

CSP 554 BIG DATA TECHNOLOGIES

PROJECT REPORT

Members: Soundarya Na (A20466037), Disha Hitesh Patel(A20452866), Abhishek Chamakura(A20457963)

Topic: Sentiment analysis of Amazon food reviews

Dataset:

Amazon Fine Food Reviews from Kaggle (<https://www.kaggle.com/snap/amazon-fine-food-reviews>)

Introduction:

Amazon is an online ecommerce website where usually users post the reviews of the products they purchase. These reviews give us an overview about the experience the users have as a result of purchasing the product. The sentiment of the user towards the product can be understood from the reviews. These help the vendor understand the user's perspective. This helps them to make changes to the products they sell accordingly and improve the user experience. These reviews are used as a dataset to understand the experience of the user and perform sentiment analysis in this project.

A regular technique to get important data is to separate the sentiment or opinion from a message. Sentiment analysis of product surveys, an application issue, has as of late become exceptionally well known in text mining and computational etymology research. AI advancements are broadly utilized in sentiment characterization in view of their capacity to gain from the preparation dataset to anticipate or uphold dynamic with generally high exactness. Apache Spark MLlib is quite possibly the most unmistakable stages for enormous information investigation which offers a bunch of phenomenal functionalities for various AI errands going from relapse, arrangement, and measurement decrease to grouping and rule extraction. Apache Spark MLlib is quite possibly the most exceptionally requested stage free and open-source libraries for huge information AI which profits by appropriated engineering and programmed data parallelization. This task uses Apache Spark MLlib for estimation investigation on Amazon food reviews.

Apache Hadoop:

Apache Hadoop is a collection of open-source software utilities that facilitates using a network of many computers to solve problems involving massive amounts of data and computation. It provides a software framework for distributed storage and processing of big data using the MapReduce programming model. Hadoop was originally designed for computer clusters built from commodity hardware, which is still the common use. It has since also found use on clusters of higher-end hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework.

The core of Apache Hadoop consists of a storage part, known as Hadoop Distributed File System (HDFS), and a processing part which is a MapReduce programming model. Hadoop splits files into large blocks and distributes them across nodes in a cluster. It then transfers packaged code into nodes to process the data in parallel. This approach takes advantage of data locality, where nodes manipulate the data, they have access to. This allows the dataset to be processed faster and more efficiently than it would be in a more conventional supercomputer architecture that relies on a parallel file system where computation and data are distributed via high-speed networking.

Apache Hive:

Apache Hive data warehouse software facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. Hive gives an SQL-like interface to query data stored in various databases and file systems that integrate with Hadoop. Traditional SQL queries must be implemented in the MapReduce Java API to execute SQL applications and queries over distributed data. Hive provides the necessary SQL abstraction to integrate SQL-like queries (HiveQL) into the underlying Java without the need to implement queries in the low-level Java API. Since most data warehousing applications work with SQL-based querying languages, Hive aids portability of SQL-based applications to Hadoop. While initially developed by Facebook, Apache.

We were planning to use Hive to store the dataset and connect it with python code to retrieve the data from the dataset accordingly, but while implementing the python code, we got “import sasl” error. When we tried importing that library using “pip install sasl” it gave another error attached below.

