# Practice lab: Deep Learning for Content-Based Filtering

In this exercise, you will implement content-based filtering using a neural network to build a recommender system for movies.

# Outline



***NOTE:** To prevent errors from the autograder, you are not allowed to edit or delete non-graded cells in this lab. Please also refrain from adding any new cells. **Once you have passed this assignment** and want to experiment with any of the non-graded code, you may follow the instructions at the bottom of this notebook.*

## 1 - Packages

We will use familiar packages, NumPy, TensorFlow and helpful routines from scikit-learn. We will also use tabulate to neatly print tables and Pandas to organize tabular data.

```
import numpy as np
import numpy.ma as ma
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split
import tabulate
```

```
from recsysNN_utils import *
pd.set_option("display.precision", 1)
```

# 2 - Movie ratings dataset

The data set is derived from the MovieLens ml-latest-small dataset.

[F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. https://doi.org/10.1145/2827872]

The original dataset has roughly 9000 movies rated by 600 users with ratings on a scale of 0.5 to 5 in 0.5 step increments. The dataset has been reduced in size to focus on movies from the years since 2000 and popular genres. The reduced dataset has $n_u = 397$ users, $n_m = 847$ movies and 25521 ratings. For each movie, the dataset provides a movie title, release date, and one or more genres. For example "Toy Story 3" was released in 2010 and has several genres: "Adventure|Animation|Children|Comedy|Fantasy". This dataset contains little information about users other than their ratings. This dataset is used to create training vectors for the neural networks described below. Let's learn a bit more about this data set. The table below shows the top 10 movies ranked by the number of ratings. These movies also happen to have high average ratings. How many of these movies have you watched?

```
top10_df = pd.read_csv("./data/content_top10_df.csv")
bygenre_df = pd.read_csv("./data/content_bygenre_df.csv")
top10_df
```

```
   movie id  num ratings  ave rating  \
0      4993          198         4.1
1      5952          188         4.0
2      7153          185         4.1
3      4306          170         3.9
4     58559          149         4.2
5      6539          149         3.8
6     79132          143         4.1
7      6377          141         4.0
8      4886          132         3.9
9      7361          131         4.2

                                              title  \
0  Lord of the Rings: The Fellowship of the Ring,...
1             Lord of the Rings: The Two Towers, The
2     Lord of the Rings: The Return of the King, The
3                                             Shrek
4                                 Dark Knight, The
5  Pirates of the Caribbean: The Curse of the Bla...
6                                         Inception
7                                      Finding Nemo
```

```
8                                       Monsters, Inc.
9                Eternal Sunshine of the Spotless Mind

                                                genres
0                                    Adventure|Fantasy
1                                    Adventure|Fantasy
2                       Action|Adventure|Drama|Fantasy
3   Adventure|Animation|Children|Comedy|Fantasy|Ro...
4                                   Action|Crime|Drama
5                      Action|Adventure|Comedy|Fantasy
6           Action|Crime|Drama|Mystery|Sci-Fi|Thriller
7                  Adventure|Animation|Children|Comedy
8          Adventure|Animation|Children|Comedy|Fantasy
9                                Drama|Romance|Sci-Fi
```

The next table shows information sorted by genre. The number of ratings per genre vary substantially. Note that a movie may have multiple genre's so the sum of the ratings below is larger than the number of original ratings.

```
bygenre_df
```

| | genre | num movies | ave rating/genre | ratings per genre |
|---|---|---|---|---|
| 0 | Action | 321 | 3.4 | 10377 |
| 1 | Adventure | 234 | 3.4 | 8785 |
| 2 | Animation | 76 | 3.6 | 2588 |
| 3 | Children | 69 | 3.4 | 2472 |
| 4 | Comedy | 326 | 3.4 | 8911 |
| 5 | Crime | 139 | 3.5 | 4671 |
| 6 | Documentary | 13 | 3.8 | 280 |
| 7 | Drama | 342 | 3.6 | 10201 |
| 8 | Fantasy | 124 | 3.4 | 4468 |
| 9 | Horror | 56 | 3.2 | 1345 |
| 10 | Mystery | 68 | 3.6 | 2497 |
| 11 | Romance | 151 | 3.4 | 4468 |
| 12 | Sci-Fi | 174 | 3.4 | 5894 |
| 13 | Thriller | 245 | 3.4 | 7659 |

# 3 - Content-based filtering with a neural network

In the collaborative filtering lab, you generated two vectors, a user vector and an item/movie vector whose dot product would predict a rating. The vectors were derived solely from the ratings.

Content-based filtering also generates a user and movie feature vector but recognizes there may be other information available about the user and/or movie that may improve the prediction. The additional information is provided to a neural network which then generates the user and movie vector as shown below.

## 3.1 Training Data

The movie content provided to the network is a combination of the original data and some 'engineered features'. Recall the feature engineering discussion and lab from Course 1, Week 2, lab 4. The original features are the year the movie was released and the movie's genre's presented as a one-hot vector. There are 14 genres. The engineered feature is an average rating derived from the user ratings.

The user content is composed of engineered features. A per genre average rating is computed per user. Additionally, a user id, rating count and rating average are available but not included in the training or prediction content. They are carried with the data set because they are useful in interpreting data.

The training set consists of all the ratings made by the users in the data set. Some ratings are repeated to boost the number of training examples of underrepresented genre's. The training set is split into two arrays with the same number of entries, a user array and a movie/item array.

Below, let's load and display some of the data.

```
# Load Data, set configuration variables
item_train, user_train, y_train, item_features, user_features,
item_vecs, movie_dict, user_to_genre = load_data()

num_user_features = user_train.shape[1] - 3  # remove userid, rating
count and ave rating during training
num_item_features = item_train.shape[1] - 1  # remove movie id at
train time
uvs = 3  # user genre vector start
ivs = 3  # item genre vector start
u_s = 3  # start of columns to use in training, user
i_s = 1  # start of columns to use in training, items
print(f"Number of training vectors: {len(item_train)}")
```

```
Number of training vectors: 50884
```

Let's look at the first few entries in the user training array.

```
pprint_train(user_train, user_features, uvs,  u_s, maxcount=5)
```

```
'<table>\n<thead>\n<tr><th style="text-align: center;"> [user id]
</th><th style="text-align: center;"> [rating count] </th><th
style="text-align: center;"> [rating ave] </th><th style="text-align:
center;"> Act ion </th><th style="text-align: center;"> Adve nture
</th><th style="text-align: center;"> Anim ation </th><th style="text-
align: center;"> Chil dren </th><th style="text-align: center;"> Com
edy </th><th style="text-align: center;"> Crime </th><th style="text-
align: center;"> Docum entary </th><th style="text-align: center;">
Drama </th><th style="text-align: center;"> Fan tasy </th><th
```

```
style="text-align: center;"> Hor ror </th><th style="text-align:
center;"> Mys tery </th><th style="text-align: center;"> Rom ance
</th><th style="text-align: center;"> Sci -Fi </th><th style="text-
align: center;"> Thri ller </th></tr>\n</thead>\n<tbody>\n<tr><td
style="text-align: center;">     2     </td><td style="text-align:
center;">       22       </td><td style="text-align: center;">     4.0
</td><td style="text-align: center;">   4.0   </td><td style="text-
align: center;">    4.2     </td><td style="text-align: center;">
0.0     </td><td style="text-align: center;">    0.0    </td><td
style="text-align: center;">   4.0   </td><td style="text-align:
center;">  4.1  </td><td style="text-align: center;">     4.0
</td><td style="text-align: center;">  4.0  </td><td style="text-
align: center;">   0.0    </td><td style="text-align: center;">   3.0
</td><td style="text-align: center;">   4.0    </td><td style="text-
align: center;">   0.0    </td><td style="text-align: center;">   3.9
</td><td style="text-align: center;">    3.9    </td></tr>\n<tr><td
style="text-align: center;">     2     </td><td style="text-align:
center;">       22       </td><td style="text-align: center;">     4.0
</td><td style="text-align: center;">   4.0   </td><td style="text-
align: center;">    4.2     </td><td style="text-align: center;">
0.0     </td><td style="text-align: center;">    0.0    </td><td
style="text-align: center;">   4.0   </td><td style="text-align:
center;">  4.1  </td><td style="text-align: center;">     4.0
</td><td style="text-align: center;">  4.0  </td><td style="text-
align: center;">   0.0    </td><td style="text-align: center;">   3.0
</td><td style="text-align: center;">   4.0    </td><td style="text-
align: center;">   0.0    </td><td style="text-align: center;">   3.9
</td><td style="text-align: center;">    3.9    </td></tr>\n<tr><td
style="text-align: center;">     2     </td><td style="text-align:
center;">       22       </td><td style="text-align: center;">     4.0
</td><td style="text-align: center;">   4.0   </td><td style="text-
align: center;">    4.2     </td><td style="text-align: center;">
0.0     </td><td style="text-align: center;">    0.0    </td><td
style="text-align: center;">   4.0   </td><td style="text-align:
center;">  4.1  </td><td style="text-align: center;">     4.0
</td><td style="text-align: center;">  4.0  </td><td style="text-
align: center;">   0.0    </td><td style="text-align: center;">   3.0
</td><td style="text-align: center;">   4.0    </td><td style="text-
align: center;">   0.0    </td><td style="text-align: center;">   3.9
</td><td style="text-align: center;">    3.9    </td></tr>\n<tr><td
style="text-align: center;">     2     </td><td style="text-align:
center;">       22       </td><td style="text-align: center;">     4.0
</td><td style="text-align: center;">   4.0   </td><td style="text-
align: center;">    4.2     </td><td style="text-align: center;">
0.0     </td><td style="text-align: center;">    0.0    </td><td
style="text-align: center;">   4.0   </td><td style="text-align:
center;">  4.1  </td><td style="text-align: center;">     4.0
</td><td style="text-align: center;">  4.0  </td><td style="text-
align: center;">   0.0    </td><td style="text-align: center;">   3.0
```

```
</td><td style="text-align: center;">   4.0    </td><td style="text-align: center;">   0.0    </td><td style="text-align: center;">   3.9
</td><td style="text-align: center;">    3.9    </td></tr>\n<tr><td style="text-align: center;">     2     </td><td style="text-align: center;">       22       </td><td style="text-align: center;">     4.0
</td><td style="text-align: center;">   4.0   </td><td style="text-align: center;">    4.2     </td><td style="text-align: center;">
0.0     </td><td style="text-align: center;">    0.0    </td><td style="text-align: center;">   4.0   </td><td style="text-align: center;">  4.1  </td><td style="text-align: center;">     4.0
</td><td style="text-align: center;">  4.0  </td><td style="text-align: center;">   0.0    </td><td style="text-align: center;">   3.0
</td><td style="text-align: center;">   4.0    </td><td style="text-align: center;">   0.0    </td><td style="text-align: center;">   3.9
</td><td style="text-align: center;">    3.9    </td></tr>\n</tbody>\n</table>'
```

Some of the user and item/movie features are not used in training. In the table above, the features in brackets "[]" such as the "user id", "rating count" and "rating ave" are not included when the model is trained and used. Above you can see the per genre rating average for user 2. Zero entries are genre's which the user had not rated. The user vector is the same for all the movies rated by a user.
Let's look at the first few entries of the movie/item array.

```
pprint_train(item_train, item_features, ivs, i_s, maxcount=5, user=False)
```

```
'<table>\n<thead>\n<tr><th style="text-align: center;"> [movie id] </th><th style="text-align: center;"> year </th><th style="text-align: center;"> ave rating </th><th style="text-align: center;"> Act ion </th><th style="text-align: center;"> Adve nture </th><th style="text-align: center;"> Anim ation </th><th style="text-align: center;"> Chil dren </th><th style="text-align: center;"> Com edy </th><th style="text-align: center;"> Crime </th><th style="text-align: center;"> Docum entary </th><th style="text-align: center;"> Drama </th><th style="text-align: center;"> Fan tasy </th><th style="text-align: center;"> Hor ror </th><th style="text-align: center;"> Mys tery </th><th style="text-align: center;"> Rom ance </th><th style="text-align: center;"> Sci -Fi </th><th style="text-align: center;"> Thri ller </th></tr>\n</thead>\n<tbody>\n<tr><td style="text-align: center;">    6874    </td><td style="text-align: center;"> 2003 </td><td style="text-align: center;">    4.0
</td><td style="text-align: center;">    1    </td><td style="text-align: center;">     0      </td><td style="text-align: center;">
0      </td><td style="text-align: center;">     0     </td><td style="text-align: center;">    0    </td><td style="text-align: center;">   1   </td><td style="text-align: center;">      0
</td><td style="text-align: center;">   0   </td><td style="text-align: center;">    0     </td><td style="text-align: center;">    0
```

```
</td><td style="text-align: center;">    0     </td><td style="text-
align: center;">    0     </td><td style="text-align: center;">    0
</td><td style="text-align: center;">     1     </td></tr>\n<tr><td
style="text-align: center;">    8798    </td><td style="text-align:
center;"> 2004 </td><td style="text-align: center;">    3.8
</td><td style="text-align: center;">    1    </td><td style="text-
align: center;">     0      </td><td style="text-align: center;">
0      </td><td style="text-align: center;">     0     </td><td
style="text-align: center;">    0    </td><td style="text-align:
center;">   1   </td><td style="text-align: center;">      0
</td><td style="text-align: center;">   1   </td><td style="text-
align: center;">    0     </td><td style="text-align: center;">    0
</td><td style="text-align: center;">    0     </td><td style="text-
align: center;">    0     </td><td style="text-align: center;">    0
</td><td style="text-align: center;">     1     </td></tr>\n<tr><td
style="text-align: center;">   46970    </td><td style="text-align:
center;"> 2006 </td><td style="text-align: center;">    3.2
</td><td style="text-align: center;">    1    </td><td style="text-
align: center;">     0      </td><td style="text-align: center;">
0      </td><td style="text-align: center;">     0     </td><td
style="text-align: center;">    1    </td><td style="text-align:
center;">   0   </td><td style="text-align: center;">      0
</td><td style="text-align: center;">   0   </td><td style="text-
align: center;">    0     </td><td style="text-align: center;">    0
</td><td style="text-align: center;">    0     </td><td style="text-
align: center;">    0     </td><td style="text-align: center;">    0
</td><td style="text-align: center;">     0     </td></tr>\n<tr><td
style="text-align: center;">   48516    </td><td style="text-align:
center;"> 2006 </td><td style="text-align: center;">    4.3
</td><td style="text-align: center;">    0    </td><td style="text-
align: center;">     0      </td><td style="text-align: center;">
0      </td><td style="text-align: center;">     0     </td><td
style="text-align: center;">    0    </td><td style="text-align:
center;">   1   </td><td style="text-align: center;">      0
</td><td style="text-align: center;">   1   </td><td style="text-
align: center;">    0     </td><td style="text-align: center;">    0
</td><td style="text-align: center;">    0     </td><td style="text-
align: center;">    0     </td><td style="text-align: center;">    0
</td><td style="text-align: center;">     1     </td></tr>\n<tr><td
style="text-align: center;">   58559    </td><td style="text-align:
center;"> 2008 </td><td style="text-align: center;">    4.2
</td><td style="text-align: center;">    1    </td><td style="text-
align: center;">     0      </td><td style="text-align: center;">
0      </td><td style="text-align: center;">     0     </td><td
style="text-align: center;">    0    </td><td style="text-align:
center;">   1   </td><td style="text-align: center;">      0
</td><td style="text-align: center;">   1   </td><td style="text-
align: center;">    0     </td><td style="text-align: center;">    0
</td><td style="text-align: center;">    0     </td><td style="text-
```

```
align: center;">    0     </td><td style="text-align: center;">    0
</td><td style="text-align: center;">     0     </td></tr>\n</tbody>\
n</table>'
```

Above, the movie array contains the year the film was released, the average rating and an indicator for each potential genre. The indicator is one for each genre that applies to the movie. The movie id is not used in training but is useful when interpreting the data.

```
print(f"y_train[:5]: {y_train[:5]}")
```

```
y_train[:5]: [4.  3.5 4.  4.  4.5]
```

The target, y, is the movie rating given by the user.

