Part one: Introduction

1. The specific task planned to carry out in this project;
2. The Importance of the task;
3. The advantages of introducing the content-based algorithm and collaborative filtering algorithm for recommender system;
4. List the references for this part;

<u>Specific tasks in order to carry out in this project:</u>

Step 1:

To start with, in order to build a effective recommender system which can recommend jokes to the target users who believe they are funny, we first must understand what methods can do these tasks for us. As a result, the first task of our project is to do researches about recommender system and to have a deep understanding about the different and popular recommender systems that are available.

Step 2:

After searching and understand the contents and working principles about recommender system models, the next task to be carried out is to choose the ones that can finish our specific goal --- to recommend jokes to whom is likely to enjoy them. In order to finish this task, we need to be able to interpret the principles of the recommender system and understand the contents as well as the structures of the dataset --- the jokes and its ratings.

Step 3:

When the working principles of the recommender system we want are chosen, the next step is to realize our ideas and designs of the recommender system. For this specific task, according to the data structure and system functions that we require, the content-based algorithm and collaborative filtering algorithm recommender system are supposed to be able to carry our tasks. Consequently, in this stage of project, we need to customize our content-based and collaborative filtering algorithms according to the structure and contents of the joke dataset in order to make our design function in the real world.

Step 4:

After we have functional content-based and collaborative filtering algorithm based recommender system programs, the next thing on the list is to test and analyze the result and accuracy of the recommender system we build for accomplishing the task. Since we have a large data set available already, there are multiple ways to test the program and result and the details will be discussed in the later parts of this report.

Step 5:

Throughout the step 4 process I just talk about, we would be able to get an general idea and results about the effectiveness and accuracy of our recommendation program. At this point, we need to fix the problems we find that affects the test results negatively. Moreover, by testing the data with the test datasets we will be able to correctly estimate the performance of the recommendation program and make improvements on the recommender system we have if possible.

Importance:

Throughout the whole process of building the recommender system. I believe that the step 2 & 3 are the most important and time consuming stages in the project. This is because in order to build a recommender system, we have to understand how the system works and what methods will make it work. That is why the research and learning process is very important in accomplishing this task. Moreover, the realization process is the crucial staging in completing the tasks. During the realization process, we need to put the understanding and knowledge we learnt into practice and solve the problem we encounter in the real world.

The advantages of content-based algorithm and collaborative filtering algorithm for recommender system:

The advantages of content-based filtering algorithm. The reason why we believe content-based filtering algorithm would work effectively for this task at the first place is because the content-based filtering algorithm uses the attributes of the content to recommend similar content. As for our dataset, the jokes are has the content that is in the most explicit format, sentences of words. Based on the nature of the content of our data, it is reasonable to assume that content-based filtering algorithm has advantage in finishing this joke recommendation task. Moreover, when applying content-based filtering method, we do not have to deal with the cold-start problem. This is because the program will directly walk through attributes or tags of the content, such as key words, consequently, the system can generate the recommendation right away when given the input in the required format.

The advantage of collaborative-filtering stands out when you can access large amount of explicit user's ratings towards the content of interest. Since we have around 600 thousands ratings available in the dataset, applying user-based collaborative-filtering algorithms would likely generate the results with satisfying performance and accuracy due to the size of the data we have. As a result, user-based collaborative-filtering has advantage in accomplishing our goals in this task. However, user-based collaborative filtering has an obvious drawback which is the cold-starting problem. For example, if there is a new joke that has not received any ratings or enough ratings yet, it will never shows up on the recommendation list generated by this algorithm, even if the target user like the content very much.

Reference for part 1:

Heige Reikeras, 2018, https://www.offerzen.com/blog/how-to-build-a-content-based-recommender-system-for-your-product

Badrul Sarwar, George Karypis, Joseph Konstan, John Riedl, http://www.ra.ethz.ch/cdstore/www10/papers/pdf/p519.pdf

Part two: Related works

During the research process of building the joke recommender system, we also find many other algorithms and methods that are supposed to complete the tasks alos. In this part, I will briefly introduce these methods that are performed by other researchers.

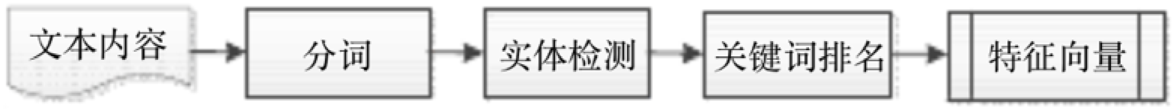
Related Work

For the other recommend system, use a News Recommend System as an example, in their system include three steps to achieved the task.