We tried install Microsoft Visual C++ in the laptop but no luck at all. I and my team did research about the issue and found out that if we try importing sasl in python2 it might work. I created a new environment in anaconda using python2 but did not work there as well. Hence, we used Hadoop Distributed File System to load the dataset and in the python script we gave the path accordingly.

```
In [4]: !pip3 install sasl3
```

```
Collecting sasl3
```

```
ERROR: Command errored out with exit status 1:
  command: 'C:\Users\nshind4\Anaconda3\python.exe' -u -c 'import sys, setuptools, tokenize; sys.argv[0] = '"'"'C:\Users\nshind4\AppData\Local\Temp\pip-install-rp85wc4q\sasl3\setup.py'"'"'; __file__='"'"'C:\Users\nshind4\AppData\Local\Temp\pip-install-rp85wc4q\sasl3\setup.py'"'"';f=getattr(tokenize, '"'"'open'"'"', open)(__file__);code=f.read().replace('"'"'\n'"'"', '"'"'\n'"'"');f.close();exec(compile(code, __file__, '"'"'exec'"'"'))' bdist_wheel -d 'C:\Users\nshind4\AppData\Local\Temp\pip-wheel-hzdnxyxb'
  cwd: C:\Users\nshind4\AppData\Local\Temp\pip-install-rp85wc4q\sasl3\
Complete output (21 lines):
running bdist_wheel
running build
running build_py
creating build
creating build\lib.win-amd64-3.8
creating build\lib.win-amd64-3.8\sasl
copying sasl\__init__.py -> build\lib.win-amd64-3.8\sasl
running egg_info
writing sasl3.egg-info\PKG-INFO
writing dependency_links to sasl3.egg-info\dependency_links.txt
writing requirements to sasl3.egg-info\requires.txt
writing top-level names to sasl3.egg-info\top_level.txt
reading manifest file 'sasl3.egg-info\SOURCES.txt'
reading manifest template 'MANIFEST.in'
writing manifest file 'sasl3.egg-info\SOURCES.txt'
copying sasl\saslwrapper.cpp -> build\lib.win-amd64-3.8\sasl
copying sasl\saslwrapper.h -> build\lib.win-amd64-3.8\sasl
copying sasl\saslwrapper.pvx -> build\lib.win-amd64-3.8\sasl
```

```

copying sasl\saslwrapper.pyx -> build\lib.win-amd64-3.8\sasl
running build_ext
building 'sasl.saslwrapper' extension
error: Microsoft Visual C++ 14.0 or greater is required. Get it with "Microsoft C++ Build Tools": https://visualstudio.microsoft.com/visual-cpp-build-tools/
-----
ERROR: Failed building wheel for sasl3
ERROR: Command errored out with exit status 1:
  command: 'C:\Users\nshind4\Anaconda3\python.exe' -u -c 'import sys, setuptools, tokenize; sys.argv[0] = '''C:\Users\nshind4\AppData\Local\Temp\pip-install-rp85wc4q\sasl3\setup.py'''; __file__ = '''C:\Users\nshind4\AppData\Local\Temp\pip-install-rp85wc4q\sasl3\setup.py'''; f=getattr(tokenize, '''open''', open)(__file__); code=f.read().replace('\r\n', '\n'); exec(code, {'__file__': __file__, '__name__': '__main__'})' install --record 'C:\Users\nshind4\AppData\Local\Temp\pip-record-w9z0r67p\install-record.txt' --single-version-externally-managed --compile --install-headers 'C:\Users\nshind4\Anaconda3\Include\sasl3'
  cwd: C:\Users\nshind4\AppData\Local\Temp\pip-install-rp85wc4q\sasl3\
Complete output (21 lines):
running install
running build
running build_py
creating build
creating build\lib.win-amd64-3.8
creating build\lib.win-amd64-3.8\sasl
copying sasl\__init__.py -> build\lib.win-amd64-3.8\sasl
running egg_info
writing sasl3.egg-info\PKG-INFO
writing dependency_links to sasl3.egg-info\dependency_links.txt
writing requirements to sasl3.egg-info\requires.txt
writing top-level names to sasl3.egg-info\top_level.txt
reading manifest file 'sasl3.egg-info\SOURCES.txt'
reading manifest template 'MANIFEST.in'
writing manifest file 'sasl3.egg-info\SOURCES.txt'
copying sasl\saslwrapper.cpp -> build\lib.win-amd64-3.8\sasl
copying sasl\saslwrapper.h -> build\lib.win-amd64-3.8\sasl
copying sasl\saslwrapper.pyx -> build\lib.win-amd64-3.8\sasl
running build_ext
building 'sasl.saslwrapper' extension
error: Microsoft Visual C++ 14.0 or greater is required. Get it with "Microsoft C++ Build Tools": https://visualstudio.microsoft.com/visual-cpp-build-tools/
-----
ERROR: Command errored out with exit status 1: 'C:\Users\nshind4\Anaconda3\python.exe' -u -c 'import sys, setuptools, tokenize; sys.argv[0] = '''C:\Users\nshind4\AppData\Local\Temp\pip-install-rp85wc4q\sasl3\setup.py'''; __file__ = '''C:\Users\nshind4\AppData\Local\Temp\pip-install-rp85wc4q\sasl3\setup.py'''; f=getattr(tokenize, '''open''', open)(__file__); code=f.read().replace('\r\n', '\n'); exec(code, {'__file__': __file__, '__name__': '__main__'})' install --record 'C:\Users\nshind4\AppData\Local\Temp\pip-record-w9z0r67p\install-record.txt' --single-version-externally-managed --compile --install-headers 'C:\Users\nshind4\Anaconda3\Include\sasl3' Check the logs for full command output.

Using cached sasl3-0.2.11.tar.gz (44 kB)
Requirement already satisfied: six in c:\users\nshind4\anaconda3\lib\site-packages (from sasl3) (1.15.0)
Building wheels for collected packages: sasl3
  Building wheel for sasl3 (setup.py): started
    Building wheel for sasl3 (setup.py): finished with status 'error'
  Running setup.py clean for sasl3
Failed to build sasl3
Installing collected packages: sasl3
  Running setup.py install for sasl3: started
    Running setup.py install for sasl3: finished with status 'error'

