4/17/2022

Recommend Movie Built on IMDb Review

Internet of Things (ISTM 6217)

Group 11 - Haoran Zhang, Jingru Xu, Jason Wyman

Table of Contents

1. Executive Summary

Reviews are a very important way to gain insight into a product or service. In machine learning tasks, text reviews play an important role in predicting or gaining insights. IMDb is one of the largest databases for films and TV shows. It is the place where people provide their valuable opinions on the world. This project utilized logistic regression, ALS regression, as well as content-based collaborative analysis based on the IMDb Review database. This research is aiming to build models to recommend movies to users, recommend users to specific movies, recommend movies using keywords (and selected specific movies), to predict the rating solely based on the reviews. It also enables similarity pattern matching, which will help the filmmakers advertise their movies. This research has successfully built an over 70% accuracy logistics regression model to predict the ratings from a review detail. In addition, the research result is able to recommend movies based on keywords like “magic, superhero”.

For each movie, the result is able to recommend the five best-matched users to it, it can also recommend the five best-matched movies for each user. The matching score will be listed for references. All the models are visualized with the SNS Seaborn graph, so the accuracy of prediction (from comments to rating) is comparable over different ratings. The research managed to tokenize the comments, remove the stop-words and apply the TF-IDF structure to the texts, so not only the frequency of the word is considered, but also the relative frequency in the whole document is included, too. For users with repeated comments on the same movie, this study removes them as duplicate values to ensure consistency. This research will aid in the recommendation algorithm of IMDb by providing a quicker, less costly, and more comprehensive model, which will improve the overall recommendation accuracy and relevance.

2. Data Description

a. Data Source

The data is from Kaggle, a subsidiary of Google LLC, which is an online community of data scientists and machine learning practitioners. The total data size is over 8 GB and it is separated into 6 parts. In this study, we mainly use the part_01.json which is about 1.2 GB.

b. Variable Description

Content	**Details**
review_id	It is generated by IMDb and unique to each review
reviewer	Public identity or username of the reviewer
movie	It represents the name of the show (can be - movie, tv-series, etc.)
rating	Rating of the movie out of 10, can be None for older reviews
review_summary	Plain summary of the review
review_date	Date of the posted review
spoiler_tag	If 1 = spoiler & 0 = not spoiler
review_detail	Details of the review
helpful	list[0] people find the review helpful out of list[1]

c. Sample Size (n) and Number of Variables (k)

There are over 10,000 reviews in the part_01.json, and the number of variables is 9.

d. Sample Observation

helpful	movie	rating	review_date	review_detail	review_id	review_summary	reviewer	spoiler_tag
[0, 0]	Destiny 2 (2017 V...	1	28 October 2020	!!!Before play th...	rw6213561	Careless to the p...	parcabral	0
[0, 6]	Tim's Vermeer (2013)	10	6 March 2014	" . . . My Master...	rw2974978	"When I Paint	cricket30	1
[0, 1]	Fatty's Faithful ...	7	16 January 2014	" . . . and even ...	rw2943175	The originator of...	cricket30	1
[0, 0]	The Gay Divorcee ...	7	7 December 2013	" . . . if you ca...	rw2918405	"Be feminine and ...	cricket30	1
[2, 5]	Racket Busters (1...	8	17 October 2019	" . . . the Mob, ...	rw5192921	U.S. voters alway...	tadpole-596-918256	1

In the first column “helpful”. Row 1 has a helpful value of [0,0], which means that 0 people think this review is helpful and total attitudes are 0. However, row 2 has a value of [0,6]. That is to say, in all six attitudes, no one finds row 2 helpful. We will treat it as a vector value and calculate the helpfulness by using helpful[0]/helpful[1], and if the helpful[1] is zero, the helpfulness will be zero too.

In the second column “movie”. It has the movie’s name that the reviewer is reviewing. For row 3, the movie name is Fatty’s Faithful … (the rest are omitted due to screen size). This study will use it as a string value, and index it to be a “movie_id” to distinguish all the movies.

In the third column “rating”. It has the rating that the reviewer gives. We will use the cast() method to change these string values into integers so we can numerically analyze this column.

