Dear student, you have not submitted the below files -

1. top\_5.p

2. top\_10.p

3. top\_20.p

4. user\_item\_matrix.p

5. project\_tests.py

Please submit the above files along with the .html and jupyter notebook. You can also use the `submit` button to submit the project from udacity workspace. All the best.

here your solution does not match with the expected results. You should have got

```Python

The 10 most similar users to user 1 are: [3933, 23, 3782, 203, 4459, 131, 3870, 46, 4201, 49]

The 5 most similar users to user 3933 are: [1, 23, 3782, 203, 4459]

The 3 most similar users to user 46 are: [4201, 23, 3782]

```

instead of -

```Python

The 10 most similar users to user 1 are: [2, 49, 3764, 98, 3697, 3782, 38, 21, 23, 5083]

The 5 most similar users to user 3933 are: [3934, 122, 3264, 126, 3172]

The 3 most similar users to user 46 are: [117, 2073, 131]

````

Below is a hint for you -

```Python

# compute similarity of each user to the provided user

similarity = []

for user in range(1, user\_item.shape[0]+1):

sim = USE NP.DOT TO COMPUTE SIMILARITY BETWEEN USER\_ID AND USER

similarity.append((user, sim))

# sort by similarity

similarity.sort(USE A LAMBDA FUNCTION HERE TO EXTRACT 2ND ELEMENT OF EACH ITEM IN SIMILARITY, reverse=True)

# create list of just the ids

most\_similar\_users = [USE A LIST COMPREHENSION TO EXTRACT 1ST ELEMENT OF EACH ITEM IN SIMILARITY]

# remove the own user's id

most\_similar\_users.remove(user\_id)

return most\_similar\_users

```

Dear student, although you have passed this test your implementation of this function is not exactly correct. In the function `get\_top\_sorted\_users` it seems you are pulling the number of unique articles a user has interacted with, but you actually want to count all interactions. For example, If user 2 interacts with article 12 three times, you need to count each of these, but currently you are counting it as one interaction (as you are using `user\_item` instead of 'df'). Below is a hint for you -

```Python

# Your code here

df\_article\_views = df.groupby('user\_id').count() # using `user\_item` will count all interactions as one interaction so we are using `df`

similarity = []

for user in range(1, user\_item.shape[0]+1):

sim = np.dot(user\_item.loc[user\_id], user\_item.loc[user])

similarity.append((user, sim))

# sort by similarity

similarity.sort(key=lambda x: x[1], reverse=True)

# create dataframe

df\_sims = pd.DataFrame()

df\_sims['user\_id'] = [x[0] for x in similarity]

df\_sims['similarity'] = [x[1] for x in similarity]

df\_sims = df\_sims.set\_index('user\_id')

# dataframe with users sorted by closest followed by most articles viewed

neighbors\_df = YOUR CODE HERE (merge `df\_sims` and `df\_article\_views`)

neighbors\_df = neighbors\_df[['similarity', 'article\_id']]

neighbors\_df = neighbors\_df.reset\_index()

neighbors\_df.columns = [['neighbor\_id', 'similarity', 'num\_articles']]

neighbors\_df = neighbors\_df.sort\_values(by=['similarity', 'num\_articles'], ascending=False)

return neighbors\_df

```

You have not answered this question. You answer this question in `words` as to why have you used the function `get\_top\_article\_ids` to recommend articles for new users? Please write down your answer in the space provided to you in the notebook as \*\*Provide your response here.\*\*

Dear student, you have not answered this question. Please answer this question in words explaining why we can use the `SVD` here, whereas in the lessons we were not able to use SVD and rather used `FunkSVD`. Please write down your answer in the space provided to you in the notebook as \*\*Provide your response here.\*\*

Dear student, here it is not clear how you got the values for \*\*a\*\*, \*\*b\*\*, \*\*c\*\* and \*\*d\*\*, Try to answer this as shown below -

```Python

a = len(test\_idx) - user\_item\_test.shape[0]

b = YOUR CODE HERE

c = user\_item\_test.shape[0]

d = YOUR CODE HERE

```

Nice write up here, but your answer is incomplete. Here you need to talk about the why this model predicts really well based on accuracy? And also you need to explain a method by which you could test how well your recommendation engine is working in practice to further engage users? Hint: use your knowledge of A/B testing

1. After fitting SVD on the user\_item\_train matrix, find out which rows match the test set from u,s,v

```Python

row\_idxs = user\_item\_train.index.isin(test\_idx)

col\_idxs = YOUR CODE HERE

u\_test = u\_train[row\_idxs, :]

vt\_test = YOUR CODE HERE

```

2. Then inside the for loop, restructure with k latent features

```Python

s\_train\_lat, u\_train\_lat, vt\_train\_lat = np.diag(s\_train[:k]), u\_train[:, :k], vt\_train[:k, :]

u\_test\_lat, vt\_test\_lat = YOUR CODE HERE

```

3. Take dot product

```Python

user\_item\_train\_preds = np.around(np.dot(np.dot(u\_train\_lat, s\_train\_lat), vt\_train\_lat))

user\_item\_test\_preds = YOUR CODE HERE

all\_errs.append(1 - ((np.sum(user\_item\_test\_preds)+np.sum(np.sum(user\_item\_test)))/(user\_item\_test.shape[0]\*user\_item\_test.shape[1])))

```

4. Compute error for each prediction to actual value

```Python

diffs\_train = np.subtract(user\_item\_train, user\_item\_train\_preds)

diffs\_test = YOUR CODE HERE

```

5. Calculate total errors and keep track of them

```Python

err\_train = np.sum(np.sum(np.abs(diffs\_train)))

err\_test = YOUR CODE HERE

```

6. Lastly plot the accuracy vs number of latent features

Once you successfully plot the points you should a get a plot like this –

Excellent first attempt! You are very close to completing this project, only one specification require changes and I am sure you will be able to successfully pass the project in your next attempt.

I have provided you hints on how to make those changes, In case you have any doubt implementing the answers please connect with your mentors on the [Study Hub](https://study-hall.udacity.com). Mentors will be happy to help you :smile:

You can also try out the `content based recommendation` system if you are interested. There are many ways to solve this problem. But you should have some basic knowledge of NLP. One such way you can try is that - you can use article titles as the content base to do NLP and tf-idf transformation for getting a article content matrix. Then use this matrix to calculate article-article similarities, below is a simple logic which you can try out -

1. Get user viewed article ids

2. Get similar articles to viewed articles (here you can use cosine similarity on the content matrix)

3. Sort by viewer #s and get top articles

4. If avg score is same then recommend based upon most viewed articles

I hope this would give you some idea.

Also, I would like to get your :star: ratings for this review, this would help us to improve our reviews in future. All The Best :thumbsup:

Congratulations! :tada: :fireworks:

You have successfully completed the project. Awesome work! But you should keep learning. There is lots of new research is going on, specially on the use of neural networks in the recommender systems. You should keep yourself updated.

Keep learning and don't forget to rate this review :smile:

All the best :thumbsup: