Big Data And Machine Learning Applications

COURSEWORK 1: The Lab Book
UP2225522

Contents

Week 1 4
- Basic Linux & Hadoop commands 4
- File transfer to cluster 4
- Implement MapReduce 5
- Week 1: Reflection 6

Week 2 7
- Writing Map and Reduce functions in Python 7
- Week 2: Reflection 13

Week 3 14
- MapReduce PageRank 14
- Week 3: Reflection 24

Week 4 25
- Basic Apache Spark 25
- Machine learning with Spark 30
- Week 4: Reflection 32

Week 5 33
- Using ALS in Spark 33
- Running Spark across the cluster 35
- Week 5: Reflection 39

Week 6 40
- Images, Tensors and Convolutions 40
- Image Recognition 41
- Transfer Learning 43
- Data Augmentation 45
- Week 6: Reflection 46

Week 7 47
- CNN From Scratch 47
- Improved Deep Learning Models 49
- A Small Project on an Astronomical Image set 50
- Exercise 53
- Week 7: Reflection 53

Week 8 54
- Spark Structured Streaming 54
- Processing Twitter Data 59
- Week 8: Reflection 60

Week 9 61
Building a Dense Artificial Neural Network from Scratch 61
Classification Using Simple Recurrent Neural Networks 63
An Improved Model Based on LSTM Recurrent Neural Networks 65
Exercise 66
Week 9: Reflection..... 68

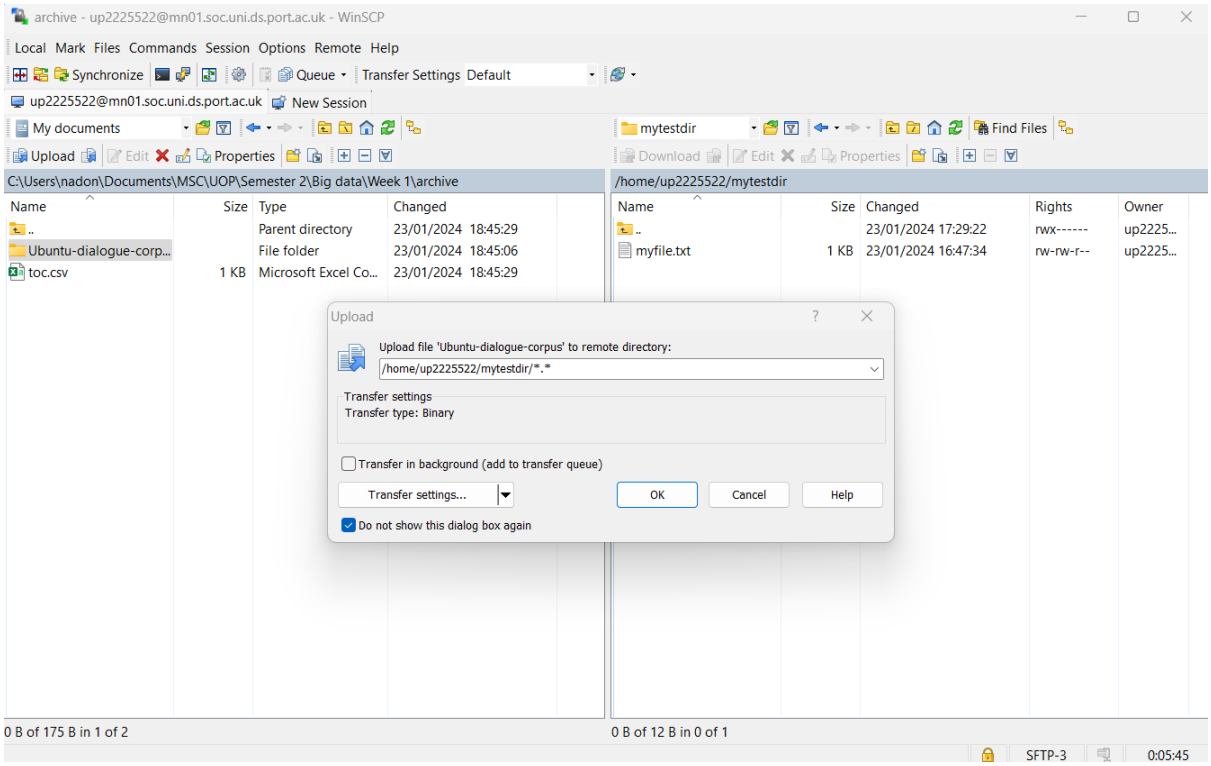
Week 1

Basic Linux & Hadoop commands

Linux uses a system of files and directories similar to Windows, with "/" in paths and the root directory being "/". The "ls" command lists directory contents, and the current working directory is found using "pwd". To create a directory, use "mkdir", change the working directory with "cd", or create a text file with an editor like "nano". File ownership and permissions can be checked using "ls -l". In HDFS, commands often begin with "hdfs dfs", and the "-ls" subcommand lists files and directories.

File transfer to cluster

WinSCP is used to transfer files from my local PC to the cluster in my working directory



The below shows that the file has been transferred.

```
[up2225522@mn01 ~]$ cd mytestdir
[up2225522@mn01 mytestdir]$ ls -l
total 8
-rw-rw-r-- 1 up2225522 up2225522   12 Jan 23 16:47 myfile.txt
drwxrwxr-x 2 up2225522 up2225522 4096 Jan 23 19:24 Ubuntu-dialogue-corpus
[up2225522@mn01 mytestdir]$
```

The below shows that the file has been transferred to the cluster.

```
n01 mytestdir]$ hdfs dfs -copyFromLocal Ubuntu-dialogue-corpus
^[[A^[[A^[[B^[[B
[up2225522@mn01 mytestdir]$
[up2225522@mn01 mytestdir]$ hdfs dfs -ls
Found 2 items
drwxr-xr-x   - up2225522 up2225522          0 2024-01-23 19:53 Ubuntu-dialogue-corpus

-rw-r--r--   3 up2225522 up2225522         12 2024-01-23 17:09 myfile.txt[u[[u[[uup[[
[up2225522@mn01 mytestdir]$
```

Implement MapReduce

YARN framework execution. The command involves running a (JAR) file, "hadoop-mapreduce-examples-2.9.2.jar," located in the specified Hadoop directory. The purpose is to perform a word count operation on "Ubuntu-dialogue-corpus" directory.

The syntax "$ yarn jar $HADOOP_PREFIX/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.2.jar wordcount Ubuntu-dialogue-corpus out" indicates the use of YARN to execute the specified JAR file with the "wordcount" example. The input data for the word count operation is assumed to be in the "Ubuntu-dialogue-corpus" directory. The results of the word count will be written to an output directory named "out."

The below command line output captures the detailed progress of a Hadoop MapReduce job, including the stages of map and reduce task execution and their respective completion percentages. The job is ultimately reported as completed successfully.

```
[up2225522@mn01 mytestdir]$ yarn jar $HADOOP_PREFIX/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.2.jar wordcount Ubuntu-dialogue-corpus out
24/01/23 19:58:14 INFO client.RMProxy: Connecting to ResourceManager at /10.3.8.1:8032
24/01/23 19:58:15 INFO input.FileInputFormat: Total input files to process : 3
24/01/23 19:58:15 INFO mapreduce.JobSubmitter: number of splits:23
24/01/23 19:58:15 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publ
isher.enabled
24/01/23 19:58:20 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1681658814718_1724
24/01/23 19:58:21 INFO impl.YarnClientImpl: Submitted application application_1681658814718_1724
24/01/23 19:58:21 INFO mapreduce.Job: The url to track the job: http://mn01.soc.uni.ds.port.ac.uk:8081/proxy/application_1681658814718_1724/
24/01/23 19:58:21 INFO mapreduce.Job: Running job: job_1681658814718_1724
24/01/23 19:58:29 INFO mapreduce.Job: Job job_1681658814718_1724 running in uber mode : false
24/01/23 19:58:29 INFO mapreduce.Job:  map 0% reduce 0%
24/01/23 19:58:43 INFO mapreduce.Job:  map 4% reduce 0%
24/01/23 19:58:44 INFO mapreduce.Job:  map 9% reduce 0%
24/01/23 19:58:45 INFO mapreduce.Job:  map 25% reduce 0%
24/01/23 19:58:46 INFO mapreduce.Job:  map 27% reduce 0%
24/01/23 19:58:47 INFO mapreduce.Job:  map 52% reduce 0%
24/01/23 19:58:51 INFO mapreduce.Job:  map 57% reduce 0%
24/01/23 19:58:52 INFO mapreduce.Job:  map 58% reduce 0%
24/01/23 19:58:53 INFO mapreduce.Job:  map 67% reduce 0%
24/01/23 19:58:54 INFO mapreduce.Job:  map 70% reduce 0%
24/01/23 19:58:55 INFO mapreduce.Job:  map 73% reduce 0%
24/01/23 19:58:56 INFO mapreduce.Job:  map 90% reduce 0%
24/01/23 19:58:57 INFO mapreduce.Job:  map 98% reduce 0%
24/01/23 19:58:59 INFO mapreduce.Job:  map 100% reduce 0%
24/01/23 19:59:00 INFO mapreduce.Job:  map 100% reduce 19%
24/01/23 19:59:06 INFO mapreduce.Job:  map 100% reduce 29%
24/01/23 19:59:12 INFO mapreduce.Job:  map 100% reduce 33%
24/01/23 19:59:18 INFO mapreduce.Job:  map 100% reduce 66%
24/01/23 19:59:24 INFO mapreduce.Job:  map 100% reduce 72%
24/01/23 19:59:30 INFO mapreduce.Job:  map 100% reduce 77%
24/01/23 19:59:36 INFO mapreduce.Job:  map 100% reduce 86%
24/01/23 19:59:42 INFO mapreduce.Job:  map 100% reduce 94%
24/01/23 19:59:48 INFO mapreduce.Job:  map 100% reduce 99%
24/01/23 19:59:51 INFO mapreduce.Job:  map 100% reduce 100%
24/01/23 19:59:51 INFO mapreduce.Job: Job job_1681658814718_1724 completed successfully
24/01/23 19:59:52 INFO mapreduce.Job: Counters: 51
```

The counters below provide detailed information about resource usage, task execution, input/output metrics, and other aspects of the MapReduce job. They are valuable for performance analysis and optimization.

```
24/01/23 19:59:52 INFO mapreduce.Job: Counters: 51
        File System Counters
                FILE: Number of bytes read=4963513686
                FILE: Number of bytes written=6824787680
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=2912346209
                HDFS: Number of bytes written=1016042803
                HDFS: Number of read operations=72
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Killed map tasks=1
                Launched map tasks=24
                Launched reduce tasks=1
                Data-local map tasks=21
                Rack-local map tasks=3
                Total time spent by all maps in occupied slots (ms)=570404
                Total time spent by all reduces in occupied slots (ms)=64639
                Total time spent by all map tasks (ms)=570404
                Total time spent by all reduce tasks (ms)=64639
                Total vcore-milliseconds taken by all map tasks=570404
                Total vcore-milliseconds taken by all reduce tasks=64639
                Total megabyte-milliseconds taken by all map tasks=584093696
                Total megabyte-milliseconds taken by all reduce tasks=66190336
```

```
        Map-Reduce Framework
                Map input records=26839034
                Map output records=269088400
                Map output bytes=3985265973
                Map output materialized bytes=1856502980
                Input split bytes=3308
                Combine input records=309732377
                Combine output records=77074216
                Reduce input groups=19000362
                Reduce shuffle bytes=1856502980
                Reduce input records=36430239
                Reduce output records=19000362
                Spilled Records=138101438
                Shuffled Maps =23
                Failed Shuffles=0
                Merged Map outputs=23
                GC time elapsed (ms)=5846
                CPU time spent (ms)=733770
                Physical memory (bytes) snapshot=7672815616
                Virtual memory (bytes) snapshot=53042331648
                Total committed heap usage (bytes)=4939841536
        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=2912342901
        File Output Format Counters
                Bytes Written=1016042803
[up2225522@mn01 mytestdir]$
```

Week 1: Reflection

The first week focused on Linux commands and Hadoop MapReduce tasks, introducing the Linux file system, command-line operations, and Hadoop commands. The experience highlighted the importance of efficient data handling, task execution, and resource management. The hands-on approach to learning emphasized the importance of mastering these tools for effective data management and analysis.

Week 2

Writing Map and Reduce functions in Python

Loading text files to the cluster and using Nano as a Python text editor

```
[up2225522@mn01 mytestdir]$ ls
MR test 2.py  MR test.py  myfile.txt  poem.txt  Ubuntu-dialogue-corpus
[up2225522@mn01 mytestdir]$ nano mr_unix_wc.py
```

```
up2225522@mn01:~/mytestd
  GNU nano 2.3.1            File: mr_unix_wc.py

from mrjob.job import MRJob

class MRUnixWC(MRJob):

    def mapper(self, key, line):
        yield "chars", len(line)
        yield "words", len(line.split())
        yield "lines", 1

    def reducer(self, key, values):
        yield key, sum(values)

if __name__ == '__main__':
    MRUnixWC.run()

                          [ Read 14 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

```
[up2225522@mn01 mytestdir]$ ls
MR test 2.py  mr_unix_wc.py  poem.txt
MR test.py    myfile.txt     Ubuntu-dialogue-corpus
```

Running the Python script for count operations and compare it with the built-in function

```
[up2225522@mn01 mytestdir]$ python mr_unix_wc.py poem.txt
No configs found; falling back on auto-configuration
No configs specified for inline runner
Running step 1 of 1...
Creating temp directory /tmp/mr_unix_wc.up2225522.20240130.154107.093102
job output is in /tmp/mr_unix_wc.up2225522.20240130.154107.093102/output
Streaming final output from /tmp/mr_unix_wc.up2225522.20240130.154107.093102/ou
tput...
"chars" 1267
"lines" 42
"words" 242
Removing temp directory /tmp/mr_unix_wc.up2225522.20240130.154107.093102...
```

```
[up2225522@mn01 mytestdir]$ wc poem.txt
  42  242 1309 poem.txt
```

MapReduce word count on one cluster:

```
up2225522@mn01:~/mytestd
  GNU nano 2.3.1            File: mr_word_count.py

from mrjob.job import MRJob

class MRWordCount(MRJob):

    def mapper(self, key, value):
        # key: line number - not used here
        # value: contents of one line
        for word in value.split() :
            yield word, 1                 # output a pair: word and count

    def reducer(self, key, values):
        yield key, sum(values)


if __name__ == '__main__':
    MRWordCount.run()

                              [ Read 16 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

```
[up2225522@mn01 mytestdir]$ nano mr_word_count.py
[up2225522@mn01 mytestdir]$ ls
MR test 2.py  mr_unix_wc.py     myfile.txt  Ubuntu-dialogue-corpus
MR test.py    mr_word_count.py  poem.txt
```

```
[up2225522@mn01 mytestdir]$ python mr_word_count.py poem.txt
No configs found; falling back on auto-configuration
No configs specified for inline runner
Running step 1 of 1...
Creating temp directory /tmp/mr_word_count.up2225522.20240130.154649.663091
job output is in /tmp/mr_word_count.up2225522.20240130.154649.663091/output
Streaming final output from /tmp/mr_word_count.up2225522.20240130.154649.663091
/output...
"(1983)"        1
"And"   3
"BY"    1
"Bong-Tree"     1
"Book"  1
"But"   1
"By"    1
"Children"      1
"EDWARD"        1
"His"   2
"House" 1
"How"   1
"I"     1
"II"    1
"III"   1
"In"    1
"LEAR"  1
"O"     2
"Owl"   3
"Owl,"  1
"Pig,"  1
"Piggy,"        1
"Piggy-wig"     1
"Poetry"        1
"Pussy!"        1
"Pussy" 3
"Pussy,"        1
"Pussy-Cat"     1
"Pussy-cat"     1
"Random"        1
```

MapReduce word count on multiple clusters:

```
[up2225522@mn01 mytestdir]$ python mr_word_count.py -r hadoop poem.txt
No configs found; falling back on auto-configuration
No configs specified for hadoop runner
Looking for hadoop binary in /home/hadoop/hadoop-2.9.2/bin...
Found hadoop binary: /home/hadoop/hadoop-2.9.2/bin/hadoop
Using Hadoop version 2.9.2
Looking for Hadoop streaming jar in /home/hadoop/hadoop-2.9.2...
Found Hadoop streaming jar: /home/hadoop/hadoop-2.9.2/share/hadoop/tools/lib/ha
doop-streaming-2.9.2.jar
Creating temp directory /tmp/mr_word_count.up2225522.20240130.155621.419045
Copying local files to hdfs:///user/up2225522/tmp/mrjob/mr_word_count.up2225522
.20240130.155621.419045/files/...
Running step 1 of 1...
  packageJobJar: [/tmp/hadoop-unjar2679985630338952803/] [] /tmp/streamjob43984
74206797308488.jar tmpDir=null
  Connecting to ResourceManager at /10.3.8.1:8032
  Connecting to ResourceManager at /10.3.8.1:8032
  Total input files to process : 1
  number of splits:2
  yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead,
 use yarn.system-metrics-publisher.enabled
  Submitting tokens for job: job_1681658814718_1848
  Submitted application application_1681658814718_1848
  The url to track the job: http://mn01.soc.uni.ds.port.ac.uk:8081/proxy/applic
ation_1681658814718_1848/
  Running job: job_1681658814718_1848
  Job job_1681658814718_1848 running in uber mode : false
   map 0% reduce 0%
   map 100% reduce 0%
   map 100% reduce 100%
  Job job_1681658814718_1848 completed successfully
  Output directory: hdfs:///user/up2225522/tmp/mrjob/mr_word_count.up2225522.20
240130.155621.419045/output
```

```
up2225522@mn01:~/mytestd
"sand," 1
"sang"  1
"sea"   1
"sell"  1
"shall" 1
"shilling"      1
"sing!" 1
"slices"        1
"small" 1
"some"  1
"spoon;"        1
"stars" 1
"stood" 1
"sweet" 1
"tarried:"      1
"the"   17
"there" 1
"they"  2
"to"    5
"too"   1
"took"  2
"up"    2
"us"    1
"we"    2
"went"  1
"were"  1
"what"  1
"where" 1
"who"   1
"will.\""       1
"willing"       1
"with"  1
"wood"  1
"year"  1
"you"   4
Removing HDFS temp directory hdfs:///user/up2225522/tmp/mrjob/mr_word_count.up2
225522.20240130.155621.419045...
Removing temp directory /tmp/mr_word_count.up2225522.20240130.155621.419045...
[up2225522@mn01 mytestdir]$
```

MapReduce word count on multiple clusters with performance monitoring and on a different file:

```
up2225522@mn01:~/mytestd

[up2225522@mn01 mytestdir]$ python mr_word_count.py -r hadoop -o hdfs:///user/up2225522/out hdfs:///user/up2225522/Ubuntu-dialogue-cor
No configs found; falling back on auto-configuration
No configs specified for hadoop runner
Looking for hadoop binary in /home/hadoop/hadoop-2.9.2/bin...
Found hadoop binary: /home/hadoop/hadoop-2.9.2/bin/hadoop
Using Hadoop version 2.9.2
Looking for Hadoop streaming jar in /home/hadoop/hadoop-2.9.2...
Found Hadoop streaming jar: /home/hadoop/hadoop-2.9.2/share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar
Creating temp directory /tmp/mr_word_count.up2225522.20240130.161656.481484
Copying local files to hdfs:///user/up2225522/tmp/mrjob/mr_word_count.up2225522.20240130.161656.481484/files/...
Running step 1 of 1...
  packageJobJar: [/tmp/hadoop-unjar7816129374047689238/] [] /tmp/streamjob5393522657318970445.jar tmpDir=null
  Connecting to ResourceManager at /10.3.8.1:8032
  Connecting to ResourceManager at /10.3.8.1:8032
  Total input files to process : 3
  number of splits:23
  yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publisher.enabled
  Submitting tokens for job: job_1681658814718_1856
  Submitted application application_1681658814718_1856
  The url to track the job: http://mn01.soc.uni.ds.port.ac.uk:8081/proxy/application_1681658814718_1856/
  Running job: job_1681658814718_1856
  Job job_1681658814718_1856 running in uber mode : false
   map 0% reduce 0%
   map 14% reduce 0%
   map 21% reduce 0%
   map 29% reduce 0%
   map 38% reduce 0%
   map 44% reduce 0%
   map 50% reduce 0%
   map 50% reduce 3%
   map 55% reduce 3%
   map 62% reduce 3%
   map 68% reduce 3%
   map 69% reduce 3%
   map 83% reduce 3%
   map 85% reduce 3%
   map 86% reduce 3%
   map 88% reduce 10%
   map 98% reduce 10%
   map 99% reduce 10%
```

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes
11	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:32768, vCores:12>

Show 100 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB
application_1681658814718_1860	up2239047	streamjob4308073658837691119.jar	MAPREDUCE	default	0	Tue Jan 30 16:20:58 +0000 2024	Tue Jan 30 16:21:19 +0000 2024	FINISHED	SUCCEEDED	N/A	N/A	N/A
application_1681658814718_1859	up2224620	streamjob6218985058554816211.jar	MAPREDUCE	default	0	Tue Jan 30 16:20:03 +0000 2024	Tue Jan 30 16:20:23 +0000 2024	FINISHED	SUCCEEDED	N/A	N/A	N/A
application_1681658814718_1858	up2224635	streamjob7635581766188187162.jar	MAPREDUCE	default	0	Tue Jan 30 16:19:38 +0000 2024	Tue Jan 30 16:19:44 +0000 2024	FINISHED	FAILED	N/A	N/A	N/A
application_1681658814718_1857	up2224635	streamjob5542228932308527316.jar	MAPREDUCE	default	0	Tue Jan 30 16:17:28 +0000 2024	Tue Jan 30 16:17:34 +0000 2024	FINISHED	FAILED	N/A	N/A	N/A
application_1681658814718_1856	up2225522	streamjob5393522657318970445.jar	MAPREDUCE	default	0	Tue Jan 30 16:17:09 +0000	N/A	RUNNING	UNDEFINED	2	2	3072

Reserved CPU VCores	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
N/A	N/A	0.0	0.0		History	0
N/A	N/A	0.0	0.0		History	0
N/A	N/A	0.0	0.0		History	0
N/A	N/A	0.0	0.0		History	0
0	0	0.9	0.9		ApplicationMaster	0

Notes

- The processing is very slow due to the Python interpreter
- As shown above, 2 CPU VCore is allocated to compute the count operation (3072MB allocated)

The two Python commands from the file `mr_word_count.py`, uses Hadoop MapReduce to count words in local files and a directory on HDFS. The difference lies in the input sources and output destinations.

Run it again with mapred.reduce.tasks = 50:

```
up2225522@mn01:~/mytestd
[up2225522@mn01 mytestdir]$ python mr_word_count.py -r hadoop -o hdfs:///user/up2225522/out -D mapred.reduce.tasks=50 hdfs:///user/up2225522/Ubuntu-dialogue
-corpus
No configs found; falling back on auto-configuration
No configs specified for hadoop runner
Looking for hadoop binary in /home/hadoop/hadoop-2.9.2/bin...
Found hadoop binary: /home/hadoop/hadoop-2.9.2/bin/hadoop
Using Hadoop version 2.9.2
Looking for Hadoop streaming jar in /home/hadoop/hadoop-2.9.2...
Found Hadoop streaming jar: /home/hadoop/hadoop-2.9.2/share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar
Creating temp directory /tmp/mr_word_count.up2225522.20240130.165214.680971
Copying local files to hdfs:///user/up2225522/tmp/mrjob/mr_word_count.up2225522.20240130.165214.680971/files/...
Detected hadoop configuration property names that do not match hadoop version 2.9.2:
The have been translated as follows
 mapred.reduce.tasks: mapreduce.job.reduces
Running step 1 of 1...
  packageJobJar: [/tmp/hadoop-unjar8363999210541471919/] [] /tmp/streamjob4704957935293154594.jar tmpDir=null
  Connecting to ResourceManager at /10.3.8.1:8032
  Connecting to ResourceManager at /10.3.8.1:8032
  Total input files to process : 3
  number of splits:23
  mapred.reduce.tasks is deprecated. Instead, use mapreduce.job.reduces
  yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publisher.enabled
  Submitting tokens for job: job_1681658814718_1871
  Submitted application application_1681658814718_1871
  The url to track the job: http://mn01.soc.uni.ds.port.ac.uk:8081/proxy/application_1681658814718_1871/
  Running job: job_1681658814718_1871
  Job job_1681658814718_1871 running in uber mode : false
   map 0% reduce 0%
   map 15% reduce 0%
   map 21% reduce 0%
   map 29% reduce 0%
   map 31% reduce 0%
   map 37% reduce 0%
   map 38% reduce 0%
   map 44% reduce 0%
   map 50% reduce 0%
   map 56% reduce 0%
   map 56% reduce 1%
   map 62% reduce 3%
   map 68% reduce 3%
```

Notes

- With 50 cores, the task runs exponentially faster than before

```
up2225522@mn01:~/mytestd
[up2225522@mn01 mytestdir]$ hdfs dfs -ls out
Found 51 items
-rw-r--r--   3 up2225522 up2225522          0 2024-01-30 16:54 out/_SUCCESS
-rw-r--r--   3 up2225522 up2225522   21437785 2024-01-30 16:54 out/part-00000
-rw-r--r--   3 up2225522 up2225522   21394986 2024-01-30 16:54 out/part-00001
-rw-r--r--   3 up2225522 up2225522   21449414 2024-01-30 16:54 out/part-00002
-rw-r--r--   3 up2225522 up2225522   21435821 2024-01-30 16:54 out/part-00003
-rw-r--r--   3 up2225522 up2225522   21454026 2024-01-30 16:54 out/part-00004
-rw-r--r--   3 up2225522 up2225522   21550313 2024-01-30 16:54 out/part-00005
-rw-r--r--   3 up2225522 up2225522   21472226 2024-01-30 16:54 out/part-00006
-rw-r--r--   3 up2225522 up2225522   21455499 2024-01-30 16:54 out/part-00007
-rw-r--r--   3 up2225522 up2225522   21511373 2024-01-30 16:54 out/part-00008
-rw-r--r--   3 up2225522 up2225522   21475159 2024-01-30 16:54 out/part-00009
-rw-r--r--   3 up2225522 up2225522   21434069 2024-01-30 16:54 out/part-00010
-rw-r--r--   3 up2225522 up2225522   21424209 2024-01-30 16:54 out/part-00011
-rw-r--r--   3 up2225522 up2225522   21483251 2024-01-30 16:54 out/part-00012
-rw-r--r--   3 up2225522 up2225522   21422285 2024-01-30 16:54 out/part-00013
-rw-r--r--   3 up2225522 up2225522   21517613 2024-01-30 16:54 out/part-00014
-rw-r--r--   3 up2225522 up2225522   21459995 2024-01-30 16:54 out/part-00015
-rw-r--r--   3 up2225522 up2225522   21467907 2024-01-30 16:54 out/part-00016
-rw-r--r--   3 up2225522 up2225522   21405384 2024-01-30 16:54 out/part-00017
-rw-r--r--   3 up2225522 up2225522   21432836 2024-01-30 16:54 out/part-00018
-rw-r--r--   3 up2225522 up2225522   21494740 2024-01-30 16:54 out/part-00019
-rw-r--r--   3 up2225522 up2225522   21446402 2024-01-30 16:54 out/part-00020
-rw-r--r--   3 up2225522 up2225522   21471914 2024-01-30 16:54 out/part-00021
-rw-r--r--   3 up2225522 up2225522   21515618 2024-01-30 16:54 out/part-00022
-rw-r--r--   3 up2225522 up2225522   21452128 2024-01-30 16:54 out/part-00023
-rw-r--r--   3 up2225522 up2225522   21462157 2024-01-30 16:54 out/part-00024
-rw-r--r--   3 up2225522 up2225522   21443069 2024-01-30 16:54 out/part-00025
-rw-r--r--   3 up2225522 up2225522   21558187 2024-01-30 16:54 out/part-00026
-rw-r--r--   3 up2225522 up2225522   21426576 2024-01-30 16:54 out/part-00027
-rw-r--r--   3 up2225522 up2225522   21409972 2024-01-30 16:54 out/part-00028
-rw-r--r--   3 up2225522 up2225522   21495712 2024-01-30 16:54 out/part-00029
-rw-r--r--   3 up2225522 up2225522   21522378 2024-01-30 16:54 out/part-00030
-rw-r--r--   3 up2225522 up2225522   21486141 2024-01-30 16:54 out/part-00031
-rw-r--r--   3 up2225522 up2225522   21422276 2024-01-30 16:54 out/part-00032
-rw-r--r--   3 up2225522 up2225522   21522670 2024-01-30 16:54 out/part-00033
-rw-r--r--   3 up2225522 up2225522   21485195 2024-01-30 16:54 out/part-00034
-rw-r--r--   3 up2225522 up2225522   21436566 2024-01-30 16:54 out/part-00035
```

The above shows the display of the output.

Week 2: Reflection

This week, I learned Python scripts for running count operations on Hadoop clusters, demonstrating how parallel processing can significantly reduce computing time and improve efficiency, thereby emphasising the importance of parallel processing.

Week 3

MapReduce PageRank

Code 1:

```
up2225522@mn01:~/mytestd
  GNU nano 2.3.1          File: mr_page_rank1.py

from mrjob.job import MRJob

N = 11

class MRPageRank(MRJob):

    def mapper(self, key, line):
        # key: line number - not used here
        # line: contents of one line
        fields = line.split()
        pageRank = float(fields [1])  # current PageRank of this page
        outLinks = fields [2:]        # outgoing links of this page

        prContrib = pageRank / len(outLinks)
        for mId in outLinks :
            yield mId, prContrib

    def reducer(self, mId, values):
        s = sum(values)
        mPageRank = 0.85 * s + 0.15 / N
        yield mId, mPageRank

if __name__ == '__main__':
    MRPageRank.run()
```

Text file:

```
up2225522@mn01:~/mytestd
  GNU nano 2.3.1          File: web-graph.txt

A  0.090909  A  B  C  D  E  F  G  H  I  J  K
B  0.090909  C
C  0.090909  B
D  0.090909  A  B
E  0.090909  B  D  F
F  0.090909  B  E
G  0.090909  B  E
H  0.090909  B  E
I  0.090909  B  E
J  0.090909  E
K  0.090909  E
```

Output:

```
[up2225522@mn01 mytestdir]$ python mr_page_rank1.py web-graph.txt
No configs found; falling back on auto-configuration
No configs specified for inline runner
Running step 1 of 1...
Creating temp directory /tmp/mr_page_rank1.up2225522.20240206.143836.696013
job output is in /tmp/mr_page_rank1.up2225522.20240206.143836.696013/output
Streaming final output from /tmp/mr_page_rank1.up2225522.20240206.143836.696
013/output...
"A"     0.059297475
"B"     0.316872975
"C"     0.0979338
"D"     0.04641869999999999
"E"     0.32975175
"F"     0.04641869999999999
"G"     0.02066115
"H"     0.02066115
"I"     0.02066115
"J"     0.02066115
"K"     0.02066115
Removing temp directory /tmp/mr_page_rank1.up2225522.20240206.143836.696013.
..
```

Adjust code to make the algorithm iterable:

```
up2225522@mn01:~/mytestd
  GNU nano 2.3.1            File: mr_page_rank.py

from mrjob.job import MRJob
from mrjob.step import MRStep
from mrjob.protocol import RawValueProtocol

N = 11  # Nodes in our graph
NITER = 50

class MRPageRank(MRJob):

    OUTPUT_PROTOCOL = RawValueProtocol
        # Default for input; use here for output so we can "input our outpu$

    def mapper(self, key, value):
        # key: line number - not used here
        # value: contents of one line
        fields = value.split()
        myId = fields [0]              # ID of this page
        pageRank = float(fields [1])   # current PageRank of this page
        outLinks = fields [2:]         # outgoing links of this page

        yield myId, outLinks

        prContrib = pageRank / len(outLinks)
        for mId in outLinks :
            yield mId, prContrib

    def reducer(self, mId, values):
        s = 0
        for val in values :
            if(isinstance(val, list)) :
                mOutLinks = val  # From *first* yield in mapper above
            else :
                s += float(val)

                                [ Read 45 lines ]
^G Get Help  ^O WriteOut  ^R Read File^Y Prev Page^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is ^V Next Page^U UnCut Tex^T To Spell
```

```
up2225522@mn01:~/mytestd
  GNU nano 2.3.1            File: mr_page_rank.py

        outLinks = fields [2:]        # outgoing links of this page

        yield myId, outLinks

        prContrib = pageRank / len(outLinks)
        for mId in outLinks :
            yield mId, prContrib

    def reducer(self, mId, values):
        s = 0
        for val in values :
            if(isinstance(val, list)) :
                mOutLinks = val  # From *first* yield in mapper above
            else :
                s += float(val)

        mPageRank = s * 0.85 + 0.15 / N
        yield None, mId + " " + str(mPageRank) + " " + " ".join(mOutLinks)
            # NB: key value irrelevant with RawValueProtocol output

    def steps(self) :
        # Repeat the same map-reduce phase NITER times
        step = MRStep(mapper=self.mapper, reducer=self.reducer)
        return [step] * NITER

if __name__ == '__main__':
    MRPageRank.run()

^G Get Help  ^O WriteOut  ^R Read File^Y Prev Page^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is ^V Next Page^U UnCut Tex^T To Spell
```

Notes

The adjustment gives a more structured approach by utilizing RawValueProtocol for output and explicitly defining the iteration process using the steps method.

Output:

```
up2225522@mn01:~/mytestd
Running step 29 of 50...
Running step 30 of 50...
Running step 31 of 50...
Running step 32 of 50...
Running step 33 of 50...
Running step 34 of 50...
Running step 35 of 50...
Running step 36 of 50...
Running step 37 of 50...
Running step 38 of 50...
Running step 39 of 50...
Running step 40 of 50...
Running step 41 of 50...
Running step 42 of 50...
Running step 43 of 50...
Running step 44 of 50...
Running step 45 of 50...
Running step 46 of 50...
Running step 47 of 50...
Running step 48 of 50...
Running step 49 of 50...
Running step 50 of 50...
job output is in /tmp/mr_page_rank.up2225522.20240206.144936.763367/output
Streaming final output from /tmp/mr_page_rank.up2225522.20240206.144936.7633
67/output...
A 0.03278149315934767 A B C D E F G H I J K
B 0.3843697808389126 C
C 0.3429414531872355 B
D 0.03908709209997012 A B
E 0.08088569323450431 B D F
F 0.03908709209997012 B E
G 0.01616947901685893 B E
H 0.01616947901685893 B E
I 0.01616947901685893 B E
J 0.01616947901685893 E
K 0.01616947901685893 E
Removing temp directory /tmp/mr_page_rank.up2225522.20240206.144936.763367..
```

New data set:

```
up2225522@mn01:~/mytestd
GNU nano 2.3.1

# A little Web graph
2  3
3  2
4  1
4  2
5  2
5  4
5  6
6  2
6  5
7  2
7  5
8  2
8  5
9  2
9  5
10  5
11  5
```

Adjust the code for pre-processing:

```
up2225522@mn01:~/mytestd
GNU nano 2.3.1          File: mr_preprocess.py                    Modified

from mrjob.job import MRJob
from mrjob.protocol import RawValueProtocol

N = 11
pageRank = 1 / N

class MRPreprocess(MRJob):

    OUTPUT_PROTOCOL = RawValueProtocol
        # Default for input; use here for output so we can "input our outpu$

    def mapper(self, key, value):
        # key: line number - not used here
        # value: contents of one line

        # Ignore blank lines and comments
        if len(value) == 0 or value [0] == '#':
            return

        fields = value.split()

        fromNode = fields [0]
        toNode   = fields [1]

        yield fromNode, toNode

    def reducer(self, mId, values):
        mOutLinks = values
        yield None, mId + " " + str(pageRank) + " " + " ".join(mOutLinks)

if __name__ == '__main__':
    MRPreprocess.run()
```

Output:

```
[up2225522@mn01 mytestdir]$ python mr_preprocess.py web-Wikipedia.txt
No configs found; falling back on auto-configuration
No configs specified for inline runner
Running step 1 of 1...
Creating temp directory /tmp/mr_preprocess.up2225522.20240206.145611.550930
job output is in /tmp/mr_preprocess.up2225522.20240206.145611.550930/output
Streaming final output from /tmp/mr_preprocess.up2225522.20240206.145611.550
930/output...
10 0.09090909090909091 5
11 0.09090909090909091 5
2 0.09090909090909091 3
3 0.09090909090909091 2
4 0.09090909090909091 1 2
5 0.09090909090909091 2 4 6
6 0.09090909090909091 2 5
7 0.09090909090909091 2 5
8 0.09090909090909091 2 5
9 0.09090909090909091 2 5
Removing temp directory /tmp/mr_preprocess.up2225522.20240206.145611.550930.
..
[up2225522@mn01 mytestdir]$
```

This code only works for normal nodes not dangling nodes.

Adjust again for the code to work with dangling nodes:

```
up2225522@mn01:~/mytestd
  GNU nano 2.3.1          File: mr_preprocess.py                  Modified

from mrjob.job import MRJob
from mrjob.protocol import RawValueProtocol

N = 11
pageRank = 1 / N
class MRPreprocess(MRJob):

    OUTPUT_PROTOCOL = RawValueProtocol
        # Default for input; use here for output so we can "input our outpu$

    def mapper(self, key, value):
        # key: line number - not used here
        # value: contents of one line

        # Ignore blank lines and comments
        if len(value) == 0 or value [0] == '#':
            return

        fields = value.split()
        fromNode = fields [0]
        toNode   = fields [1]

        yield fromNode, toNode
        yield toNode, None

    def reducer(self, mId, values):
        mOutLinks = [] ;
        for val in values :
            if not (val is None) :
                mOutLinks.append(val)
        yield None, mId + " " + str(pageRank) + " " + " ".join(mOutLinks)

if __name__ == '__main__':
    MRPreprocess.run()

^G Get Help ^O WriteOut ^R Read File^Y Prev Page^K Cut Text ^C Cur Pos
^X Exit     ^J Justify  ^W Where Is ^V Next Page^U UnCut Tex^T To Spell
```

Output:

```
[up2225522@mn01 mytestdir]$ python mr_preprocess.py web-Wikipedia.txt
No configs found; falling back on auto-configuration
No configs specified for inline runner
Running step 1 of 1...
Creating temp directory /tmp/mr_preprocess.up2225522.20240206.150129.179275
job output is in /tmp/mr_preprocess.up2225522.20240206.150129.179275/output
Streaming final output from /tmp/mr_preprocess.up2225522.20240206.150129.179
275/output...
1 0.09090909090909091
10 0.09090909090909091 5
11 0.09090909090909091 5
2 0.09090909090909091 3
3 0.09090909090909091 2
4 0.09090909090909091 1 2
5 0.09090909090909091 2 4 6
6 0.09090909090909091 2 5
7 0.09090909090909091 2 5
8 0.09090909090909091 2 5
9 0.09090909090909091 2 5
Removing temp directory /tmp/mr_preprocess.up2225522.20240206.150129.179275.
..
```

The modified code now ensures preprocessing by emitting both `fromNode` and `toNode` pairs in the mapper and accounting for all nodes, including those with no outgoing links, while aggregating outgoing links in the reducer, thereby enhancing data preparation for the PageRank algorithm.

