Parallel Movie Recommendation System

CSE4001 – Parallel and Distributed Computing

PROJECT BASED COMPONENT REPORT

by

GIRISH BALAJI - (19BCI0247)

KOLLA SRINIVASAN PRAGATHI - 19BCE2313

DHARUN KARTHICK - 19BCE2316

PANIGRAHI SHYAM - 19BCI0239

KIRANAADARSH S - 19BCI0242)

School of Computer Science and Engineering

DECEMBER 2021

DECLARATION

I hereby declare that the report entitled **"Parallel Movie Recommendation System"** submitted by me, for the CSE4001 Parallel and Distributed Computing (EPJ) to Vellore Institute of Technology is a record of bonafide work carried out by me under the supervision of Dr.N. Narayanan Prasanth.

I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for any other courses in this institute or any other institute or university.

[illegible]

Place : Vellore

Date : 8-12-2021

Signature of the Candidate

Abstract:

The goal of this project is to parallelize the process of generating recommendations to users. So, we have selected a recommendation system with machine learning techniques so it will be in a way to create there, own recommendation system for users. So, this algorithm will help to make the process simple and predict the results with good accuracy. So, to obtain this we are going to use two methods and compare the results of it, to provide the best predictions. The methodologies which we are going to use are KNN-item based collaborative filtering and Alternating Least Square (ALS) Matrix Factorization in Collaborative Filtering. By using these methods, and comparing there, accuracy we are providing a good movie recommendation system to users

Table of Contents

LIST OF FIGURES:

1. INTRODUCTION

The goal of this project is to parallelize the process of generating movie recommendations to the users. Specifically, we aim to predict, as accurately as possible, the rating a user gives to a particular product based on Spark and OpenMP. If we are able to make accurate predictions, we can recommend products to users that they have not bought yet based on a collaborative approach. We will try to accomplish this in the shortest time.

1.1 Objective

The objective of our project is to develop a movie recommendation system with most precise suggestions to the users based on the ratings and reviews. We look up to some parameters to make the recommendation more user friendly. They are:

Increase the number of movies: Users are more likely to watch movies that are close to their preferences.

Diversity: Recommender Systems are considered to be useful when the recommended items have not been seen by the user in the past. Repeated recommended popular movies can lead to reduction in movies that were watched less frequently.

Increase user's satisfaction: A well- structured Recommender System can increase the overall experience of the user with the system. The better the experience is, the longer users stay in the system.

Increase user's loyalty: Users tend to be loyal in the system that recognize them as old customers and treat them like valuable visitors.

Better understanding of user's needs: The user's preferences that are either collected explicitly or predicted by the system, can be leveraged in the future in order to improve the management of the movies.

1.2 Motivation

YouTube, Netflix, Amazon, Pinterest and a long list of other internet products all rely on recommender systems to filter millions of contents and make personalized recommendations to their users. Recommender systems are well-studied and proven to provide tremendous values to internet businesses and their consumers. As far as the existing systems deal with the large dataset. Then how about simple datasets or do you ever think about making your own recommendation systems with good accuracy and with good efficiency?

2.LITERATURE SURVEY

2.1 Movie Recommendation System Using KNN Algorithm and Collaborative Filtering

In this paper the methodology makes use of the information provided by users, analyses them and compares them with the choices of other users then recommends the movies. This article plans and executes an entire motion picture proposal framework model in light of the KNN algorithm, collaborative filtering algorithm and recommendation system technology. But this paper has mostly considered centrality for the improvement of customized proposal innovation. By using centrality improvement, the system might take much time to execute.

2.2 Book Recommendation System using KNN Algorithm

In this paper they have proposed a book recommendation system using knn algorithm. The problem of data sparsity has been solved by the use of KNN algorithm and association rule mining to achieve better performance. And in their system, they have also used matrix factorization process in order to find the missing values and then the factorized values, passed through the KNN algorithm.

Though this system helps in solving data sparsity it lacks in focusing the data of other perspectives.

2.3 Recommendation System for E-commerce using Alternating Least Squares (ALS) on Apache Spark

In this research paper they developed a recommendation system for e-commerce using alternating least squares (ALS) matrix factorization method on Apache Spark MLlib. As Apache Spark is well suited for applications which require high speed query of data, transformation and analytics results. And they have also reduced the RMSE (root mean squared error) value through ALS matrix factorization.

2.4 Optimizing Parallel Collaborative Filtering Approaches for Improving Recommendation Systems Performance

In this paper they discussed the methodologies to improve the CFA for recommendations, in order to achieve this they used rating replication methodology which was used on the graphs[large sparse bipartite graphs] and using SVD++[single value decomposition technique]and they balanced the information loss and controlled but it has increased the collaborative filtering algorithms execution. They have considered the active user ratings, rather than active users selected ratings.

2.5 Movie Recommender System Based on Collaborative Filtering Using Apache Spark

In this paper they have proposed a movie recommender system based on ALS using Apache Spark. This research focuses on the selection of parameters of ALS algorithms that can affect the performance of a building robust RS.So based on this paper for our second approach we will be able to choose the better parameters for getting less RMSE value.

2.6 Algorithmic Acceleration of Parallel ALS for Collaborative Filtering: Speeding up Distributed Big Data Recommendation in Spark

In this paper they have considered a very large dataset to achieve the parallel optimization. Where they used to accelerate the convergence of parallel ALS-based optimization methods for collaborative filtering using a nonlinear conjugate gradient (NCG) wrapper around the ALS iterations.

Though this method is very useful for further big data processing, the cost and optimization time for this methodology increases rapidly.

2.7 A Parallel Recommender System Using a Collaborative Filtering Algorithm with Correntropy for Social Networks

In this paper they have created a new parallel recommender system based on correntropy and CF. To achieve insensitive performance to outliers, the correntropy was used to compute the similarity of two items or users instead of typical metrics used in recommendation algorithms. Furthermore, we used the Spark framework to support parallel processing in order to lower the computational cost. Experiments were conducted on three datasets containing data collected from actual social networks, and the results showed that the proposed system could effectively improve the computational time and achieve satisfactory performance for social networks applications, despite the presence of invalid data.

But the disadvantage of this method is correntropy doesn't match well with alternating least squares which performs less compared to existing work.

2.8 A Movie Recommender System: MOVREC

In this paper they have introduced MovieREC, a movie recommendation recommender system, in this work. It allows a user to choose from a variety of attributes and then recommends a movie list to him based on the cumulative weight of the various attributes and the K-means algorithm. Due to the nature of our system, evaluating performance is difficult because there is no right or incorrect recommendation; it is simply a matter of opinion.

As we mentioned this paper has dealt with the serial implementation of the movie recommendation system which lacks in scalability, and also has cold start problem

2.9 An Efficient movie recommendation algorithm based on improved k-clique

In this paper they have proposed a k-clique algorithm for the movie recommendation system and compared the analysis with collaborative filtering. Though they might have achieved a better accuracy rate compared to the existing system, they lack the speed the proposed methodology takes a lot of time to execute.

2.10 Content-based Recommender System for Movie Website

In this paper they have proposed a content-based filtering method for the recommendation system, in which they have used cosine similarity to find similar features, and for weighting the features they have used the Term Frequency-Inverse Document Frequency-Normalization.

But the proposed system is hard to apply for automate feature extraction to media data by content-based filtering method. And the recommendation result only limits to items the user ever chose, which means the diversity is not so good. It is very hard to recommend for users who never choose anything.

3. TECHNICAL SPECIFICATION

To implement the movie recommender using the collaborative filtering technique using the KNN method we used python and many libraries such as:

1) Pandas,
2) scipy.sparse for csr matrix,
3) NearestNeighbors from sklearn.neighbors and
4) fuzz from fuzzywuzzy
5) We also had to import the csv files from MovieLens Datasets. It contains 27,753,444 ratings and 1,108,997 tag applications across 58,098 movies. There are 2 major csv files to import and save – movies.csv and ratings.csv. These will be used for the ratings prediction and the recommendations.

