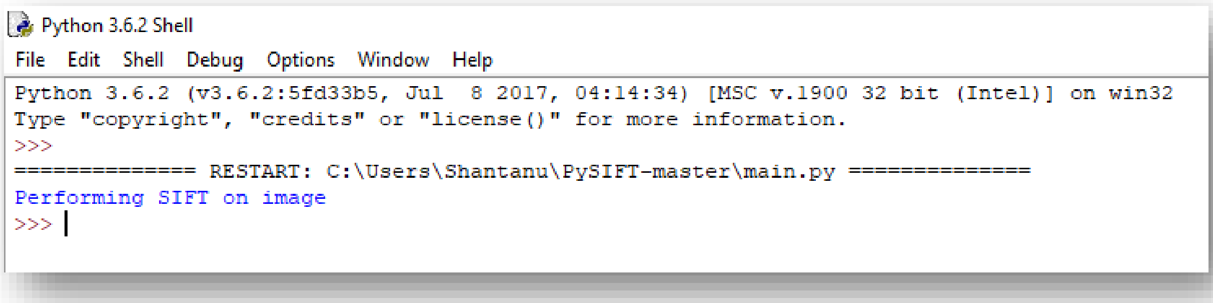
And for this image:



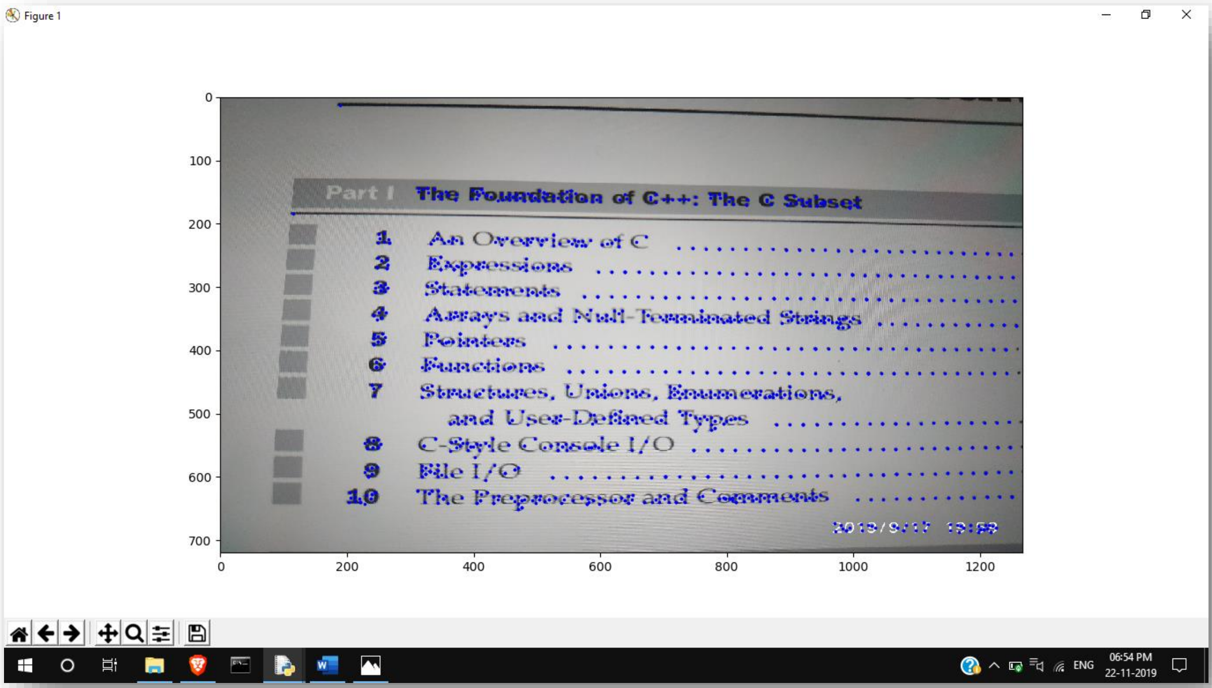
Part I The Foundation of C++: The C Subset

2019/9/17 19:59

The output is:



```
Python 3.6.2 (v3.6.2:5fd33b5, Jul  8 2017, 04:14:34) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
============== RESTART: C:\Users\Shantanu\PySIFT-master\main.py ==============
Performing SIFT on image
>>> 
```



Part I The Foundation of C++: The C Subset

INFERENCE AND CONCLUSION

The Tesseract – OCR library is successfully incorporated in the MapReduce framework along with the integration of Hadoop and OpenCV.
In the MapReduce program which gives out the output as text, the mapper divides the video input into frames. Since the video is at 15 fps (frames per second), our input video would generally give (15*60*4.24) = 3816 image files. Since it is a video which is a lyrical version of a song (purposely taken so as to extract text from video), most of the frames are empty (no text detected in it). For that, we filter them out by taking only one frame per second, which gives us (60*4.24) = 254 frames. This also makes the computation easy.

The reducer program filters out the noise in each image and thresholds it, in order to make it a binary image. This eliminates most of the noise that the Tesseract – OCR would otherwise interpret it as some character.

The output given by the reducer is encoded in utf-8 format in order for the Hadoop to interpret the character properly, otherwise it gives an encoding error.

The SIFT on individual image is performed separately. It processes the image and detects the presence of text in an image (or a frame) and marks scatter points around the text shapes that are present in the image. If the scatter points are to be joined, it will give us the text characters around which the blue points are marked.

Scope for further improvement:

For further improvement in the project the Tesseract – OCR library that was used in the MapReduce framework can be replaced by a more sophisticated version of the SIFT (Scale Invariant Feature Transform) bundled with a good machine learning model that learns the features extracted from SIFT and, technically gives us the complete functionality that is currently provided by the Tesseract – OCR.

As of now, we have ensured that, we can perform image and video processing on Hadoop framework using MapReduce scripts and Hadoop Streaming jar which helps us execute MapReduce programs written in python, which, otherwise, would have to be written in JAVA.