Step 1: News feature extraction. Extracting some features from news content for structured display of news.

Step 2: User portrait, also named user modeling, using a feature data of news that a user like (or dislike like) in the past to learn the user's preference features.

Step 3: Recommendation generation. By calculating the feature similarity between the user portrait and the candidate news, we recommend a set of news that is most match with the user's preferences.



In order to get the feature value of news, firstly need to segment the news. System use VSM method. In the model, each article is represented as an n-dimensional vector, and each dimension corresponds to one of the dictionaries. At this time, the article will be represented as a vector containing the weight of each word.

User modeling can also be seen as a process of binary classification, where each text is categorized as likes and dislikes. Therefore, we have a categorization mark C={c+, c-}, where c+ represents the positive text class and c- represents the negative text class.

PU learning is a series of probability deductions. First, we need to generate reasonable negative data, and then use these negative data to classify. There are many methods to generate negative data, such as random selection of some data from unlabeled data as negative data. Based on this, the classifier is constructed. If the negative data is selected randomly, the results are usually good. The most famous method is SPY algorithm, which classifies a small part of positive data as negative data. In this way, some common classifiers are used to classify the positive data, and unlabeled is used to classify the negative data.

Recommendation is the process of applying classifiers derived from user modeling into analyze unknown news. Good results can be achieved by recommending news whose user interest prediction value is higher than a fixed value to users.

In order to deal with complex processes, the system is divided into three modules: training module, recommendation module and server module.

First, we need to get training news resources and store them on the news server. Then, the training module gets training instructions from the news server and starts its training cycle. In training, we need to use the internal news training module and user training module.

User module needs to construct user matrix from user data according to historical browsing news data, and then construct classifier for recommendation module to recommend to users.

The news server delivers the collected news to the recommendation module. The recommendation module uses the classifier prepared by the training module to recommend users, and stores the results in the database, then feed back to the news server, as give feedback to the client.

Recommendation module needs to get data from news server, get classifier from training module, and use classifier to forecast news data. In this module, we need to deal with the problem of transforming the matrix data of training results into the available recommendation results to the news server.

References for part two:

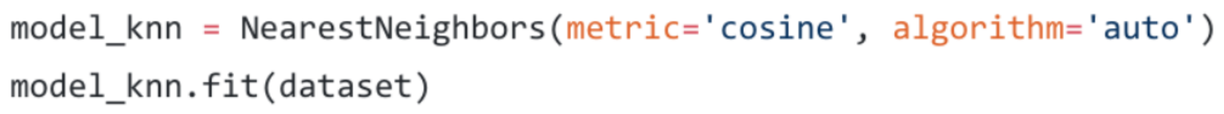
1. Sep. 2015, Dai Chenxu, & Zhou Xichen, Content-based news recommendation system. Retrieved from: https://wenku.baidu.com/view/79ed547359fb770bf78a6529647d27284b7337aa.html?rec_flag=default&sxts=1556457569908

Part 3: Methodology

The motivation of using collaborative-filtering and content-based method, as well as their advantages are discussed in detail in the last section of part 1 of this report.

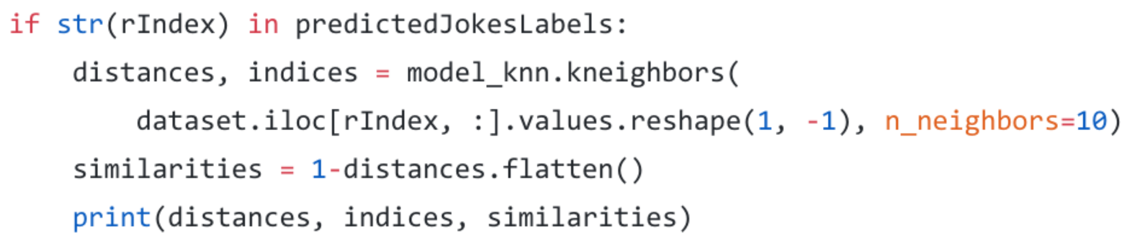
Details of the Collaborative-filtering method we apply:

For collaborative-filtering method, we choose to use user-user based collaborative-filtering method. For a joke that a user has not rated yet, we load the user rating matrix for all the users from the training dataset and iteration every line through the rating matrix (in this rating matrix, a line is a user and his/her ratings). Applying KNN method while iterating the rating matrix to find the most similar K users to the target user.



```
model_knn = NearestNeighbors(metric='cosine', algorithm='auto')
model_knn.fit(dataset)
```

To be specific, the first line of the rating matrix is the ratings of the first user, which can be treated as a vector. We apply KNN method to compare this vector with the whole rating matrix to find the k most similar users. For KNN method, we use cosine distance to measure the distance between users, and cosine distance = 1- cosine similarity.