Integrate all together (main algorithms and pre-processing):

```
up2225522@mn01:~/mytestd
  GNU nano 2.3.1          File: mr_page_rank_big.py                    Modified

from mrjob.job import MRJob
from mrjob.step import MRStep
from mrjob.protocol import RawValueProtocol

NITER = 50
N = 11
initPageRank = 1 / N

class MRPageRank(MRJob):

    OUTPUT_PROTOCOL = RawValueProtocol
        # Default for input; use here for output so we can "input our outpu$

    def preMapper(self, key, value):

        if len(value) == 0 or value [0] == '#':
            return

        fields = value.split()
        fromNode = fields [0]
        toNode   = fields [1]

        yield toNode, None
        yield fromNode, toNode

    def preReducer(self, mId, values):

        outLinks = []
        for val in values :
            if not (val is None) :
                outLinks.append(val)

        yield None, mId + " " + str(initPageRank) + " " + " ".join(outLinks)
```

```
    def mapper(self, key, value):

        fields = value.split()
        myId     = fields [0]         # ID of this page
        pageRank = float(fields [1])  # current pagerank of this page
        outLinks = fields [2:]        # outgoing links of this page

        yield myId, outLinks

        nOutLinks = len(outLinks)
        if(nOutLinks > 0) :
            prContrib = pageRank / nOutLinks
            for mId in outLinks :
                yield mId, prContrib
```

```
    def reducer(self, mId, values):
        s = 0
        for val in values :
            if(isinstance(val, list)) :
                mOutLinks = val  # From "outLinks" yield in mapper above
            else :
                s += float(val)

        mPageRank = s * 0.85 + 0.15 / N
        yield None, mId + " " + str(mPageRank) + " " + " ".join(mOutLinks)
            # NB: key value irrelevant with RawValueProtocol output

    def steps(self) :
        # preprocessing, then N times "main" MR
        pre   = MRStep(mapper=self.preMapper, reducer=self.preReducer)
        step  = MRStep(mapper=self.mapper, reducer=self.reducer)
        return [pre] + [step] * NITER

if __name__ == '__main__':
    MRPageRank.run()


^G Get Help ^O WriteOut ^R Read File^Y Prev Page^K Cut Text ^C Cur Pos
^X Exit     ^J Justify  ^W Where Is ^V Next Page^U UnCut Tex^T To Spell
```

Output:

```
Running step 51 of 51...
job output is in /tmp/mr_page_rank_big.up2225522.20240206.150932.402279/outp
ut
Streaming final output from /tmp/mr_page_rank_big.up2225522.20240206.150932.
402279/output...
1 0.027645934714267257
10 0.013636363636363636 5
11 0.013636363636363636 5
2 0.3241496975987292 3
3 0.2892207429507734 2
4 0.03296369665389087 1 2
5 0.06821411653244908 2 4 6
6 0.03296369665389087 2 5
7 0.013636363636363636 2 5
8 0.013636363636363636 2 5
9 0.013636363636363636 2 5
Removing temp directory /tmp/mr_page_rank_big.up2225522.20240206.150932.4022
79...
```

Final version to run with web-wiki.txt:

```
up2225522@mn01:~/mytestd
  GNU nano 2.3.1          File: mr_page_rank_big1.py                Modified

from mrjob.job import MRJob
from mrjob.step import MRStep
from mrjob.protocol import RawValueProtocol

NITER = 50
N = 11
initPageRank = 1 / N

class MRPageRank(MRJob):

    OUTPUT_PROTOCOL = RawValueProtocol
        # Default for input; use here for output so we can "input our outpu$

    def preMapper(self, key, value):

        if len(value) == 0 or value [0] == '#':
            return

        fields = value.split()
        fromNode = fields [0]
        toNode   = fields [1]

        yield toNode, None
        yield fromNode, toNode

    def preReducer(self, mId, values):

        outLinks = []
        for val in values :
            if not (val is None) :
                outLinks.append(val)

        yield None, mId + " " + str(initPageRank) + " " + " ".join(outLinks)
```

```
def intermapper(self, key, line):

    fields = line.split()
    myId  = fields [0]      # ID of this page

    yield myId, line        # forward directly to interreducer for outp$

    if(len(fields) == 2) :  # dangling node
        pageRank = float(fields [1])   # current pagerank of this page
        yield "accDangling", pageRank
```

```
  def interreducer(self, mId, values):

      if mId == "accDangling" :
          s = sum(values)
          yield None, "accDangling " + str(s)
      else :
          # Should only receive one value, from first yield in intermapper
          line = next(values)
          yield None, line


  def mapper(self, key, value):

      fields = value.split()
      myId     = fields [0]         # ID of this page
      pageRank = float(fields [1])  # current pagerank of this page

      if myId == "accDangling" :
          p = pageRank / N
          for i in range(N) :
              yield str(i + 1), p   # to be shared amongst *all N* nodes
      else :
          outLinks = fields [2:]    # outgoing links of this page
```

```
    def reducer(self, mId, values):
        s = 0
        for val in values :
            if(isinstance(val, list)) :
                mOutLinks = val  # From "outLinks" yield in mapper above
            else :
                s += float(val)

        mPageRank = s * 0.85 + 0.15 / N
        yield None, mId + " " + str(mPageRank) + " " + " ".join(mOutLinks)


    def steps(self) :
        # preprocessing, then N times "inter" MR followed by "main" MR
        pre   = MRStep(mapper=self.preMapper, reducer=self.preReducer)
        inter = MRStep(mapper=self.intermapper, reducer=self.interreducer)
        step  = MRStep(mapper=self.mapper, reducer=self.reducer)
        return [pre] + [inter, step] * NITER

if __name__ == '__main__':
    MRPageRank.run()



^G Get Help ^O WriteOut ^R Read File^Y Prev Page^K Cut Text ^C Cur Pos
^X Exit     ^J Justify  ^W Where Is ^V Next Page^U UnCut Tex^T To Spell
```

The final version utilizes the `preMapper` and `preReducer` functions for preprocessing to identify node connections and initialize PageRank values, while the main `mapper` and `reducer` functions iteratively update PageRank values based on incoming contributions from neighbouring nodes. The code streamlines PageRank computation through preprocessing and iterative MapReduce steps to converge on accurate PageRank values.

Output:

```
Running step 101 of 101...
job output is in /tmp/mr_page_rank_big1.up2225522.20240206.160144.023370/out
put
Streaming final output from /tmp/mr_page_rank_big1.up2225522.20240206.160144
.023370/output...
1 0.03278149315934767
10 0.01616947901685893 5
11 0.01616947901685893 5
2 0.3843697809528769 3
3 0.3429414533690357 2
4 0.03908709209997011 1 2
5 0.08088569323450433 2 4 6
6 0.03908709209997011 2 5
7 0.01616947901685893 2 5
8 0.01616947901685893 2 5
9 0.01616947901685893 2 5
Removing temp directory /tmp/mr_page_rank_big1.up2225522.20240206.160144.023
370...
[up2225522@mn01 mytestdir]$
```

The output is now the same as web-graph.txt

Unzipping web data:

```
[up2225522@mn01 mytestdir]$ gunzip web-BerkStan.txt.gz
```

/home/up2225522/mytestdir

Name	Size	Changed	Rights	Owner
..		06/02/2024 14:31:06	rwx------	up2225...
Ubuntu-dialogue-corp...		23/01/2024 19:24:29	rwxrwxr-x	up2225...
mr_page_rank.py	2 KB	06/02/2024 14:45:26	rw-rw-r--	up2225...
mr_page_rank_big.py	2 KB	06/02/2024 15:08:47	rw-rw-r--	up2225...
mr_page_rank_big1.py	3 KB	06/02/2024 15:59:15	rw-rw-r--	up2225...
mr_page_rank1.py	1 KB	06/02/2024 14:23:16	rw-rw-r--	up2225...
mr_preprocess.py	1 KB	06/02/2024 15:01:25	rw-rw-r--	up2225...
mr_unix_wc.py	1 KB	30/01/2024 15:36:34	rw-rw-r--	up2225...
mr_word_count.py	1 KB	30/01/2024 17:11:17	rw-rw-r--	up2225...
myfile.txt	1 KB	23/01/2024 16:47:34	rw-rw-r--	up2225...
poem.txt	2 KB	30/01/2024 15:32:45	rw-rw-r--	up2225...
web-BerkStan.txt	107,555 KB	05/02/2024 19:34:11	rw-rw-r--	up2225...
web-graph.txt	1 KB	06/02/2024 14:30:30	rw-rw-r--	up2225...
web-Wikipedia.txt	1 KB	06/02/2024 14:54:00	rw-rw-r--	up2225...

Running with multiple clusters:

```
up2225522@mn01:~/mytestd
txt
[up2225522@mn01 mytestdir]$ python mr_page_rank_big1.py -r hadoop -o hdfs://
/user/up2225522/out -D mapred.reduce.tasks=10 hdfs:///user/up2225522/web-Ber
kStan.txt
No configs found; falling back on auto-configuration
No configs specified for hadoop runner
Looking for hadoop binary in /home/hadoop/hadoop-2.9.2/bin...
Found hadoop binary: /home/hadoop/hadoop-2.9.2/bin/hadoop
Using Hadoop version 2.9.2
Looking for Hadoop streaming jar in /home/hadoop/hadoop-2.9.2...
Found Hadoop streaming jar: /home/hadoop/hadoop-2.9.2/share/hadoop/tools/lib
/hadoop-streaming-2.9.2.jar
Creating temp directory /tmp/mr_page_rank_big1.up2225522.20240206.162146.065
359
Copying local files to hdfs:///user/up2225522/tmp/mrjob/mr_page_rank_big1.up
2225522.20240206.162146.065359/files/...
Detected hadoop configuration property names that do not match hadoop versio
n 2.9.2:
The have been translated as follows
 mapred.reduce.tasks: mapreduce.job.reduces
Running step 1 of 11...
  packageJobJar: [/tmp/hadoop-unjar4043952498190941994/] [] /tmp/streamjob88
9935000952394654.jar tmpDir=null
  Connecting to ResourceManager at /10.3.8.1:8032
  Connecting to ResourceManager at /10.3.8.1:8032
  Total input files to process : 1
  number of splits:2
  mapred.reduce.tasks is deprecated. Instead, use mapreduce.job.reduces
  yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Inste
ad, use yarn.system-metrics-publisher.enabled
  Submitting tokens for job: job_1681658814718_2165
  Submitted application application_1681658814718_2165
  The url to track the job: http://mn01.soc.uni.ds.port.ac.uk:8081/proxy/app
lication_1681658814718_2165/
  Running job: job_1681658814718_2165
  Job job_1681658814718_2165 running in uber mode : false
   map 0% reduce 0%
   map 10% reduce 0%
   map 22% reduce 0%
```

Output:

```
[up2225522@mn01 mytestdir]$ hdfs dfs -ls out
Found 11 items
-rw-r--r--   3 up2225522 up2225522          0 2024-02-06 16:28 out/_SUCCESS
-rw-r--r--   3 up2225522 up2225522    7212411 2024-02-06 16:28 out/part-0000
0
-rw-r--r--   3 up2225522 up2225522    7187233 2024-02-06 16:28 out/part-0000
1
-rw-r--r--   3 up2225522 up2225522    7188082 2024-02-06 16:28 out/part-0000
2
-rw-r--r--   3 up2225522 up2225522    7175490 2024-02-06 16:28 out/part-0000
3
-rw-r--r--   3 up2225522 up2225522    7179994 2024-02-06 16:28 out/part-0000
4
-rw-r--r--   3 up2225522 up2225522    7197662 2024-02-06 16:28 out/part-0000
5
-rw-r--r--   3 up2225522 up2225522    7206964 2024-02-06 16:28 out/part-0000
6
-rw-r--r--   3 up2225522 up2225522    7194922 2024-02-06 16:28 out/part-0000
7
-rw-r--r--   3 up2225522 up2225522    7198215 2024-02-06 16:28 out/part-0000
8
-rw-r--r--   3 up2225522 up2225522    7203793 2024-02-06 16:28 out/part-0000
9
```

Processor monitoring:

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers
application_1681658814718_2165	up2225522	streamjob889935000952394654.jar	MAPREDUCE	default	0	Tue Feb 6 16:21:59 +0000 2024	N/A	RUNNING	UNDEFINED	10

Allocated CPU VCores	Allocated Memory MB	Reserved CPU VCores	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
10	11264	0	0	3.1	3.1		ApplicationMaster	0

It is shown that the 10 nodes are used for parallel processing

Week 3: Reflection

In this week's lab, we implemented the PageRank algorithm on a Hadoop cluster, dividing it into mappers and reducers for data parsing, updating PageRank values, and defining a pre-processor function. The algorithm efficiently processed large-scale web data, emphasizing dangling nodes and iterative convergence for accurate rankings.

Week 4

Basic Apache Spark

Opening spark

```
[up2225522@mn01 ~]$ pyspark
Python 3.6.10 |Anaconda, Inc.| (default, Jan  7 2020, 21:14:29)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
2024-02-13 14:38:30 WARN  NativeCodeLoader:62 - Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(
newLevel).
2024-02-13 14:38:31 WARN  Utils:66 - Service 'SparkUI' could not bind on port 4040
. Attempting port 4041.
2024-02-13 14:38:31 WARN  Utils:66 - Service 'SparkUI' could not bind on port 4041
. Attempting port 4042.
2024-02-13 14:38:31 WARN  Utils:66 - Service 'SparkUI' could not bind on port 4042
. Attempting port 4043.
2024-02-13 14:38:31 WARN  Utils:66 - Service 'SparkUI' could not bind on port 4043
. Attempting port 4044.
2024-02-13 14:38:31 WARN  Utils:66 - Service 'SparkUI' could not bind on port 4044
. Attempting port 4045.
2024-02-13 14:38:31 WARN  Utils:66 - Service 'SparkUI' could not bind on port 4045
. Attempting port 4046.
2024-02-13 14:38:31 WARN  Utils:66 - Service 'SparkUI' could not bind on port 4046
. Attempting port 4047.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.4.0
      /_/

Using Python version 3.6.10 (default, Jan  7 2020 21:14:29)
SparkSession available as 'spark'.
```

Read the text file and display the rows by using show():

```
>>> textFile = spark.read.text("file:/home/spark/spark-2.4.0-bin-hadoop2.7/README.
md")
>>> textFile.show()
+--------------------+
|               value|
+--------------------+
|      # Apache Spark|
|                    |
|Spark is a fast a...|
|high-level APIs i...|
|supports general ...|
|rich set of highe...|
|MLlib for machine...|
|and Spark Streami...|
|                    |
|<http://spark.apa...|
|                    |
|                    |
|## Online Documen...|
|                    |
|You can find the ...|
|guide, on the [pr...|
|This README file ...|
|                    |
|   ## Building Spark|
|                    |
+--------------------+
only showing top 20 rows
```

Trying count():

```
>>> textFile.count()
105
```

Avoid truncated rows:

```
>>> textFile.show(truncate=False)
+------------------------------------------------------------------------------+
|value
   |
+------------------------------------------------------------------------------+
|# Apache Spark
   |
|
   |
|Spark is a fast and general cluster computing system for Big Data. It provides|
|high-level APIs in Scala, Java, Python, and R, and an optimized engine that   |
|supports general computation graphs for data analysis. It also supports a     |
|rich set of higher-level tools including Spark SQL for SQL and DataFrames,    |
|MLlib for machine learning, GraphX for graph processing,
   |
|and Spark Streaming for stream processing.
   |
|
   |
|<http://spark.apache.org/>
   |
|
   |
|
   |
|## Online Documentation
```

Store as variable and utilise filter function:

```
>>> linesWithSpark = textFile.filter(textFile.value.contains("Spark"))
>>> linesWithSpark.show()
+--------------------+
|               value|
+--------------------+
|      # Apache Spark|
|Spark is a fast a...|
|rich set of highe...|
|and Spark Streami...|
|You can find the ...|
|   ## Building Spark|
|Spark is built us...|
|To build Spark an...|
|You can build Spa...|
|["Building Spark"...|
|For general devel...|
|The easiest way t...|
|Spark also comes ...|
|    ./bin/run-exa...|
|    MASTER=spark:...|
|Testing first req...|
|Spark uses the Ha...|
|Hadoop, you must ...|
|in the online doc...|
|Please review the...|
+--------------------+
```

```
>>> linesWithSpark.count()
20
```

Reading CSV file and displaying results:

```
>>> flightData2015 = spark.read.option("inferSchema", "true").option("header", "tr
ue").csv("file:/home/spark/Spark-The-Definitive-Guide-master/data/flight-data/csv/
2015-summary.csv")
>>> flightData2015.show()
+-------------------+-------------------+-----+
|   DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-------------------+-------------------+-----+
|       United States|            Romania|   15|
|       United States|            Croatia|    1|
|       United States|            Ireland|  344|
|              Egypt|      United States|   15|
|       United States|              India|   62|
|       United States|          Singapore|    1|
|       United States|            Grenada|   62|
|          Costa Rica|      United States|  588|
|             Senegal|      United States|   40|
|             Moldova|      United States|    1|
|       United States|       Sint Maarten|  325|
|       United States|   Marshall Islands|   39|
|              Guyana|      United States|   64|
|               Malta|      United States|    1|
|            Anguilla|      United States|   41|
|             Bolivia|      United States|   30|
|       United States|           Paraguay|    6|
|             Algeria|      United States|    4|
|Turks and Caicos ...|      United States|  230|
|       United States|          Gibraltar|    1|
+-------------------+-------------------+-----+
only showing top 20 rows

>>> flightData2015.describe()
DataFrame[summary: string, DEST_COUNTRY_NAME: string, ORIGIN_COUNTRY_NAME: string,
 count: string]
```

Reducing partition numbers:

```
>>> spark.conf.set("spark.sql.shuffle.partitions", "5")
```

This command configures the default number of partitions used for shuffling data during Spark SQL operations. It impacts performance, resource utilization, and parallelism. Adjusting this parameter based on workload and cluster resources can optimise query execution.

Utilising SQL commands with the dataframe:

```
>>> flightData2015.select("DEST_COUNTRY_NAME", "count").show(10)
+-----------------+-----+
|DEST_COUNTRY_NAME|count|
+-----------------+-----+
|    United States|   15|
|    United States|    1|
|    United States|  344|
|            Egypt|   15|
|    United States|   62|
|    United States|    1|
|    United States|   62|
|       Costa Rica|  588|
|          Senegal|   40|
|          Moldova|    1|
+-----------------+-----+
only showing top 10 rows
```

```
>>> flightData2015.select(flightData2015.DEST_COUNTRY_NAME, flightData2015 ["count
"]).show(10)
+-----------------+-----+
|DEST_COUNTRY_NAME|count|
+-----------------+-----+
|    United States|   15|
|    United States|    1|
|    United States|  344|
|            Egypt|   15|
|    United States|   62|
|    United States|    1|
|    United States|   62|
|       Costa Rica|  588|
|          Senegal|   40|
|          Moldova|    1|
+-----------------+-----+
only showing top 10 rows
```

Utilise aggregation function:

```
>>> flightData2015.select("DEST_COUNTRY_NAME", "count").groupBy("DEST_COUNTRY_NAME").sum("count").show(10)
+----------------+----------+
|DEST_COUNTRY_NAME|sum(count)|
+----------------+----------+
|        Anguilla|        41|
|         Russia|       176|
|       Paraguay|        60|
|        Senegal|        40|
|         Sweden|       118|
|       Kiribati|        26|
|         Guyana|        64|
|     Philippines|       134|
|       Djibouti|         1|
|       Malaysia|         2|
+----------------+----------+
only showing top 10 rows
```

The command above groups the column values by “DEST_COUNTRY_NAME” and then sums the count for each respect value.

Querying with SQL statements:

```
>>> flightData2015.createOrReplaceTempView("flight_data_2015")
>>> spark.sql("""
...       SELECT DEST_COUNTRY_NAME, sum(count)
...       FROM flight_data_2015
...       GROUP BY DEST_COUNTRY_NAME
...       """).show(10)
2024-02-13 15:42:48 WARN  ObjectStore:6666 - Version information not found in meta
store. hive.metastore.schema.verification is not enabled so recording the schema v
ersion 1.2.0
2024-02-13 15:42:49 WARN  ObjectStore:568 - Failed to get database default, return
ing NoSuchObjectException
2024-02-13 15:42:50 WARN  ObjectStore:568 - Failed to get database global_temp, re
turning NoSuchObjectException
+-----------------+----------+
|DEST_COUNTRY_NAME|sum(count)|
+-----------------+----------+
|        Anguilla|        41|
|         Russia|       176|
|       Paraguay|        60|
|        Senegal|        40|
|         Sweden|       118|
|       Kiribati|        26|
|         Guyana|        64|
|     Philippines|       134|
|       Djibouti|         1|
|       Malaysia|         2|
+-----------------+----------+
only showing top 10 rows
```

```
>>> sqlMax = spark.sql("SELECT max(count) from flight_data_2015")
>>> sqlMax.show()
+----------+
|max(count)|
+----------+
|    370002|
+----------+
```

```
>>> high5Sql = spark.sql("""
...        SELECT DEST_COUNTRY_NAME, sum(count) as destination_total
...        FROM flight_data_2015
...        GROUP BY DEST_COUNTRY_NAME
...        ORDER BY sum(count) DESC
...        LIMIT 5
...        """)
>>> high5Sql.show()
+-----------------+-----------------+
|DEST_COUNTRY_NAME|destination_total|
+-----------------+-----------------+
|    United States|           411352|
|           Canada|             8399|
|           Mexico|             7140|
|   United Kingdom|             2025|
|            Japan|             1548|
+-----------------+-----------------+
```

(Summation of the top 5 destination counts)

Same command but through spark API:

```
>>> from pyspark.sql.functions import desc
>>> flightData2015.select("DEST_COUNTRY_NAME", "count").groupBy("DEST_COUNTRY_NAME
").sum("count").sort(desc("sum(count)")).limit(5).show()
+-----------------+----------+
|DEST_COUNTRY_NAME|sum(count)|
+-----------------+----------+
|    United States|    411352|
|           Canada|      8399|
|           Mexico|      7140|
|   United Kingdom|      2025|
|            Japan|      1548|
+-----------------+----------+
```

Implementing word count:

```
>>> from pyspark.sql.functions import split, explode
>>> wordCounts = textFile.select(explode(split(textFile.value, "\s+")).alias("word
")).groupBy("word").count()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'textFile' is not defined
>>> textFile = spark.read.text("file:/home/spark/spark-2.4.0-bin-hadoop2.7/README.
md")
>>> wordCounts = textFile.select(explode(split(textFile.value, "\s+")).alias("word
")).groupBy("word").count()
>>> wordCounts.show(10)
+-------------+-----+
|         word|count|
+-------------+-----+
|       online|    1|
|       graphs|    1|
|   ["Parallel|    1|
|   ["Building|    1|
|       thread|    1|
|documentation|    3|
|     command,|    2|
|  abbreviated|    1|
|     overview|    1|
|         rich|    1|
+-------------+-----+
only showing top 10 rows
```

The command above processes a text file by splitting it into words, counting the occurrences of each word, and grouping the results by word.

Machine learning with Spark

Reading CSV & count()

```
>>> melbourne_data = spark.read.option("inferSchema", "true").option("header", "tr
ue").csv("file:/home/spark/melb_data.csv")
>>> melbourne_data.show(10)
+----------+------------------+-----+----+---------+------+-------+---------+-----
---+--------+--------+--------+---+--------+------------+---------+-----------+---
------+----------+--------------------+-------------+
|    Suburb|           Address|Rooms|Type|    Price|Method|SellerG|     Date|Dista
nce|Postcode|Bedroom2|Bathroom|Car|Landsize|BuildingArea|YearBuilt|CouncilArea|Lat
titude|Longtitude|          Regionname|Propertycount|
+----------+------------------+-----+----+---------+------+-------+---------+-----
---+--------+--------+--------+---+--------+------------+---------+-----------+---
------+----------+--------------------+-------------+
|Abbotsford|      85 Turner St|    2|   h|1480000.0|     S| Biggin|3/12/2016|
2.5|  3067.0|     2.0|     1.0|1.0|   202.0|        null|     null|      Yarra| -3
7.7996|  144.9984|Northern Metropol...|       4019.0|
|Abbotsford|   25 Bloomburg St|    2|   h|1035000.0|     S| Biggin|4/02/2016|
2.5|  3067.0|     2.0|     1.0|0.0|   156.0|        79.0|   1900.0|      Yarra| -3
```

```
>>> melbourne_data.count()
13580
```

Data cleaning by dropping rows with N/A:

```
>>> melbourne_data_clean = melbourne_data.dropna()
>>> melbourne_data.count()
13580
>>> melbourne_data_clean.count()
6196
```

Vector assemble:

```
>>> from pyspark.ml.feature import VectorAssembler
>>> vectorAssembler = VectorAssembler().setInputCols(["Rooms", "Bathroom", "Landsi
ze", "Lattitude", "Longtitude"]).setOutputCol("features")
>>> melbourne_data_assembled = vectorAssembler.transform(melbourne_data_clean)
>>> melbourne_data_assembled.first()
Row(Suburb='Abbotsford', Address='25 Bloomburg St', Rooms=2, Type='h', Price=10350
00.0, Method='S', SellerG='Biggin', Date='4/02/2016', Distance=2.5, Postcode=3067.
0, Bedroom2=2.0, Bathroom=1.0, Car=0.0, Landsize=156.0, BuildingArea=79.0, YearBui
lt=1900.0, CouncilArea='Yarra', Lattitude=-37.8079, Longtitude=144.9934, Regionnam
e='Northern Metropolitan', Propertycount=4019.0, features=DenseVector([2.0, 1.0, 1
56.0, -37.8079, 144.9934]))
```

The command above combines the specified input columns into a single output column named “features” and then is applied to the clean data.

Splitting test/train data:

```
>>> (trainingData, testData) = melbourne_data_assembled.randomSplit([0.7, 0.3])
```

Importing a ML library then training & predict and then displaying the predictions and actual results:

```
>>> from pyspark.ml.regression import RandomForestRegressor
>>> rf = RandomForestRegressor(labelCol="Price",featuresCol="features")
>>> model = rf.fit(trainingData)
>>> predictions = model.transform(testData)
>>> predictions.select("prediction", "Price").show()
+------------------+---------+
|        prediction|    Price|
+------------------+---------+
| 810446.2911798055| 750000.0|
|  858756.890841562|1012500.0|
| 1240184.801543386|1165000.0|
|1269722.7024533513|1195000.0|
| 861109.8121975254|1180000.0|
| 1305154.636661316|1465000.0|
| 964622.8356986992|1100000.0|
| 843721.8586911479| 870000.0|
| 855873.2231102416|1636000.0|
| 1409479.620945933|1200000.0|
| 401199.6003831327| 380000.0|
| 1761205.866666052|1740000.0|
|1067215.4306054027|1190000.0|
|1504687.4966072044|1646000.0|
| 740049.3316649504| 752000.0|
| 752860.2496172766| 700000.0|
| 745197.5397649952| 685000.0|
| 779893.3005065501|1042000.0|
| 779893.3005065501| 715000.0|
| 765905.5707868002| 840000.0|
+------------------+---------+
only showing top 20 rows
```

Importing and Displaying evaluation metrics:

```
>>> from pyspark.ml.evaluation import RegressionEvaluator
>>> evaluator = RegressionEvaluator(labelCol="Price", predictionCol="prediction",
metricName="mae")
>>> mae = evaluator.evaluate(predictions)
>>> print("Mean Absolute Error (MAE) on test data = " + str(mae))
Mean Absolute Error (MAE) on test data = 281287.13461936236
```

The model does not perform well as reflected by the high MSE

Adjusting Max_depth parameter to 10:

```
>>> rf = RandomForestRegressor(labelCol="Price",featuresCol="features", maxDepth=1
0)
>>> model = rf.fit(trainingData)
>>> predictions = model.transform(testData)
>>> predictions.select("prediction", "Price").show(10)
+------------------+---------+
|        prediction|    Price|
+------------------+---------+
|  790281.798997449| 750000.0|
|1031760.0306788102|1012500.0|
|1257006.0718732025|1165000.0|
|1286595.0687939494|1195000.0|
|1038463.8168468463|1180000.0|
|1433722.2295428808|1465000.0|
|1005981.2021852263|1100000.0|
| 766300.5210523049| 870000.0|
|1012790.3190821239|1636000.0|
|1476990.3086913382|1200000.0|
+------------------+---------+
only showing top 10 rows

>>> evaluator = RegressionEvaluator(labelCol="Price", predictionCol="prediction",
metricName="mae")
>>> mae = evaluator.evaluate(predictions)
>>> print("Mean Absolute Error (MAE) on test data = " + str(mae))
Mean Absolute Error (MAE) on test data = 219546.89412667693
```

Adjusting Max_depth parameter to 20:

```
>>> rf = RandomForestRegressor(labelCol="Price",featuresCol="features", maxDepth=
0)
>>> model = rf.fit(trainingData)
predictions = model.transform(testData)
>>> predictions = model.transform(testData)
>>> predictions.select("prediction", "Price").show(10)
+------------------+---------+
|        prediction|    Price|
+------------------+---------+
| 795289.7021974057| 750000.0|
|1066249.8882217656|1012500.0|
|1123081.7706982975|1165000.0|
| 1157564.295950823|1195000.0|
| 988750.9217155051|1180000.0|
|1267186.0791155747|1465000.0|
|1051984.8161764706|1100000.0|
| 784300.6216530525| 870000.0|
| 1140136.194438697|1636000.0|
|1282530.5179946595|1200000.0|
+------------------+---------+
only showing top 10 rows

>>> evaluator = RegressionEvaluator(labelCol="Price", predictionCol="prediction",
metricName="mae")
>>> mae = evaluator.evaluate(predictions)
>>> print("Mean Absolute Error (MAE) on test data = " + str(mae))
Mean Absolute Error (MAE) on test data = 205647.15301988836
```

It may be the case that the more we increase the max-depth parameter, the MAE decreases.

Week 4: Reflection

In this week's lab session, we delved into Apache Spark, a powerful platform for handling big data. Spark offers a wide range of capabilities, including SQL functions, streaming, and machine learning. We explored its MLlib, a fast and scalable machine learning library inspired by Scikit-learn. By leveraging Spark, we can process large datasets efficiently and apply sophisticated algorithms for advanced analytics and predictions.

Week 5

Using ALS in Spark

Data pre-processing:

```
>>> lines = spark.read.text("file:///home/spark/spark-2.4.0-bin-hadoop2.7/d
ata/mllib/als/sample_movielens_ratings.txt")
>>> from pyspark.sql.functions import split
>>> parts = lines.withColumn("value", split("value", "::"))
>>> ratings = parts.select(parts.value.getItem(0).cast("int").alias("userId
"), parts.value.getItem(1).cast("int").alias("movieId"), parts.value.getIte
m(2).cast("float").alias("rating"))
>>> ratings.show(10)
+------+-------+------+
|userId|movieId|rating|
+------+-------+------+
|     0|      2|   3.0|
|     0|      3|   1.0|
|     0|      5|   2.0|
|     0|      9|   4.0|
|     0|     11|   1.0|
|     0|     12|   2.0|
|     0|     15|   1.0|
|     0|     17|   1.0|
|     0|     19|   1.0|
|     0|     21|   1.0|
+------+-------+------+
only showing top 10 rows
```

Training & Predicting:

```
>>> (training, test) = ratings.randomSplit([0.8, 0.2])
>>> from pyspark.ml.recommendation import ALS
>>> als = ALS(maxIter=10, regParam=0.05, coldStartStrategy="drop", userCol=
"userId", itemCol="movieId", ratingCol="rating")
>>> model = als.fit(training)
2024-02-20 15:25:22 WARN  BLAS:61 - Failed to load implementation from: com
.github.fommil.netlib.NativeSystemBLAS
2024-02-20 15:25:22 WARN  BLAS:61 - Failed to load implementation from: com
.github.fommil.netlib.NativeRefBLAS
2024-02-20 15:25:23 WARN  LAPACK:61 - Failed to load implementation from: c
om.github.fommil.netlib.NativeSystemLAPACK
2024-02-20 15:25:23 WARN  LAPACK:61 - Failed to load implementation from: c
om.github.fommil.netlib.NativeRefLAPACK
>>> als = ALS(maxIter=10, regParam=0.05, coldStartStrategy="drop", userCol=
"userId", itemCol="movieId", ratingCol="rating")
>>> model = als.fit(training)
>>> predictions = model.transform(test)
>>> predictions.show(10)
+------+-------+------+----------+
|userId|movieId|rating|prediction|
+------+-------+------+----------+
|    12|     31|   4.0|0.93509126|
|     6|     85|   3.0|  2.492552|
|    28|     65|   1.0| 1.0401498|
|    16|     65|   1.0| 0.8211255|
|     3|     65|   2.0| 0.7515473|
|    19|     65|   1.0| 1.5756226|
|    23|     65|   5.0| 1.7785054|
|     2|     65|   1.0| 0.3817751|
|     9|     53|   1.0|0.30779952|
|     8|     53|   5.0| 1.6135321|
+------+-------+------+----------+
only showing top 10 rows
```

Model Evaluation:

```
>>> from pyspark.ml.evaluation import RegressionEvaluator
>>> evaluator = RegressionEvaluator(metricName="rmse", labelCol="rating", p
redictionCol="prediction")
>>> rmse = evaluator.evaluate(predictions)
>>> print("Root-mean-squared error = " + str(rmse))
Root-mean-squared error = 1.181645490815753
```

Obtaining top ten recommendations for each of the 3 users:

```
>>> users = ratings.select("userID").distinct().limit(3)
>>> userSubsetRecs = model.recommendForUserSubset(users, 10)
>>> userSubsetRecs.show(truncate=False)
[Stage 257:=====================================>              (73 + 24) /
[Stage 257:=============================================>      (85 + 15) /

+------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
------------------------+
|userId|recommendations

                        |
+------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
------------------------+
|28    |[[92, 4.8227577], [81, 4.563517], [12, 4.466758], [40, 3.9481149],
[49, 3.8323007], [89, 3.7613683], [82, 3.509053], [94, 3.4520905], [46, 3.3
656921], [41, 3.032614]]|
|26    |[[51, 5.2844276], [23, 4.947725], [94, 4.933737], [88, 4.8844595],
[22, 4.836465], [92, 4.674999], [53, 4.6018496], [24, 4.535016], [68, 3.887
6326], [54, 3.8321948]] |
|27    |[[18, 3.858637], [32, 3.8220105], [8, 3.3317208], [49, 3.2185998],
[7, 3.1757782], [34, 2.9050868], [51, 2.8881083], [27, 2.8269284], [83, 2.7
37769], [30, 2.7016382]]|
+------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
------------------------+
```

The scores represent the model's confidence in the user's preferences.

Running Spark across the cluster

Create a python file with minor modifications:

```
from pyspark.sql import SparkSession

from pyspark.sql.functions import split

from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.recommendation import ALS

import time


spark = SparkSession.builder.appName("ALSExample").getOrCreate()

startTime = time.time()

lines = spark.read.text("spark-examples/sample_movielens_ratings.txt")

parts = lines.withColumn("value", split("value", "::"))
ratings = parts.select(parts.value.getItem(0).cast("int").alias("userId"),
                       parts.value.getItem(1).cast("int").alias("movieId"),
                       parts.value.getItem(2).cast("float").alias("rating"$

(training, test) = ratings.randomSplit([0.8, 0.2])

# Build the recommendation model using ALS on the training data
# Note we set cold start strategy to 'drop' to ensure we don't get NaN eva$
als = ALS(maxIter=10, regParam=0.05, coldStartStrategy="drop",
          userCol="userId", itemCol="movieId", ratingCol="rating")
model = als.fit(training)

# Evaluate the model by computing the RMSE on the test data
predictions = model.transform(test)
evaluator = RegressionEvaluator(metricName="rmse", labelCol="rating",
                                predictionCol="prediction")
rmse = evaluator.evaluate(predictions)
```

```
rmse = evaluator.evaluate(predictions)
print("Root-mean-squared error = " + str(rmse))

# Generate top 10 movie recommendations for a specified set of users
users = ratings.select(als.getUserCol()).distinct().limit(3)
userSubsetRecs = model.recommendForUserSubset(users, 10)

userSubsetRecs.show(truncate=False)

endTime = time.time()

print("Elapsed time = " + str(endTime - startTime))

spark.stop()
```

The Spark session object, spark, is created automatically in Python. The script line

```
spark = SparkSession.builder.appName("ALSExample").getOrCreate()
```

clearly creates it.

The script modifies the path argument for "spark.read.text" after reading data from HDFS. Spark performance is assessed by timing the program's start and finish.