```

Apache Spark MLlib 2.0:

Apache Spark is a highly scalable, fast, and in-memory big data processing engine and it offers an ability to develop distributed applications using Java, Python, Scala, and R programming languages. It comes with four major libraries, including Apache Spark Streaming, Apache Spark SQL, Apache Spark GraphX, and Apache Spark ML-lib. Apache Spark MLlib is a big data analytics library which provides more than 55 scalable machine learning algorithms that benefit from both data and process parallelization. The library includes implementation of a variety of machine learning strategies, such as classification, clustering, regression, dimension reduction, and rule extraction which enables easy and fast in-practice development of large-scale machine learning applications.

Apache Spark MLlib offers fast, flexible, and scalable implementations of a variety of machine learning components, ranging from ensemble learning and principal component analysis (PCA) to optimization and clustering analysis. Apache Spark MLlib also offer options for distributed processing by parallel processing and support of big data tools that utilize distributed architectures. These criteria will decrease the processing time required and, at the same time, increase the time available to interpret analytics results. This becomes very important when the machine learning task has many predictions to calculate. The distributed architecture can also take advantages of some of the big data tool sets available to help break apart the machine learning component to improve overall running time. Integration is another advantage of Apache Spark MLlib, meaning that the MLlib gains from several software components available in the

Spark ecosystem, such as Spark GraphX, Spark SQL, and Spark Streaming, and a wide range of well-organized documentations, including code samples are publicly and freely available to the machine learning community. Given that, Apache Spark is well-suited for querying and trying to make sense of very, very large data sets. The software offers many advanced machine learning and econometrics tools, although these tools are used only partially because very large data sets require too much time when the data sets get too large. The software is not well-suited for projects that are not big data in size. The graphics and analytical output are subpar compared to other tools.

Proposed Approach:

1. Load data into the Hadoop environment: this requires AWS S3 and Hadoop clusters.
2. Clean and transform the dataset: collect the relevant columns, clean, and tokenize the reviews for sentiment analysis.
3. Use Machine Learning libraries in PySpark for sentiment analysis.

DataSet Source:

This dataset consists of reviews of Fine Foods from Amazon. The food reviews are from Oct 1999 to Oct 2012 (approx. 10 years). There is total of 568,454 food reviews from 256,059 users on 74,258 products. There are 260 users with more than 50 reviews. Reviews include product and user information, ratings, and a plain text review. The ratings given by the user for each product is on a scale of 1 to 5, 1 being the lowest. The data is present in the CSV format. Some significant columns of the dataset are as follows:

- Text: complete review text from the user
- Score: the rating of product ranging from 1 to 5
- Summary: Brief text of the review
- Helpfulness Numerator: number of users who found the review helpful.
- Helpfulness Denominator: number of users who indicated whether they found the review helpful or not.
- Product Id: unique identifier for each product.

Implementation Details:

The project implementation is done in the following phases:

➤ **Data Cleaning:**

The initial dataset was a comma-separated CSV format. And, to retain the 'Text' column's contents which consists of commas.

➤ **Data Selection:**

The data is read into RDD using Spark Context and is split with a tab separator. The data is then converted into a data frame. Filtered out the data which has Score=3 as it is a neutral score (i.e. neither comes under a positive nor negative review). Further, every word in the Text column is changed to lower case for the better analysis. Mainly, 'Score' and 'Text' columns will be used for performing the analysis.

➤ **Text Mining:**

All review texts are in lower case and all words that are not alphabets are removed. Stop Words in Text column are removed, and words are tokenized using built-in libraries StopWordsRemover and Tokenizer from spark MLlib. Removal of stopwords (i.e. frequently used words) enables to focus on the more important words of the review text provided by the user.

➤ **Storage:**

The data is stored in the Spark SQL table and then cached for enhancing the performance of analysis.

➤ **Data Analysis:**

Segregate words into positive word list and negative words list from most positive review (Score=5) and the most negative review (Score=1). Sort the words with respect to its frequency to know most used words in positive and negative reviews. Plot a word cloud of the positive and negative words list. Choose specific set of negative and positive words. Count the frequency of these words appearing in the reviews with Score 5 and 1. Select word 'bad' as a typical negative words and word 'great' as a typical positive. Find the frequency of 'bad' in the most negative reviews and the frequency of 'great' in the most positive reviews.

Results:

We get to see the following results.

Word Cloud for the most positive reviews



Word Cloud for the most negative reviews



With the selected positive and negative words as below:

```
selected_neg_words = set(['bad', 'hate', 'horrible', 'terrible', 'worst', 'dislike',
'disappointed', 'disappointing', 'never', 'waste', 'awful'])
```

```
selected_pos_words = set(['awesome', 'amazing', 'best', 'good', 'great', 'love', 'loves',  
'like', 'wonderful', 'tasty', 'fresh', 'organic', 'happy' ])
```

The count of the selected positive and negative words appearing in the most positive and negative reviews:

```
pos_word_count = 13 , neg_word_count = 11
```

The count of the word 'bad' appearing in the reviews with Score=1 is 135.

The count of the word 'great' appearing in the reviews with Score=5 is 349.

Conclusion:

Through the successful execution of this project, we understand that Spark is capable of

1. Handling large scale data
2. Manipulation of data frames and tables
3. Application of MLlib for data science research and experiments.
4. Distributed computing

Spark can be used for any aspect of Big Data analysis without depending on additional packages, modules or tools.

The code for the project is made available as a separate attachment, both as a py file and as a pdf. Relevant screen shots of execution of the project in Hadoop is shown below.

Now we look at setting up the Hadoop Cluster



```

_ | _ | _
_| ( /
_| \_|_|

```

Amazon Linux 2 AMI

EEEEEEEEEEEEEEEEEEEE	MMMMMMMM	MMMMMMMM	RRRRRRRRRRRRRR		
E:::E:::E:::E:::E:::E	M:::M	M:::M	R:::R:::R:::R:::R		
EEEEEEEEEEEEEEEE::E	M:::M	M:::M	R:::RRRRRR:::R		
E:::E	EEEE	M:::M	M:::M	RR::R	R:::R
E:::E	M:::M	M:::M	M:::M	R::R	R:::R
E:::EEEEEEEEEEEE	M:::M	M:::M	M:::M	R::RRRRRR:::R	
E:::EEEEEEEEEEEE	M:::M	M:::M	M:::M	R::RRRRRR::RR	
E:::EEEEEEEEEEEE	M:::M	M:::M	M:::M	R::RRRRRR:::R	
E:::E	M:::M	M:::M	M:::M	R::R	R:::R
E:::E	EEEE	M:::M	M:::M	R::R	R:::R
EEEEEEEEEEEEEEEE::E	M:::M	M:::M	M:::M	R::R	R:::R
E:::EEEEEEEEEEEE	M:::M	M:::M	M:::M	RR::R	R:::R
EEEEEEEEEEEEEEEEEEEE	MMMMMMMM	MMMMMMMM	RRRRRR	RRRRRR	