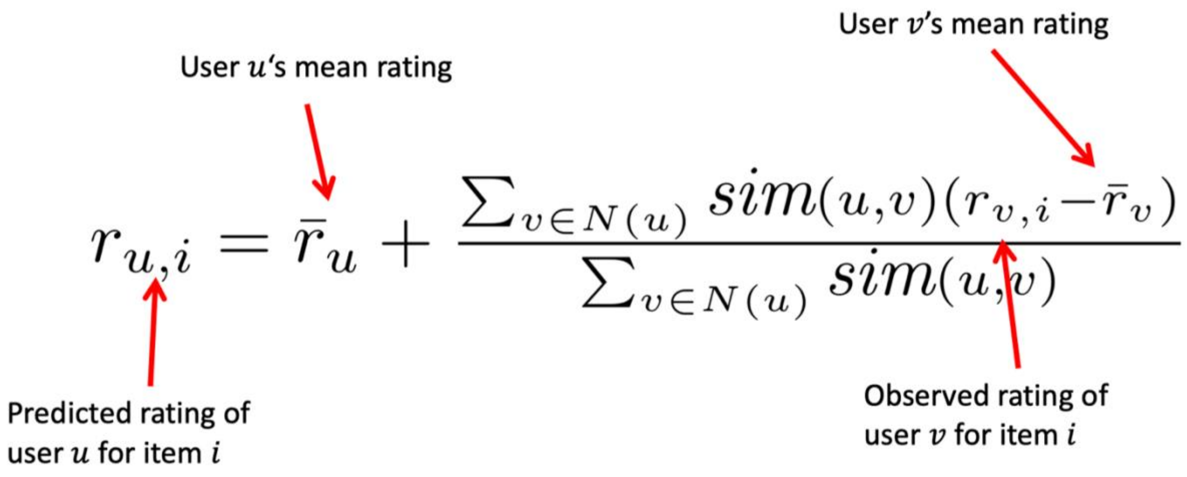


```
if str(rIndex) in predictedJokesLabels:
    distances, indices = model_knn.kneighbors(
        dataset.iloc[rIndex, :].values.reshape(1, -1), n_neighbors=10)
    similarities = 1-distances.flatten()
    print(distances, indices, similarities)
```

For our example, assume K is 10, with cosine distance applied, we will find the “10” most similar users to the target user when using the target user’s rating (the vector) as input. It is noteworthy that the 10 most similar users include himself and the cosine distance would be 0 in this case. Consequently, we will find the 9 most similar users and have their cosine distance to the target user, as well as their cosine similarity compared to the target user. The next step is to predict the rating that would be rated by the target user:

Assuming the target user is user0 and the rest 9 users are (user1 to user9), given a specific film A. User0’s predicted rating on film A is calculated as:

User0’ mean rating + {(User1’s similarity * (user1’s rating on A – user1’s mean rating) +user2’s similarity *(user2’s rating on A - user2’s mean rating) +…. User9’s similarity *(user9’s rating on A – user9’s mean rating)) / 9 most similar users’ sum of similarity.



$$r_{u,i} = \bar{r}_u + \frac{\sum_{v \in N(u)} sim(u,v)(r_{v,i} - \bar{r}_v)}{\sum_{v \in N(u)} sim(u,v)}$$

User u's mean rating
User v's mean rating
Predicted rating of user u for item i
Observed rating of user v for item i

With this method applied, we can predict all the unrated jokes' rating for all of our targeted users.

```
        product = ratings_diff * (similarities[i])
        wtd_sum = wtd_sum + product
prediction = round((mean_rating + (wtd_sum/sum_wt)), 2)
dataset_test.at[rIndex, label] = prediction
```

Details of the Content-based method we apply:

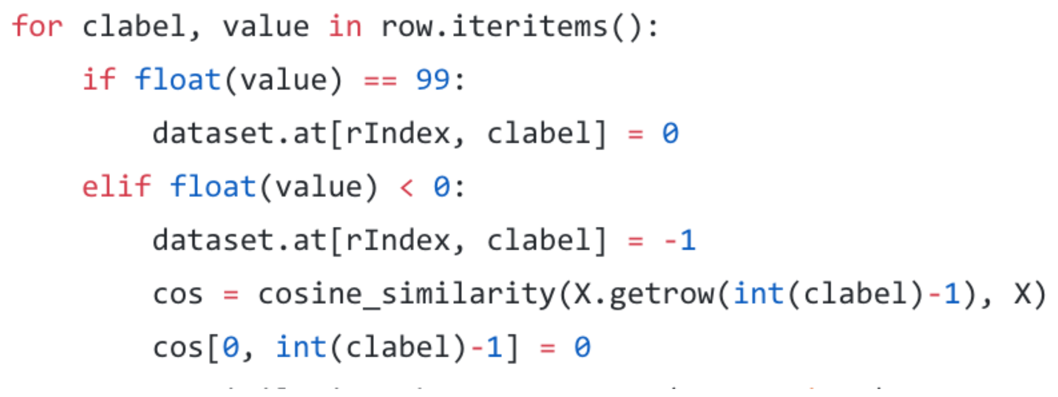
Firstly, extract the jokes and remove the stop-words. Then, transform the jokes into a tf-idf matrix.

```
vectorizer = TfidfVectorizer(stop_words='english')
X = vectorizer.fit_transform(jokes)
```

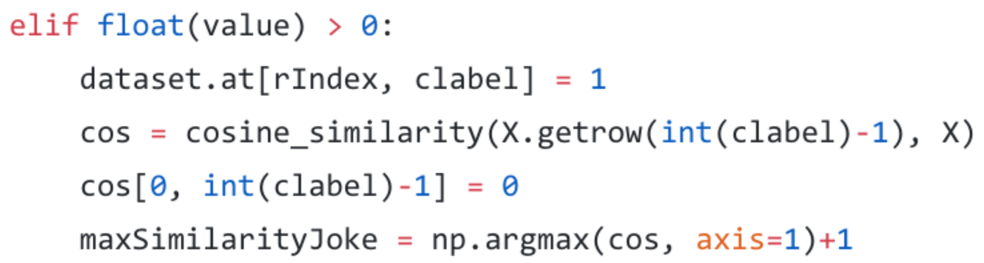
Taking user1 for example, if his rating for joke1 is greater than 0, we assume he likes this joke and locate the joke in the tf-idf matrix, which is a line in the tf-idf matrix.

After that, we compare this line with all the lines in the tf-idf matrix by calculating the cosine similarity. We extract the one with the highest cosine similarity and treat it as "liked" and locate it in the new matrix that indicates the ratings of the users but contains only zeros. In this new rating matrix, we mark the ratings of the joke we just find as “1”, which represents “liked”. Similarly, if the user rates one joke below zero, we treat this joke as “dislike” and find the most similar one from the tf-idf matrix , find its corresponding location at the user’s rating matrix and mark the “0” to “-1”. When the rating of a joke is “99”, we do nothing so at the corresponding location of this joke in the user’s rating matrix is 0 as still.

After iterating through all the jokes that rated by the user, we have updated the user’s rating matrix and it contains only 1, 0, -1. In this new matrix we updated, 1 stands for “like”, 0 stands for “unrated” and -1 stands for “dislike”.



```
for clabel, value in row.iteritems():
    if float(value) == 99:
        dataset.at[rIndex, clabel] = 0
    elif float(value) < 0:
        dataset.at[rIndex, clabel] = -1
        cos = cosine_similarity(X.getrow(int(clabel)-1), X)
        cos[0, int(clabel)-1] = 0
```



```
elif float(value) > 0:
    dataset.at[rIndex, clabel] = 1
    cos = cosine_similarity(X.getrow(int(clabel)-1), X)
    cos[0, int(clabel)-1] = 0
    maxSimilarityJoke = np.argmax(cos, axis=1)+1
```

Next, we transform the original rating matrix to a matrix only contains 1, 0, -1 also. The rules are the same, >0 =1; 99 =0; <0 =-1.

Next step: Starting Iterating each line of the two matrix at the same time, find all the “0” in the two matrix and remove them.

Now we have two matrix that have the same dimension and size, we can compare the similarity between the two matrix and get the evaluations of the performance of our content-based recommendation system.

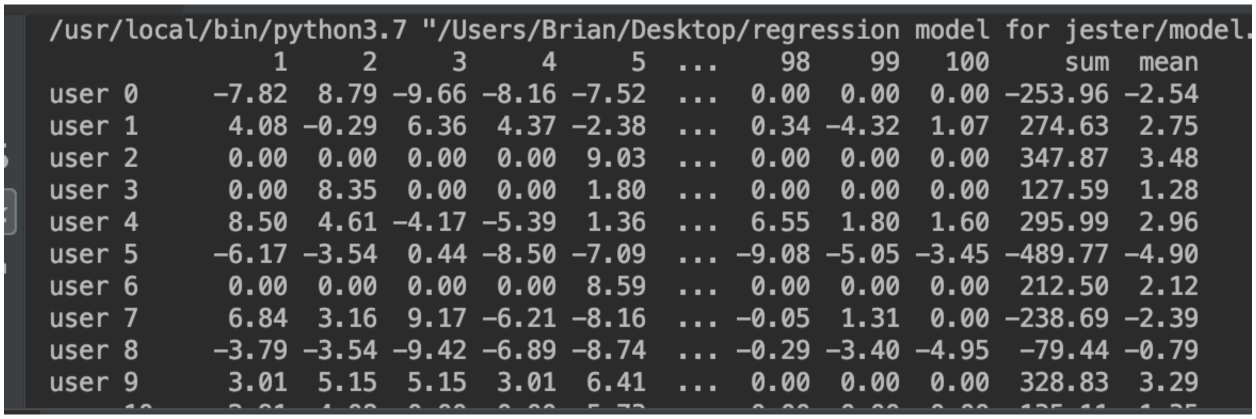
Part 4: Data Description

- Introduce the details of the datasets, e.g., data statistics;
- Explain how to partition the original datasets into the training data and test data;

For this project , we choose to use the Dataset one that is provided by Berkeley's website. This dataset contains 4.1 million continuous ratings ranges from (-10 to 10) which are collected from 73421 users from April 1999 to May 2003. The dataset is further divided into three sub datasets that include approximately the same amount of users. To be specific, the first sub-dataset has the ratings from 24983 users who rated 36 or more jokes (a matrix with dimensions 24893 * 101). The second has ratings from 23500 users who rated 36 or more jokes. The third one has the ratings from 24938 users who has rated between 15 and 35 jokes.

Looking carefully at the dataset, the first column shows how many jokes out of 100 have been rated by the user, the rest of the row indicate the ratings of the jokes from 1 to 100 rated by the user. The value "99" under a joke means this user has not rated this joke. Given a rated joke, the ratings ranges from -10 to 10.

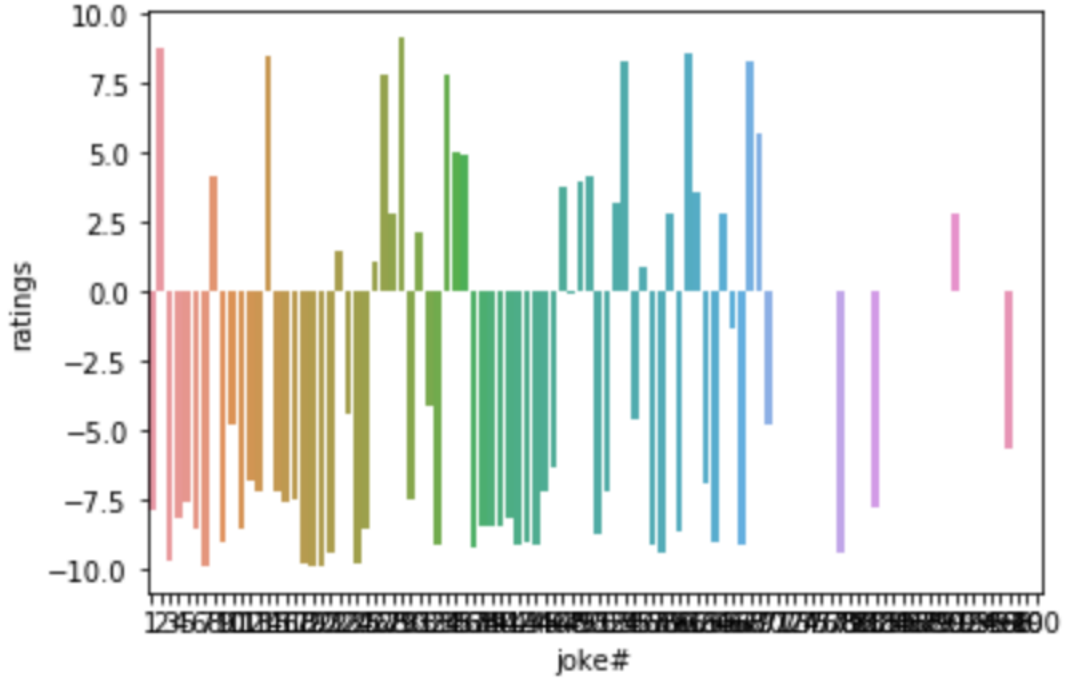
To better understand the dataset, I replaced all the "99" with "0" in the dataset 1 in order to show some relative statistics about the dataset.



```
/usr/local/bin/python3.7 "/Users/Brian/Desktop/regression model for jester/model.
             1     2     3     4     5  ...    98    99   100     sum  mean
user 0   -7.82  8.79 -9.66 -8.16 -7.52  ...  0.00  0.00  0.00 -253.96 -2.54
user 1    4.08 -0.29  6.36  4.37 -2.38  ...  0.34 -4.32  1.07  274.63  2.75
user 2    0.00  0.00  0.00  0.00  9.03  ...  0.00  0.00  0.00  347.87  3.48
user 3    0.00  8.35  0.00  0.00  1.80  ...  0.00  0.00  0.00  127.59  1.28
user 4    8.50  4.61 -4.17 -5.39  1.36  ...  6.55  1.80  1.60  295.99  2.96
user 5   -6.17 -3.54  0.44 -8.50 -7.09  ... -9.08 -5.05 -3.45 -489.77 -4.90
user 6    0.00  0.00  0.00  0.00  8.59  ...  0.00  0.00  0.00  212.50  2.12
user 7    6.84  3.16  9.17 -6.21 -8.16  ... -0.05  1.31  0.00 -238.69 -2.39
user 8   -3.79 -3.54 -9.42 -6.89 -8.74  ... -0.29 -3.40 -4.95  -79.44 -0.79
user 9    3.01  5.15  5.15  3.01  6.41  ...  0.00  0.00  0.00  328.83  3.29
```