Above, we can see that movie 6874 is an Action/Crime/Thriller movie released in 2003. User 2 rates action movies as 3.9 on average. MovieLens users gave the movie an average rating of 4. 'y' is 4 indicating user 2 rated movie 6874 as a 4 as well. A single training example consists of a row from both the user and item arrays and a rating from y_train.

### 3.2 Preparing the training data

Recall in Course 1, Week 2, you explored feature scaling as a means of improving convergence. We'll scale the input features using the scikit learn StandardScaler. This was used in Course 1, Week 2, Lab 5. Below, the inverse_transform is also shown to produce the original inputs. We'll scale the target ratings using a Min Max Scaler which scales the target to be between -1 and 1. scikit learn MinMaxScaler

```
# scale training data
item_train_unscaled = item_train
user_train_unscaled = user_train
y_train_unscaled    = y_train

scalerItem = StandardScaler()
scalerItem.fit(item_train)
item_train = scalerItem.transform(item_train)

scalerUser = StandardScaler()
scalerUser.fit(user_train)
user_train = scalerUser.transform(user_train)

scalerTarget = MinMaxScaler((-1, 1))
scalerTarget.fit(y_train.reshape(-1, 1))
y_train = scalerTarget.transform(y_train.reshape(-1, 1))
#ynorm_test = scalerTarget.transform(y_test.reshape(-1, 1))

print(np.allclose(item_train_unscaled,
scalerItem.inverse_transform(item_train)))
```

```
print(np.allclose(user_train_unscaled,
scalerUser.inverse_transform(user_train)))

True
True
```

To allow us to evaluate the results, we will split the data into training and test sets as was discussed in Course 2, Week 3. Here we will use sklearn train_test_split to split and shuffle the data. Note that setting the initial random state to the same value ensures item, user, and y are shuffled identically.

```
item_train, item_test = train_test_split(item_train, train_size=0.80,
shuffle=True, random_state=1)
user_train, user_test = train_test_split(user_train, train_size=0.80,
shuffle=True, random_state=1)
y_train, y_test       = train_test_split(y_train,    train_size=0.80,
shuffle=True, random_state=1)
print(f"movie/item training data shape: {item_train.shape}")
print(f"movie/item test data shape: {item_test.shape}")

movie/item training data shape: (40707, 17)
movie/item test data shape: (10177, 17)
```

The scaled, shuffled data now has a mean of zero.

```
pprint_train(user_train, user_features, uvs, u_s, maxcount=5)

'<table>\n<thead>\n<tr><th style="text-align: center;"> [user id]
</th><th style="text-align: center;"> [rating count] </th><th
style="text-align: center;"> [rating ave] </th><th style="text-align:
center;"> Act ion </th><th style="text-align: center;"> Adve nture
</th><th style="text-align: center;"> Anim ation </th><th style="text-
align: center;"> Chil dren </th><th style="text-align: center;"> Com
edy </th><th style="text-align: center;"> Crime </th><th style="text-
align: center;"> Docum entary </th><th style="text-align: center;">
Drama </th><th style="text-align: center;"> Fan tasy </th><th
style="text-align: center;"> Hor ror </th><th style="text-align:
center;"> Mys tery </th><th style="text-align: center;"> Rom ance
</th><th style="text-align: center;"> Sci -Fi </th><th style="text-
align: center;"> Thri ller </th></tr>\n</thead>\n<tbody>\n<tr><td
style="text-align: center;">     1     </td><td style="text-align:
center;">       0        </td><td style="text-align: center;">     -
1.0     </td><td style="text-align: center;">  -0.8   </td><td
style="text-align: center;">    -0.7    </td><td style="text-align:
center;">    0.1     </td><td style="text-align: center;">   -0.0
</td><td style="text-align: center;">  -1.2   </td><td style="text-
align: center;"> -0.4  </td><td style="text-align: center;">     0.6
</td><td style="text-align: center;"> -0.5  </td><td style="text-
align: center;">   -0.5   </td><td style="text-align: center;">  -0.1
```

```
</td><td style="text-align: center;">   -0.6   </td><td style="text-
align: center;">   -0.6   </td><td style="text-align: center;">  -0.7
</td><td style="text-align: center;">   -0.7    </td></tr>\n<tr><td
style="text-align: center;">     0     </td><td style="text-align:
center;">       1        </td><td style="text-align: center;">     -
0.7     </td><td style="text-align: center;">  -0.5   </td><td
style="text-align: center;">    -0.7    </td><td style="text-align:
center;">    -0.1    </td><td style="text-align: center;">   -0.2
</td><td style="text-align: center;">  -0.6   </td><td style="text-
align: center;"> -0.2  </td><td style="text-align: center;">     0.7
</td><td style="text-align: center;"> -0.5  </td><td style="text-
align: center;">   -0.8   </td><td style="text-align: center;">   0.1
</td><td style="text-align: center;">   -0.0   </td><td style="text-
align: center;">   -0.6   </td><td style="text-align: center;">  -0.5
</td><td style="text-align: center;">   -0.4    </td></tr>\n<tr><td
style="text-align: center;">    -1     </td><td style="text-align:
center;">       -1       </td><td style="text-align: center;">     -
0.2     </td><td style="text-align: center;">   0.3   </td><td
style="text-align: center;">    -0.4    </td><td style="text-align:
center;">    0.4     </td><td style="text-align: center;">    0.5
</td><td style="text-align: center;">   1.0   </td><td style="text-
align: center;">  0.6  </td><td style="text-align: center;">     -1.2
</td><td style="text-align: center;"> -0.3  </td><td style="text-
align: center;">   -0.6   </td><td style="text-align: center;">  -2.3
</td><td style="text-align: center;">   -0.1   </td><td style="text-
align: center;">   0.0    </td><td style="text-align: center;">   0.4
</td><td style="text-align: center;">   -0.0    </td></tr>\n<tr><td
style="text-align: center;">     0     </td><td style="text-align:
center;">       -1       </td><td style="text-align: center;">     0.6
</td><td style="text-align: center;">   0.5   </td><td style="text-
align: center;">    0.5     </td><td style="text-align: center;">
0.2     </td><td style="text-align: center;">    0.6    </td><td
style="text-align: center;">  -0.1   </td><td style="text-align:
center;">  0.5  </td><td style="text-align: center;">     -1.2
</td><td style="text-align: center;">  0.9  </td><td style="text-
align: center;">   1.2    </td><td style="text-align: center;">  -2.3
</td><td style="text-align: center;">   -0.1   </td><td style="text-
align: center;">   0.0    </td><td style="text-align: center;">   0.2
</td><td style="text-align: center;">    0.3    </td></tr>\n<tr><td
style="text-align: center;">    -1     </td><td style="text-align:
center;">       0        </td><td style="text-align: center;">     0.7
</td><td style="text-align: center;">   0.6   </td><td style="text-
align: center;">    0.5     </td><td style="text-align: center;">
0.3     </td><td style="text-align: center;">    0.5    </td><td
style="text-align: center;">   0.4   </td><td style="text-align:
center;">  0.6  </td><td style="text-align: center;">     1.0
</td><td style="text-align: center;">  0.6  </td><td style="text-
align: center;">   0.3    </td><td style="text-align: center;">   0.8
</td><td style="text-align: center;">   0.8    </td><td style="text-
```

```
align: center;">   0.4    </td><td style="text-align: center;">   0.7    </td><td style="text-align: center;">    0.7    </td></tr>\n</tbody>\n</table>'
```

# 4 - Neural Network for content-based filtering

Now, let's construct a neural network as described in the figure above. It will have two networks that are combined by a dot product. You will construct the two networks. In this example, they will be identical. Note that these networks do not need to be the same. If the user content was substantially larger than the movie content, you might elect to increase the complexity of the user network relative to the movie network. In this case, the content is similar, so the networks are the same.

## Exercise 1

- Use a Keras sequential model
  - The first layer is a dense layer with 256 units and a relu activation.
  - The second layer is a dense layer with 128 units and a relu activation.
  - The third layer is a dense layer with `num_outputs` units and a linear or no activation.

The remainder of the network will be provided. The provided code does not use the Keras sequential model but instead uses the Keras functional api. This format allows for more flexibility in how components are interconnected.

```
# GRADED_CELL
# UNQ_C1

num_outputs = 32
tf.random.set_seed(1)
user_NN = tf.keras.models.Sequential([
    ### START CODE HERE ###
    tf.keras.layers.Dense(256,activation='relu'),
    tf.keras.layers.Dense(128,activation='relu'),
    tf.keras.layers.Dense(num_outputs)
    ### END CODE HERE ###
])

item_NN = tf.keras.models.Sequential([
    ### START CODE HERE ###
    tf.keras.layers.Dense(256,activation='relu'),
    tf.keras.layers.Dense(128,activation='relu'),
    tf.keras.layers.Dense(num_outputs)
    ### END CODE HERE ###
])
```

```python
# create the user input and point to the base network
input_user = tf.keras.layers.Input(shape=(num_user_features))
vu = user_NN(input_user)
vu = tf.linalg.l2_normalize(vu, axis=1)

# create the item input and point to the base network
input_item = tf.keras.layers.Input(shape=(num_item_features))
vm = item_NN(input_item)
vm = tf.linalg.l2_normalize(vm, axis=1)

# compute the dot product of the two vectors vu and vm
output = tf.keras.layers.Dot(axes=1)([vu, vm])

# specify the inputs and output of the model
model = tf.keras.Model([input_user, input_item], output)

model.summary()
```

```
Model: "model"
______________________________________________________________________
____________________________
Layer (type)                    Output Shape         Param #
Connected to
======================================================================
============================
input_3 (InputLayer)            [(None, 14)]         0

______________________________________________________________________
____________________________
input_4 (InputLayer)            [(None, 16)]         0

______________________________________________________________________
____________________________
sequential_2 (Sequential)       (None, 32)           40864
input_3[0][0]
______________________________________________________________________
____________________________
sequential_3 (Sequential)       (None, 32)           41376
input_4[0][0]
______________________________________________________________________
____________________________
tf_op_layer_l2_normalize_2/Squa [(None, 32)]         0
sequential_2[0][0]
______________________________________________________________________
____________________________
tf_op_layer_l2_normalize_3/Squa [(None, 32)]         0
sequential_3[0][0]
______________________________________________________________________
____________________________
```

```
tf_op_layer_l2_normalize_2/Sum  [(None, 1)]         0
tf_op_layer_l2_normalize_2/Square
______________________________________________________________________
____________________________
tf_op_layer_l2_normalize_3/Sum  [(None, 1)]         0
tf_op_layer_l2_normalize_3/Square
______________________________________________________________________
____________________________
tf_op_layer_l2_normalize_2/Maxi [(None, 1)]         0
tf_op_layer_l2_normalize_2/Sum[0]
______________________________________________________________________
____________________________
tf_op_layer_l2_normalize_3/Maxi [(None, 1)]         0
tf_op_layer_l2_normalize_3/Sum[0]
______________________________________________________________________
____________________________
tf_op_layer_l2_normalize_2/Rsqr [(None, 1)]         0
tf_op_layer_l2_normalize_2/Maximu
______________________________________________________________________
____________________________
tf_op_layer_l2_normalize_3/Rsqr [(None, 1)]         0
tf_op_layer_l2_normalize_3/Maximu
______________________________________________________________________
____________________________
tf_op_layer_l2_normalize_2 (Ten [(None, 32)]        0
sequential_2[0][0]

tf_op_layer_l2_normalize_2/Rsqrt[
______________________________________________________________________
____________________________
tf_op_layer_l2_normalize_3 (Ten [(None, 32)]        0
sequential_3[0][0]

tf_op_layer_l2_normalize_3/Rsqrt[
______________________________________________________________________
____________________________
dot_1 (Dot)                     (None, 1)           0
tf_op_layer_l2_normalize_2[0][0]

tf_op_layer_l2_normalize_3[0][0]
======================================================================
============================
Total params: 82,240
Trainable params: 82,240
Non-trainable params: 0
______________________________________________________________________
____________________________

# Public tests
from public_tests import *
```

```
test_tower(user_NN)
test_tower(item_NN)

All tests passed!
All tests passed!
```

We will use a mean squared error loss and an Adam optimizer.

```
tf.random.set_seed(1)
cost_fn = tf.keras.losses.MeanSquaredError()
opt = keras.optimizers.Adam(learning_rate=0.01)
model.compile(optimizer=opt,
              loss=cost_fn)

tf.random.set_seed(1)
model.fit([user_train[:, u_s:], item_train[:, i_s:]], y_train,
epochs=30)

Train on 40707 samples
Epoch 1/30
40707/40707 [==============================] - 5s 128us/sample - loss:
0.1232
Epoch 2/30
40707/40707 [==============================] - 5s 116us/sample - loss:
0.1146
Epoch 3/30
40707/40707 [==============================] - 5s 117us/sample - loss:
0.1089
Epoch 4/30
40707/40707 [==============================] - 5s 114us/sample - loss:
0.1039
Epoch 5/30
40707/40707 [==============================] - 5s 117us/sample - loss:
0.1001
Epoch 6/30
40707/40707 [==============================] - 5s 116us/sample - loss:
0.0973
Epoch 7/30
40707/40707 [==============================] - 5s 116us/sample - loss:
0.0956
Epoch 8/30
40707/40707 [==============================] - 5s 115us/sample - loss:
0.0935
Epoch 9/30
40707/40707 [==============================] - 5s 116us/sample - loss:
0.0916
Epoch 10/30
40707/40707 [==============================] - 5s 117us/sample - loss:
0.0897
Epoch 11/30
```

```
40707/40707 [==============================] - 5s 116us/sample - loss:
0.0880
Epoch 12/30
40707/40707 [==============================] - 5s 116us/sample - loss:
0.0865
Epoch 13/30
40707/40707 [==============================] - 5s 115us/sample - loss:
0.0852
Epoch 14/30
40707/40707 [==============================] - 5s 116us/sample - loss:
0.0839
Epoch 15/30
40707/40707 [==============================] - 5s 116us/sample - loss:
0.0830
Epoch 16/30
40707/40707 [==============================] - 5s 118us/sample - loss:
0.0815
Epoch 17/30
40707/40707 [==============================] - 5s 116us/sample - loss:
0.0807
Epoch 18/30
40707/40707 [==============================] - 5s 116us/sample - loss:
0.0796
Epoch 19/30
40707/40707 [==============================] - 5s 116us/sample - loss:
0.0786
Epoch 20/30
40707/40707 [==============================] - 5s 116us/sample - loss:
0.0776
Epoch 21/30
40707/40707 [==============================] - 5s 116us/sample - loss:
0.0769
Epoch 22/30
40707/40707 [==============================] - 5s 117us/sample - loss:
0.0761
Epoch 23/30
40707/40707 [==============================] - 5s 116us/sample - loss:
0.0755
Epoch 24/30
40707/40707 [==============================] - 5s 116us/sample - loss:
0.0746
Epoch 25/30
40707/40707 [==============================] - 5s 117us/sample - loss:
0.0741
Epoch 26/30
40707/40707 [==============================] - 5s 116us/sample - loss:
0.0733
Epoch 27/30
40707/40707 [==============================] - 5s 118us/sample - loss:
```

```
0.0728
Epoch 28/30
40707/40707 [==============================] - 5s 120us/sample - loss:
0.0723
Epoch 29/30
40707/40707 [==============================] - 5s 118us/sample - loss:
0.0717s - loss: 0.071 - ETA: 0s - loss: 0.0
Epoch 30/30
40707/40707 [==============================] - 5s 120us/sample - loss:
0.0713

<tensorflow.python.keras.callbacks.History at 0x7f2ba76b6110>
```

Evaluate the model to determine loss on the test data.

```
model.evaluate([user_test[:, u_s:], item_test[:, i_s:]], y_test)

10177/10177 [==============================] - 0s 36us/sample - loss:
0.0815

0.08146006993124337
```

It is comparable to the training loss indicating the model has not substantially overfit the training data.