 version 2.4.7-amzn-1

```
>>> conf = (SparkConf().setAppName("project")
... .set("spark.shuffle.service.enabled","false")
... .set("spark.dynamicAllocation.enabled","false"))
>>> s = SparkContext.getOrCreate();
>>>
```

```

myProj.py Reviews.csv
[hadoop@ip-172-31-23-23 ~]$ spark-submit myProj.py
21/05/12 00:50:48 INFO SparkContext: Running Spark version 2.4.7-amzn-1
21/05/12 00:50:48 INFO SparkContext: Submitted application: myProj.py
21/05/12 00:50:48 INFO SecurityManager: Changing view acls to: hadoop
21/05/12 00:50:48 INFO SecurityManager: Changing modify acls to: hadoop
21/05/12 00:50:49 INFO SecurityManager: Changing view acls groups to:
21/05/12 00:50:49 INFO SecurityManager: Changing modify acls groups to:
21/05/12 00:50:49 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(hadoop); groups with view permissions: Set(); users with modify permissions: Set(hadoop); groups with modify permissions: Set()
21/05/12 00:50:49 INFO SparkEnv: Successfully started service 'sparkDriver' on port 38775.
21/05/12 00:50:49 INFO SparkEnv: Registering MapOutputTracker
21/05/12 00:50:49 INFO SparkEnv: Registering BlockManagerMaster
21/05/12 00:50:49 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
21/05/12 00:50:49 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
21/05/12 00:50:49 INFO DiskBlockManager: Created local directory at /mnt/tmp/blockmgr-445a7264-232a-42b2-a690-a7169622cc40
21/05/12 00:50:49 INFO MemoryStore: MemoryStore started with capacity 912.3 MB
21/05/12 00:50:49 INFO SparkEnv: Registering OutputCommitCoordinator
21/05/12 00:50:49 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
21/05/12 00:50:49 WARN Utils: Service 'SparkUI' could not bind on port 4041. Attempting port 4042.
21/05/12 00:50:49 WARN Utils: Service 'SparkUI' could not bind on port 4042. Attempting port 4043.
21/05/12 00:50:49 INFO Utils: Successfully started service 'SparkUI' on port 4043.
21/05/12 00:50:49 INFO SparkEnv: Bound SparkEnv to 0.0.0.0, and started at http://ip-172-31-23-23.ec2.internal:4043
21/05/12 00:50:49 INFO Utils: Using initial executors = 50, max of spark.dynamicAllocation.initialExecutors, spark.dynamicAllocation.minExecutors and spark.executor.instances
21/05/12 00:50:50 INFO RMProxy: Connecting to ResourceManager at ip-172-31-23-23.ec2.internal/172.31.23.23:8032
21/05/12 00:50:50 INFO Client: Requesting a new application from cluster with 1 NodeManagers
21/05/12 00:50:50 INFO Configuration: resource-types.xml not found
21/05/12 00:50:51 INFO ResourceUtils: Unable to find 'resource-types.xml'.
21/05/12 00:50:51 INFO ResourceUtils: Adding resource type - name = memory-mb, units = Mi, type = COUNTABLE
21/05/12 00:50:51 INFO ResourceUtils: Adding resource type - name = vcores, units = , type = COUNTABLE
21/05/12 00:50:51 INFO Client: Verifying our application has not requested more than the maximum memory capability of the cluster (12288 MB per container)
21/05/12 00:50:51 INFO Client: Will allocate AM container, with 896 MB memory including 384 MB overhead
21/05/12 00:50:51 INFO Client: Setting up container launch context for our AM
21/05/12 00:50:51 INFO Client: Setting up the launch environment for our AM container
21/05/12 00:50:51 INFO Client: Preparing resources for our AM container