To implement the movie recommender system using the Alternating Least Square (ALS) Matrix Factorization in Collaborative Filtering we had to use many libraries and technologies like:

1) Spark and Hadoop
2) ALS from pyspark.ml.recommendation
3) RegressionEvaluator from pyspark.ml.evaluation
4) SparkSession from pyspark.sql
5) We used the same dataset that we used in the previous method so that the comparison of time between the 2 implementations is done without any bias.

4. DESIGN

4.1 KNN- collaborative filtering approach

Methodology

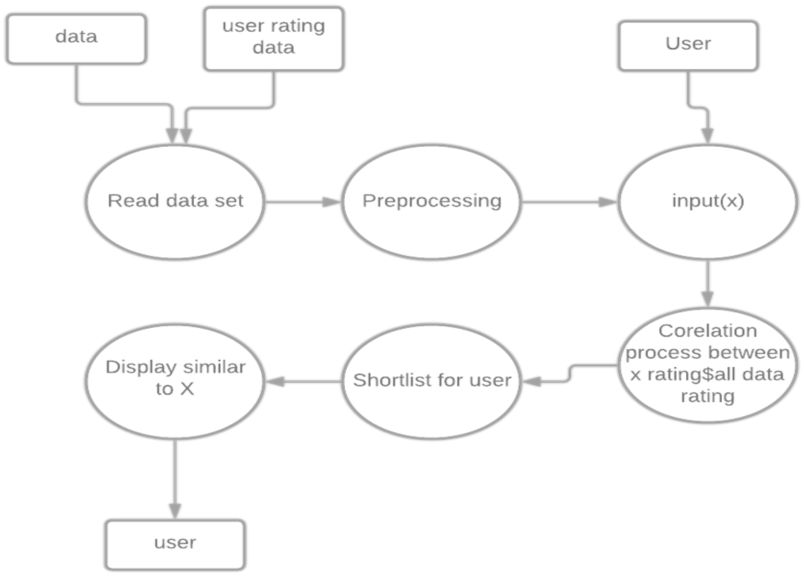


Fig 1: (Methodology)

KNN is a ideal model and is also a perfect baseline for a recommendation system. It is a lazy an investigating method. Uses database when data points are subdivided into several sets to index's new samples. KNN does not take any consideration in the distribution of basic data but relies on feature similarity. When KNN makes a reference about a movie, it will calculate the “distance” between the target movie and all other movies on its website, then measure its distances and return the top-notch movies as K as the most similar movie recommendations.

First, we will collect data sets knowing that many models make recommendations based on user rating patterns. In order to remove noisy patterns and avoid “memory error” due to large databases, we will filter our rating data.

Modeling:

1) Collaborative filtering programs use user actions to recommend other movies. Generally, they are not user-based or object-based. Object-based approach is oftenpreferred over a user-based approach.
2) For the use of integrated object-based filters, KNN is the perfect model to go to and is an excellent basis for the development of a recommendation system.

3) KNN does not make any consideration in the distribution of basic data but relies on feature similarity.
4) When KNN makes a reference to an item, KNN will calculate the "distance" between the target object and everything else on its website, then measure its distances and return the K to the neighbouring objects as recommendations for very similar items.
5) So later we convert the data frame into a KNN-accepted format and then a scant matrix into a spicy scant matrix.
6) Next, we will use the cosine similarity to avoid the curse of size. We also train our data once the data has been pre-processed and prepare our KNN model with the correct parameters. And make recommendations.

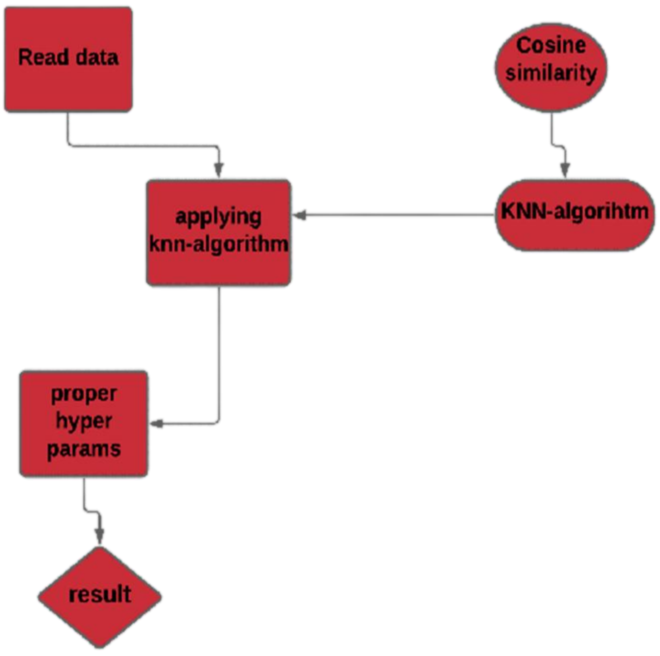


Fig 2 :(Flow Diagram of KNN-CF approach)

4.2 Alternating Least Square (ALS) Matrix Factorization-C

In collaborative filtering, matrix factorization is a modern solution to the ambiguous data problem.

Matrix Factorization:

Matrix factorization is simply a family of mathematical computations in matrices in line algebra. Specifically, the factorization of the matrix is the factorization of the matrix into the product of matrices.

Alternating Least Square (ALS) is also a matrix factorization algorithm and operates in a similar manner. ALS is used in the Apache Spark ML and is designed for large-scale filtering problem. ALS does an excellent job of resolving shortcoming like scalability and sparseness of rating data, and it is simple and well scaled to very large databases.

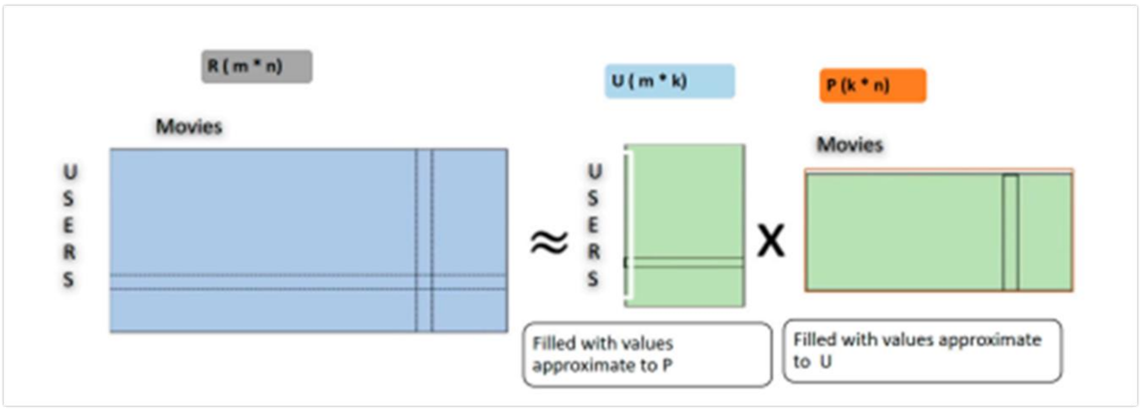

R (m * n)
Movies
USERS
≈
U (m * k)
USERS
X
P (k * n)
Movies
Filled with values approximate to P
Filled with values approximate to U

Fig 3: (Working model of ALS general)

In order to produce a model, we need to build a work flow around the model. The normal ML workflow probably begins with data processing with a pre-defined set of ETL tasks, offline / online model training, and then imports trained models into web services to produce. In our case, we're going to implement a very small version of the parallel recommender that just does the job of movie recommendation.

ALS matrix factorisation attempts to measure matrix R values as a product of two low-level matrices, X and Y, i.e., X * Yt = R. These measurements are commonly referred to as 'factor' matrices. The normal pattern repeats itself. During each multiplication, one factor matrices are constantly captured, while the other is solved using small squares. The newly resolved factor matrix is then held regularly while resolving another factor matrix.

