

Amazon Customer Reviews (US Based Dataset)

CIS 4130 Big Data Technologies

Semester Project

Prepared By
Pranesh Chitrakar
pranesh.chitrakar@baruchmail.cuny.edu

Main Dataset
<https://www.kaggle.com/datasets/cynthiarempel/amazon-us-customer-reviews-dataset>

Milestone 1

Introduction:

Amazon Customer Reviews dataset is a collection of millions of Amazon's user reviews from 1995 to 2015. Over hundred million customers have conveyed their experiences regarding the merchandises on the Amazon.com website. The data helps us to study the properties and evolution of customer reviews on how the users express their views on a product/s.

This dataset consists of 15 columns:

1. marketplace- 2 letters country code (AU for Australia, US for United States)
2. customer_id- random identifier that can be used to aggregate reviews by a single user
3. review_id- unique ID of the review
4. product_id- unique Product ID the review pertains to
5. product_parent- random identifier that can be used to aggregate reviews for the same product.
6. product_title- title of the product
7. product_category- broad product category that can be used to group reviews (
8. star_rating- 1-5 star rating of the review
9. helpful_votes- no. of helpful votes
10. total_vote- no. of helpful votes received
11. vine- review was written as part of the Vine program.
12. verified_purchase- the review is on a verified purchase.
13. review_headline- the title of the review.
14. review_body- the review text.
15. review_date- the date the review was written

This dataset has a bunch of opportunities to that come in had for Amazon Sellers, Customers as well as the site itself. By addressing the reviews, products can be recommended to certain parties and forecast(predict) the demand.

My focus for this project will be predicting the review ratings for respective products from the dataset.

Milestone 2

For this project, I will be using Amazon Web Service (AWS) Management Console and its platform like EC2, S3, and so on to manage and monitor the dataset. The Amazon Customer Reviews dataset is approximately 22GB, and to download and re-upload manually from Kaggle(kaggle.com/) would take me hours. I am using Amazon S3(Simple Storage Service) to create a bucket where the dataset can be extracted and stored in less amount of time.

The steps I followed are below:

1. Set up AWS Command Line Interface (CLI) in my EC2 instance using my Access Key ID and Secret Access Key.
2. In Amazon EC2 instance with Amazon Linux, I first created a bucket named 'my-data-bucket-pc' to store the dataset.

```
$ aws s3api create-bucket --bucket my-data-bucket-pc --region us-east-2 \
--create-bucket-configuration LocationConstraint=us-east-2
```

3. Followed the steps for downloading data directly from Kaggle to Amazon S3 using Kaggle API
4. After the installation and configuration, I used `$ kaggle datasets list` to search the data set

```
[ec2-user@ip-172-31-39-117 ~]$ kaggle datasets list
```

id	title	size	lastUpdated
iamsouravbanerjee/world-population-dataset	World Population Dataset	17KB	2022-08-31 11:20:04
whenamancodes/hr-employee-attrition	Employee Analysis Attrition Report	50KB	2022-09-12 10:46:33
pantanjali/unemployment-dataset	Unemployment dataset	17KB	2022-09-08 08:26:10

5. To download my data, I used `$ kaggle datasets download` command to fetch the dataset and to pipe the output and direct it to my S3 bucket, I used `aws s3 cp` command.

```
$ kaggle datasets download --quiet -d cynthiarempel/amazon-us-customer-reviews-dataset -p - | aws s3 cp - s3://my-data-bucket-pc/data.zip
```

6. To check my S3 bucket to see if the file was downloaded

```
$ aws s3 ls s3://my-data-bucket-pc/
```

```
[ec2-user@ip-172-31-39-117 ~]$ aws s3 ls s3://my-data-bucket-pc/
2022-10-01 18:15:23      403822 archive.zip
2022-10-01 18:16:39  22497731749 data.zip
```

Milestone 3

7. Testing the Boto3 module in Python

```
import boto3
s3 = boto3.resource('s3')
for bucket in s3.buckets.all():

    print(bucket.name)
```

```
>>> import boto3
>>> s3 = boto3.resource('s3')
>>> for bucket in s3.buckets.all():
...     print(bucket.name)
...
amazon-customer-review
my-data-bucket-pc
```

As we can see above, 'amazon-customer-review', and 'my-data-bucket-pc' are the bucket I created for this project.