In the fourth column “review_detail”. It has detailed texts that the reviewer gives about the movie.

In the fifth column “review_id”. It is the unique string value id for all the reviews. All reviews have different review_id, even though the review texts are identical. This study will use the cast() method to transform it into integer values so we can use the ALS model to distinguish the movies.

In the sixth column “review_summary”. It provides an overview of the review for the reader’s convenience. This study will not use it as our input vector because the texts in the “review_summary” is too simplified that can’t provide enough insights about the reviewer's preferences.

In the seventh column “reviewer”. It is the unique string value if for all the reviewers. All reviewers have different reviewer_id. This study will use the cast() method to transform it into integer values so we can use the ALS model to distinguish the movies.
In the eighth column “spoiler_tag”. It represents whether this reviewer is a spoiler. If the reviewer is a spoiler, the value will be 1. For example, rows 2-5 are considered spoilers.

e. Data Link

Link of dataset:

https://www.kaggle.com/datasets/ebiswas/imdb-review-dataset?datasetId=1092024

3. Research Questions

Q1: Can we predict the rating solely based on the reviews?

Q2: Can we recommend the top 5 (or more) movies (users) to users (movies)?

Q3: Can we recommend the best-matched movies list based on keywords like "Superhero and Magic" or one particular movie?

4. Methodology

a. Data Mining Techniques

In order to predict the rating, this study utilized logistics regression as well as ALS regression.

Alternating Least Square (ALS) is a matrix factorization algorithm and it runs itself in a parallel fashion. It is implemented in Apache Spark ML and built for large-scale collaborative filtering problems. ALS is very popular at solving the scalability and sparseness of the Rating data, and it's simple and scales well to very large datasets. The ALS fit into our datasets very well.

Logistic regression is easier to implement, interpret, and very efficient to train. It makes no assumptions about distributions of classes in feature space. The logistics regression is strong running on the TF-IDF of the review token and it generates over 70% of accuracy with 1.17 RMSE.

5. Results and Findings

a. Question 1

```
# Build Logistic Regression Model
from pyspark.ml.classification import LogisticRegression
log_reg = LogisticRegression(featuresCol='features', labelCol='rating')
logr_model = log_reg.fit(training)
results = logr_model.transform(test)
results.select('review_detail','rating','prediction').show()
```

```
+--------------------+------+----------+
|       review_detail|rating|prediction|
+--------------------+------+----------+
|You got rid of EV...|     4|       4.0|
|Movie shows frien...|    10|      10.0|
|This is just a ba...|     6|       6.0|
|13 Hours, a movie...|     9|       9.0|
|One of the better...|     9|       9.0|
|The plot itself w...|     6|       6.0|
|During my Lasik s...|     9|       9.0|
|In high school, t...|     8|       8.0|
```

This study has successfully built a logistics regression model to predict the rating of the user. Overall the prediction is pretty accurate, to visualize our prediction, we draw out the confusion matrix, too.