Work flow of ALS approach is:

The flow of ALS system function is:

1) A new user submits his / her favourite preferences, and the system creates new samples for interaction with movie users for model users.
2) The system also trains the ALS model in data and new inputs
3) The system creates a recommendation data for me to understand (in my case, I used a sample of all movies from the data)
4) The system generates rating predictions on all that user's recommendations
5) The system issues N movie recommendations based on user rating   estimates ratings

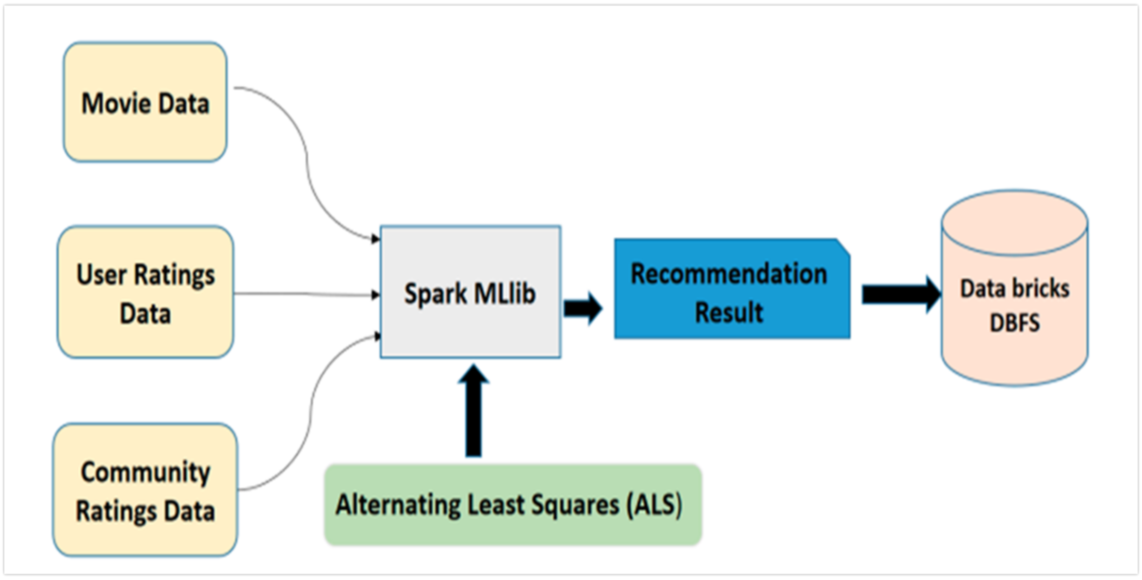


Fig 4 (Block diagram of ALS-with Apache Spark-Mlib)

5.PROPOSED SYSTEM

Our proposed system is to implement a movie recommendation system using collaborative filtering approach. These recommendations should be based on a user's previous watch history as well as movies that similar users have watched and also the ratings. Therefore, computing how similar two users are is an essential part of the recommendation process. Good recommendations benefit both customers, who are suggested to watch movies that are better suited to their liking and are able to save time, as they are able to host a greater number of movies, successfully market new movies, and obtain customer loyalty as watching new movies increases the quality of recommended movies.

There are few drawbacks in the KNN method based on which the ALS system was proposed:

1) popularity bias: refers to system recommends the movies with the most interactions without any personalization
2) item cold-start problem: refers to when movies added to the catalogue have either none or very little interactions while recommender rely on the movie's interactions to make recommendations
3) scalability issue: refers to lack of the ability to scale to much larger sets of data when more and more users and movies added into our database

Thus the proposed ALS system will overcome these problems using the Matrix factorization method. This increases the scalability as well as the speed of the recommender systems.

From the above discussion we can see that the existing system is not really accurate and also has many disadvantages. On top of these it is slow compared to the ALS approach. Thus, in order to parallelize the KNN based approach we introduce another method using ALS based on which the system will be parallelized.

In the proposed parallelized approach, we will parallelize ALS by parallelizing the updates of U and of the Matrix M.

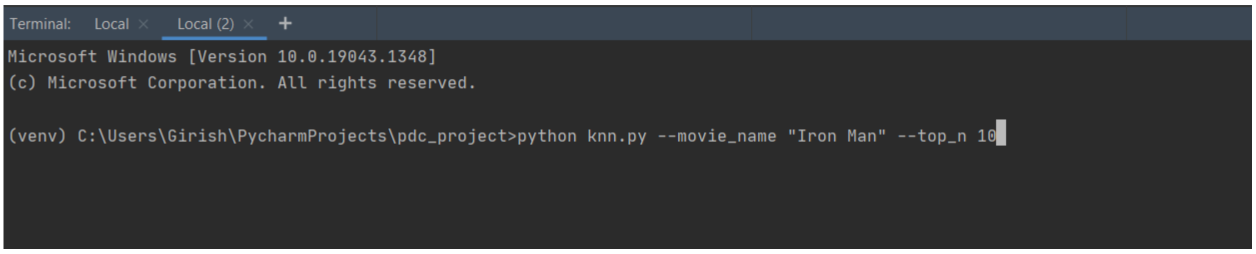
1. First we split, clean and tune the data
2. set model parameters for pyspark.ml.recommendation.ALS

3. We must Return the closest matches of the input movie via SQL regex. If no match found, return None.
4. Append a user's movie ratings to ratingsDF
5. Create a user with all movies except ones were rated for inferencing
6. Implement the grid search function to select the best model based on RMSE of validation data and return The best fitted ALS model with lowest RMSE score on validation data
7. Get the ALS model, train the ALS model and evaluate the model by computing the RMSE on the validation data
8. Then finally display the top n movies after getting the inputs from user

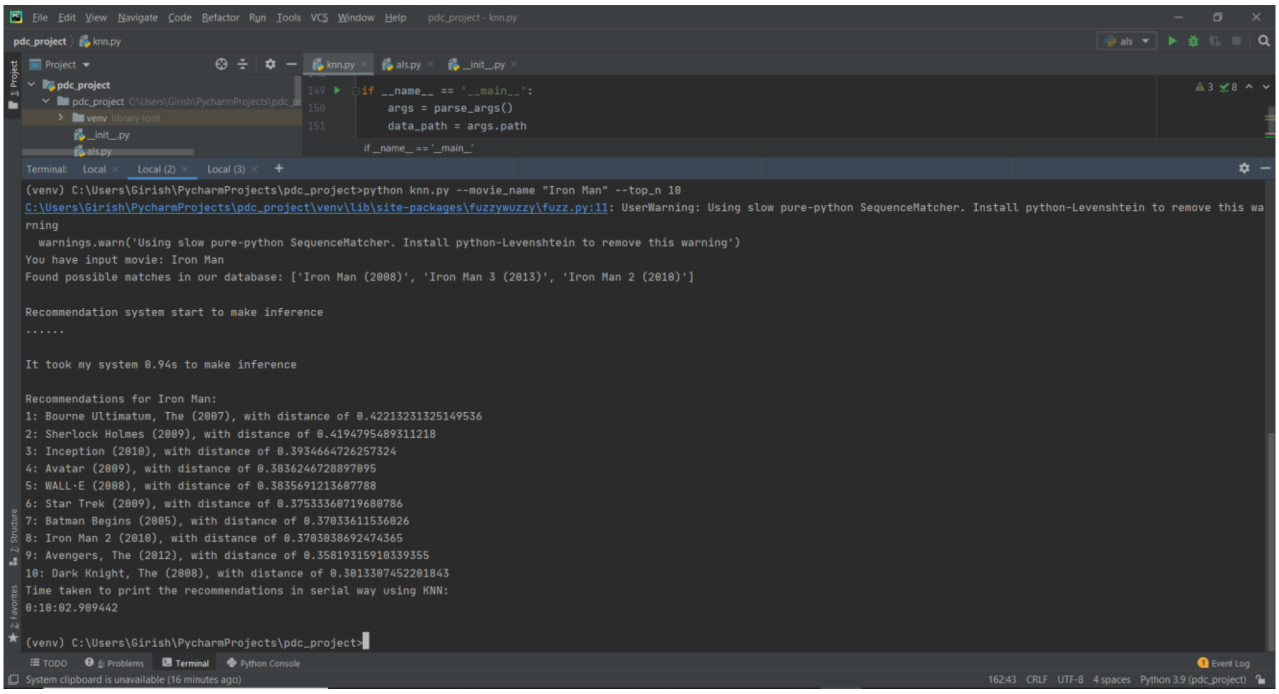
This is how the proposed system is implemented and works.