8. Unzipping files with S3

```
>>> import zipfile
>>> import boto3
>>> from io import BytesIO
>>> bucket="my-data-bucket-pc"
>>> zipfile_to_unzip="data.zip"
>>> s3_client= boto3.client('s3', use_ssl=False)
>>> s3_resource= boto3.resource('s3')
>>> zip_obj = s3_resource.Object(bucket_name=bucket, key=zipfile_to_unzip)
>>> buffer = BytesIO(zip_obj.get() ["Body"].read())
>>> z = zipfile.ZipFile(buffer)
>>> for filename in z.namelist():
...     print('Working on ' + filename)
...     s3_resource.meta.client.upload_fileobj(z.open(filename), Bucket=bucket, Key=f'{filename}')
...
Working on amazon_reviews_multilingual_US_v1_00.tsv
Working on amazon_reviews_us_Apparel_v1_00.tsv
Working on amazon_reviews_us_Automotive_v1_00.tsv
Working on amazon_reviews_us_Baby_v1_00.tsv
Working on amazon_reviews_us_Beauty_v1_00.tsv
Working on amazon_reviews_us_Books_v1_00.tsv
```

In this step, I imported the ZIP file into my EC2 instance. While doing so, I had to change the instance type from t2.micro to t2.xlarge(32GB).

```
import pandas as pd
df=pd.read_table('s3://my-data-bucket-pc/amazon_reviews_multilingual_US_v1_00.tsv',lineterminator='\n',error_bad_lines=False)
```

```
>>> df= pd.read_table('s3://my-data-bucket-pc/amazon_reviews_multilingual_US_v1_00.tsv',lineterminator='\n',error_bad_lines=False)
sys:1: FutureWarning: The error_bad_lines argument has been deprecated and will be removed in a future version.
```

Here, I had to use the "lineterminator='\n',error_bad_lines=False)" in order to skip the lines that were causing errors to load so that we can move with our pandas DataFrame.

```
>>> print(df.info)
<bound method DataFrame.info of
0      US      53096384 ... this is the first 8 issues of the series. it i... 1995-08-13      review_body review_date
1      US      53096399 ... I've always been partial to immutable laws. Th... 1995-08-17
2      US      53096332 ... This is a book about first contact with aliens... 1995-08-30
3      US      53096335 ... This is quite possibly *the* funniest book I h... 1995-09-11
4      US      51747709 ... The story behind the book is almost better tha... 1995-10-17
...      ...      ...      ...      ...
6900881    US      52303 ... very fun game 2015-08-31
6900882    US      565563 ... Great 2015-08-31
6900883    US      254421 ... This is my first book from Jodi, and she has b... 2015-08-31
6900884    US      146004 ... Good movie! 2015-08-31
6900885    US      146004 ... Amazing show that runs close to history. Love ... 2015-08-31
[6900886 rows x 15 columns]>
```

df.dtypes

Ten of the variables are categorical(which are labelled as 'object') while the remaining variables are numerical (which are labelled as 'int64'(2) and 'float64'(3)).

```
dtypes: float64(3), int64(2), object(10)
```

```
>>> df.dtypes
marketplace      object
customer_id      int64
review_id        object
product_id       object
product_parent   int64
product_title    object
product_category object
star_rating      float64
helpful_votes    float64
total_votes      float64
vine            object
verified_purchase object
review_headline  object
review_body      object
review_date      object
dtype: object
```

Grouping the data

```
results_star_rating =
df.groupby('customer_id').star_rating.agg(['count', 'min',
'max', 'mean', 'std'])
>>> print(results_star_rating)
```

```
>>> results_star_rating = df.groupby('customer_id').star_rating.agg(['count', 'min', 'max', 'mean', 'std'])
>>> print(results_star_rating)
      count  min  max  mean  std
customer_id
10001      2  5.0  5.0   5.0  0.000000
10018      1  4.0  4.0   4.0      NaN
10019      1  5.0  5.0   5.0      NaN
10020      1  5.0  5.0   5.0      NaN
10022      2  5.0  5.0   5.0  0.000000
...      ...  ...  ...   ...   ...
53096567    1  1.0  1.0   1.0      NaN
53096575    1  1.0  1.0   1.0      NaN
53096582    2  4.0  4.0   4.0  0.000000
53096584    4  2.0  5.0   4.0  1.414214
53096589    1  5.0  5.0   5.0      NaN
```

```
>>> results_helpful_votes =
df.groupby('customer_id').helpful_votes.agg(['count', 'min',
'max', 'mean', 'std'])
```

```
>>> print(results_helpful_votes)
>>> results_helpful_votes = df.groupby('customer_id').helpful_votes.agg(['count', 'min', 'max', 'mean', 'std'])
>>> print(results_helpful_votes)
```

