Neha Patel

15 Followers About Follow

Recommendation system in python using ALS algorithm and Apache Spark

Neha Patel Apr 7, 2019 · 4 min read

Recommendation system has become integral part of many online platforms given it's use in increasing sales and customer retention.Amazon, Netflix, Facebook, Linkedin and many more make use of recommendation system on their platform to recommend products,movies,people,posts respectively to users.Ever wondered how these companies provides recommendations.

Given it's popularity there are three main techniques used for providing recommendations online-Collaborative filtering,Content-based and Hybrid technique.Here we will be making use of Alternating least square matrix factorization method,a collaborative filtering algorithm.

Jupyter Notebook.You can make this simple recommendation model as a mini project in college or as a personal project.

Requirements:

Your machine should have latest version of Python,Apache Spark and Jupyter Notebook installed. You also need to connect pyspark with Jupyter Notebook and many tutorials are available out there to do the same.

Dataset used is downloaded from the link given below:

> **Amazon review data**
> This dataset contains product reviews and metadata from Amazon, including 142.8 million reviews spanning May 1996 ...
>
> jmcauley.ucsd.edu

The data used here is of Musical Instruments for training the model.

Terminologies:

There are certain terminologies which needs to be understood before moving forward.

1. **Apache Spark:** Apache Spark is an open-source distributed general-purpose cluster-computing framework.It can be used with Hadoop too.

2. **Collaborative filtering:** Collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users. Consider example if a person A likes item 1, 2, 3 and B like 2,3,4 then they have similar interests and A should like item 4 and B should like item 1.

3. **Alternating least square(ALS) matrix factorization:** The idea is basically to take a large (or potentially huge) matrix and factor it into some smaller representation of the original matrix through alternating least squares. We end up with two or more lower dimensional matrices whose product equals the original one.ALS comes inbuilt in Apache Spark.

Get started Open in app

So let's start making our recommendation model in jupyter notebook.

1.Initialize spark session:

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('Recommendation_system')
.getOrCreate()
```

2. Load Dataset in Apache Spark

```
df = spark.read.json("Musical_Instruments_5.json")
df.show(100,truncate=True)
```

Your dataframe will look like following:

```
+----------+--------+-------+--------------------+-----------+--------------+--------------------+--------------------+-------
-------+
|      asin| helpful|overall|          reviewText| reviewTime|    reviewerID|        reviewerName|             summary|unixRev
iewTime|
+----------+--------+-------+--------------------+-----------+--------------+--------------------+--------------------+-------
-------+
|1384719342|  [0, 0]|    5.0|Not much to write...|02 28, 2014|A2IBPI20UZIR0U|cassandra tu "Yea...|                good|    139
3545600|
|1384719342|[13, 14]|    5.0|The product does ...|03 16, 2013|A14VAT5EAX3D9S|                Jake|                Jake|    136
3392000|
|1384719342|  [1, 1]|    5.0|The primary job o...|08 28, 2013|A195EZSQDW3E21|Rick Bennette "Ri...|It Does The Job Well|    137
7648000|
|1384719342|  [0, 0]|    5.0|Nice windscreen p...|02 14, 2014|A2C00NNG1ZQQG2|RustyBill "Sunday...|GOOD WINDSCREEN F...|    139
2336000|
|1384719342|  [0, 0]|    5.0|This pop filter i...|02 21, 2014| A94QU4C90B1AX|       SEAN MASLANKA|No more pops when...|    139
2940800|
|B00004Y2UT|  [0, 0]|    5.0|So good that I bo...|12 21, 2012|A2A039TZMZHH9Y| Bill Lewey "blewey"|      The Best Cable|    135
6048000|
|B00004Y2UT|  [0, 0]|    5.0|I have used monst...|01 19, 2014|A1UPZM995ZAH90|               Brian|Monster Standard ...|    139
0089600|
|B00004Y2UT|  [0, 0]|    3.0|I now use this ca...|11 16, 2012| AJNFQI3YR6XJ5|  Fender Guy "Rick"|Didn't fit my 199...|    135
3024000|
```

Original dataframe

3. Select appropriate columns