From the dataset above, we can find that the joke ratings from the users are very subjective, some users generally like the 100 jokes more while others do not, which can be judged from the positive and negative sign of the “sum” or “mean” column.

Take user0’ s ratings of the 100 jokes for example:



Partition the original datasets into the training data and test data:

For the three sub-datasets, we split each dataset into 90% and 10% size of the original dataset. Taking the 90% part as training dataset and the 10% part for testing. The reason why we apply this partition method is because the data amount in each sub-dataset is large, as a result, we can use only 10% of each dataset as the testing dataset but still making sure that a considerable amount of data are tested for our recommendation system. Moreover, since there are three sub-datasets in dataset1, we can evaluate our recommendation system triple times in order to better understand the performance of the recommendation system.

Part 5: Results

{Include:

- Provide model settings that you used in the experiments;
- Show the results on all databases in terms of some evaluation metrics;
- Explain and discuss the results to demonstrate the effectiveness of the algorithms.}

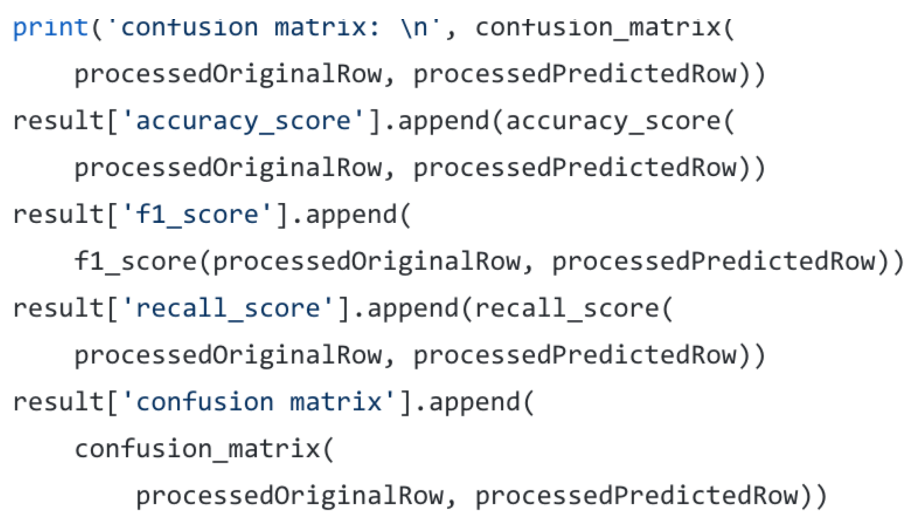
For content-based filtering method:

Taking the 10% percent of each dataset as testing dataset:

When applying the method I just discuss above in part 3, we will have a predicted new rating matrix based on joke content similarity (tf-idf), showing only “-1”, “0” and “1” which represents the target user’s attitude towards the 100 jokes.