# 5 - Predictions

Below, you'll use your model to make predictions in a number of circumstances.

## 5.1 - Predictions for a new user

First, we'll create a new user and have the model suggest movies for that user. After you have tried this on the example user content, feel free to change the user content to match your own preferences and see what the model suggests. Note that ratings are between 0.5 and 5.0, inclusive, in half-step increments.

```
new_user_id = 5000
new_rating_ave = 0.0
new_action = 0.0
new_adventure = 5.0
new_animation = 0.0
new_childrens = 0.0
new_comedy = 0.0
new_crime = 0.0
new_documentary = 0.0
new_drama = 0.0
new_fantasy = 5.0
```

```
new_horror = 0.0
new_mystery = 0.0
new_romance = 0.0
new_scifi = 0.0
new_thriller = 0.0
new_rating_count = 3

user_vec = np.array([[new_user_id, new_rating_count, new_rating_ave,
                      new_action, new_adventure, new_animation,
new_childrens,
                      new_comedy, new_crime, new_documentary,
                      new_drama, new_fantasy, new_horror, new_mystery,
                      new_romance, new_scifi, new_thriller]])
```

The new user enjoys movies from the adventure, fantasy genres. Let's find the top-rated movies for the new user.
Below, we'll use a set of movie/item vectors, `item_vecs` that have a vector for each movie in the training/test set. This is matched with the new user vector above and the scaled vectors are used to predict ratings for all the movies.

```
# generate and replicate the user vector to match the number movies in
the data set.
user_vecs = gen_user_vecs(user_vec,len(item_vecs))

# scale our user and item vectors
suser_vecs = scalerUser.transform(user_vecs)
sitem_vecs = scalerItem.transform(item_vecs)

# make a prediction
y_p = model.predict([suser_vecs[:, u_s:], sitem_vecs[:, i_s:]])

# unscale y prediction
y_pu = scalerTarget.inverse_transform(y_p)

# sort the results, highest prediction first
sorted_index = np.argsort(-y_pu,axis=0).reshape(-1).tolist()  #negate
to get largest rating first
sorted_ypu   = y_pu[sorted_index]
sorted_items = item_vecs[sorted_index]  #using unscaled vectors for
display

print_pred_movies(sorted_ypu, sorted_items, movie_dict, maxcount = 10)
```

```
'<table>\n<thead>\n<tr><th style="text-align: right;">  y_p</th><th
style="text-align: right;">  movie id</th><th style="text-align:
right;">  rating ave</th><th>title
</th><th>genres                         </th></tr>\n</thead>\
n<tbody>\n<tr><td style="text-align: right;">  4.5</td><td
style="text-align: right;">     98809</td><td style="text-align:
right;">        3.8</td><td>Hobbit: An Unexpected Journey, The (2012)
```

```
</td><td>Adventure|Fantasy               </td></tr>\n<tr><td
style="text-align: right;">  4.4</td><td style="text-align: right;">
8368</td><td style="text-align: right;">         3.9</td><td>Harry
Potter and the Prisoner of Azkaban (2004)    </td><td>Adventure|
Fantasy               </td></tr>\n<tr><td style="text-align: right;">
4.4</td><td style="text-align: right;">     54001</td><td style="text-
align: right;">         3.9</td><td>Harry Potter and the Order of the
Phoenix (2007)   </td><td>Adventure|Drama|Fantasy         </td></tr>\
n<tr><td style="text-align: right;">  4.3</td><td style="text-align:
right;">     40815</td><td style="text-align: right;">
3.8</td><td>Harry Potter and the Goblet of Fire (2005)
</td><td>Adventure|Fantasy|Thriller      </td></tr>\n<tr><td
style="text-align: right;">  4.3</td><td style="text-align: right;">
106489</td><td style="text-align: right;">         3.6</td><td>Hobbit:
The Desolation of Smaug, The (2013)        </td><td>Adventure|Fantasy
</td></tr>\n<tr><td style="text-align: right;">  4.3</td><td
style="text-align: right;">     81834</td><td style="text-align:
right;">         4  </td><td>Harry Potter and the Deathly Hallows:
Part 1 (2010)</td><td>Action|Adventure|Fantasy        </td></tr>\
n<tr><td style="text-align: right;">  4.3</td><td style="text-align:
right;">     59387</td><td style="text-align: right;">         4
</td><td>Fall, The (2006)
</td><td>Adventure|Drama|Fantasy         </td></tr>\n<tr><td
style="text-align: right;">  4.3</td><td style="text-align: right;">
5952</td><td style="text-align: right;">         4  </td><td>Lord of
the Rings: The Two Towers, The (2002)      </td><td>Adventure|Fantasy
</td></tr>\n<tr><td style="text-align: right;">  4.3</td><td
style="text-align: right;">      5816</td><td style="text-align:
right;">         3.6</td><td>Harry Potter and the Chamber of Secrets
(2002)     </td><td>Adventure|Fantasy               </td></tr>\
n<tr><td style="text-align: right;">  4.3</td><td style="text-align:
right;">     54259</td><td style="text-align: right;">
3.6</td><td>Stardust (2007)
</td><td>Adventure|Comedy|Fantasy|Romance</td></tr>\n</tbody>\n</table
>'
```