Transferring file from Linux to Hadoop:

```
[up2225522@mn01 ~]$ hdfs dfs -mkdir spark-examples
[up2225522@mn01 ~]$ hdfs dfs -put /home/spark/spark-2.4.0-bin-hadoop2.7/dat
a/mllib/als/sample_movielens_ratings.txt spark-examples
[up2225522@mn01 ~]$ hdfs dfs -ls spark-examples
Found 1 items
-rw-r--r--   3 up2225522 up2225522      32363 2024-02-20 15:36 spark-exampl
es/sample_movielens_ratings.txt
```

Running on Hadoop:

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus
application_1681658814718_3543	up2225522	ALSExample	SPARK	default	0	Tue Feb 20 15:49:02 +0000 2024	Tue Feb 20 15:49:43 +0000 2024	FINISHED	SUCCEEDED

The job is finished fairly quickly and predictions for the 3 users are as follows:

```
|userId|recommendations
                          |
+------+--------------------------------------------------------------------
-------------------------------------------------------------------------------
--------------------------+
|28    |[[92, 4.6931562], [81, 4.339291], [89, 3.7562668], [82, 3.4041328],
 [53, 2.8697495], [23, 2.8452682], [19, 2.8166122], [0, 2.8030622], [57, 2.
7577586], [4, 2.5388455]] |
|26    |[[88, 5.027465], [94, 4.9876475], [51, 4.9089813], [22, 4.8899665],
 [92, 4.823366], [23, 4.5981426], [7, 4.397109], [24, 4.3939877], [68, 4.04
14715], [75, 4.015758]]   |
|27    |[[18, 3.6258378], [32, 3.5159216], [30, 3.3378725], [23, 3.0879343]
, [80, 3.040659], [66, 2.9910355], [27, 2.943649], [19, 2.8636105], [33, 2.
8099527], [75, 2.7957644]]|
+------+--------------------------------------------------------------------
-------------------------------------------------------------------------------
--------------------------+
```

Run it again with a larger dataset (100k), as this time it is a tabular data the code is adjusted:

Trasfer u.data to Hadoop:

```
[up2225522@mn01 ~]$ hdfs dfs -put /home/spark/datasets/ml-100k/u.data spark
-examples
[up2225522@mn01 ~]$ ls
als_example.py  als_ml-100k.py  derby.log  metastore_db  mytestdir
[up2225522@mn01 ~]$ hdfs dfs -ls spark-examples
Found 2 items
-rw-r--r--   3 up2225522 up2225522      32363 2024-02-20 15:36 spark-exampl
es/sample_movielens_ratings.txt
-rw-r--r--   3 up2225522 up2225522    1979173 2024-02-20 16:21 spark-exampl
es/u.data
```

Adjust Python code:

```
spark = SparkSession.builder.appName("ALS100k").getOrCreate()

startTime = time.time()

lines = spark.read.text("spark-examples/u.data")

parts = lines.withColumn("value", split("value", "\t"))
```

Run on Hadoop:

```
NativeMethodAccessorImpl.java:0, took 0.306319 s
+------+-----------------------------------------------------------------
-----------------------------------------------------------------------
-------------------------------------+
|userId|recommendations

                                     |
+------+-----------------------------------------------------------------
-----------------------------------------------------------------------
-------------------------------------+
|471   |[[1496, 6.0460267], [1242, 5.9857626], [263, 5.711217], [791, 5.671
297], [1620, 5.633374], [35, 5.6002], [833, 5.3431053], [862, 5.2531276], [
761, 5.1976514], [869, 5.17398]]      |
|463   |[[958, 4.9676476], [887, 4.775561], [793, 4.517633], [1129, 4.46519
3], [1251, 4.3891892], [1154, 4.3727307], [408, 4.2932816], [1149, 4.282136
], [1085, 4.2340627], [318, 4.220999]]|
|148   |[[1589, 5.4992814], [1240, 5.4826493], [1368, 5.3734155], [1143, 5.
285393], [119, 5.25959], [173, 5.254262], [61, 5.1691804], [958, 5.161663],
 [613, 5.1152043], [589, 5.09351]]    |
+------+-----------------------------------------------------------------
-----------------------------------------------------------------------
-------------------------------------+

Elapsed time = 34.38858366012573
```

up2225522	ALS100k	SPARK	default	0	Tue Feb 20 16:26:22 +0000 2024	Tue Feb 20 16:27:16 +0000 2024	FINISHED	SUCCEEDED

The job took longer than the previous query

Trying an even larger dataset(20m):

Adjust the code:

```
spark = SparkSession.builder.appName("ALS20m").getOrCreate()

startTime = time.time()

ratings = spark.read.option("inferSchema", "true").option("header", "true").csv("spark-examples/ratings.csv")
(training, test) = ratings.randomSplit([0.8, 0.2])
```

Load the file to Hadoop:

```
[up2225522@mn01 ~]$ hdfs dfs -put /home/spark/datasets/ml-20m/ratings.csv s
park-examples
[up2225522@mn01 ~]$ hdfs dfs -ls spark-examples
Found 3 items
-rw-r--r--   3 up2225522 up2225522  533444411 2024-02-20 16:37 spark-exampl
es/ratings.csv
-rw-r--r--   3 up2225522 up2225522      32363 2024-02-20 15:36 spark-exampl
es/sample_movielens_ratings.txt
-rw-r--r--   3 up2225522 up2225522    1979173 2024-02-20 16:21 spark-exampl
es/u.data
```

Run it with an increasing number of executors:

2 executors:

User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Reserved CPU VCores	Reserved Memory MB	% of Queue	% of Cluster
up2225522	ALS20m	SPARK	default	0	Tue Feb 20 16:39:36 +0000 2024	N/A	RUNNING	UNDEFINED	3	3	5120	0	0	1.4	1.4

```
+------+----------------------
```

8 executors:

User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Reserved CPU VCores	Reserved Memory MB	% of Queue	% of Cluster
up2225522	ALS20m	SPARK	default	0	Tue Feb 20 16:52:27 +0000 2024	N/A	RUNNING	UNDEFINED	9	9	17408	0	0	4.8	4.8

```
+------+-----------------------
```

Week 6

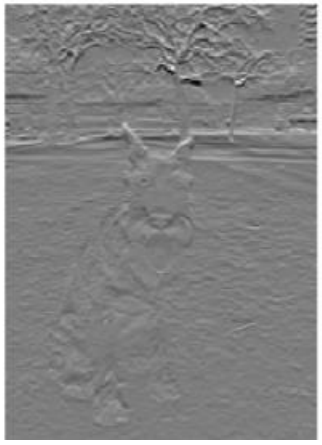
Images, Tensors and Convolutions

Set the working directory to Kaggle and import the pre-trained model then test loading images:

```
import sys
sys.path.append('/home/kaggle/learntools-master')
%matplotlib inline

from learntools.deep_learning.exercise_1 import load_my_image, apply_conv_to_image, show

original_image = load_my_image("/home/kaggle/dog-breed-identification/train/0246f44bb123ce3f91c939861eb97fb7.jpg")
show(original_image)
```

Detect bright over dark pixels:

```
conv = [[1, 1],
        [-1, -1]]

filtered_image = apply_conv_to_image(conv, original_image)
show(filtered_image)
```



Display images based on file path:

Image Recognition

In [5]:
```
from IPython.display import Image, display

img_paths = ['/home/kaggle/dog-breed-identification/test/000621fb3cbb32d8935728e48679680e.jpg',
             '/home/kaggle/dog-breed-identification/test/5441b8726acbf1158999122ba0729721.jpg',
             '/home/kaggle/dog-breed-identification/test/e8914f74848c23b70b520887b3ff2e8a.jpg']

#img_paths = ['/home/kaggle/seefood/test/not_hot_dog/13023.jpg',
#              '/home/kaggle/seefood/test/not_hot_dog/44578.jpg',
#              '/home/kaggle/seefood/test/not_hot_dog/75638.jpg']

display(Image(img_paths [0]))
display(Image(img_paths [1]))
display(Image(img_paths [2]))
```

Image pre-processing :

```
In [6]: import numpy as np
        from tensorflow.python.keras.applications.resnet50 import preprocess_input
        from tensorflow.python.keras.preprocessing.image import load_img, img_to_array

        image_size = 224

        def read_and_prep_images(img_paths, img_height=image_size, img_width=image_size):
            imgs = [load_img(img_path, target_size=(img_height, img_width)) for img_path in img_paths]
            img_array = np.array([img_to_array(img) for img in imgs])
            output = preprocess_input(img_array)
            return(output)
```

```
/home/anaconda3/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:523: FutureWarning: Passing (type, 1) or '1ty
pe' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/home/anaconda3/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:524: FutureWarning: Passing (type, 1) or '1ty
pe' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/home/anaconda3/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:525: FutureWarning: Passing (type, 1) or '1ty
pe' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/home/anaconda3/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:526: FutureWarning: Passing (type, 1) or '1ty
pe' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/home/anaconda3/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:527: FutureWarning: Passing (type, 1) or '1ty
pe' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/home/anaconda3/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:532: FutureWarning: Passing (type, 1) or '1ty
pe' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

The code preprocesses images using 'ResNet50' preprocessing and resizes them to a specified size.

Loading a pre-trained 'ResNet50' model with weights loaded from a specified file, preprocesses test images, and generates predictions using the model:

```
In [7]: from tensorflow.python.keras.applications import ResNet50

        my_model = ResNet50(weights='/home/kaggle/resnet50/resnet50_weights_tf_dim_ordering_tf_kernels.h5')

        test_data = read_and_prep_images(img_paths)
        preds = my_model.predict(test_data)
```

Displaying the top predicted labels for each image along with the corresponding image itself:

```
In [8]: from tensorflow.python.keras.applications.imagenet_utils import decode_predictions
        from IPython.display import Image, display

        #most_likely_labels = decode_predictions(preds, top=3, class_list_path='/home/kaggle/resnet50/imagenet_class_index.json')
        most_likely_labels = decode_predictions(preds, top=3)

        for i, img_path in enumerate(img_paths):
          display(Image(img_paths[i]))
          print(most_likely_labels[i])
```

```
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json
40960/35363 [==================================] - 0s 0us/step
```

```
[('n02088364', 'beagle', 0.9008499), ('n02089973', 'English_foxhound', 0.03226827), ('n02089867', 'Walker_hound', 0.00840528)]
```

```
[('n02115641', 'dingo', 0.99699044), ('n02105162', 'malinois', 0.0010566617), ('n02091244', 'Ibizan_hound', 0.0006247607)]
```

Transfer Learning

In [9]:
```
from tensorflow.python.keras.applications import ResNet50
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense, Flatten, GlobalAveragePooling2D

# num_classes is the number of categories your model chooses between for each prediction
num_classes = 2
resnet_weights_path = '/home/kaggle/resnet50/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5'
    # Note this is *different* to weights file above.

my_new_model = Sequential()
my_new_model.add(ResNet50(include_top=False, pooling='avg', weights=resnet_weights_path))
my_new_model.add(Dense(num_classes, activation='softmax'))

# We don't want to train the first layers - ResNet50 is already trained
my_new_model.layers[0].trainable = False
```

The code defines a new sequential model with a ResNet50 base, followed by a dense layer with SoftMax activation for classification into two categories. It freezes the weights of the ResNet50 layers to prevent training.

Compiling the model with various functions and evaluation metric:

In [10]:
```
my_new_model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
In [11]: from tensorflow.python.keras.applications.resnet50 import preprocess_input
         from tensorflow.python.keras.preprocessing.image import ImageDataGenerator

         image_size = 224
         data_generator = ImageDataGenerator(preprocessing_function=preprocess_input)

         train_generator = data_generator.flow_from_directory(
                 '/home/kaggle/rural_and_urban_photos/train',
                 target_size=(image_size, image_size),
                 batch_size=24,
                 class_mode='categorical')

         validation_generator = data_generator.flow_from_directory(
                 '/home/kaggle/rural_and_urban_photos/val',
                 target_size=(image_size, image_size),
                 class_mode='categorical')

         my_new_model.fit_generator(
                 train_generator,
                 steps_per_epoch=3,
                 validation_data=validation_generator,
                 validation_steps=1)

         Found 72 images belonging to 2 classes.
         Found 20 images belonging to 2 classes.
         Epoch 1/1
         3/3 [==============================] - 7s 2s/step - loss: 0.7116 - acc: 0.5278 - val_loss: 0.5622 - val_acc: 0.6500

Out[11]: <tensorflow.python.keras.callbacks.History at 0x7f89b4358048>
```

The above creates training and validation data generators, applies preprocessing using ResNet50, and fits the model using the training data generator, specifying epoch and validation steps.

Taking a look at predictions from our validation set:

```
In [12]: from os import listdir
         from os.path import join, isdir

         image_dir = '/home/kaggle/rural_and_urban_photos/val'
         classes = listdir(image_dir)
         class_dirs = [join(image_dir, dirname) for dirname in classes if isdir(join(image_dir, dirname))]
         img_paths = [join(class_dir, filename) for class_dir in class_dirs for filename in listdir(class_dir)]

         # uncomment following if you want to be convinced the Python os code above is finding the right paths.
         # print(img_paths)

         test_data = read_and_prep_images(img_paths)
         preds = my_new_model.predict(test_data)
         class_indices = validation_generator.class_indices  # see documentation for ImageDataGenerator class referenced above
         print(class_indices)

         for i, img_path in enumerate(img_paths):
             display(Image(img_path))
             print(preds[i])

         {'rural': 0, 'urban': 1}
```

```
         [0.3725394 0.6274606]
```

Overall performance summary:

```
In [13]: # Evaluate the model on the validation generator
         evaluation = my_new_model.evaluate_generator(validation_generator, steps=len(validation_generator))

         # Extract the evaluation metrics
         loss = evaluation[0]
         accuracy = evaluation[1]

         # Print the evaluation metrics
         print("Validation Loss:", loss)
         print("Validation Accuracy:", accuracy)

         Validation Loss: 0.5621657967567444
         Validation Accuracy: 0.6499999761581421
```

The model's validation dataset results show moderate validation loss and accuracy, indicating moderate performance in making accurate predictions on unseen data.

Data Augmentation

Data augmentation by adding a slight shift in the pre-processing step and adding 2 extra epochs parameter:

```
In [14]: from tensorflow.python.keras.applications.resnet50 import preprocess_input
         from tensorflow.python.keras.preprocessing.image import ImageDataGenerator

         image_size = 224

         data_generator_with_aug = ImageDataGenerator(preprocessing_function=preprocess_input,
                                                      horizontal_flip=True,
                                                      width_shift_range = 0.2,
                                                      height_shift_range = 0.2)

         train_generator = data_generator_with_aug.flow_from_directory(
                 '/home/kaggle/rural_and_urban_photos/train',
                 target_size=(image_size, image_size),
                 batch_size=24,
                 class_mode='categorical')

         data_generator_no_aug = ImageDataGenerator(preprocessing_function=preprocess_input)
         validation_generator = data_generator_no_aug.flow_from_directory(
                 '/home/kaggle/rural_and_urban_photos/val',
                 target_size=(image_size, image_size),
                 class_mode='categorical')

         my_new_model.fit_generator(
                 train_generator,
                 steps_per_epoch=3,
                 epochs=2,
                 validation_data=validation_generator,
                 validation_steps=1)

         Found 72 images belonging to 2 classes.
         Found 20 images belonging to 2 classes.
         Epoch 1/2
         3/3 [==============================] - 4s 1s/step - loss: 0.3544 - acc: 0.9028 - val_loss: 0.4176 - val_acc: 0.7500
         Epoch 2/2
         3/3 [==============================] - 3s 1s/step - loss: 0.2514 - acc: 0.9306 - val_loss: 0.3648 - val_acc: 0.8500

Out[14]: <tensorflow.python.keras.callbacks.History at 0x7f89b4198f98>
```

Printing the predictions of the 20 validation images:

```
In [15]: preds = my_new_model.predict(test_data)
         class_indices = validation_generator.class_indices  # see documentation for ImageDataGenerator class referenced above
         print(class_indices)

         for i, img_path in enumerate(img_paths):
             display(Image(img_path))
             print(preds[i])

         {'rural': 0, 'urban': 1}
```

```
         [0.14575192 0.85424805]
```

Evaluation metrics for the augmented data:

```
In [16]: # Evaluate the model on the validation generator
         evaluation = my_new_model.evaluate_generator(validation_generator, steps=len(validation_generator))

         # Extract the evaluation metrics
         loss = evaluation[0]
         accuracy = evaluation[1]

         # Print the evaluation metrics
         print("Validation Loss:", loss)
         print("Validation Accuracy:", accuracy)

         Validation Loss: 0.36479324102401733
         Validation Accuracy: 0.8500000238418579
```

It is shown from the data augmentation the performance has improved as reflected from validation loss & accuracy.

Week 6: Reflection

This week's exercise involved developing and training deep learning models using TensorFlow and Keras libraries. The model was trained on a dataset of rural and urban photos, using techniques like preprocessing, augmentation, and image data generators. The model's performance was evaluated on a validation dataset, highlighting the importance of preprocessing, augmentation, and evaluation in building accurate models.

Week 7

CNN From Scratch

Load csv:

In [1]:
```
import pandas as pd

train_file = "/home/kaggle/digit-recognizer/train.csv"
raw_data = pd.read_csv(train_file)

raw_data.head()
```

Out[1]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pi
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	

5 rows × 785 columns

Data preparation:

In [2]:
```
from tensorflow.python import keras
import numpy as np

img_rows, img_cols = 28, 28
num_classes = 10

def data_prep(raw):
    #out_y = keras.utils.to_categorical(raw.label, num_classes)
    out_y = keras.utils.to_categorical(raw.label)

    num_images = raw.shape[0]
    x_as_array = raw.values[:,1:]
    x_shaped_array = x_as_array.reshape(num_images, img_rows, img_cols, 1)
    out_x = x_shaped_array / 255
    return out_x, out_y

x, y = data_prep(raw_data)

# print out parts of x and y to see what is in them...
print(y[:9,:])
np.set_printoptions(precision=1, linewidth=120)
print(x[3,:,:,0])
```

```
/home/anaconda3/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:523: FutureWarning: Passing (type, 1) or '1ty
pe' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/home/anaconda3/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:524: FutureWarning: Passing (type, 1) or '1ty
pe' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/home/anaconda3/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:525: FutureWarning: Passing (type, 1) or '1ty
```

The code utilizes TensorFlow and NumPy to convert image data into categorical format, reshape input images, normalize pixel values, and print out a portion for inspection.

Output:

```
[[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]]
[[0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.9 0.7 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.3 0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.1 1.  0.1 0.  0.  0.  0.  0.  0.  0.  0.  0.1 0.8 0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.9 0.6 0.  0.  0.  0.  0.  0.  0.  0.  0.1 1.  0.2 0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.6 0.8 0.  0.  0.  0.  0.  0.  0.  0.  0.1 1.  0.3 0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.5 1.  0.1 0.  0.  0.  0.  0.  0.  0.  0.1 0.9 0.3 0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.3 1.  0.1 0.  0.  0.  0.  0.  0.  0.  0.  0.8 0.3 0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.3 1.  0.1 0.  0.  0.  0.  0.  0.  0.  0.  0.8 0.6 0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.2 1.  0.1 0.  0.  0.  0.  0.  0.  0.  0.  0.8 0.9 0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.9 0.6 0.  0.  0.  0.  0.  0.  0.  0.  0.5 0.9 0.1 0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.3 1.  0.2 0.  0.  0.  0.  0.  0.  0.  0.4 1.  0.3 0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.9 0.7 0.  0.  0.  0.  0.  0.  0.  0.  0.1 1.  0.3 0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.9 0.7 0.  0.  0.  0.  0.  0.  0.  0.  0.8 0.7 0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.6 0.8 0.  0.  0.  0.1 0.3 0.5 0.8 0.8 1.  1.  0.5 0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.2 1.  0.5 0.4 0.7 1.  1.  0.7 0.3 0.2 0.5 1.  0.3 0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.1 0.8 0.9 0.9 0.6 0.4 0.2 0.  0.  0.  0.1 1.  0.3 0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.1 1.  0.3 0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.1 1.  0.3 0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.8 0.4 0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.7 0.6 0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.8 0.4 0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]]
```

Model building:

```
In [3]: from tensorflow.python.keras.models import Sequential
        from tensorflow.python.keras.layers import Dense, Flatten, Conv2D, Dropout

        model = Sequential()
        model.add(Conv2D(20, kernel_size=(3, 3),
                         activation='relu',
                         input_shape=(img_rows, img_cols, 1)))
        model.add(Conv2D(20, kernel_size=(3, 3), activation='relu'))
        model.add(Flatten())
        model.add(Dense(128, activation='relu'))
        model.add(Dense(num_classes, activation='softmax'))

        model.compile(loss=keras.losses.categorical_crossentropy,
                      optimizer='adam',
                      metrics=['accuracy'])
```

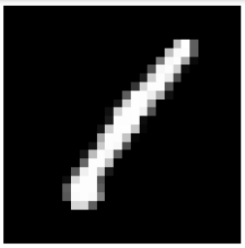
The code creates a CNN model using TensorFlow's Keras API, incorporating ReLU activation, flattening, and dense layers, and compiling it with various metrics.

Fitting:

```
In [4]: model.fit(x, y,
                  batch_size=128,
                  epochs=2,
                  validation_split = 0.2)

        Train on 33600 samples, validate on 8400 samples
        Epoch 1/2
        33600/33600 [==============================] - 43s 1ms/step - loss: 0.2479 - acc: 0.9261 - val_loss: 0.0909 - val_acc: 0.9736
        Epoch 2/2
        33600/33600 [==============================] - 40s 1ms/step - loss: 0.0635 - acc: 0.9808 - val_loss: 0.0631 - val_acc: 0.9801

Out[4]: <tensorflow.python.keras.callbacks.History at 0x7f0cc593fba8>
```

Evaluation:

```
In [5]: import matplotlib.pyplot as plt
        %matplotlib inline
        np.set_printoptions(precision=5, suppress=True)
        preds = model.predict(x [0:11,:,:,:])
        for i, pred in enumerate(preds):
            plt.imshow(255 * x [i,:,:,0], cmap='gray')
            plt.axis('off')
            plt.show()
            print(pred)
```



```
[0.00016 0.99287 0.00079 0.00014 0.00068 0.00007 0.00088 0.00008 0.00432 0.00001]
```

Each element in pred is the probability that the input image belongs to the respective class.