```
nd=df.select(df['asin'],df['overall'],df['reviewerID'])
nd.show()
```

We do not need all the columns present in the dataframe .Only asin which is

```
+----------+-------+--------------+
|      asin|overall|    reviewerID|
+----------+-------+--------------+
|1384719342|    5.0|A2IBPI20UZIR0U|
|1384719342|    5.0|A14VAT5EAX3D9S|
|1384719342|    5.0|A195EZSQDW3E21|
|1384719342|    5.0|A2C00NNG1ZQQG2|
|1384719342|    5.0| A94QU4C90B1AX|
|B00004Y2UT|    5.0|A2A039TZMZHH9Y|
|B00004Y2UT|    5.0|A1UPZM995ZAH90|
|B00004Y2UT|    3.0| AJNFQI3YR6XJ5|
|B00004Y2UT|    5.0|A3M1PLEYNDEYO8|
|B00004Y2UT|    5.0| AMNTZU1YQN1TH|
|B00004Y2UT|    5.0|A2NYK9KWFMJV4Y|
|B00005ML71|    4.0|A35QFQI0M46LWO|
|B00005ML71|    3.0|A2NIT6BKW11XJQ|
|B00005ML71|    5.0|A1C0O09LOLVI39|
|B00005ML71|    5.0|A17SLR18TUMULM|
|B00005ML71|    2.0|A2PD27UKAD3Q00|
|B000068NSX|    4.0| AKSFZ4G1AXYFC|
|B000068NSX|    5.0| A67OJZLHBBUQ9|
|B000068NSX|    5.0|A2EZWZ8MBEDOLN|
|B000068NSX|    5.0|A1CL807EOUPVP1|
+----------+-------+--------------+
only showing top 20 rows
```

4. Importing important modules

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

5. Converting String to index

Before making an ALS model it needs to be clear that ALS only accepts integer value as parameters. Hence we need to convert asin and reviewerID column in index form.

```
from pyspark.ml.feature import StringIndexer
from pyspark.ml import Pipeline
from pyspark.sql.functions import col

indexer = [StringIndexer(inputCol=column,
outputCol=column+"_index") for column in list(set(nd.columns)-
set(['overall'])) ]
```

```
transformed.show()
```

asin	overall	reviewerID	reviewerID_index	asin_index
1384719342	5.0	A2IBPI20UZIR0U	72.0	781.0
1384719342	5.0	A14VAT5EAX3D9S	359.0	781.0
1384719342	5.0	A195EZSQDW3E21	436.0	781.0
1384719342	5.0	A2C00NNG1ZQQG2	1216.0	781.0
1384719342	5.0	A94QU4C90B1AX	1137.0	781.0
B00004Y2UT	5.0	A2A039TZMZHH9Y	54.0	629.0
B00004Y2UT	5.0	A1UPZM995ZAH90	348.0	629.0
B00004Y2UT	3.0	AJNFQI3YR6XJ5	324.0	629.0
B00004Y2UT	5.0	A3M1PLEYNDEYO8	12.0	629.0
B00004Y2UT	5.0	AMNTZU1YQN1TH	185.0	629.0
B00004Y2UT	5.0	A2NYK9KWFMJV4Y	4.0	629.0
B00005ML71	4.0	A35QFQI0M46LWO	425.0	870.0
B00005ML71	3.0	A2NIT6BKW11XJQ	652.0	870.0
B00005ML71	5.0	A1C0O09LOLVI39	55.0	870.0
B00005ML71	5.0	A17SLR18TUMULM	651.0	870.0
B00005ML71	2.0	A2PD27UKAD3Q00	290.0	870.0
B000068NSX	4.0	AKSFZ4G1AXYFC	93.0	538.0
B000068NSX	5.0	A67OJZLHBBUQ9	258.0	538.0
B000068NSX	5.0	A2EZWZ8MBEDOLN	3.0	538.0
B000068NSX	5.0	A1CL807EOUPVP1	31.0	538.0

only showing top 20 rows

String to index conversion

6. Creating training and test data

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

7. Creating ALS model and fitting data