## 5.2 - Predictions for an existing user.

Let's look at the predictions for "user 2", one of the users in the data set. We can compare the predicted ratings with the model's ratings.

```
uid = 2
# form a set of user vectors. This is the same vector, transformed and
repeated.
user_vecs, y_vecs = get_user_vecs(uid, user_train_unscaled, item_vecs,
user_to_genre)
```

```python
# scale our user and item vectors
suser_vecs = scalerUser.transform(user_vecs)
sitem_vecs = scalerItem.transform(item_vecs)

# make a prediction
y_p = model.predict([suser_vecs[:, u_s:], sitem_vecs[:, i_s:]])

# unscale y prediction
y_pu = scalerTarget.inverse_transform(y_p)

# sort the results, highest prediction first
sorted_index = np.argsort(-y_pu,axis=0).reshape(-1).tolist()  #negate
to get largest rating first
sorted_ypu   = y_pu[sorted_index]
sorted_items = item_vecs[sorted_index]  #using unscaled vectors for
display
sorted_user  = user_vecs[sorted_index]
sorted_y     = y_vecs[sorted_index]

#print sorted predictions for movies rated by the user
print_existing_user(sorted_ypu, sorted_y.reshape(-1,1), sorted_user,
sorted_items, ivs, uvs, movie_dict, maxcount = 50)
```

```
'<table>\n<thead>\n<tr><th style="text-align: right;">  y_p</th><th
style="text-align: right;">  y</th><th style="text-align: right;">
user</th><th>user genre ave           </th><th style="text-align:
right;">  movie rating ave</th><th style="text-align: right;">  movie
id</th><th>title
</th><th>genres
</th></tr>\n</thead>\n<tbody>\n<tr><td style="text-align: right;">
4.5</td><td style="text-align: right;">5.0</td><td style="text-align:
right;">     2</td><td>[4.0]                    </td><td style="text-
align: right;">               4.3</td><td style="text-align: right;">
80906</td><td>Inside Job (2010)
</td><td>Documentary                              </td></tr>\n<tr><td
style="text-align: right;">  4.2</td><td style="text-align:
right;">3.5</td><td style="text-align: right;">
2</td><td>[4.0,4.0]                </td><td style="text-align:
right;">               3.9</td><td style="text-align: right;">
99114</td><td>Django Unchained (2012)
</td><td>Action|Drama                             </td></tr>\n<tr><td
style="text-align: right;">  4.1</td><td style="text-align:
right;">4.5</td><td style="text-align: right;">
2</td><td>[4.0,4.0]                </td><td style="text-align:
right;">               4.1</td><td style="text-align: right;">
68157</td><td>Inglourious Basterds (2009)
</td><td>Action|Drama                             </td></tr>\n<tr><td
style="text-align: right;">  4.1</td><td style="text-align:
right;">3.5</td><td style="text-align: right;">
2</td><td>[4.0,3.9,3.9]            </td><td style="text-align:
```

```
right;">               3.9</td><td style="text-align: right;">
115713</td><td>Ex Machina (2015)
</td><td>Drama|Sci-Fi|Thriller                     </td></tr>\n<tr><td
style="text-align: right;">  4.0</td><td style="text-align:
right;">4.0</td><td style="text-align: right;">
2</td><td>[4.0,4.1,4.0,4.0,3.9,3.9]</td><td style="text-align:
right;">               4.1</td><td style="text-align: right;">
79132</td><td>Inception (2010)
</td><td>Action|Crime|Drama|Mystery|Sci-Fi|Thriller</td></tr>\n<tr><td
style="text-align: right;">  4.0</td><td style="text-align:
right;">4.0</td><td style="text-align: right;">
2</td><td>[4.1,4.0,3.9]            </td><td style="text-align:
right;">               4.3</td><td style="text-align: right;">
48516</td><td>Departed, The (2006)
</td><td>Crime|Drama|Thriller                      </td></tr>\n<tr><td
style="text-align: right;">  4.0</td><td style="text-align:
right;">4.5</td><td style="text-align: right;">
2</td><td>[4.0,4.1,4.0]            </td><td style="text-align:
right;">               4.2</td><td style="text-align: right;">
58559</td><td>Dark Knight, The (2008)
</td><td>Action|Crime|Drama                        </td></tr>\n<tr><td
style="text-align: right;">  4.0</td><td style="text-align:
right;">4.0</td><td style="text-align: right;">
2</td><td>[4.0,4.1,3.9]            </td><td style="text-align:
right;">               4.0</td><td style="text-align: right;">
6874</td><td>Kill Bill: Vol. 1 (2003)
</td><td>Action|Crime|Thriller                     </td></tr>\n<tr><td
style="text-align: right;">  4.0</td><td style="text-align:
right;">3.5</td><td style="text-align: right;">
2</td><td>[4.0,4.1,4.0,3.9]        </td><td style="text-align:
right;">               3.8</td><td style="text-align: right;">
8798</td><td>Collateral (2004)
</td><td>Action|Crime|Drama|Thriller               </td></tr>\n<tr><td
style="text-align: right;">  3.9</td><td style="text-align:
right;">5.0</td><td style="text-align: right;">
2</td><td>[4.0,4.1,4.0]            </td><td style="text-align:
right;">               3.9</td><td style="text-align: right;">
106782</td><td>Wolf of Wall Street, The (2013)
</td><td>Comedy|Crime|Drama                        </td></tr>\n<tr><td
style="text-align: right;">  3.9</td><td style="text-align:
right;">3.5</td><td style="text-align: right;">
2</td><td>[4.0,4.2,4.1]            </td><td style="text-align:
right;">               4.0</td><td style="text-align: right;">
91529</td><td>Dark Knight Rises, The (2012)
</td><td>Action|Adventure|Crime                    </td></tr>\n<tr><td
style="text-align: right;">  3.9</td><td style="text-align:
right;">4.0</td><td style="text-align: right;">
2</td><td>[4.0,4.0,3.9]            </td><td style="text-align:
right;">               4.0</td><td style="text-align: right;">
```

```
74458</td><td>Shutter Island (2010)
</td><td>Drama|Mystery|Thriller                    </td></tr>\n<tr><td
style="text-align: right;">  3.9</td><td style="text-align:
right;">4.5</td><td style="text-align: right;">
2</td><td>[4.1,4.0,3.9]            </td><td style="text-align:
right;">              4.0</td><td style="text-align: right;">
80489</td><td>Town, The (2010)
</td><td>Crime|Drama|Thriller                      </td></tr>\n<tr><td
style="text-align: right;">  3.8</td><td style="text-align:
right;">4.0</td><td style="text-align: right;">     2</td><td>[4.0]
</td><td style="text-align: right;">               4.0</td><td
style="text-align: right;">    112552</td><td>Whiplash (2014)
</td><td>Drama                                     </td></tr>\n<tr><td
style="text-align: right;">  3.8</td><td style="text-align:
right;">3.0</td><td style="text-align: right;">     2</td><td>[3.9]
</td><td style="text-align: right;">               4.0</td><td
style="text-align: right;">    109487</td><td>Interstellar (2014)
</td><td>Sci-Fi                                    </td></tr>\n<tr><td
style="text-align: right;">  3.8</td><td style="text-align:
right;">5.0</td><td style="text-align: right;">     2</td><td>[4.0]
</td><td style="text-align: right;">               3.7</td><td
style="text-align: right;">     89774</td><td>Warrior (2011)
</td><td>Drama                                     </td></tr>\n<tr><td
style="text-align: right;">  3.7</td><td style="text-align:
right;">3.0</td><td style="text-align: right;">
2</td><td>[4.0,4.0,3.0]            </td><td style="text-align:
right;">              3.9</td><td style="text-align: right;">
71535</td><td>Zombieland (2009)
</td><td>Action|Comedy|Horror                      </td></tr>\n<tr><td
style="text-align: right;">  3.7</td><td style="text-align:
right;">5.0</td><td style="text-align: right;">
2</td><td>[4.0,4.2,3.9,3.9]        </td><td style="text-align:
right;">              3.8</td><td style="text-align: right;">
122882</td><td>Mad Max: Fury Road (2015)
</td><td>Action|Adventure|Sci-Fi|Thriller          </td></tr>\n<tr><td
style="text-align: right;">  3.5</td><td style="text-align:
right;">5.0</td><td style="text-align: right;">     2</td><td>[4.0]
</td><td style="text-align: right;">               3.6</td><td
style="text-align: right;">     60756</td><td>Step Brothers (2008)
</td><td>Comedy                                    </td></tr>\n<tr><td
style="text-align: right;">  3.5</td><td style="text-align:
right;">2.5</td><td style="text-align: right;">
2</td><td>[4.0,3.9]                </td><td style="text-align:
right;">              3.5</td><td style="text-align: right;">
91658</td><td>Girl with the Dragon Tattoo, The (2011)
</td><td>Drama|Thriller                            </td></tr>\n<tr><td
style="text-align: right;">  3.1</td><td style="text-align:
right;">3.0</td><td style="text-align: right;">
2</td><td>[4.0,4.0]                </td><td style="text-align:
```

```
right;">                4.0</td><td style="text-align: right;">
77455</td><td>Exit Through the Gift Shop (2010)
</td><td>Comedy|Documentary                        </td></tr>\n<tr><td
style="text-align: right;">  3.1</td><td style="text-align:
right;">4.0</td><td style="text-align: right;">
2</td><td>[4.0,4.0]                </td><td style="text-align:
right;">                3.2</td><td style="text-align: right;">
46970</td><td>Talladega Nights: The Ballad of Ricky Bobby
(2006)</td><td>Action|Comedy                              </td></tr>\
n</tbody>\n</table>'
```

The model prediction is generally within 1 of the actual rating though it is not a very accurate predictor of how a user rates specific movies. This is especially true if the user rating is significantly different than the user's genre average. You can vary the user id above to try different users. Not all user id's were used in the training set.

## 5.3 - Finding Similar Items

The neural network above produces two feature vectors, a user feature vector $v_u$, and a movie feature vector, $v_m$. These are 32 entry vectors whose values are difficult to interpret. However, similar items will have similar vectors. This information can be used to make recommendations. For example, if a user has rated "Toy Story 3" highly, one could recommend similar movies by selecting movies with similar movie feature vectors.