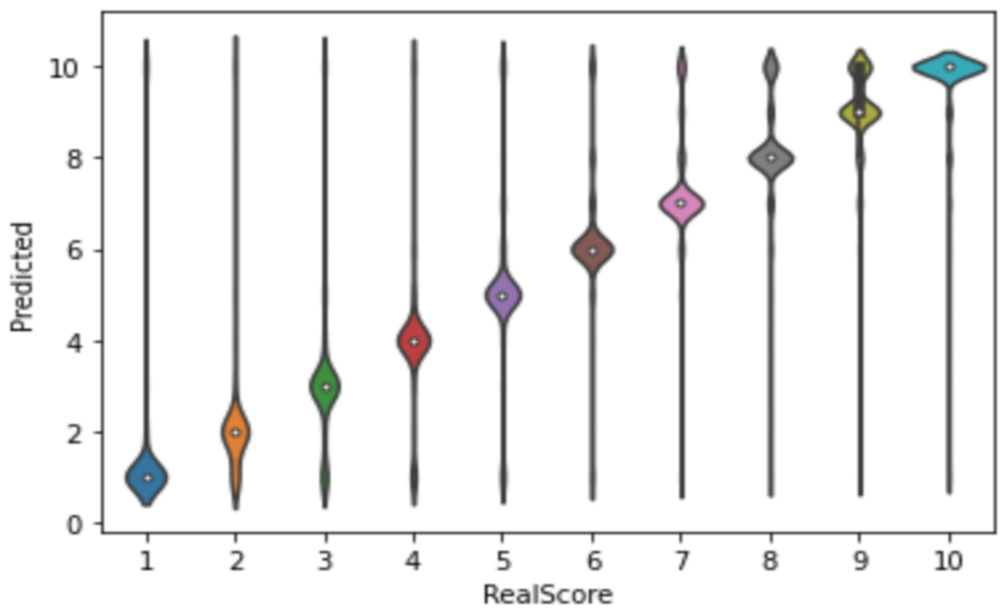
```
# confusion Matrix
from sklearn.metrics import confusion_matrix
y_true = results.select("rating")
y_true = y_true.toPandas()

y_pred = results.select("prediction")
y_pred = y_pred.toPandas()

cnf_matrix = confusion_matrix(y_true, y_pred)
print(cnf_matrix)
```

```
[[17857   758   549   359   358   261   188   186   149   951]
 [ 1630  4953   354   262   258   158   126   114    69   330]
 [ 1216   353  5387   273   372   244   204   142    79   351]
 [  838   265   330  5759   398   334   243   173   115   395]
 [  728   288   341   373  7729   601   504   344   185   618]
 [  487   209   275   351   597  9839   978   636   340   884]
 [  400   137   210   247   505   906 13136  1476   726  1987]
 [  364   136   146   232   371   766  1445 14833  1410  4469]
 [  294   101   112   160   235   389   803  1526 11847  6521]
 [  668   170   205   186   281   431   937  1874  2242 42302]]
```

From the confusion matrix, we can see that the shape of the matrix is 10*10. It represents a rating from 1 to 10. The correct predictions numbers sum are in the diagonal of the confusion matrix. To see how the model performs across different ratings, we use the sns package from seaborn. For different actual ratings, the model accurately predicts the real ratings. Overall, the accuracy of the model is 0.7077221263119989, and the RMSE value of 1.75.




```
<AxesSubplot:xlabel='RealScore', ylabel='Predicted'>
```

```
from pyspark.ml.evaluation import RegressionEvaluator
evaluator = RegressionEvaluator(metricName="rmse", labelCol="rating",
                                predictionCol="prediction")
rmse = evaluator.evaluate(results)
print("Root-mean-square error = " + str(rmse))
```

```
Root-mean-square error = 1.7534409270761768
```

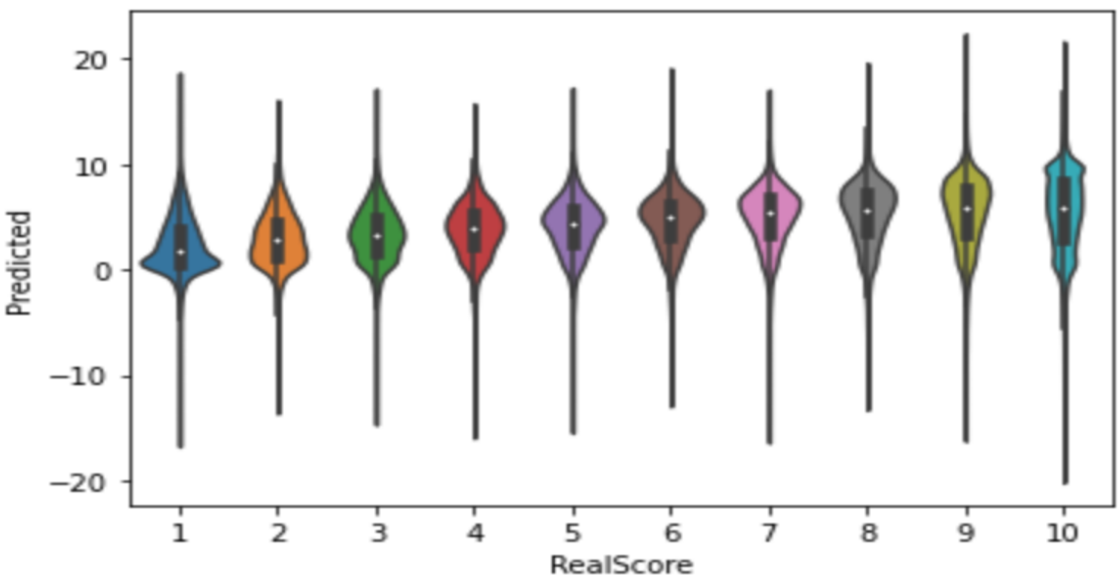
b. Question 2

Compared to the logistics regression, the ALS regression has a slightly higher RMSE value which is 4.15.

```
evaluator = RegressionEvaluator(metricName="rmse", labelCol="rating",
                                predictionCol="prediction")
rmse = evaluator.evaluate(predictions)
print("Root-mean-square error = " + str(rmse))
```

```
Root-mean-square error = 4.146000199820348
```

We visualize the prediction by SNS seaborn, too.




```
from pyspark.sql.functions import round
predictions.select("reviewer","rating",'
```

```
+---------------+-------+-----------+
|       reviewer|rating|prediction|
+---------------+-------+-----------+
|  treakle_1978|      6| 6.3247538|
|   ejlif-89392|      3| 5.0283165|
|        rmarkd|      7|  8.388969|
|gwnightscream|      6| 4.7466173|
|     legonerdy|      7| 7.2181487|
|    ompreetdas|     10| 4.8024955|
|TheEthosDiary|      1|  2.695867|
|         zetes|      8| 4.4966145|
|    leemeldrum|      6|  5.632532|
|      fmwongmd|      6|  5.259736|
+---------------+-------+-----------+
```

Though not as good as logistic regression, the ALS model is still doing reliable work in predicting the ratings across the different actual scores. See above to compare the predictions with the real results.

This study has successfully recommended the top 5 movies (users) to users (movies) with similarities listed.

```
# Generate top 5 movie recommendations for each user
userRecs = model.recommendForAllUsers(5)
userRecs.show(truncate=False)
```

```
/databricks/spark/python/pyspark/sql/context.py:134: FutureWarning: Deprecated in 3.0.0. Use SparkSession.builder.getOr
  warnings.warn(
+-----------+--------------------------------------------------------------------------------------------------------+
|reviewer_id|recommendations                                                                                         |
+-----------+--------------------------------------------------------------------------------------------------------+
|31         |[{51879, 14.2137165}, {28438, 13.946489}, {124930, 13.752805}, {37604, 13.561127}, {119488, 13.377673}]|
|34         |[{22500, 9.552204}, {49536, 9.23136}, {12118, 9.132296}, {55218, 9.062408}, {60056, 9.017928}]         |
|53         |[{43482, 14.610702}, {56981, 12.968922}, {55033, 12.897483}, {57596, 12.858547}, {92770, 12.832756}]   |
|65         |[{32630, 16.28215}, {99354, 15.571305}, {134541, 15.4662695}, {31752, 15.236786}, {20475, 15.02924}]   |
|78         |[{31752, 13.408895}, {99354, 12.682316}, {68488, 12.331946}, {28932, 12.304062}, {138114, 12.268056}]  |
|85         |[{118174, 18.522505}, {32630, 17.949858}, {72230, 17.89101}, {18521, 17.67452}, {10343, 16.592459}]    |
|108        |[{32280, 15.996279}, {32630, 15.714592}, {22679, 15.098007}, {134541, 15.029295}, {71430, 14.75822}]   |
|133        |[{20475, 15.5128975}, {58675, 15.308478}, {31752, 15.201431}, {32630, 15.019698}, {54975, 14.650212}]  |
|137        |[{52549, 6.1451263}, {135578, 6.1430054}, {56784, 5.7177863}, {125561, 5.6911697}, {71430, 5.6116605}] |
|148        |[{32630, 16.158834}, {24676, 16.002758}, {56320, 15.534097}, {48370, 15.150261}, {31752, 15.122065}]   |
|155        |[{32630, 14.68884}, {99118, 14.416712}, {58675, 14.349509}, {71430, 14.075901}, {134541, 14.006577}]   |
|193        |[{55033, 15.3821}, {99399, 15.006073}, {138114, 14.622332}, {100419, 14.597843}, {56320, 14.497939}]   |
|211        |[{23192, 14.781394}, {57272, 14.554753}, {11633, 14.230761}, {40087, 14.199314}, {133295, 14.1793}]    |
|243        |[{58675, 22.146229}, {31752, 20.7259}, {71430, 20.196457}, {97719, 19.494766}, {35374, 19.475224}]     |
|251        |[{59739, 17.821718}, {48370, 17.197374}, {59773, 15.727671}, {116375, 15.469708}, {126991, 15.371632}] |
|255        |[{32630, 13.306288}, {31752, 12.81796}, {99354, 12.798188}, {56422, 12.54983}, {35374, 12.545101}]     |
```

```
# Generate top 5 user recommendations for each movie
movieRecs = model.recommendForAllItems(5)
movieRecs.show(truncate=False)
```

```
+--------+-----------------------------------------------------------------------------------------------------------+
|movie_id|recommendations                                                                                            |
+--------+-----------------------------------------------------------------------------------------------------------+
|31      |[{37898, 11.833333}, {32740, 11.115079}, {13334, 10.883379}, {95717, 10.878005}, {58878, 10.878005}]      |
|34      |[{58922, 14.044959}, {71956, 13.639737}, {48104, 13.556645}, {13085, 13.03353}, {8053, 12.702527}]        |
|53      |[{25266, 19.968468}, {48104, 16.701607}, {85149, 16.569502}, {22235, 16.283482}, {58922, 15.820237}]      |
|65      |[{48104, 12.33483}, {76407, 12.188332}, {6913, 11.934265}, {12475, 11.601177}, {53351, 11.306722}]        |
|78      |[{74685, 11.944761}, {76407, 11.803977}, {48104, 11.678318}, {44780, 11.416073}, {3636, 11.399761}]       |
|85      |[{50694, 13.106741}, {17652, 10.447769}, {53420, 10.150984}, {53108, 10.150984}, {7537, 10.02775}]        |
|108     |[{48104, 12.032226}, {283393, 11.996016}, {279462, 11.996016}, {179641, 11.996016}, {167016, 11.996016}]|
|133     |[{13085, 17.516663}, {16412, 16.498945}, {22235, 16.077477}, {45065, 14.908003}, {71956, 14.881508}]      |
|137     |[{48104, 18.318403}, {22235, 17.669733}, {76407, 17.113052}, {25266, 16.898746}, {58922, 16.606459}]      |
|148     |[{76407, 19.750208}, {48104, 19.199781}, {45065, 19.038744}, {25266, 18.814602}, {13085, 18.72463}]       |
|155     |[{48104, 13.652329}, {76407, 12.724522}, {10199, 12.667502}, {62957, 12.523996}, {9969, 12.467998}]       |
|193     |[{11512, 12.28585}, {59240, 11.871207}, {50694, 11.675144}, {14764, 10.942196}, {79690, 10.795575}]       |
|211     |[{36375, 15.117417}, {48104, 14.2475195}, {90884, 14.107857}, {62957, 14.042645}, {58922, 13.753921}]     |
|243     |[{19654, 17.965366}, {25254, 17.325216}, {27757, 16.032671}, {14317, 15.587824}, {13345, 14.572837}]      |
|251     |[{22235, 17.023344}, {48104, 16.59499}, {58922, 15.515896}, {26498, 15.128635}, {54709, 15.128635}]       |
|255     |[{84710, 15.327112}, {89493, 14.981108}, {8500, 14.517617}, {30553, 13.940929}, {56688, 13.618892}]       |
|296     |[{48104, 14.809673}, {9465, 12.775314}, {62957, 12.600001}, {50139, 12.560595}, {9595, 12.551518}]        |
|321     |[{42563, 13.608915}, {61074, 13.1800995}, {76407, 12.812789}, {8500, 12.693701}, {84710, 12.679026}]      |
```