```
als=ALS(maxIter=5,regParam=0.09,rank=25,userCol="reviewerID_index
",itemCol="asin_index",ratingCol="overall",coldStartStrategy="dro
p",nonnegative=True)

model=als.fit(training)
```

8. Generate predictions and evaluate rmse

```
evaluator=RegressionEvaluator(metricName="rmse",labelCol="overall
",predictionCol="prediction")

predictions=model.transform(test)
rmse=evaluator.evaluate(predictions)

print("RMSE="+str(rmse))
predictions.show()
```

RMSE=1.168435412679992

asin	overall	reviewerID	reviewerID_index	asin_index	prediction
B000CCJP4I	5.0	AWYXB9L41T82S	858.0	148.0	3.0042644
B000CCJP4I	5.0	A3J8U952XAL34Z	502.0	148.0	3.715278
B000CCJP4I	3.0	A1YR3RVSBZK8CW	30.0	148.0	4.1241045
B000KIPTE4	4.0	A3F49ZMUC1GSRP	1165.0	463.0	3.7388468
B000KIPTE4	5.0	A2EZWZ8MBEDOLN	3.0	463.0	4.2314043
B000KIPTE4	5.0	A2AH7HRHDTQENH	399.0	463.0	4.4930964
B001OLZYUU	3.0	AXXYMIJBD0J9G	530.0	471.0	3.5948217
B001OLZYUU	5.0	A2IBPI20UZIR0U	72.0	471.0	5.033518
B0002GZ052	5.0	A26HM2R5529NYY	822.0	496.0	4.2211905
B0002GZ052	5.0	A1T4U9CAQ25IBR	750.0	496.0	4.59568
B000BKY8CU	4.0	A3PGQWCSJPCYDH	64.0	243.0	3.317696
B000RPUMII	5.0	A223S6N0DBQBHP	236.0	392.0	3.5661318
B000RPUMII	1.0	ANAKK5KNUAP17	663.0	392.0	4.1016426
B000RPUMII	5.0	A3OJ0RGAECAGH8	381.0	392.0	2.6469836
B000KUCQXY	4.0	A2Q6KC2KU2T0OL	1315.0	540.0	3.1362884
B000KUCQXY	5.0	A2O8BAXJPDSV0M	365.0	540.0	4.13888
B001GGYF4E	4.0	A3S737ZGWE1GKY	1346.0	623.0	3.714615
B001GGYF4E	3.0	A8DCZN408QYKC	953.0	623.0	3.6110063
B009EOKTCM	5.0	A1SD1C8XK3Z3V1	6.0	858.0	4.6142297
B0002GLCRC	5.0	A24VCDADYAIHAM	481.0	31.0	4.6885085

only showing top 20 rows

Predictions for test data

9. **Providing Recommendations**

```
user_recs=model.recommendForAllUsers(20).show(10)
```

```
|reviewerID_index|     recommendations|
+----------------+--------------------+
|             471|[[746, 5.981454],...|
|            1342|[[412, 6.348332],...|
|             463|[[898, 6.0316505]...|
|             833|[[426, 6.379773],...|
|             496|[[426, 5.978744],...|
|             148|[[898, 6.291563],...|
|            1088|[[426, 4.7776713]...|
|            1238|[[579, 5.840673],...|
|             540|[[898, 4.944048],...|
|             392|[[710, 5.531966],...|
+----------------+--------------------+
only showing top 10 rows
```

Recommendations

10. Converting back to string form

As seen in above image the results are in integer form we need to convert it back to its original name.The code is little bit longer given so many conversions.