A similarity measure is the squared distance between the two vectors $ \mathbf{v_m^{(k)}}$ and $v_m^{(i)}$ :

$$\left\| v_m^{(k)} - v_m^{(i)} \right\|^2 = \sum_{l=1}^{n} \left( v_{m_l}^{(k)} - v_{m_l}^{(i)} \right)^2$$

## Exercise 2

Write a function to compute the square distance.

```
# GRADED_FUNCTION: sq_dist
# UNQ_C2
def sq_dist(a,b):
    """
    Returns the squared distance between two vectors
    Args:
      a (ndarray (n,)): vector with n features
      b (ndarray (n,)): vector with n features
    Returns:
      d (float) : distance
    """
```

```
    ### START CODE HERE ###
    d = np.sum((a-b)**2)
    ### END CODE HERE ###
    return d

a1 = np.array([1.0, 2.0, 3.0]); b1 = np.array([1.0, 2.0, 3.0])
a2 = np.array([1.1, 2.1, 3.1]); b2 = np.array([1.0, 2.0, 3.0])
a3 = np.array([0, 1, 0]);       b3 = np.array([1, 0, 0])
print(f"squared distance between a1 and b1: {sq_dist(a1, b1):0.3f}")
print(f"squared distance between a2 and b2: {sq_dist(a2, b2):0.3f}")
print(f"squared distance between a3 and b3: {sq_dist(a3, b3):0.3f}")

squared distance between a1 and b1: 0.000
squared distance between a2 and b2: 0.030
squared distance between a3 and b3: 2.000
```

**Expected Output**:

squared distance between a1 and b1: 0.000
squared distance between a2 and b2: 0.030
squared distance between a3 and b3: 2.000

```
# Public tests
test_sq_dist(sq_dist)

All tests passed!
```

A matrix of distances between movies can be computed once when the model is trained and then reused for new recommendations without retraining. The first step, once a model is trained, is to obtain the movie feature vector, $v_m$, for each of the movies. To do this, we will use the trained `item_NN` and build a small model to allow us to run the movie vectors through it to generate $v_m$.

```
input_item_m = tf.keras.layers.Input(shape=(num_item_features))    # 
input layer
vm_m = item_NN(input_item_m)                                       # 
use the trained item_NN
vm_m = tf.linalg.l2_normalize(vm_m, axis=1)                        # 
incorporate normalization as was done in the original model
model_m = tf.keras.Model(input_item_m, vm_m)

model_m.summary()

Model: "model_1"
________________________________________________________________________
________________________________
Layer (type)                    Output Shape         Param #
Connected to
========================================================================
```

```
============================
input_5 (InputLayer)            [(None, 16)]         0

______________________________________________________________________
____________________________
sequential_3 (Sequential)       (None, 32)           41376
input_5[0][0]
______________________________________________________________________
____________________________
tf_op_layer_l2_normalize_4/Squa [(None, 32)]         0
sequential_3[1][0]
______________________________________________________________________
____________________________
tf_op_layer_l2_normalize_4/Sum  [(None, 1)]          0
tf_op_layer_l2_normalize_4/Square
______________________________________________________________________
____________________________
tf_op_layer_l2_normalize_4/Maxi [(None, 1)]          0
tf_op_layer_l2_normalize_4/Sum[0]
______________________________________________________________________
____________________________
tf_op_layer_l2_normalize_4/Rsqr [(None, 1)]          0
tf_op_layer_l2_normalize_4/Maximu
______________________________________________________________________
____________________________
tf_op_layer_l2_normalize_4 (Ten [(None, 32)]         0
sequential_3[1][0]

tf_op_layer_l2_normalize_4/Rsqrt[
======================================================================
============================
Total params: 41,376
Trainable params: 41,376
Non-trainable params: 0
______________________________________________________________________
____________________________
```

Once you have a movie model, you can create a set of movie feature vectors by using the model to predict using a set of item/movie vectors as input. `item_vecs` is a set of all of the movie vectors. It must be scaled to use with the trained model. The result of the prediction is a 32 entry feature vector for each movie.

```
scaled_item_vecs = scalerItem.transform(item_vecs)
vms = model_m.predict(scaled_item_vecs[:,i_s:])
print(f"size of all predicted movie feature vectors: {vms.shape}")
```

```
size of all predicted movie feature vectors: (847, 32)
```

Let's now compute a matrix of the squared distance between each movie feature vector and all other movie feature vectors:

We can then find the closest movie by finding the minimum along each row. We will make use of numpy masked arrays to avoid selecting the same movie. The masked values along the diagonal won't be included in the computation.

```
count = 50  # number of movies to display
dim = len(vms)
dist = np.zeros((dim,dim))

for i in range(dim):
    for j in range(dim):
        dist[i,j] = sq_dist(vms[i, :], vms[j, :])

m_dist = ma.masked_array(dist, mask=np.identity(dist.shape[0]))  #
mask the diagonal

disp = [["movie1", "genres", "movie2", "genres"]]
for i in range(count):
    min_idx = np.argmin(m_dist[i])
    movie1_id = int(item_vecs[i,0])
    movie2_id = int(item_vecs[min_idx,0])
    disp.append( [movie_dict[movie1_id]['title'],
movie_dict[movie1_id]['genres'],
                  movie_dict[movie2_id]['title'],
movie_dict[movie1_id]['genres']]
               )
table = tabulate.tabulate(disp, tablefmt='html', headers="firstrow")
table
```

```
'<table>\n<thead>\n<tr><th>movie1
</th><th>genres
</th><th>movie2
</th><th>genres
</th></tr>\n</thead>\n<tbody>\n<tr><td>Save the Last Dance (2001)
</td><td>Drama|Romance
</td><td>Mona Lisa Smile (2003)
</td><td>Drama|Romance
</td></tr>\n<tr><td>Wedding Planner, The (2001)
</td><td>Comedy|Romance
</td><td>Mr. Deeds (2002)
</td><td>Comedy|Romance
</td></tr>\n<tr><td>Hannibal (2001)
</td><td>Horror|Thriller
</td><td>Final Destination 2 (2003)
</td><td>Horror|Thriller
</td></tr>\n<tr><td>Saving Silverman (Evil Woman) (2001)
</td><td>Comedy|Romance
</td><td>Down with Love (2003)
```

```
</td><td>Comedy|Romance
</td></tr>\n<tr><td>Down to Earth (2001)
</td><td>Comedy|Fantasy|Romance
</td><td>Bewitched (2005)
</td><td>Comedy|Fantasy|Romance
</td></tr>\n<tr><td>Mexican, The (2001)
</td><td>Action|Comedy
</td><td>Rush Hour 2 (2001)
</td><td>Action|Comedy
</td></tr>\n<tr><td>15 Minutes (2001)
</td><td>Thriller
</td><td>Panic Room (2002)
</td><td>Thriller
</td></tr>\n<tr><td>Enemy at the Gates (2001)
</td><td>Drama
</td><td>Kung Fu Hustle (Gong fu) (2004)
</td><td>Drama
</td></tr>\n<tr><td>Heartbreakers (2001)
</td><td>Comedy|Crime|Romance
</td><td>Fun with Dick and Jane (2005)
</td><td>Comedy|Crime|Romance
</td></tr>\n<tr><td>Spy Kids (2001)
</td><td>Action|Adventure|Children|Comedy
</td><td>Tuxedo, The (2002)
</td><td>Action|Adventure|Children|Comedy
</td></tr>\n<tr><td>Along Came a Spider (2001)
</td><td>Action|Crime|Mystery|Thriller
</td><td>Insomnia (2002)
</td><td>Action|Crime|Mystery|Thriller
</td></tr>\n<tr><td>Blow (2001)
</td><td>Crime|Drama
</td><td>25th Hour (2002)
</td><td>Crime|Drama
</td></tr>\n<tr><td>Bridget Jones&#x27;s Diary (2001)
</td><td>Comedy|Drama|Romance
</td><td>Punch-Drunk Love (2002)
</td><td>Comedy|Drama|Romance
</td></tr>\n<tr><td>Joe Dirt (2001)
</td><td>Adventure|Comedy|Mystery|Romance
</td><td>Polar Express, The (2004)
</td><td>Adventure|Comedy|Mystery|Romance
</td></tr>\n<tr><td>Crocodile Dundee in Los Angeles (2001)
</td><td>Comedy|Drama
</td><td>Bewitched (2005)
</td><td>Comedy|Drama
</td></tr>\n<tr><td>Mummy Returns, The (2001)
</td><td>Action|Adventure|Comedy|Thriller
</td><td>Rundown, The (2003)
</td><td>Action|Adventure|Comedy|Thriller
```

```
</td></tr>\n<tr><td>Knight&#x27;s Tale, A (2001)
</td><td>Action|Comedy|Romance
</td><td>Legally Blonde (2001)
</td><td>Action|Comedy|Romance
</td></tr>\n<tr><td>Shrek (2001)
</td><td>Adventure|Animation|Children|Comedy|Fantasy|Romance</td><td>T
angled (2010)                                    </td><td>Adventure|
Animation|Children|Comedy|Fantasy|Romance</td></tr>\n<tr><td>Moulin
Rouge (2001)                      </td><td>Drama|Romance
</td><td>Notebook, The (2004)
</td><td>Drama|Romance
</td></tr>\n<tr><td>Pearl Harbor (2001)
</td><td>Action|Drama|Romance
</td><td>Bridget Jones: The Edge of Reason (2004)
</td><td>Action|Drama|Romance
</td></tr>\n<tr><td>Animal, The (2001)
</td><td>Comedy
</td><td>Dumb and Dumberer: When Harry Met Lloyd (2003)
</td><td>Comedy
</td></tr>\n<tr><td>Evolution (2001)
</td><td>Comedy|Sci-Fi
</td><td>Behind Enemy Lines (2001)
</td><td>Comedy|Sci-Fi
</td></tr>\n<tr><td>Swordfish (2001)
</td><td>Action|Crime|Drama
</td><td>We Were Soldiers (2002)
</td><td>Action|Crime|Drama
</td></tr>\n<tr><td>Atlantis: The Lost Empire (2001)
</td><td>Adventure|Animation|Children|Fantasy
</td><td>Cloudy with a Chance of Meatballs (2009)
</td><td>Adventure|Animation|Children|Fantasy
</td></tr>\n<tr><td>Lara Croft: Tomb Raider (2001)
</td><td>Action|Adventure
</td><td>National Treasure: Book of Secrets (2007)
</td><td>Action|Adventure
</td></tr>\n<tr><td>Dr. Dolittle 2 (2001)
</td><td>Comedy
</td><td>Legally Blonde 2: Red, White &amp; Blonde (2003)
</td><td>Comedy
</td></tr>\n<tr><td>Fast and the Furious, The (2001)
</td><td>Action|Crime|Thriller
</td><td>xXx (2002)
</td><td>Action|Crime|Thriller
</td></tr>\n<tr><td>A.I. Artificial Intelligence (2001)
</td><td>Adventure|Drama|Sci-Fi
</td><td>Bubba Ho-tep (2002)
</td><td>Adventure|Drama|Sci-Fi
</td></tr>\n<tr><td>Cats &amp; Dogs (2001)
</td><td>Children|Comedy
```

```
</td><td>Robots (2005)
</td><td>Children|Comedy
</td></tr>\n<tr><td>Scary Movie 2 (2001)
</td><td>Comedy
</td><td>Orange County (2002)
</td><td>Comedy
</td></tr>\n<tr><td>Final Fantasy: The Spirits Within
(2001)</td><td>Adventure|Animation|Fantasy|Sci-Fi
</td><td>Madagascar: Escape 2 Africa (2008)
</td><td>Adventure|Animation|Fantasy|Sci-Fi
</td></tr>\n<tr><td>Legally Blonde (2001)
</td><td>Comedy|Romance
</td><td>Serendipity (2001)
</td><td>Comedy|Romance
</td></tr>\n<tr><td>Score, The (2001)
</td><td>Action|Drama
</td><td>Punisher, The (2004)
</td><td>Action|Drama
</td></tr>\n<tr><td>Jurassic Park III (2001)
</td><td>Action|Adventure|Sci-Fi|Thriller
</td><td>Men in Black II (a.k.a. MIIB) (a.k.a. MIB 2)
(2002)</td><td>Action|Adventure|Sci-Fi|Thriller
</td></tr>\n<tr><td>America&#x27;s Sweethearts (2001)
</td><td>Comedy|Romance
</td><td>Maid in Manhattan (2002)
</td><td>Comedy|Romance
</td></tr>\n<tr><td>Ghost World (2001)
</td><td>Comedy|Drama
</td><td>Station Agent, The (2003)
</td><td>Comedy|Drama
</td></tr>\n<tr><td>Planet of the Apes (2001)
</td><td>Action|Adventure|Drama|Sci-Fi
</td><td>Day After Tomorrow, The (2004)
</td><td>Action|Adventure|Drama|Sci-Fi
</td></tr>\n<tr><td>Princess Diaries, The (2001)
</td><td>Children|Comedy|Romance
</td><td>Lake House, The (2006)
</td><td>Children|Comedy|Romance
</td></tr>\n<tr><td>Rush Hour 2 (2001)
</td><td>Action|Comedy
</td><td>Mexican, The (2001)
</td><td>Action|Comedy
</td></tr>\n<tr><td>American Pie 2 (2001)
</td><td>Comedy
</td><td>Rat Race (2001)
</td><td>Comedy
</td></tr>\n<tr><td>Others, The (2001)
</td><td>Drama|Horror|Mystery|Thriller
</td><td>The Machinist (2004)
```

```
</td><td>Drama|Horror|Mystery|Thriller
</td></tr>\n<tr><td>Rat Race (2001)
</td><td>Comedy
</td><td>American Pie 2 (2001)
</td><td>Comedy
</td></tr>\n<tr><td>Jay and Silent Bob Strike Back (2001)
</td><td>Adventure|Comedy
</td><td>Mexican, The (2001)
</td><td>Adventure|Comedy
</td></tr>\n<tr><td>Training Day (2001)
</td><td>Crime|Drama|Thriller
</td><td>Frailty (2001)
</td><td>Crime|Drama|Thriller
</td></tr>\n<tr><td>Zoolander (2001)
</td><td>Comedy
</td><td>Old School (2003)
</td><td>Comedy
</td></tr>\n<tr><td>Serendipity (2001)
</td><td>Comedy|Romance
</td><td>Legally Blonde (2001)
</td><td>Comedy|Romance
</td></tr>\n<tr><td>Mulholland Drive (2001)
</td><td>Crime|Drama|Mystery|Thriller
</td><td>Prisoners (2013)
</td><td>Crime|Drama|Mystery|Thriller
</td></tr>\n<tr><td>From Hell (2001)
</td><td>Crime|Horror|Mystery|Thriller
</td><td>Identity (2003)
</td><td>Crime|Horror|Mystery|Thriller
</td></tr>\n<tr><td>Waking Life (2001)
</td><td>Animation|Drama|Fantasy
</td><td>Warm Bodies (2013)
</td><td>Animation|Drama|Fantasy
</td></tr>\n<tr><td>K-PAX (2001)
</td><td>Drama|Fantasy|Mystery|Sci-Fi
</td><td>Gosford Park (2001)
</td><td>Drama|Fantasy|Mystery|Sci-Fi
</td></tr>\n</tbody>\n</table>'
```

The results show the model will generally suggest a movie with similar genre's.

# 6 - Congratulations!

You have completed a content-based recommender system.

This structure is the basis of many commercial recommender systems. The user content can be greatly expanded to incorporate more information about the user if it is available. Items are not limited to movies. This can be used to recommend any item, books, cars or items that are similar to an item in your 'shopping cart'.