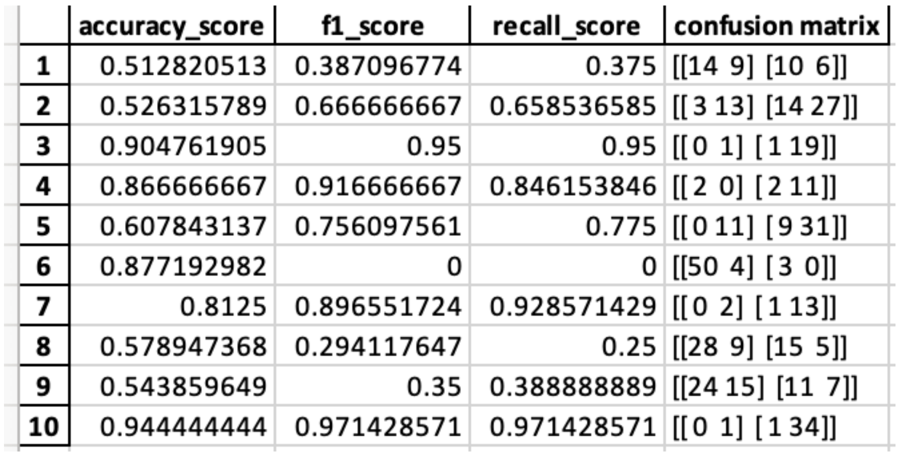
To evaluate the performance of the recommendation system, we transform the user’s original rating matrix into the form that is as same as our predicted matrix.

The next step is to compare the two matrix to evaluate the system’s performance:



```
print('confusion matrix: \n', confusion_matrix(
    processedOriginalRow, processedPredictedRow))
result['accuracy_score'].append(accuracy_score(
    processedOriginalRow, processedPredictedRow))
result['f1_score'].append(
    f1_score(processedOriginalRow, processedPredictedRow))
result['recall_score'].append(recall_score(
    processedOriginalRow, processedPredictedRow))
result['confusion matrix'].append(
    confusion_matrix(
        processedOriginalRow, processedPredictedRow))
```

Processing the testing dataset: (Screenshot of the first ten users' rating prediction)



	accuracy_score	f1_score	recall_score	confusion matrix
1	0.512820513	0.387096774	0.375	[[14 9] [10 6]]
2	0.526315789	0.666666667	0.658536585	[[3 13] [14 27]]
3	0.904761905	0.95	0.95	[[0 1] [1 19]]
4	0.866666667	0.916666667	0.846153846	[[2 0] [2 11]]
5	0.607843137	0.756097561	0.775	[[0 11] [9 31]]
6	0.877192982	0	0	[[50 4] [3 0]]
7	0.8125	0.896551724	0.928571429	[[0 2] [1 13]]
8	0.578947368	0.294117647	0.25	[[28 9] [15 5]]
9	0.543859649	0.35	0.388888889	[[24 15] [11 7]]
10	0.944444444	0.971428571	0.971428571	[[0 1] [1 34]]

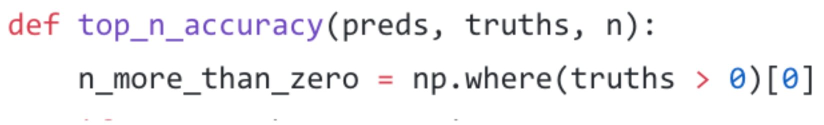
*From the Accuracy score, F1score and recall score we can have a general idea of how the system perform on making recommendation to users based on the content he/she likes. With "1.00" being 100% correct recommendation.

Collaborative Filtering (User-User based):
After building the recommendation system, taking the 10% as the training dataset:

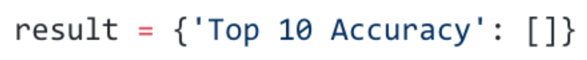
With the method mentioned in part 3 applied, we can predict all the unrated jokes' rating for all of our targeted users.

To evluate the performance of the recommendation system, we process the 10% users' rating with our recommendation system to generate a new rating matrix. By comparing the generated matrix with the original matrix, we can have a result of the accuracy performance of the collaborative filtering system.

Top k accuracy:

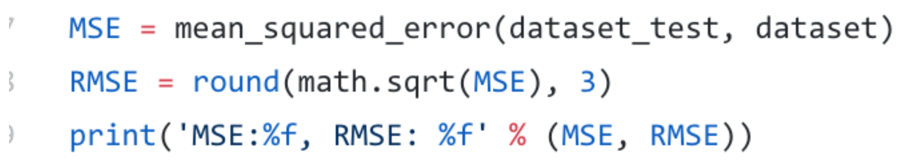


```
def top_n_accuracy(preds, truths, n):
    n_more_than_zero = np.where(truths > 0)[0]
```

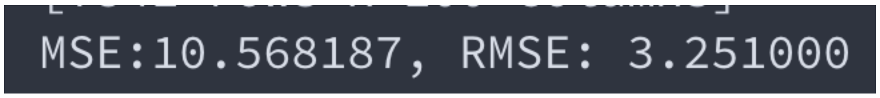


```
result = {'Top 10 Accuracy': []}
```

And MSE, RMSE:



```
MSE = mean_squared_error(dataset_test, dataset)
RMSE = round(math.sqrt(MSE), 3)
print('MSE:%f, RMSE: %f' % (MSE, RMSE))
```



```
MSE:10.568187, RMSE: 3.251000
```

For the rating range of the data in this dataset (-10 : 10), the absolute measure of fitting the original dataset is : 3.251000.