```
import pandas as pd
recs=model.recommendForAllUsers(10).toPandas()
nrecs=recs.recommendations.apply(pd.Series) \
            .merge(recs, right_index = True, left_index = True) \
            .drop(["recommendations"], axis = 1) \
            .melt(id_vars = ['reviewerID_index'], value_name = 
"recommendation") \
            .drop("variable", axis = 1) \
            .dropna()
nrecs=nrecs.sort_values('reviewerID_index')
nrecs=pd.concat([nrecs['recommendation'].apply(pd.Series), 
nrecs['reviewerID_index']], axis = 1)
nrecs.columns = [

        'ProductID_index',
        'Rating',
        'UserID_index'

     ]
md=transformed.select(transformed['reviewerID'],transformed['revi
ewerID_index'],transformed['asin'],transformed['asin_index'])
md=md.toPandas()
dict1 =dict(zip(md['reviewerID_index'],md['reviewerID']))
dict2=dict(zip(md['asin_index'],md['asin']))
nrecs['reviewerID']=nrecs['UserID_index'].map(dict1)
nrecs['asin']=nrecs['ProductID_index'].map(dict2)
```

```
new['recommendations'] = list(zip(new.asin, new.Rating))
res=new[['reviewerID','recommendations']]
res_new=res['recommendations'].groupby([res.reviewerID]).apply(li
st).reset_index()

print(res_new)
```

```
               reviewerID                                    recommendations
0    A00625243BI8W1SSZNLMD  [(B002MWKOAA, 6.359242916107178), (B0002GWXKC,...
1           A10044ECXDUVKS  [(B000T9PE9E, 5.138420104980469), (B002MWKOAA,...
2           A102MU6ZC9H1N6  [(B009E3EWPI, 5.896670818328857), (B0009RLE5Y,...
3           A109JTUZXO61UY  [(B001C9R5P6, 5.873091697692871), (B0009RLE5Y,...
4           A109ME7C09HM2M  [(B003LZ2IT2, 5.653217315673828), (B000RY68PA,...
5           A10APIDAZISWQF  [(B0002GIRP2, 5.017032146453857), (B0002E4Z8M,...
6           A10B2J2IRQXBWA  [(B000RWJQRE, 5.008950233459473), (B000RKAFIU,...
7           A10E3QH2FQUBLF  [(B000W00X1Y, 4.837876319885254), (B000SZVYLQ,...
8           A10FM4ILBIMJJ7  [(B0009IEB0I, 5.778720378875732), (B001V5K2S8,...
9           A10H2F00ZOT8S2  [(B009E3EWPI, 6.008238792419434), (B0009RLE5Y,...
10          A10HYGDU2NITYQ  [(B00095VIMU, 4.6403045654296875), (B0002DV7ZM...
11          A10KH8EN77ZKWH  [(B0002GWXKC, 5.347064018249512), (B0002GIRP2,...
12          A10N243R7A5ZW3  [(B000SZVYLQ, 5.4041852951049805), (B00063678K...
13          A10NJEIG56RHN5  [(B0009IEB0I, 5.517179489135742), (B0002GIRP2,...
14          A10VG94SAKVSC0  [(B0002CZVHI, 4.81657075881958), (B000GUR8V8, ...
15          A10ZSXTQA264C7  [(B0000AQRST, 5.012159824371338), (B004OK1G64,...
16          A110ZEDSNASVCO  [(B0002GZQ1U, 5.608520030975342), (B001V5K2S8,...
17          A118PM0B1PGWDA  [(B0001FTVD6, 3.4961419105529785), (B001RMFSDE...
18          A11E4FWMN9BXJD  [(B0002E1O7M, 4.04860258102417), (B0006H92QK, ...
19          A11INIL2YFJ137  [(B001C9R5P6, 5.950793743133545), (B0009RLE5Y,...
20          A120FZ2ESIMA63  [(B0002GZQ1U, 5.798271656036377), (B001C9R5P6,...
21          A121QRWXZIO6UP  [(B0009RLE5Y, 5.578435897827148), (B000SZVYLQ,...
22          A126XEMCLHPBNZ  [(B001C9R5P6, 5.322956085205078), (B00CK2FOZM,...
23          A127K5WGHNUUH3  [(B001V5K2S8, 5.666537284851074), (B00CK2FOZM,...
24          A12ABV9NU02O29  [(B0006H92QK, 5.467516899108887), (B002MWKOAA,...
25          A12DQZKRKTNF5E  [(B0009RLE5Y, 5.362525939941406), (B000EPVXWU,...
```

Final Result

Data Science Recommendation System Apache Spark Python

Voila!!

You just made an recommendation application.