6.RESULTS AND DISCUSSION

1. KNN recommender system

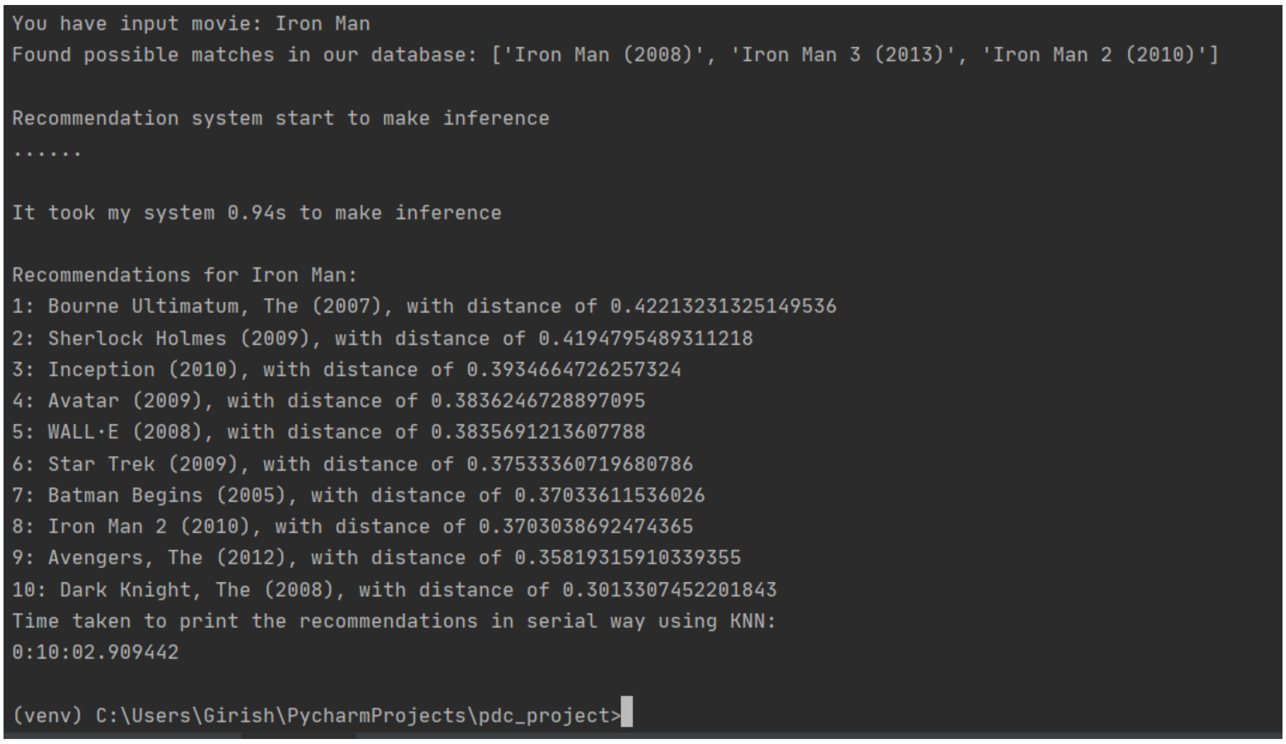


```
Microsoft Windows [Version 10.0.19043.1348]
(c) Microsoft Corporation. All rights reserved.

(venv) C:\Users\Girish\PycharmProjects\pdc_project>python knn.py --movie_name "Iron Man" --top_n 10
```