Part 6: Limitation of current study and future works

Limitation of current study:

First of all, our project uses a single system approach to do the recommendation, content-based or collaborative filtering. But for each recommendation must be some shortcomings. (May. 2015, F.O. Isinkaye &Y. O. Folajimo). For content-base, the disadvantage is that the content need to be easily extracted into meaningful features. Feature content required to have good structured and user interests must be expressed in feature form. And for collaborative filtering, the system initially recommended bad jokes to new users, also the sparsity of data will degrade the recommendation quality. So, using the single recommendation system is a limitation of our current study.

Secondly, the data set is also relatively simple, one is all grades, one is all jokes. If the dataset can have some dimensions （For example, the type of picture or text of the joke or the scope of the joke) .For the multidimensional dataset, developers can easily use it to describe complex real-world structures without ignoring real-world problems or forcing them into a technically manageable form, and the multidimensional data model greatly reduces the time to perform complex processing. The various aspects of the data set can be immediately mapped to each other, the acquisition of data is extremely fast, and due to the elimination of redundant data, the multidimensional dataset become simpler and more economical. (Jan. 2017, Xiang Zhou, Dan Xu & Jiang Ding-Ding)

Future improvements:

If there is a larger project scale further, we can adopt a larger dataset to do the project. ex. Use API to get data source from other companies (Dec. 2014, The advantage of APIs). Secondly, choose the Hybrid recommendation system instead of the single recommend system that mentioned above (Robin Burke). Since all kinds of recommendation methods have advantages and disadvantages, Hybrid Recommendation is usually adopted in practice. The combination of content recommendation and collaborative filtering recommendation has been studied and applied most. The simplest approach is to use content-based methods and collaborative filtering recommendation methods respectively to generate a recommendation prediction result, and then combine the results. Although there are many methods of recommendation combination in theory, they are not always effective in a specific problem. One of the best advantages of recommendation system is can avoid or remedy the weaknesses of each single recommendation.

Reference for part 6:

1. Robin Burke, Hybrid recommendation system. Retrieved from: https://link.springer.com/chapter/10.1007/978-3-540-72079-9_12
2. May. 2015, F.O. Isinkaye &Y. O. Folajimo. Recommendation systems: Principles, methods and evaluation. Retrieved from: https://www.sciencedirect.com/science/article/pii/S1110866515000341
3. Dec. 2014, The advantage of APIs. Retrieved from: https://www.jisc.ac.uk/guides/the-advantage-of-apis
4. Jan. 2017, Xiang Zhou, Dan Xu & Jiang Ding-Ding, Simplifying multidimensional fermentation dataset analysis and visualization: One step closer to capturing high-quality mutant strains. Retrieved from: https://www.nature.com/articles/srep39875/tables/

Reference:

Reference for part 1:

1.Heige Reikeras, 2018, Retrieved from:https://www.offerzen.com/blog/how-to-build-a-content-based-recommender-system-for-your-product

2.Badrul Sarwar, George Karypis, Joseph Konstan, John Riedl, Retrieved from:http://www.ra.ethz.ch/cdstore/www10/papers/pdf/p519.pdf

References for part 2:

1. Sep. 2015, Dai Chenxu, & Zhou Xichen, Content-based news recommendation system.

Retrieved from:

https://wenku.baidu.com/view/79ed547359fb770bf78a6529647d27284b7337aa.html?rec_flag=default&sxts=1556457569908

Reference for part 6:

1. Robin Burke, Hybrid recommendation system. Retrieved from: https://link.springer.com/chapter/10.1007/978-3-540-72079-9_12
2. May. 2015, F.O. Isinkaye &Y. O. Folajimo. Recommendation systems: Principles, methods and evaluation. Retrieved from: https://www.sciencedirect.com/science/article/pii/S1110866515000341

3. Dec. 2014, The advantage of APIs. Retrieved from: https://www.jisc.ac.uk/guides/the-advantage-of-apis
4. Jan. 2017, Xiang Zhou, Dan Xu & Jiang Ding-Ding,
Simplifying multidimensional fermentation dataset analysis and visualization: One step closer to capturing high-quality mutant strains. Retrieved from:
https://www.nature.com/articles/srep39875/tables/