The model can also recommend users based on a specific movie by id.

```
# Generate top 5 user recommendations for a specified set of movies defined by you
from pyspark.sql.types import IntegerType
from pyspark.sql.functions import col
df = spark.createDataFrame([1, 2, 3], IntegerType())
df = df.select(col("value").alias("movie_id"))
df.show()

movieSubSetRecs = model.recommendForItemSubset(df, 5)
movieSubSetRecs.show(truncate=False)

+--------+
|movie_id|
+--------+
|       1|
|       2|
|       3|
+--------+

+--------+----------------------------------------------------------------------------------------------+
|movie_id|recommendations                                                                               |
+--------+----------------------------------------------------------------------------------------------+
|1       |[{48104, 13.30162}, {58922, 12.653847}, {71956, 12.31076}, {22235, 12.020835}, {37898, 12.015402}] |
|3       |[{25266, 12.004211}, {58922, 10.5654745}, {36457, 10.515282}, {48104, 10.498585}, {71257, 10.4135}]|
|2       |[{58171, 10.565419}, {22235, 10.447314}, {25266, 10.17498}, {98005, 10.047649}, {56233, 10.000598}]|
+--------+----------------------------------------------------------------------------------------------+
```

c. Question 3

With the word2vec package, this study successfully build the model to enable a phrase similarity test. Below is a test for the "great" word and the top 5 similar words are listed.

```
#test similarity between words
synonyms = model.findSynonyms("great", 5)
synonyms.show(5)

/databricks/spark/python/pyspark/sql/context
  warnings.warn(
+---------+------------------+
|     word|        similarity|
+---------+------------------+
|excellent|0.8339200019836426|
|fantastic|0.8228392601013184|
|  amazing|0.7955614328384399|
| terrific|0.7669351100921631|
|     good|0.7557264566421509|
+---------+------------------+
```

With this model, we can proceed to provide a similar movies list based on a review of a particular movie:

```
similarity = reviews_w2v.select('movie','review_id', cossim_udf('result').alias("similarity"), 'review_detail')
similarity = similarity.orderBy("similarity", ascending = False)
display(similarity)
```

	movie	review_id	similarity	review_detail
1	Scam 1992: The Harshad Mehta Story (2020)	rw6205775	1	! A perfect movie Great story , great legend , inspiring Maja aagya bhi dekh k ki bha inspiring h
2	Dil Bechara (2020)	rw5931473	0.9248063	I am crying Dude. Ye Apne accha nhi kiya avi bohot kuch dekhna baki tha what a bri
3	Coolie No. 1 (2020)	rw6409955	0.9186001	Itni ghatiya movie kabi ni deki, ek star k bi layak nahi h ye , uss s nichee khuc hota t
4	Laxmii (2020)	rw6255298	0.91178596	Iss Diwali Garib k Ghar Diya Jalao. 🔥🔱 Bollywood K Diya Bujhao.📛 #BoycottLaxmiE
5	Coolie No. 1 (2020)	rw6397741	0.91178155	Comedy krne me v acting ki zarurta hoti h syd in ko koi batna bhul gya ....ghar ki hi
6	Laxmii (2020)	rw6250439	0.9116229	Is movie me Musalman aur uske culture ko jabardasti acha dikhane ki koshis ki ja rh vahiyat chije h unko koi samne nhi lata... Baki movie me to koi dum nhi h... 1 commu ka Propaganda chala rhe h bc... Log
7	Laxmii (2020)	rw6253501	0.91120076	Agar aapki zindagi bhot achi chal ri h to ye movie zaroor dekhen kuki Kabhi Kabhi m

Truncated results, showing first 1000 rows

Some movies will repeat in the results, to have a distinct movie view, this study utilized the SQL GROUP BY to rule out duplicate movies so only one movie and similarity are displayed.

```
%sql
SELECT movie, MAX(similarity) FROM similarity_table GROUP BY movie ORDER BY MAX(similarity) DESC
```

	movie	max(similarity)
1	Scam 1992: The Harshad Mehta Story (2020)	1
2	Dil Bechara (2020)	0.9248063
3	Coolie No. 1 (2020)	0.9186001
4	Laxmii (2020)	0.91178596
5	Mirzapur (2018–)	0.90740514
6	Kaalchakra (2016)	0.9073137