customer_id	count	min	max	mean	std
10001	2	0.0	1.0	0.5	0.707107
10018	1	0.0	0.0	0.0	NaN
10019	1	0.0	0.0	0.0	NaN
10020	1	0.0	0.0	0.0	NaN
10022	2	0.0	0.0	0.0	0.000000
...
53096567	1	0.0	0.0	0.0	NaN
53096575	1	4054.0	4054.0	4054.0	NaN
53096582	2	6.0	1062.0	534.0	746.704761
53096584	4	0.0	209.0	52.5	104.334398
53096589	1	3.0	3.0	3.0	NaN

```
[4097784 rows x 5 columns]
```

```
>>> results_total_votes =
df.groupby('customer_id').total_votes.agg(['count', 'min',
'max', 'mean', 'std'])
>>> print(results_total_votes)
```

```
>>> results_total_votes = df.groupby('customer_id').total_votes.agg(['count', 'min', 'max', 'mean', 'std'])
>>> print(results_total_votes)
```

customer_id	count	min	max	mean	std
10001	2	0.0	2.0	1.00	1.414214
10018	1	0.0	0.0	0.00	NaN
10019	1	0.0	0.0	0.00	NaN
10020	1	0.0	0.0	0.00	NaN
10022	2	0.0	0.0	0.00	0.000000
...
53096567	1	0.0	0.0	0.00	NaN
53096575	1	4756.0	4756.0	4756.00	NaN
53096582	2	6.0	1114.0	560.00	783.474314
53096584	4	0.0	246.0	61.75	122.834238
53096589	1	3.0	3.0	3.00	NaN

```
[4097784 rows x 5 columns]
```

Finding out the total NaN values throughout the dataframe

```
>>> total_nan_value = df.isna().sum().sum()
>>> print(total_nan_value)
```

```
>>> total_nan_value = df.isna().sum().sum()
>>> print(total_nan_value)
480
```

For the review_date variable, since it is categorical, I still need to convert it to datetime format to work with finding the min and max date using python function. The review dates in my dataframe are in ascending order, therefore in the image below, we can see the min and max date.

```
>>> df.loc[:, "review_date"]
0      1995-08-13
1      1995-08-17
2      1995-08-30
3      1995-09-11
4      1995-10-17
...
6900881 2015-08-31
6900882 2015-08-31
6900883 2015-08-31
6900884 2015-08-31
6900885 2015-08-31
Name: review_date, Length: 6900886, dtype: object
```

Milestone 4

For this milestone, I used Amazon EMR to create cluster that implements Apache Hadoop and Spark.

Launching the EMR Cluster

I created a Cluster named "My cluster4" on Amazon EMR

General Configuration

Cluster name

☒ Logging ⓘ

S3 folder

Launch mode ☒ Cluster ⓘ ☐ Step execution ⓘ

Software configuration

Release ⓘ

Applications

- ☒ Core Hadoop: Hadoop 3.2.1 with Hive 3.1.3, Hue 4.10.0, Pig 0.17.0 and Tez 0.9.2
- ☐ HBase: HBase 2.4.12 with Hadoop 3.2.1, Hive 3.1.3, Hue 4.10.0, Phoenix 5.1.2, and ZooKeeper 3.5.10
- ☐ Presto: Presto 0.273.3 with Hadoop 3.2.1 HDFS and Hive 3.1.3 Metastore
- ☐ Spark: Spark 3.3.0 on Hadoop 3.2.1 YARN with and Zeppelin 0.10.1
- ☐ Trino: Trino 388 with Hadoop 3.2.1 HDFS and Hive 3.1.3 Metastore

☐ Use AWS Glue Data Catalog for table metadata ⓘ

Hardware configuration

Instance type ⓘ The selected instance type adds 64 GiB of GP2 EBS storage per instance by default. [Learn more](#)

Number of instances (1 master and 2 core nodes)

Then navigated the instances from EC2

Connect to instance [Info](#)
Connect to your instance i-030dcf85de6ab620b using any of these options

EC2 Instance Connect

Session Manager

SSH client

EC2 serial console

Instance ID
i-030dcf85de6ab620b

Public IP address
3.16.23.200

User name

Connect using a custom user name, or use the default user name root for the AMI used to launch the instance.