```
In [6]: accuracy = model.evaluate(x, y)[1]
        print("Accuracy:", accuracy)

        42000/42000 [==============================] - 15s 369us/step
        Accuracy: 0.9874761904761905
```

Based on the accuracy metric, the model perform fairly well.

Improved Deep Learning Models

```
In [7]: model = Sequential()
        model.add(Conv2D(30, kernel_size=(3, 3),
                         strides=2,
                         activation='relu',
                         input_shape=(img_rows, img_cols, 1)))
        model.add(Dropout(0.5))
        model.add(Conv2D(30, kernel_size=(3, 3), strides=2, activation='relu'))
        model.add(Dropout(0.5))
        model.add(Flatten())
        model.add(Dense(128, activation='relu'))
        model.add(Dense(num_classes, activation='softmax'))

        model.compile(loss=keras.losses.categorical_crossentropy,
                      optimizer='adam',
                      metrics=['accuracy'])
        model.fit(x, y,
                  batch_size=128,
                  epochs=2,
                  validation_split = 0.2)

        Train on 33600 samples, validate on 8400 samples
        Epoch 1/2
        33600/33600 [==============================] - 8s 249us/step - loss: 0.6459 - acc: 0.7992 - val_loss: 0.2407 - val_acc: 0.9312
        Epoch 2/2
        33600/33600 [==============================] - 6s 173us/step - loss: 0.2907 - acc: 0.9113 - val_loss: 0.1522 - val_acc: 0.9558

Out[7]: <tensorflow.python.keras.callbacks.History at 0x7f0ba44dd080>
```

The second model enhances performance and generalization by incorporating deeper convolutional layers, 2 stride, and dropout layers for better feature extraction and performance.

Evaluation:

```
In [9]: accuracy = model.evaluate(x, y)[1]
        print("Accuracy:", accuracy)

        42000/42000 [==============================] - 4s 106us/step
        Accuracy: 0.956047619047619
```

There is a drop in accuracy however, the model fitting time is significantly faster.

Before:

```
Train on 33600 samples, validate on 8400 samples
Epoch 1/2
33600/33600 [==============================] - 42s 1ms/step
Epoch 2/2
33600/33600 [==============================] - 54s 2ms/step
```

After:

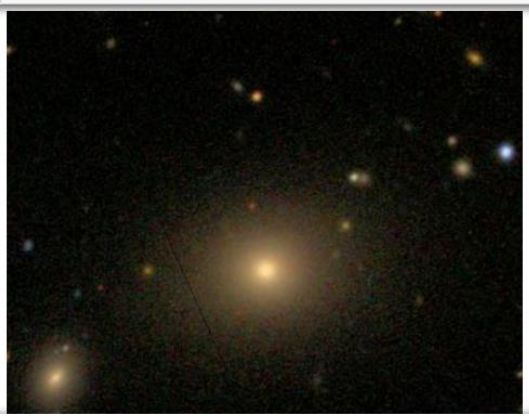
```
Train on 33600 samples, validate on 8400 samples
Epoch 1/2
33600/33600 [==============================] - 8s 249us/step
Epoch 2/2
33600/33600 [==============================] - 6s 173us/step
```

A Small Project on an Astronomical Image set

Aiming to map the universe, the Sloan Digital Sky Survey (SDSS) faces difficulties in distinguishing the form of galaxies. The Galaxy Zoo project uses crowdsourcing to classify SDSS photos.

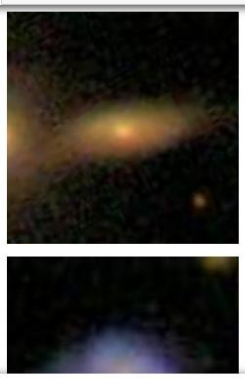
Loading/reading image data:

```
In [14]: from os import listdir
         from os.path import join, isdir
         from IPython.display import Image, display


         image_dir = '/home/kaggle/galaxy-zoo-the-galaxy-challenge/images_training_rev1'
         files = listdir(image_dir)
         img_paths = [join(image_dir, filename) for filename in files [0:9]]  # full paths to first 10 files

         for i, img_path in enumerate(img_paths):
             display(Image(img_path))
```



Viewing the first 10 files of the training data:

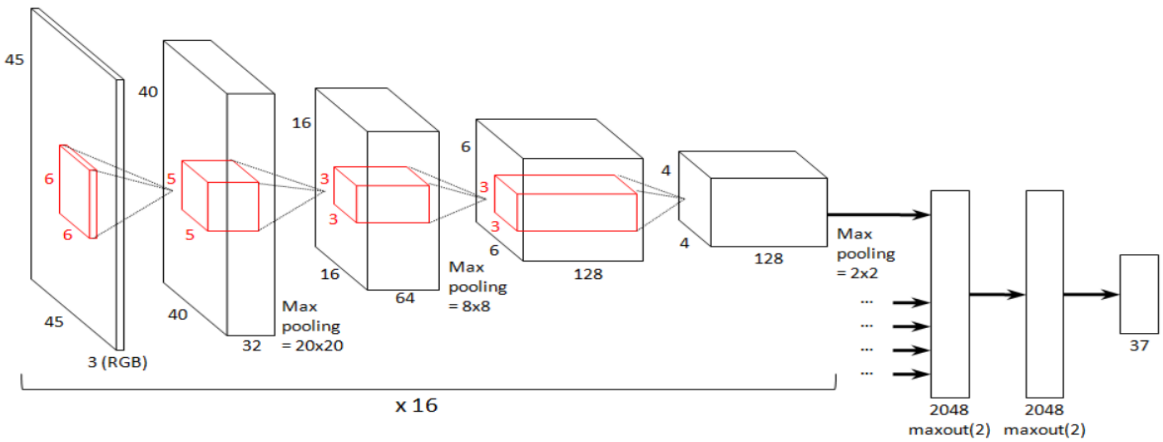
```
In [15]: from IPython.display import Image, display

         image_dir = '/home/kaggle/galaxy-zoo-the-galaxy-challenge/train/disk'
         files = listdir(image_dir)
         img_paths = [join(image_dir, filename) for filename in files [0:9]]  # full paths to first 10 files

         for img_path in img_paths:
             display(Image(img_path))
```



Building the model:

```
In [16]: from tensorflow.python import keras
         from tensorflow.python.keras.models import Sequential
         from tensorflow.python.keras.layers import Dense, Flatten, Conv2D, Dropout

         image_size = 212
         img_rows, img_cols = image_size, image_size
         num_classes = 3

         model = Sequential()

         model.add(Conv2D(30, kernel_size=(5, 5),
                          strides=2,
                          activation='relu',
                          input_shape=(img_rows, img_cols, 3)))
         model.add(Conv2D(60, kernel_size=(5, 5), strides=2, activation='relu'))
         model.add(Conv2D(120, kernel_size=(5, 5), activation='relu'))
         model.add(Flatten())
         model.add(Dense(256, activation='relu'))
         model.add(Dense(num_classes, activation='softmax'))

         model.compile(loss=keras.losses.categorical_crossentropy,
                       optimizer='adam',
                       metrics=['accuracy'])
```

The structure is inspired by this architecture:

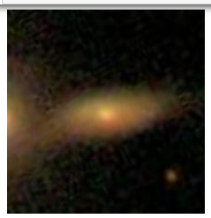


Train and fit the model:

```
In [4]: from tensorflow.python.keras.preprocessing.image import ImageDataGenerator

        def my_preprocess(x) :
            return x / 255

        data_generator = ImageDataGenerator(preprocessing_function=my_preprocess)

        train_generator = data_generator.flow_from_directory(
                '/home/kaggle/galaxy-zoo-the-galaxy-challenge/train',
                target_size=(image_size, image_size),
                batch_size=24,
                class_mode='categorical')

        validation_generator = data_generator.flow_from_directory(
                '/home/kaggle/galaxy-zoo-the-galaxy-challenge/test',
                target_size=(image_size, image_size),
                batch_size=32,
                class_mode='categorical')

        model.fit_generator(
                train_generator,
                steps_per_epoch=100,
                validation_data=validation_generator,
                validation_steps=30)

        Found 43122 images belonging to 3 classes.
        Found 18456 images belonging to 3 classes.
        Epoch 1/1
        100/100 [==============================] - 141s 1s/step - loss: 0.6666 - acc: 0.6518 - val_loss: 0.6133 - val_acc: 0.7031

Out[4]: <tensorflow.python.keras.callbacks.History at 0x7fdf377a2358>
```

I have adjusted the epoch and validation steps as the original would take more than 1 hour to run.

```
In [5]: import pandas as pd

        import numpy as np
        from tensorflow.python.keras.preprocessing.image import load_img, img_to_array

        def read_and_prep_images(img_paths, img_height=image_size, img_width=image_size):
            imgs = [load_img(img_path, target_size=(img_height, img_width)) for img_path in img_paths]
            img_array = np.array([img_to_array(img) for img in imgs])
            output = my_preprocess(img_array)
            return(output)

        test_data = read_and_prep_images(img_paths)
        preds = model.predict(test_data)

        class_indices = validation_generator.class_indices  # see documentation for ImageDataGenerator class referenced above

        for i, img_path in enumerate(img_paths):
            display(Image(img_path))
            #print(true_probs [i])
            print(preds [i] [class_indices ['smooth']], preds [i] [class_indices ['disk']], preds [i] [class_indices ['artefact']])
```



```
0.3808231 0.61740404 0.0017727779
```

The probability refers the likelihood of the image will be for each class.

```
In [6]: # Evaluate the model on the validation data using evaluate_generator
        evaluation = model.evaluate_generator(validation_generator, steps=30)

        # Extract accuracy from the evaluation result
        accuracy = evaluation[1]

        # Print the accuracy
        print("Validation Accuracy:", accuracy)

        Validation Accuracy: 0.703125
```

Exercise

A 240 layer and max pooling layer are added for experimentation:

```
In [3]: from tensorflow.python import keras
        from tensorflow.python.keras.models import Sequential
        from tensorflow.python.keras.layers import Dense, Flatten, Conv2D, Dropout, BatchNormalization, MaxPooling2D
        from tensorflow.python.keras.preprocessing.image import ImageDataGenerator

        image_size = 212
        img_rows, img_cols = image_size, image_size
        num_classes = 3

        model = Sequential()

        model.add(Conv2D(30, kernel_size=(5, 5),
                         strides=2,
                         activation='relu',
                         input_shape=(img_rows, img_cols, 3)))
        model.add(Conv2D(60, kernel_size=(5, 5), strides=2, activation='relu'))
        model.add(Conv2D(120, kernel_size=(5, 5), activation='relu'))
        model.add(Conv2D(240, kernel_size=(5, 5), activation='relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))  # Added MaxPooling2D Layer
        model.add(Flatten())
        model.add(Dense(256, activation='relu'))
        model.add(Dense(num_classes, activation='softmax'))


        model.compile(loss=keras.losses.categorical_crossentropy,
                      optimizer='adam',
                      metrics=['accuracy'])
```

Output:

```
Found 43122 images belonging to 3 classes.
Found 18456 images belonging to 3 classes.
Epoch 1/1
100/100 [==============================] - 196s 2s/step - loss: 0.7049 - acc: 0.5800 - val_loss: 0.5905 - val_acc: 0.7125

Out[3]: <tensorflow.python.keras.callbacks.History at 0x7f16e0ab0a90>
```

```
In [4]: # Evaluate the model on the validation data using evaluate_generator
        evaluation = model.evaluate_generator(validation_generator, steps=30)

        # Extract accuracy from the evaluation result
        accuracy = evaluation[1]

        # Print the accuracy
        print("Validation Accuracy:", accuracy)

        Validation Accuracy: 0.7125
```

This results in a one per cent improvement in the accuracy of the model.

Week 7: Reflection

This lab session focused on deep learning model development of CNNs for image classification tasks. It covered CNN architecture design, strategies for improving performance, hyperparameter tuning and data generator utilisation. The session highlighted the iterative nature of model development, emphasising experimentation, evaluation, and refinement for optimal performance.

Week 8

Spark Structured Streaming

Starting session and reading CSV file:

```
In [1]: from pyspark.sql import SparkSession

        spark = SparkSession.builder.appName("StreamingExamples").getOrCreate()
```

```
In [2]: spark.conf.set("spark.sql.shuffle.partitions", "5")
```

```
In [3]: staticDataFrame = (spark.read.format("csv").option("header", "true")
        .option("inferSchema", "true")
        .load("file:///home/spark/Spark-The-Definitive-Guide-master/data/retail-data/by-day/*.csv"))
```

Printing the schema of a static DataFrame in Apache Spark:

```
In [4]: staticDataFrame.printSchema()

root
 |-- InvoiceNo: string (nullable = true)
 |-- StockCode: string (nullable = true)
 |-- Description: string (nullable = true)
 |-- Quantity: integer (nullable = true)
 |-- InvoiceDate: timestamp (nullable = true)
 |-- UnitPrice: double (nullable = true)
 |-- CustomerID: double (nullable = true)
 |-- Country: string (nullable = true)
```

Assigning the schema of a static DataFrame to a variable named staticSchema and show top 20 rows:

```
In [5]: staticSchema = staticDataFrame.schema
```

```
In [6]: print(staticDataFrame.count())

        staticDataFrame.show()

541909
+---------+---------+--------------------+--------+-------------------+---------+----------+--------------+
|InvoiceNo|StockCode|         Description|Quantity|        InvoiceDate|UnitPrice|CustomerID|       Country|
+---------+---------+--------------------+--------+-------------------+---------+----------+--------------+
|   580538|    23084|  RABBIT NIGHT LIGHT|      48|2011-12-05 08:38:00|     1.79|   14075.0|United Kingdom|
|   580538|    23077| DOUGHNUT LIP GLOSS |      20|2011-12-05 08:38:00|     1.25|   14075.0|United Kingdom|
|   580538|    22906|12 MESSAGE CARDS ...|      24|2011-12-05 08:38:00|     1.65|   14075.0|United Kingdom|
|   580538|    21914|BLUE HARMONICA IN...|      24|2011-12-05 08:38:00|     1.25|   14075.0|United Kingdom|
|   580538|    22467|   GUMBALL COAT RACK|       6|2011-12-05 08:38:00|     2.55|   14075.0|United Kingdom|
|   580538|    21544|SKULLS  WATER TRA...|      48|2011-12-05 08:38:00|     0.85|   14075.0|United Kingdom|
|   580538|    23126|FELTCRAFT GIRL AM...|       8|2011-12-05 08:38:00|     4.95|   14075.0|United Kingdom|
|   580538|    21833|CAMOUFLAGE LED TORCH|      24|2011-12-05 08:38:00|     1.69|   14075.0|United Kingdom|
|   580539|    21479|WHITE SKULL HOT W...|       4|2011-12-05 08:39:00|     4.25|   18180.0|United Kingdom|
|   580539|   84030E|ENGLISH ROSE HOT ...|       4|2011-12-05 08:39:00|     4.25|   18180.0|United Kingdom|
|   580539|    23355|HOT WATER BOTTLE ...|       4|2011-12-05 08:39:00|     4.95|   18180.0|United Kingdom|
|   580539|    22111|SCOTTIE DOG HOT W...|       3|2011-12-05 08:39:00|     4.95|   18180.0|United Kingdom|
|   580539|    21115|ROSE CARAVAN DOOR...|       8|2011-12-05 08:39:00|     1.95|   18180.0|United Kingdom|
|   580539|    21411|GINGHAM HEART  DO...|       8|2011-12-05 08:39:00|     1.95|   18180.0|United Kingdom|
|   580539|    23235|STORAGE TIN VINTA...|      12|2011-12-05 08:39:00|     1.25|   18180.0|United Kingdom|
|   580539|    23239|SET OF 4 KNICK KN...|       6|2011-12-05 08:39:00|     1.65|   18180.0|United Kingdom|
|   580539|    22197|      POPCORN HOLDER|      36|2011-12-05 08:39:00|     0.85|   18180.0|United Kingdom|
|   580539|    22693|GROW A FLYTRAP OR...|      24|2011-12-05 08:39:00|     1.25|   18180.0|United Kingdom|
|   580539|    22372|AIRLINE BAG VINTA...|       4|2011-12-05 08:39:00|     4.25|   18180.0|United Kingdom|
|   580539|    22375|AIRLINE BAG VINTA...|       4|2011-12-05 08:39:00|     4.25|   18180.0|United Kingdom|
+---------+---------+--------------------+--------+-------------------+---------+----------+--------------+
only showing top 20 rows
```

Next, the code below calculates the total cost for each transaction in a DataFrame and displays the results.