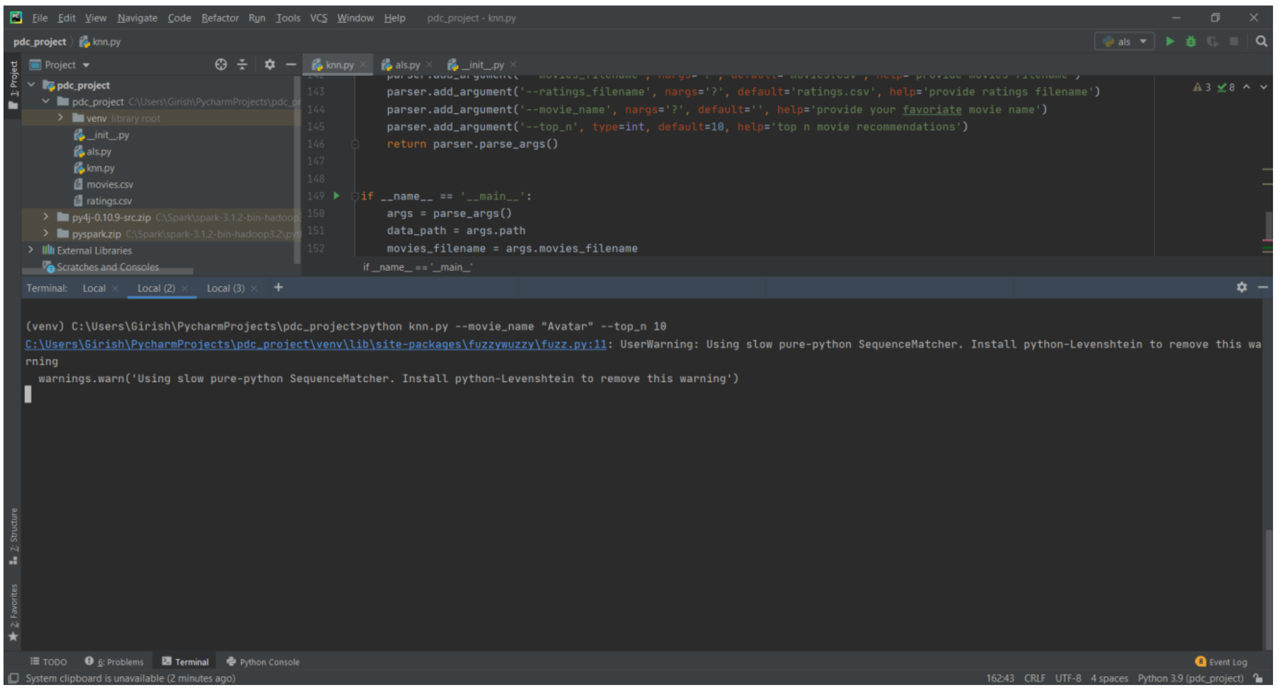

File Edit View Navigate Code Refactor Run Tools VCS Window Help pdc_project - knn.py
pdc_project knn.py
Project
knn.py
als.py
__init__.py
pdc_project
venv library root
Terminal: Local Local (2) Local (3)
TODO
6: Problems
Terminal
Python Console
Event Log
System clipboard is unavailable (16 minutes ago)
162:43 CRLF UTF-8 4 spaces Python 3.9 (pdc_project)

```
if __name__ == '__main__':
    args = parse_args()
    data_path = args.path
```

```
(venv) C:\Users\Girish\PycharmProjects\pdc_project>python knn.py --movie_name "Iron Man" --top_n 10
C:\Users\Girish\PycharmProjects\pdc_project\venv\lib\site-packages\fuzzywuzzy\fuzz.py:11: UserWarning: Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this wa
rning
  warnings.warn('Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this warning')
You have input movie: Iron Man
Found possible matches in our database: ['Iron Man (2008)', 'Iron Man 3 (2013)', 'Iron Man 2 (2010)']

Recommendation system start to make inference
......

It took my system 0.94s to make inference

Recommendations for Iron Man:
1: Bourne Ultimatum, The (2007), with distance of 0.42213231325149536
2: Sherlock Holmes (2009), with distance of 0.4194795489311218
3: Inception (2010), with distance of 0.3934664726257324
4: Avatar (2009), with distance of 0.3836246728897095
5: WALL·E (2008), with distance of 0.3835691213607788
6: Star Trek (2009), with distance of 0.37533360719680786
7: Batman Begins (2005), with distance of 0.37033611536026
8: Iron Man 2 (2010), with distance of 0.3703038692474365
9: Avengers, The (2012), with distance of 0.35819315910339355
10: Dark Knight, The (2008), with distance of 0.3013307452201843
Time taken to print the recommendations in serial way using KNN:
0:10:02.909442

(venv) C:\Users\Girish\PycharmProjects\pdc_project>
```

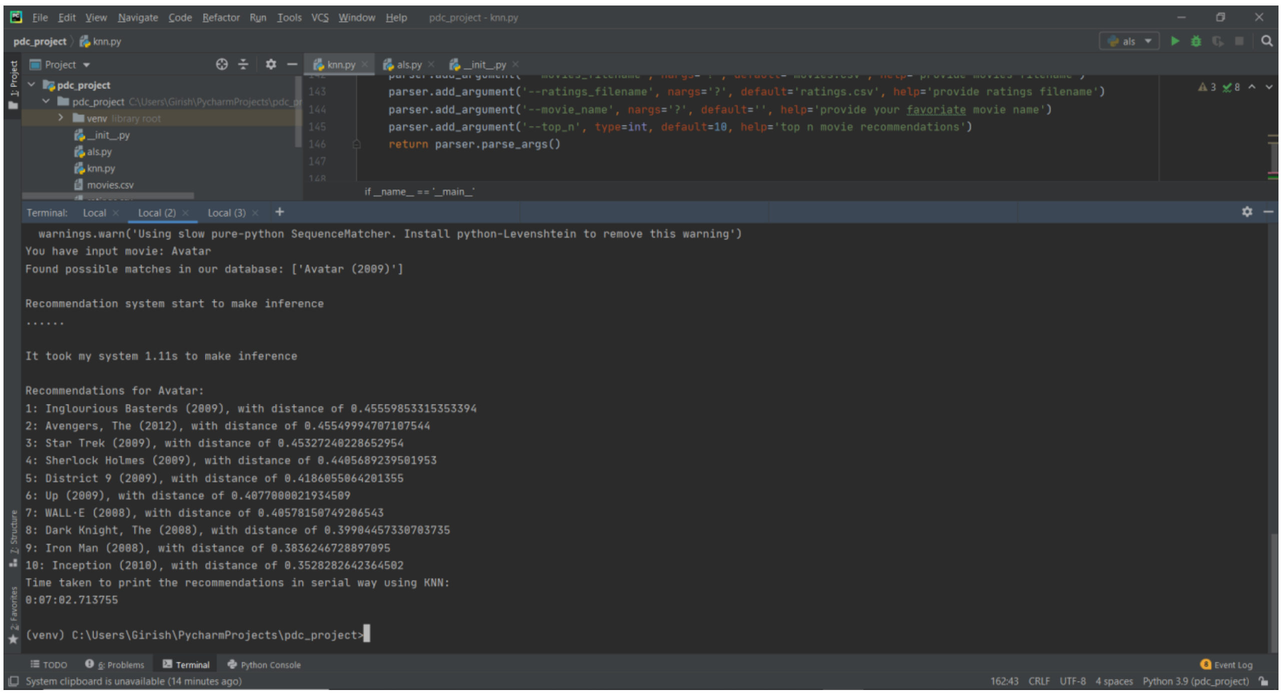


```
You have input movie: Iron Man
Found possible matches in our database: ['Iron Man (2008)', 'Iron Man 3 (2013)', 'Iron Man 2 (2010)']

Recommendation system start to make inference
......

It took my system 0.94s to make inference

Recommendations for Iron Man:
1: Bourne Ultimatum, The (2007), with distance of 0.42213231325149536
2: Sherlock Holmes (2009), with distance of 0.4194795489311218
3: Inception (2010), with distance of 0.3934664726257324
4: Avatar (2009), with distance of 0.3836246728897095
5: WALL·E (2008), with distance of 0.3835691213607788
6: Star Trek (2009), with distance of 0.37533360719680786
7: Batman Begins (2005), with distance of 0.37033611536026
8: Iron Man 2 (2010), with distance of 0.3703038692474365
9: Avengers, The (2012), with distance of 0.35819315910339355
10: Dark Knight, The (2008), with distance of 0.3013307452201843
Time taken to print the recommendations in serial way using KNN:
0:10:02.909442

(venv) C:\Users\Girish\PycharmProjects\pdc_project>
```