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Writing Hadoop MapReduce program in Python

```
[hadoop@ip-172-31-23-236 ~]$ ls -l
total 0
[hadoop@ip-172-31-23-236 ~]$ nano mapper.py
[hadoop@ip-172-31-23-236 ~]$ nano reducer.py
[hadoop@ip-172-31-23-236 ~]$ ls -l
total 8
-rw-rw-r-- 1 hadoop hadoop 423 Dec 16 20:43 mapper.py
-rw-rw-r-- 1 hadoop hadoop 921 Dec 16 20:47 reducer.py
```

Making a new directory folder on HDFS

```
[hadoop@ip-172-31-23-236 ~]$ hdfs dfs -mkdir hdfs:///bigdata
```

Now we are going to use the dataset from S3 bucket

```
s3-dist-cp --src s3://my-data-bucket-pc/amazon_reviews_multilingual_US_v1_00.tsv -
--dest hdfs:///bigdata
```

```
[hadoop@ip-172-31-23-236 ~]$ s3-dist-cp --src s3://my-data-bucket-pc/amazon_reviews_multilingual_US_v1_00.tsv --dest hdfs:///bigdata
2022-12-16 21:18:38,827 INFO s3distcp.Main: Running with args: -libjars /usr/share/aws/emr/s3-dist-cp/lib/aopalliance-1.0.jar,/usr/share/aws/emr/s3-dist-cp/lib/guava-18.0.jar,/usr/share/aws/emr/s3-dist-cp/lib/guice-4.1.0.jar,/usr/share/aws/emr/s3-dist-cp/lib/guice-servlet-4.2.3.jar,/usr/share/aws/emr/s3-dist-cp/lib/javax.inject-1.jar,/usr/share/aws/emr/s3-dist-cp/lib/s3-dist-cp-2.22.0.jar,/usr/share/aws/emr/s3-dist-cp/lib/s3-dist-cp.jar --src s3://my-data-bucket-pc/amazon_reviews_multilingual_US_v1_00.tsv --dest hdfs:///bigdata
2022-12-16 21:18:39,127 INFO s3distcp.S3DistCp: S3DistCp args: --src s3://my-data-bucket-pc/amazon_reviews_multilingual_US_v1_00.tsv --dest hdfs:///bigdata
2022-12-16 21:18:39,132 INFO s3distcp.S3DistCp: Using output path 'hdfs:/tmp/41c2b6ff-1880-44dc-b5e5-74442a636796/output'
2022-12-16 21:18:39,132 INFO s3distcp.S3DistCp: Try to recursively delete with throw Exceptionhdfs:/tmp/41c2b6ff-1880-44dc-b5e5-74442a636796/files
2022-12-16 21:18:39,582 INFO s3distcp.S3DistCp: Try to recursively delete with throw Exceptionhdfs:/tmp/41c2b6ff-1880-44dc-b5e5-74442a636796/output
2022-12-16 21:18:42,198 INFO s3distcp.S3ClientFactory: Create Amazon S3 client with endpoint override
2022-12-16 21:18:42,710 INFO s3distcp.S3ClientFactory: DefaultAWSCredentialsProviderChain is used to create AmazonS3Client.
2022-12-16 21:18:42,710 INFO s3distcp.S3ClientFactory: Overrides Amazon S3 Endpoint with s3.us-east-2.amazonaws.com
2022-12-16 21:18:42,977 INFO s3distcp.S3DistCp: Get src prefix s3://my-data-bucket-pc/amazon_reviews_multilingual_US_v1_00.tsv endWithSlash false
2022-12-16 21:18:42,977 INFO s3distcp.S3DistCp: Only a single prefix is got
2022-12-16 21:18:42,981 INFO s3distcp.S3DistCp: listing objects in bucket my-data-bucket-pc target prefix is amazon_reviews_multilingual_US_v1_00.tsv
2022-12-16 21:18:43,006 WARN cred.CredentialsLegacyConfigLocationProvider: Found the legacy config profiles file at [/home/hadoop/.aws/config]. Please move it to the latest default location [~/.aws/credentials].
2022-12-16 21:18:43,295 INFO s3distcp.S3DistCp: Got object summary
Bucket = my-data-bucket-pc, Key = amazon_reviews_multilingual_US_v1_00.tsv
```



```

2022-12-16 21:18:45,018 INFO mapreduce.Job: Running job: job_1671223184898_0001
2022-12-16 21:18:52,338 INFO mapreduce.Job: Job job_1671223184898_0001 running in uber mode : false
2022-12-16 21:18:52,339 INFO mapreduce.Job: map 0% reduce 0%
2022-12-16 21:18:58,386 INFO mapreduce.Job: map 100% reduce 0%
2022-12-16 21:19:04,414 INFO mapreduce.Job: map 100% reduce 33%
2022-12-16 21:19:05,419 INFO mapreduce.Job: map 100% reduce 67%
2022-12-16 21:19:16,458 INFO mapreduce.Job: map 100% reduce 100%
2022-12-16 21:20:45,730 INFO mapreduce.Job: Job job_1671223184898_0001 completed successfully
2022-12-16 21:20:45,814 INFO mapreduce.Job: Counters: 59