```
In [7]: staticTotals=staticDataFrame.selectExpr("CustomerId", "(UnitPrice * Quantity) as total_cost", "InvoiceDate")

        staticTotals.show()
```

```
+----------+------------------+-------------------+
|CustomerId|        total_cost|        InvoiceDate|
+----------+------------------+-------------------+
|   14075.0|             85.92|2011-12-05 08:38:00|
|   14075.0|              25.0|2011-12-05 08:38:00|
|   14075.0|39.599999999999994|2011-12-05 08:38:00|
|   14075.0|              30.0|2011-12-05 08:38:00|
|   14075.0|15.299999999999999|2011-12-05 08:38:00|
|   14075.0|              40.8|2011-12-05 08:38:00|
|   14075.0|              39.6|2011-12-05 08:38:00|
|   14075.0|             40.56|2011-12-05 08:38:00|
|   18180.0|              17.0|2011-12-05 08:39:00|
|   18180.0|              17.0|2011-12-05 08:39:00|
|   18180.0|              19.8|2011-12-05 08:39:00|
|   18180.0|14.850000000000001|2011-12-05 08:39:00|
|   18180.0|              15.6|2011-12-05 08:39:00|
|   18180.0|              15.6|2011-12-05 08:39:00|
|   18180.0|              15.0|2011-12-05 08:39:00|
|   18180.0| 9.899999999999999|2011-12-05 08:39:00|
|   18180.0|30.599999999999998|2011-12-05 08:39:00|
|   18180.0|              30.0|2011-12-05 08:39:00|
|   18180.0|              17.0|2011-12-05 08:39:00|
|   18180.0|              17.0|2011-12-05 08:39:00|
+----------+------------------+-------------------+
only showing top 20 rows
```

The code below groups the DataFrame staticTotals by CustomerId, calculates the sum of total_cost for each group, and displays the results:

```
In [8]: from pyspark.sql.functions import col
        staticTotals.groupBy(col("CustomerId")).sum("total_cost").show()
```

```
+----------+------------------+
|CustomerId|   sum(total_cost)|
+----------+------------------+
|   18180.0|           1843.75|
|   12782.0|           2273.32|
|   17402.0|           2633.85|
|   16642.0|            679.83|
|   16811.0|            547.07|
|   15053.0|            697.79|
|   12913.0|           2483.63|
|   12628.0|            769.61|
|   14401.0| 4039.980000000001|
|   16851.0|342.45000000000005|
|   17511.0| 88125.37999999999|
|   18044.0|           2085.65|
|   18198.0| 5083.060000000001|
|   13001.0|           9797.92|
|   16379.0|2131.9000000000005|
|   13230.0|2763.4099999999994|
|   17757.0| 5585.489999999999|
|   17404.0|30300.820000000007|
|   16705.0|13946.130000000001|
|   12957.0|           3979.04|
+----------+------------------+
only showing top 20 rows
```

```
In [9]: from pyspark.sql.functions import window
        staticTotals.groupBy(window(col("InvoiceDate"), "1 day")).sum("total_cost").show(truncate=False)
```

```
+----------------------------------------+------------------+
|window                                  |sum(total_cost)   |
+----------------------------------------+------------------+
|[2010-12-06 00:00:00, 2010-12-07 00:00:00]|53860.180000000015|
|[2011-11-20 00:00:00, 2011-11-21 00:00:00]|34902.01000000018 |
|[2011-11-10 00:00:00, 2011-11-11 00:00:00]|68956.24000000003 |
|[2011-09-21 01:00:00, 2011-09-22 01:00:00]|42944.07000000001 |
|[2010-12-10 00:00:00, 2010-12-11 00:00:00]|57404.91000000017 |
|[2011-11-27 00:00:00, 2011-11-28 00:00:00]|20571.500000000084|
|[2011-10-27 01:00:00, 2011-10-28 01:00:00]|47480.1499999999  |
|[2011-09-13 01:00:00, 2011-09-14 01:00:00]|54828.45000000007 |
|[2011-10-12 01:00:00, 2011-10-13 01:00:00]|29302.850000000173|
|[2010-12-02 00:00:00, 2010-12-03 00:00:00]|46207.27999999991 |
|[2011-03-17 00:00:00, 2011-03-18 00:00:00]|38804.25000000005 |
|[2011-06-20 01:00:00, 2011-06-21 01:00:00]|33493.4000000001  |
|[2011-03-29 01:00:00, 2011-03-30 01:00:00]|70531.46999999978 |
|[2011-03-07 00:00:00, 2011-03-08 00:00:00]|30525.58000000021 |
|[2011-07-07 01:00:00, 2011-07-08 01:00:00]|31357.72000000012 |
|[2011-07-25 01:00:00, 2011-07-26 01:00:00]|26687.650000000176|
|[2011-05-17 01:00:00, 2011-05-18 01:00:00]|53603.82999999985 |
|[2011-04-19 01:00:00, 2011-04-20 01:00:00]|23837.650000000183|
|[2011-11-01 00:00:00, 2011-11-02 00:00:00]|28741.55000000017 |
|[2011-07-06 01:00:00, 2011-07-07 01:00:00]|26279.580000000133|
+----------------------------------------+------------------+
only showing top 20 rows
```

This code groups the DataFrame staticTotals by a window of 1-day intervals based on InvoiceDate, calculates the sum of total_cost for each window, and displays the results without truncation.

```
In [10]: staticTotals.groupBy(col("CustomerId"), window(col("InvoiceDate"), "1 day")).sum("total_cost").show(truncate=False)
```

```
+----------+----------------------------------------+------------------+
|CustomerId|window                                  |sum(total_cost)   |
+----------+----------------------------------------+------------------+
|14075.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|316.78000000000003|
|18180.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|310.73            |
|15358.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|830.0600000000003 |
|15392.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|304.40999999999997|
|15290.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|263.02000000000004|
|16811.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|232.3             |
|12748.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|363.7899999999999 |
|16500.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|52.74000000000001 |
|16873.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|1854.8300000000002|
|14060.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|297.47999999999996|
|14649.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|513.9899999999998 |
|16904.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|349.0200000000001 |
|17857.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|2979.6            |
|14083.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|446.5700000000001 |
|14777.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|-2.95             |
|16684.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|5401.979999999999 |
|13685.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|5.48              |
|15159.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|1730.84           |
|18015.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|120.03            |
|13305.0   |[2011-12-05 00:00:00, 2011-12-06 00:00:00]|213.15999999999997|
+----------+----------------------------------------+------------------+
only showing top 20 rows
```

By comparing the previous code, The first one groups DataFrame staticTotals by InvoiceDate window intervals, calculates total_cost sums, and displays results without truncation, while this one group DataFrame staticTotals by CustomerId and window intervals.

In the next part, this code reads streaming data from CSV files with dynamic schema enforcement and a maximum of one file per trigger.

```
In [11]: streamingDataFrame = (spark.readStream.schema(staticSchema).option("maxFilesPerTrigger", 1)
         .format("csv").option("header", "true")
         .load("file:///home/spark/Spark-The-Definitive-Guide-master/data/retail-data/by-day/*.csv"))
```

Then the code below computes the total purchase cost per customer per day, streams the result to an in-memory table named "customer_purchases" with complete output mode, and finally queries the table to display the top 5 customers by total purchase cost and the streaming process is ended:

```
In [12]: purchaseByCustomerPerHour = (streamingDataFrame.selectExpr("CustomerId", "(UnitPrice * Quantity) as total_cost", "InvoiceDate")
         .groupBy(col("CustomerId"), window(col("InvoiceDate"), "1 day")).sum("total_cost"))

In [13]: purchaseQuery =  (purchaseByCustomerPerHour.writeStream.format("memory")
         .queryName("customer_purchases").outputMode("complete").start())

In [14]: spark.sql("SELECT * FROM customer_purchases ORDER BY `sum(total_cost)` DESC").show(5)

+----------+--------------------+------------------+
|CustomerId|              window|   sum(total_cost)|
+----------+--------------------+------------------+
|   17450.0|[2011-09-20 01:00...|          71601.44|
|      null|[2011-03-29 01:00...| 33521.39999999998|
|      null|[2011-11-29 00:00...|23744.250000000055|
|   12931.0|[2011-08-04 01:00...|19045.480000000003|
|   14646.0|[2011-03-29 01:00...|           18247.5|
+----------+--------------------+------------------+
only showing top 5 rows

In [15]: purchaseQuery.stop()
```

A JSON file is read then displayed the schema and first 20 rows:

```
In [16]: static = spark.read.json("file:///home/spark/Spark-The-Definitive-Guide-master/data/activity-data")

In [17]: static.printSchema()

root
 |-- Arrival_Time: long (nullable = true)
 |-- Creation_Time: long (nullable = true)
 |-- Device: string (nullable = true)
 |-- Index: long (nullable = true)
 |-- Model: string (nullable = true)
 |-- User: string (nullable = true)
 |-- gt: string (nullable = true)
 |-- x: double (nullable = true)
 |-- y: double (nullable = true)
 |-- z: double (nullable = true)

In [18]: static.show()

+-------------+-------------------+--------+-----+------+----+-----+-------------+------------+------------+
| Arrival_Time|      Creation_Time|  Device|Index| Model|User|   gt|            x|           y|           z|
+-------------+-------------------+--------+-----+------+----+-----+-------------+------------+------------+
|1424686735090|1424686733090638193|nexus4_1|   18|nexus4|   g|stand|  3.356934E-4|-5.645752E-4|-0.018814087|
|1424686735292|1424688581345918092|nexus4_2|   66|nexus4|   g|stand| -0.005722046| 0.029083252| 0.005569458|
|1424686735500|1424686733498505625|nexus4_1|   99|nexus4|   g|stand|    0.0078125|-0.017654419| 0.010025024|
|1424686735691|1424688581745026978|nexus4_2|  145|nexus4|   g|stand| -3.814697E-4|   0.0184021|-0.013656616|
|1424686735890|1424688581945252808|nexus4_2|  185|nexus4|   g|stand| -3.814697E-4|-0.031799316| -0.00831604|
|1424686736094|1424686734097840342|nexus4_1|  218|nexus4|   g|stand| -7.324219E-4|-0.013381958|  0.01109314|
|1424686736294|1424688582347932252|nexus4_2|  265|nexus4|   g|stand| -0.005722046| 0.015197754| 0.022659302|
|1424686736495|1424688582549592408|nexus4_2|  305|nexus4|   g|stand| -3.814697E-4|0.0087890625|0.0034332275|
|1424686736697|1424688582750703248|nexus4_2|  345|nexus4|   g|stand|  0.002822876|-0.008300781|-0.015792847|
|1424686736898|1424688582952241334|nexus4_2|  385|nexus4|   g|stand|  6.866455E-4|-0.008300781| 0.004501343|
|1424686737100|1424686735109928643|nexus4_1|  418|nexus4|   g|stand|  0.003540039|-0.010177612|-0.026290894|
|1424686737300|1424688583355164918|nexus4_2|  465|nexus4|   g|stand|  0.002822876|0.0045166016|-0.014724731|
|1424686737505|1424686735512935017|nexus4_1|  498|nexus4|   g|stand| 0.0024719238|-0.010177612|-0.017745972|
|1424686737707|1424686735709254597|nexus4_1|  537|nexus4|   g|stand|-0.0028686523|-0.003768921| 0.020706177|
|1424686737908|1424686735915675495|nexus4_1|  578|nexus4|   g|stand|-0.0028686523| 0.026138306| 0.007888794|
|1424686738109|1424688584160372793|nexus4_2|  625|nexus4|   g|stand| -3.814697E-4| 2.441406E-4| 0.033340454|
|1424686738326|1424688584381747305|nexus4_2|  661|nexus4|   g|stand| 0.0017547607| 0.019470215|-0.011520386|
|1424686738529|1424686736534938191|nexus4_1|  701|nexus4|   g|stand| 0.0024719238|-0.033676147|0.0068206787|
|1424686738744|1424688584799723655|nexus4_2|  744|nexus4|   g|stand| -3.814697E-4|-0.002960205|-0.027542114|
|1424686738935|1424686736943477009|nexus4_1|  782|nexus4|   g|stand| -0.009277344|-0.009109497|  -0.0690155|
+-------------+-------------------+--------+-----+------+----+-----+-------------+------------+------------+
only showing top 20 rows
```

Lastly, the code below reads streaming JSON data with dynamic schema enforcement and a maximum of one file per trigger, calculates activity counts grouped by "gt" field, streams the result to an in-memory table named "activity_counts" with complete output mode, then iteratively queries and displays the content of the "activity_counts" table five times with one-second delay between each iteration followed by ending the session.

```
In [19]: streaming = (spark.readStream.schema(static.schema).option("maxFilesPerTrigger", 1)
         .json("file:///home/spark/Spark-The-Definitive-Guide-master/data/activity-data"))
```

```
In [20]: activityCounts = streaming.groupBy("gt").count()
```

```
In [21]: activityQuery=activityCounts.writeStream.queryName("activity_counts").format("memory").outputMode("complete").start()
```

```
In [22]: from time import sleep

         for x in range(5) :
           spark.sql("SELECT * FROM activity_counts").show()
           sleep(1)
```

Output:

```
+----------+------+
|        gt| count|
+----------+------+
|       sit|320033|
|     stand|296006|
|stairsdown|243446|
|      walk|344627|
|  stairsup|271906|
|      null|271607|
|      bike|280702|
+----------+------+

+----------+------+
|        gt| count|
+----------+------+
|       sit|356959|
|     stand|330158|
|stairsdown|271532|
|      walk|384394|
|  stairsup|303282|
|      null|302947|
|      bike|313093|
+----------+------+

+----------+------+
|        gt| count|
+----------+------+
|       sit|393889|
|     stand|364315|
|stairsdown|299626|
|      walk|424156|
|  stairsup|334650|
|      null|334287|
|      bike|345480|
+----------+------+

+----------+------+
|        gt| count|
+----------+------+
|       sit|418508|
|     stand|387085|
|stairsdown|318350|
|      walk|450665|
|  stairsup|355568|
|      null|355180|
|      bike|367072|
+----------+------+

+----------+------+
|        gt| count|
+----------+------+
|       sit|455436|
|     stand|421239|
|stairsdown|346445|
|      walk|490428|
|  stairsup|386938|
|      null|386520|
|      bike|399459|
+----------+------+
```

```
In [23]: activityQuery.stop()
```

Processing Twitter Data

A Twitter data in the form of JSON is read then displayed the schema:

```
In [24]: static = spark.read.json("file:///home/spark/twitter-stream")

In [25]: static.schema

Out[25]: StructType(List(StructField(_corrupt_record,StringType,true),StructField(contributors,StringType,true),StructField(coordinate
s,StringType,true),StructField(created_at,StringType,true),StructField(display_text_range,ArrayType(LongType,true),true),Stru
ctField(entities,StructType(List(StructField(hashtags,ArrayType(StructType(List(StructField(indices,ArrayType(LongType,true),
true),StructField(text,StringType,true))),true),true),StructField(media,ArrayType(StructType(List(StructField(display_url,Str
ingType,true),StructField(expanded_url,StringType,true),StructField(id,LongType,true),StructField(id_str,StringType,true),Str
uctField(indices,ArrayType(LongType,true),true),StructField(media_url,StringType,true),StructField(media_url_https,StringTyp
e,true),StructField(sizes,StructType(List(StructField(large,StructType(List(StructField(h,LongType,true),StructField(resize,S
tringType,true),StructField(w,LongType,true))),true),StructField(medium,StructType(List(StructField(h,LongType,true),StructFi
eld(resize,StringType,true),StructField(w,LongType,true))),true),StructField(small,StructType(List(StructField(h,LongType,tru
e),StructField(resize,StringType,true),StructField(w,LongType,true))),true),StructField(thumb,StructType(List(StructField(h,L
ongType,true),StructField(resize,StringType,true),StructField(w,LongType,true))),true))),true),StructField(source_status_id,L
ongType,true),StructField(source_status_id_str,StringType,true),StructField(source_user_id,LongType,true),StructField(source_
user_id_str,StringType,true),StructField(type,StringType,true),StructField(url,StringType,true))),true),true),StructField(sym
bols,ArrayType(StructType(List(StructField(indices,ArrayType(LongType,true),true),StructField(text,StringType,true))),true),t
rue),StructField(urls,ArrayType(StructType(List(StructField(display_url,StringType,true),StructField(expanded_url,StringType,
true),StructField(indices,ArrayType(LongType,true),true),StructField(url,StringType,true))),true),true),StructField(user_ment
ions,ArrayType(StructType(List(StructField(id,LongType,true),StructField(id_str,StringType,true),StructField(indices,ArrayTyp
e(LongType,true),true),StructField(name,StringType,true),StructField(screen_name,StringType,true))),true),true))),true),Struc
tField(extended_entities,StructType(List(StructField(media,ArrayType(StructType(List(StructField(display_url,StringType,tru
```

Then the code below reads streaming JSON data from a directory named "twitter-stream", selects the "text" field from the streaming DataFrame, streams the selected text fields to an in-memory table named "text_fields" with append output mode, starts the streaming query, and finally queries and displays the content of the "text_fields" table with no truncation, showing up to 1000 rows:

```
In [26]: streaming = spark.readStream.schema(static.schema).json("file:///home/spark/twitter-stream")

In [27]: textFields = streaming.select("text")

In [28]: textQuery=textFields.writeStream.queryName("text_fields").format("memory").outputMode("append").start()

In [29]: spark.sql("SELECT * FROM text_fields").show(1000,truncate=False)
+-----------------------
```

```
In [30]: user = streaming.select(streaming.user.getItem('name').alias('name'))

In [31]: userQuery=user.writeStream.queryName("user_query").format("memory").start()

In [32]: spark.sql("SELECT * FROM user_query").show(1000)
```

```
+-----------------------------------+
|                               name|
+-----------------------------------+
|                              HELLO|
|                        BOT Kitty 🐈|
|                Aaron ''Midlife C...|
|                Tweet Whisperer D...|
|                       Gaurav Sharma|
|                               Guru|
|                Healthcare meets ...|
|                             Sadeesh|
|                             Goutham|
|                     OpenSource Orgs|
|                HOMEWORK ° ASSIGN...|
|                              Halden|
|                Aaron ''Midlife C...|
|                              HELLO|
|                Free Online Cours...|
|                           we movin'|
```

Lastly, the below code reads streaming JSON data, selects the "location" field from the nested "user" field, calculates the count of occurrences for each location, streams the result to an in-memory table named "loc_counts" with complete output mode, starts the streaming query, queries and displays the content of the "loc_counts" table and finally stops the streaming query.

```
In [33]: locations = streaming.select(streaming.user.getItem('location').alias('location')).groupBy('location').count()

         locQuery=locations.writeStream.queryName("loc_counts").format("memory").outputMode("complete").start()

In [34]: spark.sql("SELECT * FROM loc_counts").show(1000)
```

```
+----------------------------+-----+
|                    location|count|
+----------------------------+-----+
|          Amazon Web Services|    2|
|         Barranquilla, Col...|    1|
|                  Nice, France|    1|
|         San José, Costa Rica|    1|
|             Bengalore, India|    1|
|                      Soweto|    1|
|                   写真ブログ|    1|
|                     Cancún|    2|
|          1    |... المملكة العربية|
|         Dhahran, Saudi Ar...|    1|
|                 Chicago, IL|    1|
|            Waseda University|    1|
|         Right where you t...|    1|
|                   Neural Net|    1|
|                   CCS. VZLA |    1|
|                     Malaysia|    2|
```

```
In [35]: locQuery.stop()
```

Week 8: Reflection

The lab session on stream processing uses pyspark library to connect to Apache Spark and show various aspects of handling streaming data. It covers reading data from various sources, schema inference and enforcement, data transformation and aggregation operations, streaming query configuration, interactive analysis with Spark SQL, handling continuous data streams, and query termination. The given examples demonstrate how to read streaming data from CSV and JSON files, configure parameters like output mode, query name, and format, and perform interactive analysis with Spark SQL. They also demonstrate how to handle continuous data streams and terminates streaming queries to prevent resource leakage. This hands-on experience helps me gain a deeper understanding of stream processing fundamentals and develop proficiency in using Apache Spark for real-time data analytics.

Week 9

Building a Dense Artificial Neural Network from Scratch

Data is imported using Google Colab:

```
from keras.datasets import imdb
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=10000)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 [==============================] - 0s 0us/step
```

Then the data and labels are inspected:

```
[2] train_data[0]
```

```
[1,
 14,
 22,
 16,
 43,
 530,
 973,
```

```
train_labels[0]
```

```
1
```

(0 are negative reviews, 1 is positive reviews)

A function is then created that converts a list of sequences into a matrix where each row represents a sequence, and each column indicates whether a specific word or token from a vocabulary is present in that sequence.