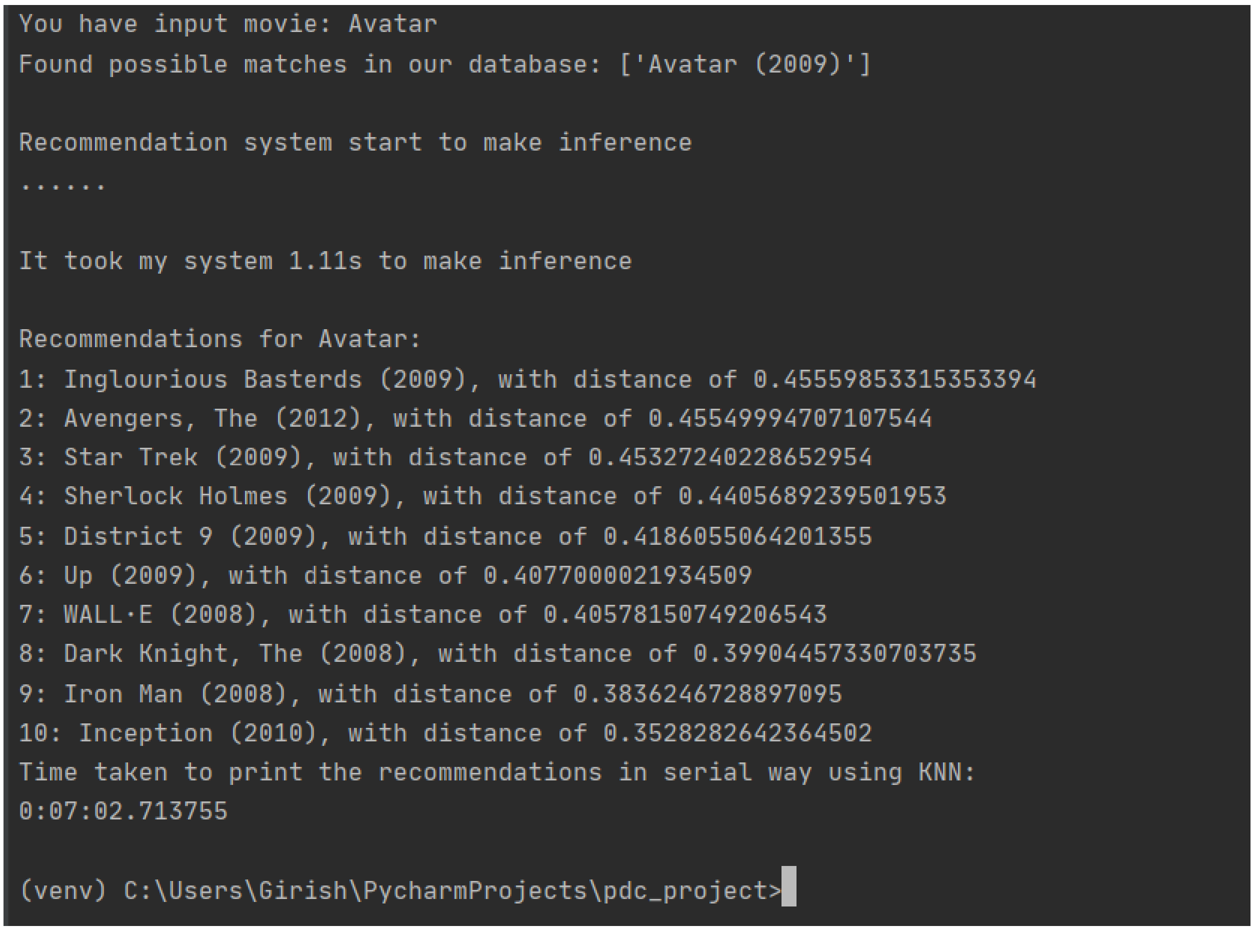
```python
    parser.add_argument('--ratings_filename', nargs='?', default='ratings.csv', help='provide ratings filename')
    parser.add_argument('--movie_name', nargs='?', default='', help='provide your favoriate movie name')
    parser.add_argument('--top_n', type=int, default=10, help='top n movie recommendations')
    return parser.parse_args()


if __name__ == '__main__':
    args = parse_args()
    data_path = args.path
    movies_filename = args.movies_filename
```

```
(venv) C:\Users\Girish\PycharmProjects\pdc_project>python knn.py --movie_name "Avatar" --top_n 10
C:\Users\Girish\PycharmProjects\pdc_project\venv\lib\site-packages\fuzzywuzzy\fuzz.py:11: UserWarning: Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this wa
rning
  warnings.warn('Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this warning')
```



```python
    parser.add_argument('--ratings_filename', nargs='?', default='ratings.csv', help='provide ratings filename')
    parser.add_argument('--movie_name', nargs='?', default='', help='provide your favoriate movie name')
    parser.add_argument('--top_n', type=int, default=10, help='top n movie recommendations')
    return parser.parse_args()
```

```
  warnings.warn('Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this warning')
You have input movie: Avatar
Found possible matches in our database: ['Avatar (2009)']

Recommendation system start to make inference
......

It took my system 1.11s to make inference

Recommendations for Avatar:
1: Inglourious Basterds (2009), with distance of 0.45559853315353394
2: Avengers, The (2012), with distance of 0.45549994707107544
3: Star Trek (2009), with distance of 0.45327240228652954
4: Sherlock Holmes (2009), with distance of 0.4405689239501953
5: District 9 (2009), with distance of 0.4186055064201355
6: Up (2009), with distance of 0.4077000021934509
7: WALL·E (2008), with distance of 0.40578150749206543
8: Dark Knight, The (2008), with distance of 0.39904457330703735
9: Iron Man (2008), with distance of 0.3836246728897095
10: Inception (2010), with distance of 0.3528282642364502
Time taken to print the recommendations in serial way using KNN:
0:07:02.713755

(venv) C:\Users\Girish\PycharmProjects\pdc_project>
```

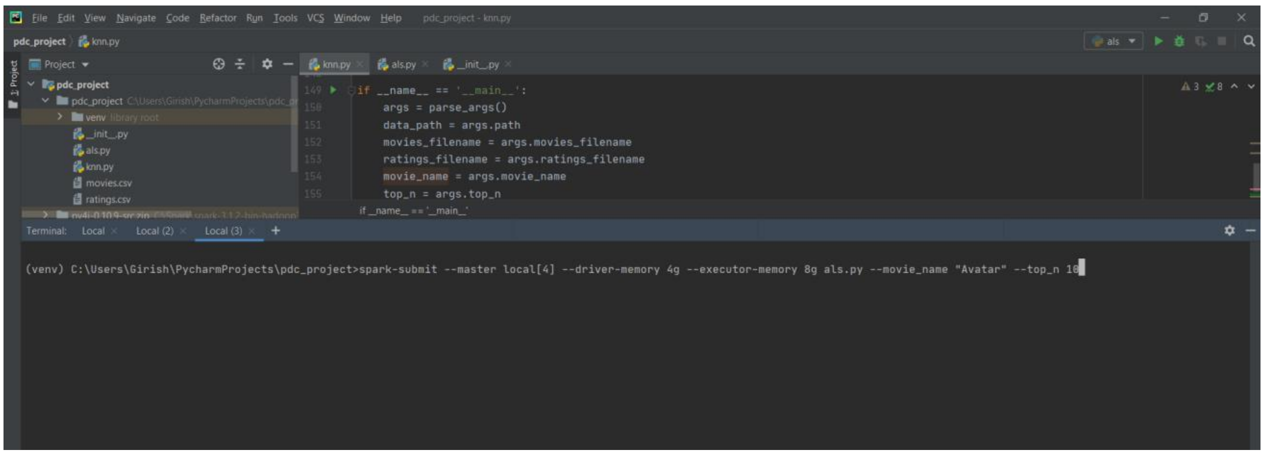


```
You have input movie: Avatar
Found possible matches in our database: ['Avatar (2009)']

Recommendation system start to make inference
......

It took my system 1.11s to make inference

Recommendations for Avatar:
1: Inglourious Basterds (2009), with distance of 0.45559853315353394
2: Avengers, The (2012), with distance of 0.45549994707107544
3: Star Trek (2009), with distance of 0.45327240228652954
4: Sherlock Holmes (2009), with distance of 0.4405689239501953
5: District 9 (2009), with distance of 0.4186055064201355
6: Up (2009), with distance of 0.4077000021934509
7: WALL·E (2008), with distance of 0.40578150749206543
8: Dark Knight, The (2008), with distance of 0.39904457330703735
9: Iron Man (2008), with distance of 0.3836246728897095
10: Inception (2010), with distance of 0.3528282642364502
Time taken to print the recommendations in serial way using KNN:
0:07:02.713755

(venv) C:\Users\Girish\PycharmProjects\pdc_project>
```

Thus from the above pictures we can see that the recommendations were successful and we can see the in the output the 10 recommendations for the given input movie with the distance from the input movie.

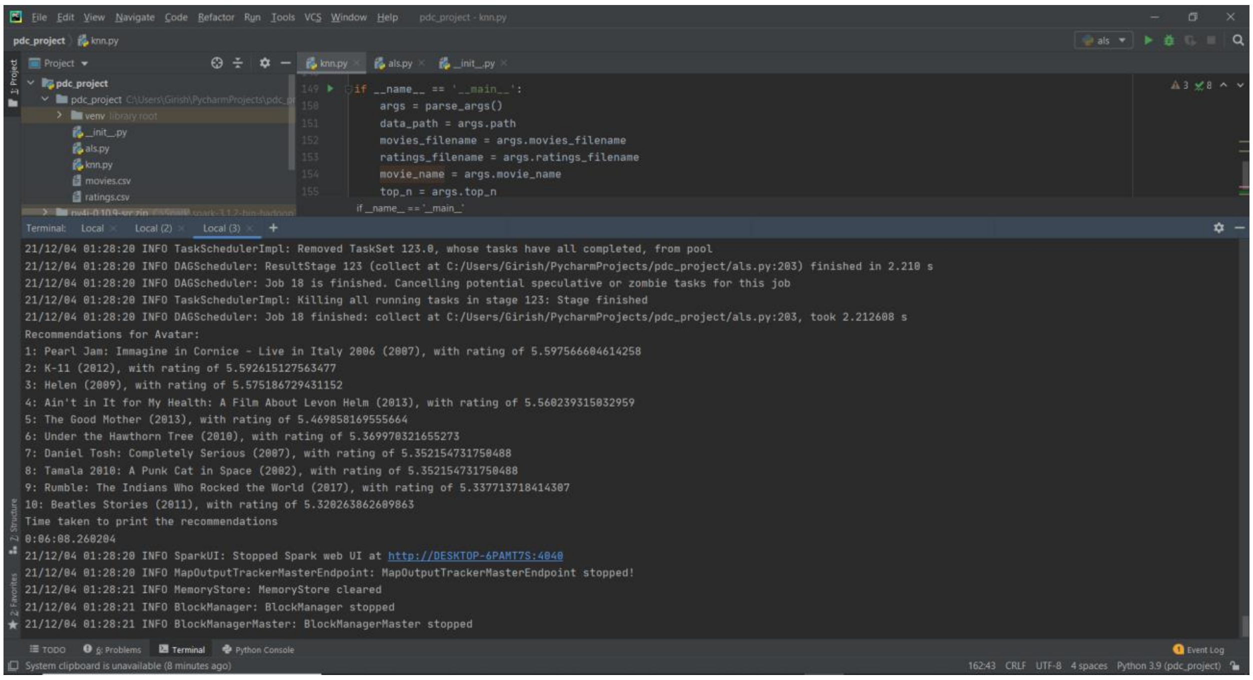
From the above screenshots we can see that the time taken to give the user the recommendation was 10 mins 2 secs 909 ms in the first case and 7 mins 2 secs 713 ms in the second case.