```

Below, I have pasted the screenshot of the counters and Shuffle Errors that it gave me as a result

File System Counters	Map-Reduce Framework
FILE: Number of bytes read=163	Map input records=1
FILE: Number of bytes written=979031	Map output records=1
FILE: Number of read operations=0	Map output bytes=177
FILE: Number of large read operations=0	Map output materialized bytes=151
FILE: Number of write operations=0	Input split bytes=169
HDFS: Number of bytes read=426	Combine input records=0
HDFS: Number of bytes written=3629753164	Combine output records=0
HDFS: Number of read operations=20	Reduce input groups=1
HDFS: Number of large read operations=0	Reduce shuffle bytes=151
HDFS: Number of write operations=7	Reduce input records=1
HDFS: Number of bytes read erasure-coded=0	Reduce output records=0
S3: Number of bytes read=3629753164	Spilled Records=2
S3: Number of bytes written=0	Shuffled Maps =3
S3: Number of read operations=0	Failed Shuffles=0
S3: Number of large read operations=0	Merged Map outputs=3
S3: Number of write operations=0	GC time elapsed (ms)=650
	CPU time spent (ms)=44560
	Physical memory (bytes) snapshot=2159296512
	Virtual memory (bytes) snapshot=25683906560
	Total committed heap usage (bytes)=2070937600
	Peak Map Physical memory (bytes)=498593792
	Peak Map Virtual memory (bytes)=4397223936
	Peak Reduce Physical memory (bytes)=1107648512
	Peak Reduce Virtual memory (bytes)=7113846784

Shuffle Errors	Job Counters
BAD_ID=0	Launched map tasks=1
CONNECTION=0	Launched reduce tasks=3
IO_ERROR=0	Data-local map tasks=1
WRONG_LENGTH=0	Total time spent by all maps in occupied slots (ms)=386976
WRONG_MAP=0	Total time spent by all reduces in occupied slots (ms)=21051648
WRONG_REDUCE=0	Total time spent by all map tasks (ms)=4031
	Total time spent by all reduce tasks (ms)=109644
File Input Format Counters	Total vcore-milliseconds taken by all map tasks=4031
Bytes Read=257	Total vcore-milliseconds taken by all reduce tasks=109644
File Output Format Counters	Total megabyte-milliseconds taken by all map tasks=12383232
Bytes Written=0	Total megabyte-milliseconds taken by all reduce tasks=673652736

Checking the resulting file on HDFS:

```

[hadoop@ip-172-31-23-236 ~]$ hdfs dfs -ls hdfs:///bigdata/amazon_reviews_multilingual_US_v1_00.tsv
-rw-r--r-- 1 hadoop hdfsadmin 3629753164 2022-12-16 21:20 hdfs:///bigdata/amazon_reviews_multilingual_US_v1_00.tsv

```

```

$ hdfs fsck hdfs:///bigdata/amazon_reviews_multilingual_US_v1_00.tsv -files -blocks -locations