```
import numpy as np

def vectorize_sequences(sequences, dimension=10000):
    #Creates an all-zero matrix
    results = np.zeros((len(sequences), dimension))
    #Set approperiate indices of the tensor result to 1s
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.
    return results

#Turn Python lists into NumPy tensors
x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)

# print out parts of x_train to see what is in it...
print(x_train[:9])
```

```
[[0. 1. 1. ... 0. 0. 0.]
 [0. 1. 1. ... 0. 0. 0.]
 [0. 1. 1. ... 0. 0. 0.]
 ...
 [0. 1. 1. ... 0. 0. 0.]
 [0. 1. 1. ... 0. 0. 0.]
 [0. 1. 1. ... 0. 0. 0.]]
```

Trained and test data is converted to a float:

```
[8] y_train = np.asarray(train_labels).astype('float32')
    y_test = np.asarray(test_labels).astype('float32')

    # print out parts of y_train to see what is in it...
    print(y_train[:9])

    [1. 0. 0. 1. 0. 0. 1. 0. 1.]
```

A 3 layer model is then built:

```
[9] from keras import models
    from keras import layers

    model = models.Sequential()
    model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
    model.add(layers.Dense(16, activation='relu'))
    model.add(layers.Dense(1, activation='sigmoid'))

    model.compile(optimizer='rmsprop',
                  loss='binary_crossentropy',
                 metrics=['accuracy'])
```

This code above shows two hidden layers with 16 neurons each and ReLU activation, and an output layer with one neuron and a sigmoid activation function. It then configures the model for training, specifying the optimizer, loss function, and evaluation metric.

The model is then trained and test:

```
44s
#Setting aside a validation set
x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]

#Training the model
model.fit(partial_x_train,
          partial_y_train,
          epochs=20,
          batch_size=512,
          validation_data=(x_val, y_val))

Epoch 1/20
30/30 [==============================] - 3s 82ms/step - loss: 0.5418 - accuracy: 0.7724 - val_loss: 0.4084 - val_accuracy: 0.8609
Epoch 2/20
30/30 [==============================] - 1s 36ms/step - loss: 0.3403 - accuracy: 0.8867 - val_loss: 0.3489 - val_accuracy: 0.8603
Epoch 3/20
30/30 [==============================] - 1s 35ms/step - loss: 0.2525 - accuracy: 0.9173 - val_loss: 0.2965 - val_accuracy: 0.8812
Epoch 4/20
30/30 [==============================] - 2s 57ms/step - loss: 0.2037 - accuracy: 0.9324 - val_loss: 0.2762 - val_accuracy: 0.8883
Epoch 5/20
30/30 [==============================] - 1s 48ms/step - loss: 0.1712 - accuracy: 0.9422 - val_loss: 0.2824 - val_accuracy: 0.8857
Epoch 6/20
30/30 [==============================] - 1s 35ms/step - loss: 0.1454 - accuracy: 0.9509 - val_loss: 0.2840 - val_accuracy: 0.8846
Epoch 7/20
30/30 [==============================] - 1s 40ms/step - loss: 0.1258 - accuracy: 0.9607 - val_loss: 0.2944 - val_accuracy: 0.8825
Epoch 8/20
30/30 [==============================] - 1s 49ms/step - loss: 0.1082 - accuracy: 0.9671 - val_loss: 0.3051 - val_accuracy: 0.8830
```

```
[11] #Evaluation
     results = model.evaluate(x_test, y_test)

     782/782 [==============================] - 2s 3ms/step - loss: 0.6135 - accuracy: 0.8582
```

```
[12] from prettytable import PrettyTable
     #Prediction
     preds = model.predict(x_test [0:10])

     t = PrettyTable(['Real', 'Predicted'])
     for i, pred in enumerate(preds):
         t.add_row([y_test [i], round(preds [i,0],2)])
     print(t)

     1/1 [==============================] - 0s 164ms/step
     +------+-----------+
     | Real | Predicted |
     +------+-----------+
     | 0.0  |    0.02   |
     | 1.0  |    1.0    |
     | 1.0  |    0.74   |
     | 0.0  |    0.93   |
     | 1.0  |    0.99   |
     | 1.0  |    1.0    |
     | 1.0  |    1.0    |
     | 0.0  |    0.0    |
     | 0.0  |    0.98   |
     | 1.0  |    1.0    |
     +------+-----------+
```

The model resulted in an 85.8% accuracy, indicating a good performance in predicting the test data.

Classification Using Simple Recurrent Neural Networks

This code below sets parameters for data processing, loads IMDb movie review data, and pads the sequences to a maximum length of 500 words for both training and testing sets:

```
from keras.datasets import imdb
from keras.utils import pad_sequences

max_features = 10000
maxlen = 500
batch_size = 32

#Load data and pad it
(input_train, y_train), (input_test, y_test) = imdb.load_data(num_words=max_features)
input_train = pad_sequences(input_train, maxlen=maxlen)
input_test = pad_sequences(input_test, maxlen=maxlen)
```

A simple RNN model is then built:

```
#Define model
model = models.Sequential()
model.add(Embedding(max_features, 32))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])

#Train the model
model.fit(input_train, y_train,
          epochs=15,
          batch_size=128,
          validation_split=0.2)
```

```
Epoch 1/15
157/157 [==============================] - 32s 184ms/step - loss: 0.6111 - acc: 0.6584 - val_loss: 0.5074 - val_acc: 0.7724
Epoch 2/15
157/157 [==============================] - 28s 177ms/step - loss: 0.4006 - acc: 0.8298 - val_loss: 0.4121 - val_acc: 0.8200
Epoch 3/15
157/157 [==============================] - 22s 143ms/step - loss: 0.3218 - acc: 0.8702 - val_loss: 0.3725 - val_acc: 0.8418
Epoch 4/15
157/157 [==============================] - 24s 154ms/step - loss: 0.2623 - acc: 0.8978 - val_loss: 0.3643 - val_acc: 0.8580
Epoch 5/15
157/157 [==============================] - 25s 157ms/step - loss: 0.2176 - acc: 0.9172 - val_loss: 0.4296 - val_acc: 0.8468
Epoch 6/15
157/157 [==============================] - 25s 157ms/step - loss: 0.1781 - acc: 0.9339 - val_loss: 0.3779 - val_acc: 0.8636
Epoch 7/15
157/157 [==============================] - 23s 149ms/step - loss: 0.1272 - acc: 0.9557 - val_loss: 0.4546 - val_acc: 0.8534
Epoch 8/15
157/157 [==============================] - 24s 155ms/step - loss: 0.0908 - acc: 0.9700 - val_loss: 0.4949 - val_acc: 0.8190
```

This code above sets up a sequential model with an embedding layer followed by a SimpleRNN layer with 32 units and a dense output layer with a sigmoid activation function. It then compiles the model with the RMSprop optimizer, binary cross-entropy loss, and accuracy metric, and trains the model on the input training data for 15 epochs with a batch size of 128 and a validation split of 0.2.

The evaluation of the model is then shown as follows:

```
[15] #Evaluation
     results = model.evaluate(input_test, y_test)

     782/782 [==============================] - 20s 25ms/step - loss: 0.7460 - acc: 0.8312
```

The evaluation metric indicates a strong performance from the model.

More layers are then added to the model:

```
#Define model
model = models.Sequential()
model.add(Embedding(max_features, 32))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])

#Train the model
model.fit(input_train, y_train,
          epochs=15,
          batch_size=128,
          validation_split=0.2)
```

```
Epoch 1/15
157/157 [==============================] - 94s 578ms/step - loss: 0.6757 - acc: 0.5662 - val_loss: 0.5595 - val_acc: 0.7162
Epoch 2/15
157/157 [==============================] - 90s 570ms/step - loss: 0.4299 - acc: 0.8071 - val_loss: 0.4130 - val_acc: 0.8180
Epoch 3/15
157/157 [==============================] - 94s 599ms/step - loss: 0.3120 - acc: 0.8735 - val_loss: 0.4139 - val_acc: 0.8128
Epoch 4/15
157/157 [==============================] - 92s 589ms/step - loss: 0.2273 - acc: 0.9109 - val_loss: 0.4722 - val_acc: 0.8428
Epoch 5/15
157/157 [==============================] - 98s 622ms/step - loss: 0.1618 - acc: 0.9407 - val_loss: 0.5102 - val_acc: 0.7962
Epoch 6/15
157/157 [==============================] - 89s 566ms/step - loss: 0.0984 - acc: 0.9653 - val_loss: 0.5392 - val_acc: 0.8450
Epoch 7/15
```

However, the accuracy decreases indicating in this case adding more layers hinders the model performance:

```
[17] #Evaluation
     results = model.evaluate(input_test, y_test)

     782/782 [==============================] - 70s 89ms/step - loss: 1.0035 - acc: 0.7949
```

An Improved Model Based on LSTM Recurrent Neural Networks

The code below sets up a sequential model with an embedding layer followed by an LSTM layer with 32 units and a dense output layer with a sigmoid activation function. It then compiles the model with the RMSprop optimizer, binary cross-entropy loss, and accuracy metric, and trains the model on the input training data for 20 epochs with a batch size of 128 and a validation split of 0.2.

```
from keras.layers import LSTM

#Define LSTM network
model = models.Sequential()
model.add(Embedding(max_features, 32))
model.add(LSTM(32))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])

#Train the model
model.fit(input_train, y_train,
          epochs=20,
          batch_size=128,
          validation_split=0.2)

Epoch 1/20
157/157 [==============================] - 54s 328ms/step - loss: 0.5827 - acc: 0.6780 - val_loss: 0.4361 - val_acc: 0.8250
Epoch 2/20
157/157 [==============================] - 52s 330ms/step - loss: 0.3534 - acc: 0.8554 - val_loss: 0.3765 - val_acc: 0.8364
Epoch 3/20
157/157 [==============================] - 49s 315ms/step - loss: 0.2706 - acc: 0.8935 - val_loss: 0.2978 - val_acc: 0.8762
Epoch 4/20
157/157 [==============================] - 51s 327ms/step - loss: 0.2351 - acc: 0.9121 - val_loss: 0.3007 - val_acc: 0.8760
Epoch 5/20
157/157 [==============================] - 49s 312ms/step - loss: 0.2038 - acc: 0.9247 - val_loss: 0.4668 - val_acc: 0.8522
Epoch 6/20
157/157 [==============================] - 49s 314ms/step - loss: 0.1829 - acc: 0.9347 - val_loss: 0.2984 - val_acc: 0.8806
```

The evaluation metrics are as follows:

```
[19] #Evaluation
     results = model.evaluate(input_test, y_test)

     782/782 [==============================] - 33s 42ms/step - loss: 0.5965 - acc: 0.8538
```

The high accuracy indicates a great performance from the model, also the loss value of this function is the lowest of all the models indicating that the predictions deviate the least compared to others.

Exercise

To improve the model I tried increasing model complexity, regularizing to prevent overfitting, experimenting with optimizers, and adjusting learning rates for faster convergence:

```
from keras.layers import LSTM, Dropout

# Define LSTM network with dropout
model = models.Sequential()
model.add(Embedding(max_features, 32))
model.add(LSTM(64, dropout=0.2, recurrent_dropout=0.2))  # Increased LSTM units and added dropout
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',  # Changed optimizer to Adam
              loss='binary_crossentropy',
              metrics=['acc'])

# Train the model
history = model.fit(input_train, y_train,
                    epochs=7,
                    batch_size=128,
                    validation_split=0.2)
```

```
Epoch 1/7
157/157 [==============================] - 191s 1s/step - loss: 0.5353 - acc: 0.7131 - val_loss: 0.3807 - val_acc: 0.8302
Epoch 2/7
157/157 [==============================] - 177s 1s/step - loss: 0.2915 - acc: 0.8861 - val_loss: 0.3117 - val_acc: 0.8710
Epoch 3/7
157/157 [==============================] - 173s 1s/step - loss: 0.2096 - acc: 0.9222 - val_loss: 0.3092 - val_acc: 0.8808
Epoch 4/7
157/157 [==============================] - 177s 1s/step - loss: 0.1927 - acc: 0.9286 - val_loss: 0.3761 - val_acc: 0.8674
Epoch 5/7
157/157 [==============================] - 177s 1s/step - loss: 0.1499 - acc: 0.9456 - val_loss: 0.3298 - val_acc: 0.8616
Epoch 6/7
157/157 [==============================] - 175s 1s/step - loss: 0.1254 - acc: 0.9564 - val_loss: 0.3507 - val_acc: 0.8770
Epoch 7/7
157/157 [==============================] - 175s 1s/step - loss: 0.0978 - acc: 0.9671 - val_loss: 0.3721 - val_acc: 0.8626
```

This resulted in an improvement in accuracy and a lower loss score:

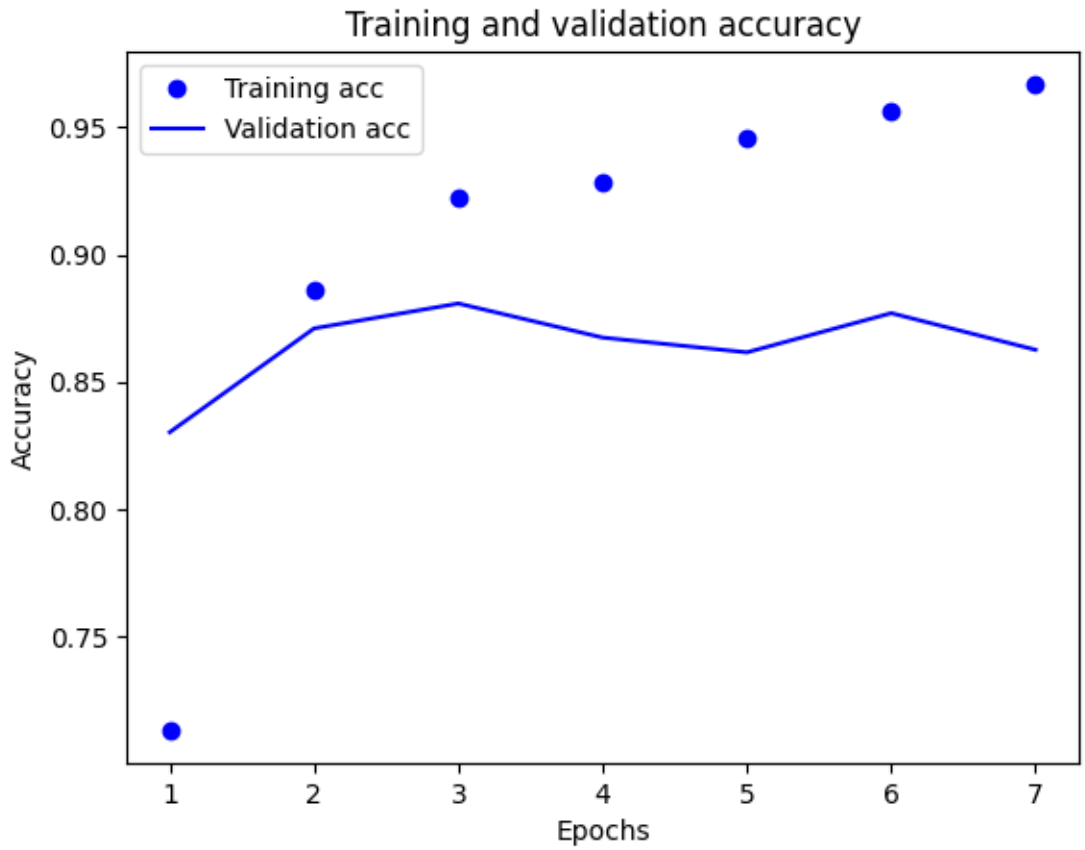
```
[22] #Evaluation
     results = model.evaluate(input_test, y_test)
```

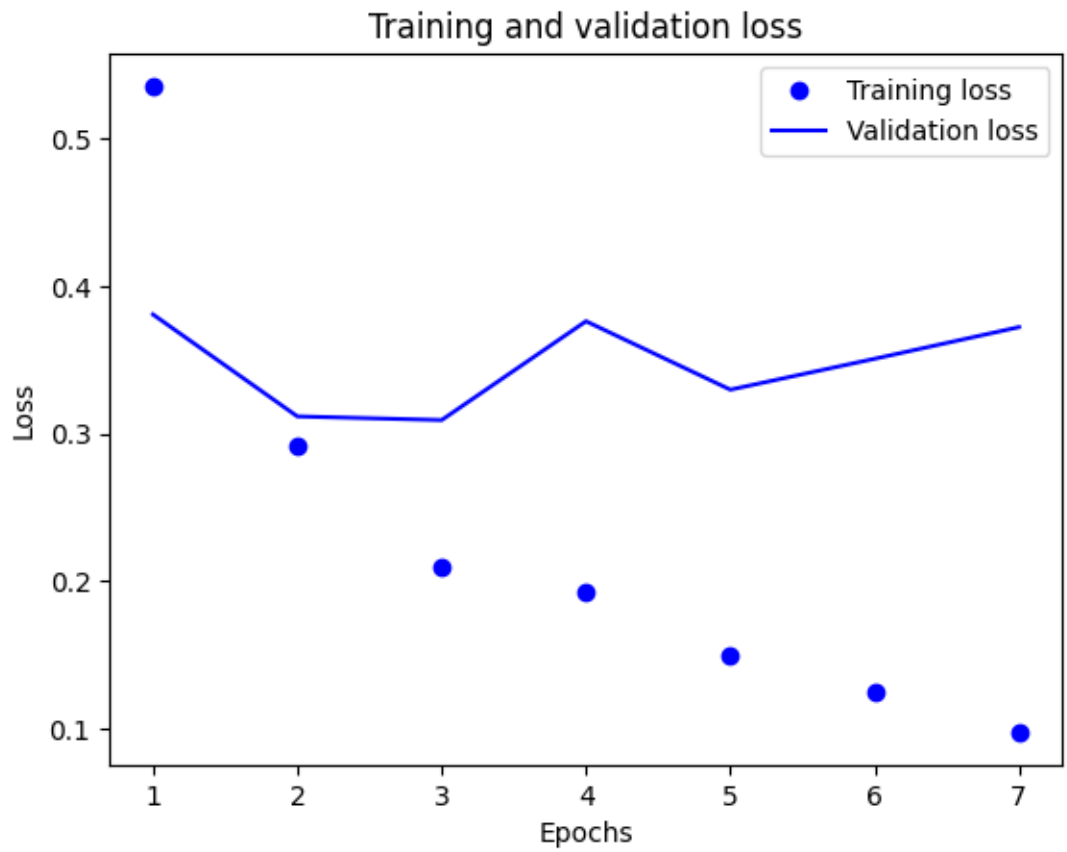
```
782/782 [==============================] - 65s 83ms/step - loss: 0.3946 - acc: 0.8549
```

I have also visualise this last models performance for each epoch of training:

```
import matplotlib.pyplot as plt


# Get training history
train_loss = history.history['loss']
val_loss = history.history['val_loss']
train_acc = history.history['acc']
val_acc = history.history['val_acc']
epochs = range(1, len(train_loss) + 1)

# Plotting loss
plt.plot(epochs, train_loss, 'bo', label='Training loss'
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

# Plotting accuracy
plt.plot(epochs, train_acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

Output:


Training and validation accuracy
Training acc
Validation acc
Accuracy
0.95
0.90
0.85
0.80
0.75
1
2
3
4
5
6
7
Epochs


Training and validation loss
Training loss
Validation loss
Loss
0.5
0.4
0.3
0.2
0.1
1
2
3
4
5
6
7
Epochs

Based on the graph above after epoch 3, the model overfits. However, due to only using 7 epochs more epochs are needed to investigate if the validation curve continues to diverge.

Week 9: Reflection

In this week's RNN lab, we focused on recurrent neural networks (RNNs) in natural language processing tasks, specifically sentiment analysis of IMDb movie reviews. Data was pre-processed, converted into sequences, and trained using Keras. Model architectures, hyperparameters, and optimization algorithms were experimented with to improve performance. Visualizing training history also helped monitor progress and deepen understanding of neural networks. Overall this lab helps me understand RNN in more depth and develops my skill in improving deep learning models.