2. ALS recommender system



```python
if __name__ == '__main__':
    args = parse_args()
    data_path = args.path
    movies_filename = args.movies_filename
    ratings_filename = args.ratings_filename
    movie_name = args.movie_name
    top_n = args.top_n
```

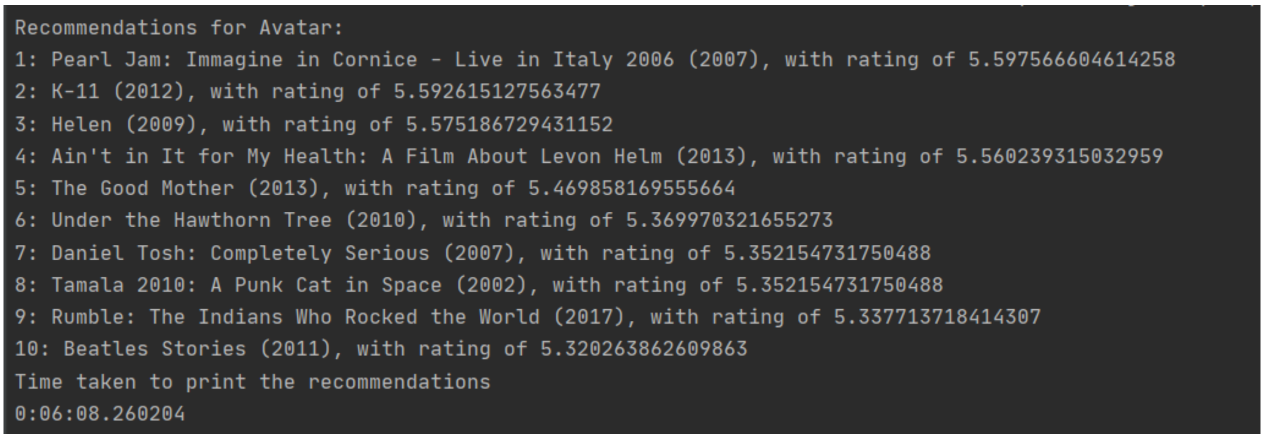
```
(venv) C:\Users\Girish\PycharmProjects\pdc_project>spark-submit --master local[4] --driver-memory 4g --executor-memory 8g als.py --movie_name "Avatar" --top_n 10
```



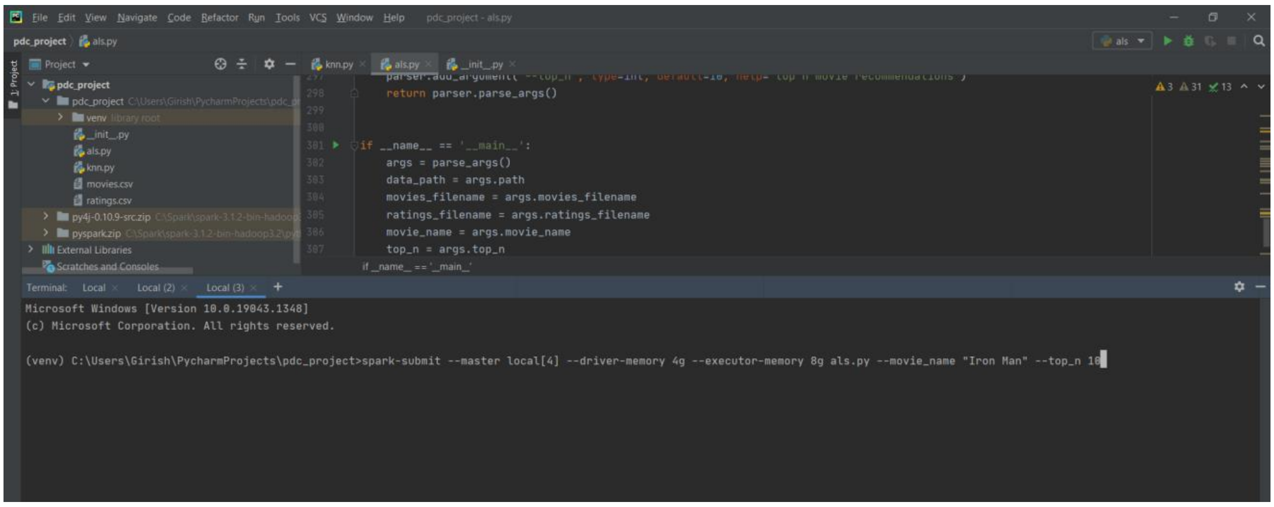
```python
if __name__ == '__main__':
    args = parse_args()
    data_path = args.path
    movies_filename = args.movies_filename
    ratings_filename = args.ratings_filename
    movie_name = args.movie_name
    top_n = args.top_n
```

```
21/12/04 01:28:20 INFO TaskSchedulerImpl: Removed TaskSet 123.0, whose tasks have all completed, from pool
21/12/04 01:28:20 INFO DAGScheduler: ResultStage 123 (collect at C:/Users/Girish/PycharmProjects/pdc_project/als.py:203) finished in 2.210 s
21/12/04 01:28:20 INFO DAGScheduler: Job 18 is finished. Cancelling potential speculative or zombie tasks for this job
21/12/04 01:28:20 INFO TaskSchedulerImpl: Killing all running tasks in stage 123: Stage finished
21/12/04 01:28:20 INFO DAGScheduler: Job 18 finished: collect at C:/Users/Girish/PycharmProjects/pdc_project/als.py:203, took 2.212608 s
Recommendations for Avatar:
1: Pearl Jam: Immagine in Cornice - Live in Italy 2006 (2007), with rating of 5.597566604614258
2: K-11 (2012), with rating of 5.592615127563477
3: Helen (2009), with rating of 5.575186729431152
4: Ain't in It for My Health: A Film About Levon Helm (2013), with rating of 5.560239315032959
5: The Good Mother (2013), with rating of 5.469858169555664
6: Under the Hawthorn Tree (2010), with rating of 5.369970321655273
7: Daniel Tosh: Completely Serious (2007), with rating of 5.352154731750488
8: Tamala 2010: A Punk Cat in Space (2002), with rating of 5.352154731750488
9: Rumble: The Indians Who Rocked the World (2017), with rating of 5.337713718414307
10: Beatles Stories (2011), with rating of 5.320263862609863
Time taken to print the recommendations
0:06:08.260204
21/12/04 01:28:20 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-6PAMT7S:4040
21/12/04 01:28:20 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
21/12/04 01:28:21 INFO MemoryStore: MemoryStore cleared
21/12/04 01:28:21 INFO BlockManager: BlockManager stopped
21/12/04 01:28:21 INFO BlockManagerMaster: BlockManagerMaster stopped
```