```

```
Status: HEALTHY
Number of data-nodes: 2
Number of racks: 1
Total dirs: 0
Total symlinks: 0

Replicated Blocks:
Total size: 3629753164 B
Total files: 1
Total blocks (validated): 28 (avg. block size 129634041 B)
Minimally replicated blocks: 28 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Missing blocks: 0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)

Erasure Coded Block Groups:
Total size: 0 B
Total files: 0
Total block groups (validated): 0
Minimally erasure-coded block groups: 0
Over-erasure-coded block groups: 0
Under-erasure-coded block groups: 0
Unsatisfactory placement block groups: 0
Average block group size: 0.0
Missing block groups: 0
Corrupt block groups: 0
Missing internal blocks: 0
```

```
>>> from pyspark.sql.functions import col, isnan, when, count, udf
>>> sc.setLogLevel("ERROR")
>>> bucket= 'my-data-bucket-pc/'
>>> filename='amazon_reviews_multilingual_US_v1_00.tsv'
>>> file_path = 's3a://' + bucket + filename
```

```
>>> print(file_path)
s3a://my-data-bucket-pc/amazon_reviews_multilingual_US_v1_00.tsv
>>> sdf = spark.read.csv(file_path, sep='\t', header=True, inferSchema=True)
>>> sdf.printSchema()
root
|-- marketplace: string (nullable = true)
|-- customer_id: integer (nullable = true)
|-- review_id: string (nullable = true)
|-- product_id: string (nullable = true)
|-- product_parent: integer (nullable = true)
|-- product_title: string (nullable = true)
|-- product_category: string (nullable = true)
|-- star_rating: integer (nullable = true)
|-- helpful_votes: integer (nullable = true)
|-- total_votes: integer (nullable = true)
|-- vine: string (nullable = true)
|-- verified_purchase: string (nullable = true)
|-- review_headline: string (nullable = true)
|-- review_body: string (nullable = true)
|-- review_date: timestamp (nullable = true)
```

Below are some summaries of the data after dropping some of the records where the column are empty(null)

```
>>> sdf.count()
6931166
>>> sdf = sdf.na.drop(subset=["star_rating", "review_body", "review_date"])
>>> sdf.select("star_rating", "helpful_votes", "total_votes").summary("count", "min", "max", "mean").show()
```

summary	star_rating	helpful_votes	total_votes
count	6930482	6930482	6930482
min	1	0	0
max	5	27550	28727
mean	4.306747784641819	2.0379774451473938	3.2418290675886614

```
>>> sdf.summary().show()
```

summary	marketplace	customer_id	review_id	product_id	product_parent	product_title
rating	helpful_votes	total_votes	vine	verified_purchase	review_headline	review_body
count	6930482	6930482	6930482	6930482	6930482	6930482
min	1	0	0	0	0	0
max	5	27550	28727	28727	28727	28727
mean	4.306747784641819	2.0379774451473938	3.2418290675886614	3.2418290675886614	3.2418290675886614	3.2418290675886614
stddev	1.5654661763964374E7	1.5654661763964374E7	1.5654661763964374E7	1.5654661763964374E7	1.5654661763964374E7	1.5654661763964374E7
1	US	10001	R10000WAG70YS2	0001046314	225472	225472
25%	US	15005223	0	3.45803485E8	249510863	249510863
50%	US	28788910	0	6.34049011E8	496520461	496520461
75%	US	44131558	0	1.419553429E9	744807844	744807844
max	US	53096589	RZZZZKQ8A3VTL	B06XXNYHRQ	999988076	999988076

As we see that, there are some emotes/characters we don't need. Therefore to ignore these character (ascii characters), I am using the user-defined function(udf) below.

```
>>> def ascii_only(mystring):
...     if mystring:
...         return mystring.encode('ascii', 'ignore').decode('ascii')
...     else:
...         return None
...
>>> ascii_udf = udf(ascii_only)
>>> sdf= sdf.withColumn("clean_review_headline", ascii_udf('review_headline'))
>>> sdf= sdf.withColumn("clean_review_body", ascii_udf('review_body'))
>>> sdf.select("clean_review_headline", "clean_review_body").summary("count", "min", "max").show()
```

summary	clean_review_headline	clean_review_body
count	6930471	6930482
min		
max	~~~~~ A MOVIE F...	~~~~~...