Truncated results, showing first 1000 rows

We also successfully build the model to recommend the movie based on keywords like “Superhero” and “Magic”. Again, we rule out the duplicated with SQL, too.

```
# Recommend Movie based on Keyword
key_word = "superhero"
docvecs = reviews_w2v
x = spark.createDataFrame([('newreviewid', key_word)]).\
    withColumnRenamed('_1', 'review_id').\
    withColumnRenamed('_2', 'review_detail')
```

```
similarity2 = reviews_w2v.select('movie','review_id', cossim_udf('result').alias("similarity"), 'review_detail')
similarity2 = similarity2.orderBy("similarity", ascending = False)
display(similarity2)
```

	movie	review_id	similarity	review_detail
				and Supervillains.
2	Captain America: The Winter Soldier (2014)	rw6072377	0.795051	It is one of the super mov
3	Smallville (2001–2011)	rw5939647	0.78102165	Really good, recommend
4	Wonder Woman 1984 (2020)	rw6402606	0.77668035	Superhero is one thing, bu
5	小丑 (2019)	rw5169617	0.77379185	It's a masterpiece. It's not
6	蝙蝠女侠 (2019–)	rw5170693	0.7672426	We get the message and
7	The Dark Knight (2008)	rw6080661	0.76454085	Even after over a decade holds up as one of the gre
8	The Boys (2019–)	rw6355589	0.7645134	The "heroes" in the series

```
%sql
SELECT movie, MAX(similarity) FROM similarity2_table GROUP BY movie ORDER BY MAX(similarity) DESC
```

	movie	max(similarity)
2	Captain America: The Winter Soldier (2014)	0.795051
3	Smallville (2001–2011)	0.78102165
4	Wonder Woman 1984 (2020)	0.77668035
5	小丑 (2019)	0.77379185
6	蝙蝠女侠 (2019–)	0.7672426
7	The Dark Knight (2008)	0.76454085
8	Deadpool (2016)	0.76287955

Truncated results, showing first 1000 rows

```
# Recommend Movie based on Keyword
key_word = "magic"
docvecs = reviews_w2v
x = spark.createDataFrame([('newreviewid', key_word)]).\
    withColumnRenamed('_1', 'review_id').\
    withColumnRenamed('_2', 'review_detail')
x.show()
```

	movie	review_id	similarity	review_detail
1	Harry Potter and the Sorcerer's Stone (2001)	rw5759536	0.75622505	The magical magic movie brought me great attraction. The the protagonist always moves me.
2	The Trouble with Angels (1966)	rw2947728	0.6925906	. . . THE TROUBLE WITH ANGELS is a "scathingly brilliant" riding a train to boarding school. The magic of attending an cigarettes in the Girls' Room and cigars in the cellar. The m days a week. The magic of forbidden hallways. The magic c snow sifting through the ...
3	Wanda and the Alien: Pumpkins (2014) Season 1, Episode 3	rw6322646	0.6886538	One of my favourite episodes. Magic + pumpkins = magic p
4	Cursed (2020–)	rw6086358	0.6714393	Magic swords, witches and wizards, Fey creatures of anoth
5	Just Add Magic: Just Add Codes (2019) Season 3, Episode 3	rw5763308	0.65995693	If mama P forgets about magic, how can she remember abc
6	Jingle Jangle: A Christmas Journey (2020)	rw6270783	0.65707356	Amazing film...reminds you how Christmas can be magic.

	movie	max(similarity)
1	Harry Potter and the Sorcerer's Stone (2001)	0.75622505
2	The Trouble with Angels (1966)	0.6925906
3	Wanda and the Alien: Pumpkins (2014) Season 1, Episode 3	0.6886538
4	Cursed (2020–)	0.6714393
5	Just Add Magic: Just Add Codes (2019) Season 3, Episode 3	0.65995693
6	Jingle Jangle: A Christmas Journey (2020)	0.65707356

Truncated results, showing first 1000 rows

6. Appendix

The code and results link:

https://drive.google.com/file/d/1lf4UR0BYEhnXIZK-bamNW4zKyiLTlP4j/view?usp=sharing

How to replicate the results:

- Register an account in the databricks, and log in.
- Upload the datasets to the databricks dbfs.
- Open the table with a notebook
- Import the notebook to the new notebook
- Click “run all” to see the results.