```


Results:

```
21/05/12 00:51:21 INFO DAGScheduler: ResultStage 6 (showString at NativeMethodAccessorImpl.java:0) finished in 0.117 s
21/05/12 00:51:21 INFO DAGScheduler: Job 5 finished: showString at NativeMethodAccessorImpl.java:0, took 0.125640 s
+-----+
|Score|      Text|
+-----+
|5|      I have bought sev...|
|1|      Product arrived ...|
|5|      Great taffy at a ...|
|5|      This saltwater ta...|
|5|      This taffy is so ...|
|5|      Right now I'm mos...|
|5|      This is a very he...|
|5|      I don't know if i...|
|5|      One of my boys ne...|
|1|      My cats have been...|
+-----+
only showing top 10 rows
```

```
21/05/12 00:51:21 INFO DAGScheduler: Job 6 finished: showString at NativeMethodAccessorImpl.java:0, took 0.266264 s
+-----+
|      text_lower|Score|
+-----+
|i have bought sev...| 5|
|product arrived l...| 1|
|great taffy at a ...| 5|
|this saltwater ta...| 5|
|this taffy is so ...| 5|
|right now i m mos...| 5|
|this is a very he...| 5|
|i don t know if i...| 5|
|one of my boys ne...| 5|
|my cats have been...| 1|
|the strawberry tw...| 5|
|my daughter loves...| 5|
|i am very satisfi...| 5|
|twizzlers strawbe...| 5|
|candy was deliver...| 5|
+-----+
only showing top 15 rows
21/05/12 00:51:23 INFO DAGScheduler: Job 8 finished: showString at NativeMethodAccessorImpl.java:0, took 0.14804 s
+-----+
|      filtered_words|Score|
+-----+
|[bought, several,...| 5|
|[product, arrived...| 1|
|[great, taffy, gr...| 5|
|[saltwater, taffy...| 5|
|[taffy, good, sof...| 5|
+-----+
only showing top 5 rows
```

```
21/05/12 00:51:23 INFO DAGScheduler: Job 9 finished: showString at NativeMethodAccessorImpl.java:0, took 0.335965 s
+-----+
|      words|Score|
+-----+
|bought several vi...| 5|
|product arrived l...| 1|
|great taffy great...| 5|
|saltwater taffy g...| 5|
|taffy good soft c...| 5|
+-----+
only showing top 5 rows

root
|-- words: string (nullable = true)
|-- Score: string (nullable = true)

21/05/12 00:51:24 INFO DAGScheduler: Job 11 finished: showString at NativeMethodAccessorImpl.java:0, took 0.246175 s
+-----+
|      words|Score|
+-----+
|bought several vi...| 5|
|product arrived l...| 1|
|great taffy great...| 5|
|saltwater taffy g...| 5|
|taffy good soft c...| 5|
+-----+
only showing top 5 rows

root
|-- words: string (nullable = true)
|-- Score: string (nullable = true)
```

```
21/05/12 00:51:31 INFO DAGScheduler: Job 12 finished: runJob at PythonRDD.scala:153, took 5.863885 s
root
|-- Score: string (nullable = true)
|-- word: array (nullable = true)
|   |-- element: string (containsNull = true)

21/05/12 00:51:31 INFO CodeGenerator: Code generated in 19.238313 ms
21/05/12 00:51:31 INFO SparkContext: Starting job: showString at NativeMethodAccessorImpl.java:0
21/05/12 00:51:31 INFO DAGScheduler: Got job 13 (showString at NativeMethodAccessorImpl.java:0) with 1 output partitions
21/05/12 00:51:31 INFO DAGScheduler: Final stage: ResultStage 14 (showString at NativeMethodAccessorImpl.java:0)
21/05/12 00:51:31 INFO DAGScheduler: Parents of final stage: List()
21/05/12 00:51:31 INFO DAGScheduler: Missing parents: List()
21/05/12 00:51:31 INFO DAGScheduler: Submitting ResultStage 14 (MapPartitionsRDD[94] at showString at NativeMethodAccessorImpl.java:0), which has no missing parents
21/05/12 00:51:31 INFO MemoryStore: Block broadcast_25 stored as values in memory (estimated size 45.8 KB, free 911.1 MB)
21/05/12 00:51:31 INFO MemoryStore: Block broadcast_25_piece0 stored as bytes in memory (estimated size 23.4 KB, free 911.1 MB)
21/05/12 00:51:31 INFO BlockManagerInfo: Added broadcast_25_piece0 in memory on ip-172-31-23-23.ec2.internal:33907 (size: 23.4 KB, free: 912.2 MB)
21/05/12 00:51:31 INFO SparkContext: Created broadcast 25 from broadcast at DAGScheduler.scala:1280
21/05/12 00:51:31 INFO DAGScheduler: Submitting 1 missing tasks from ResultStage 14 (MapPartitionsRDD[94] at showString at NativeMethodAccessorImpl.java:0) (first 15 tasks are for partitions Vector(0))
21/05/12 00:51:31 INFO YarnScheduler: Adding task set 14.0 with 1 tasks
21/05/12 00:51:31 INFO TaskSetManager: Starting task 0.0 in stage 14.0 (TID 22, ip-172-31-22-112.ec2.internal, executor 1, partition 0, PROCESS_LOCAL, 8577 bytes)
21/05/12 00:51:31 INFO BlockManagerInfo: Added broadcast_25_piece0 in memory on ip-172-31-22-112.ec2.internal:40635 (size: 23.4 KB, free: 2.2 GB)
21/05/12 00:51:31 INFO TaskSetManager: Finished task 0.0 in stage 14.0 (TID 22) in 124 ms on ip-172-31-22-112.ec2.internal (executor 1) (1/1)
21/05/12 00:51:31 INFO YarnScheduler: Removed TaskSet 14.0, whose tasks have all completed, from pool
21/05/12 00:51:31 INFO DAGScheduler: ResultStage 14 (showString at NativeMethodAccessorImpl.java:0) finished in 0.142 s
21/05/12 00:51:31 INFO DAGScheduler: Job 13 finished: showString at NativeMethodAccessorImpl.java:0, took 0.148582 s
-----+-----+
|Score|word|
+-----+-----+
|5|[bought, several,...]|1|[product, arrived...]|
|5|[great, taffy, gr...]|5|[saltwater, taffy...]|
|5|[taffy, good, sof...]|
+-----+-----+
only showing top 5 rows
```