System clipboard is unavailable (8 minutes ago)

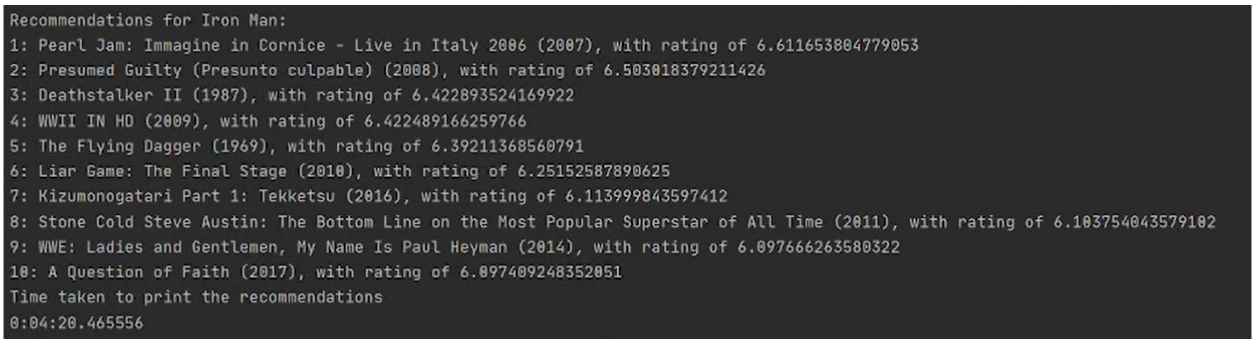


```
Recommendations for Avatar:
1: Pearl Jam: Immagine in Cornice - Live in Italy 2006 (2007), with rating of 5.597566604614258
2: K-11 (2012), with rating of 5.592615127563477
3: Helen (2009), with rating of 5.575186729431152
4: Ain't in It for My Health: A Film About Levon Helm (2013), with rating of 5.560239315032959
5: The Good Mother (2013), with rating of 5.469858169555664
6: Under the Hawthorn Tree (2010), with rating of 5.369970321655273
7: Daniel Tosh: Completely Serious (2007), with rating of 5.352154731750488
8: Tamala 2010: A Punk Cat in Space (2002), with rating of 5.352154731750488
9: Rumble: The Indians Who Rocked the World (2017), with rating of 5.337713718414307
10: Beatles Stories (2011), with rating of 5.320263862609863
Time taken to print the recommendations
0:06:08.260204
```



```python
    parser.add_argument('--top_n', type=int, default=10, help='top n movie recommendations')
    return parser.parse_args()


if __name__ == '__main__':
    args = parse_args()
    data_path = args.path
    movies_filename = args.movies_filename
    ratings_filename = args.ratings_filename
    movie_name = args.movie_name
    top_n = args.top_n
```

```
Microsoft Windows [Version 10.0.19043.1348]
(c) Microsoft Corporation. All rights reserved.

(venv) C:\Users\Girish\PycharmProjects\pdc_project>spark-submit --master local[4] --driver-memory 4g --executor-memory 8g als.py --movie_name "Iron Man" --top_n 10
```



```
Recommendations for Iron Man:
1: Pearl Jam: Immagine in Cornice - Live in Italy 2006 (2007), with rating of 6.611653804779053
2: Presumed Guilty (Presunto culpable) (2008), with rating of 6.503018379211426
3: Deathstalker II (1987), with rating of 6.422893524169922
4: WWII IN HD (2009), with rating of 6.422489166259766
5: The Flying Dagger (1969), with rating of 6.39211368560791
6: Liar Game: The Final Stage (2010), with rating of 6.25152587890625
7: Kizumonogatari Part 1: Tekketsu (2016), with rating of 6.113999843597412
8: Stone Cold Steve Austin: The Bottom Line on the Most Popular Superstar of All Time (2011), with rating of 6.103754043579102
9: WWE: Ladies and Gentlemen, My Name Is Paul Heyman (2014), with rating of 6.097666263580322
10: A Question of Faith (2017), with rating of 6.097409248352051
Time taken to print the recommendations
0:04:20.465556
```

Thus from the above pictures we can see that the recommendations were successful and we can see the in the output the 10 recommendations for the given input movie with the distance from the input movie.

KNN implementation(execution time)	**ALS/parallel implementation(execution time)**
6min 8sec 260ms	4min 20sec 465ms
7min 10sec 160ms	5min 12sec 120ms
6min 10sec 365ms	3min 45sec 32ms

From the above results we can see the clear difference in the time taken to generate the recommendations. We can see that the ALS has proved to be faster in both the cases. Execution time has been **decreased** in the second method approximately 1x times compared to the first method. Thus based on this we can strongly say that we have improved the performance of the regular recommendation systems by a great factor.

7.CONCLUSIONS

We stared with the importance of recommendation systems and the importance and characteristics of a good recommender system. We initially implemented the regular, existing KNN based collaborative filtering based approach to recommend top “n” movies closest to the movie entered by the user.

Then we discussed about the shortcomings and drawbacks of the existing model and arrived at the ALS matrix factorization method for speeding up the process of movie recommendation to the user. We implemented the movie recommendation system using this method and finally we compared the effectiveness, efficiency and most importantly the increase in speed of recommendations after using the ALS method. We saw that the ALS method was clearly faster by a factor of around 1.5 times (speedup) than the regular recommendation system. This way we proved that the ALS method for recommendation was faster as well as more scalable. Based on the results achieved we would like to conclude our project by saying that we achieved to implement a faster and more scalable recommendation system.

REFERENCES

1.Caroline,Ira Srivastava , Harsh Raj Singh , Vikram Chaurasiya , “ Movie Recommendation system KNN Algorithm and Collaborative Filtering”, JETIR, October 2018,Volume 5 , Issuse 10

2. N. Bhagirathi, P. Kiran , “Book Recommendation system using KNN algorithm”, International journal of Research in Engineering Science and Mangement[IJRESM],Volume 2,Issuse-6,June-2019

3. Subasish Gosh, Nazmun Nahar , Mohammad Abdul Wahab, Munmun Biswas, Mohammad Shahadat Hossain, and Karl Andersson, “ Recommendation System for E-commerce using Alternating Least Squares (ALS) on Apache Spark”.

4. Sardianos, Christos & Papadatos, Grigorios & Varlamis, Iraklis. “Optimizing Parallel Collaborative Filtering Approaches for Improving Recommendation Systems Performance”. Information. [10. 155. 10.3390/info10050155] 2019.

5. Fadhel Aljunid, Mohammed & D H, Manjaiah. Movie Recommender System Based on Collaborative Filtering Using Apache Spark.[10.1007/978-981-13-1274-8_22].2019

6. Anthony Lawrence Caterini, Manda Winlaw , Michael B Hynes, Hans De Sterck, “Algorithmic Acceleration of Parallel ALS for Collaborative Filtering: Speeding up Distributed Big Data Recommendation in Spark”, August 2015

7. J. Sun *et al*., "A Parallel Recommender System Using a Collaborative Filtering Algorithm with Correntropy for Social Networks", IEEE Transactions on Network Science and Engineering ,in I vol. 7, no. 1, pp. 91-103, 1 Jan.-March 2020.

8. Manoj Kumar, Yadav ,Ankur ,Vijay Kr. Gupta, “A Movie Recommender System: MOVREC”, International Journal of Computer Applications ,Volume 124 – No.3, August 2015

9. Vilakone, P., Park, DS., Xinchang, K. et al. “An Efficient movie recommendation algorithm based on improved k-clique”. Human Centric Computing Information Science. 8, article no-38 (2018).

10. Ke Ma,” Content-based Recommender System for Movie Website”, 2016