

Day 2

Please run the following code to update the reflow package

In [122]:

```
!pip install recoflow --upgrade  
import recoflow
```

Requirement already up-to-date: reflow in /usr/local/lib/python3.6/dist-packages (0.0.7)
Requirement already satisfied, skipping upgrade: matplotlib<4.0,>=3.1 in /usr/local/lib/python3.6/dist-packages (from reflow) (3.1.1)
Requirement already satisfied, skipping upgrade: numpy<2.0,>=1.17 in /usr/local/lib/python3.6/dist-packages (from reflow) (1.17.2)
Requirement already satisfied, skipping upgrade: altair<4.0,>=3.2 in /usr/local/lib/python3.6/dist-packages (from reflow) (3.2.0)
Requirement already satisfied, skipping upgrade: scikit-learn<0.22.0,>=0.21.3 in /usr/local/lib/python3.6/dist-packages (from reflow) (0.21.3)
Requirement already satisfied, skipping upgrade: keras<3.0,>=2.3 in /usr/local/lib/python3.6/dist-packages (from reflow) (2.3.0)
Requirement already satisfied, skipping upgrade: requests<3.0,>=2.22 in /usr/local/lib/python3.6/dist-packages (from reflow) (2.22.0)
Requirement already satisfied, skipping upgrade: pandas<0.26.0,>=0.25.1 in /usr/local/lib/python3.6/dist-packages (from reflow) (0.25.1)
Requirement already satisfied, skipping upgrade: cycloper>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib<4.0,>=3.1->reflow) (0.10.0)
Requirement already satisfied, skipping upgrade: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib<4.0,>=3.1->reflow) (2.8.0)
Requirement already satisfied, skipping upgrade: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib<4.0,>=3.1->reflow) (1.1.0)
Requirement already satisfied, skipping upgrade: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib<4.0,>=3.1->reflow) (2.4.2)
Requirement already satisfied, skipping upgrade: six in /usr/lib/python3/dist-packages (from altair<4.0,>=3.2->reflow) (1.11.0)
Requirement already satisfied, skipping upgrade: jsonschema in /usr/local/lib/python3.6/dist-packages (from altair<4.0,>=3.2->reflow) (3.0.2)
Requirement already satisfied, skipping upgrade: toolz in /usr/local/lib/python3.6/dist-packages (from altair<4.0,>=3.2->reflow) (0.10.0)
Requirement already satisfied, skipping upgrade: jinja2 in /usr/local/lib/python3.6/dist-packages (from altair<4.0,>=3.2->reflow) (2.10.1)
Requirement already satisfied, skipping upgrade: entrypoints in /usr/local/lib/python3.6/dist-packages (from altair<4.0,>=3.2->reflow) (0.3)
Requirement already satisfied, skipping upgrade: scipy>=0.17.0 in /usr/local/lib/python3.6/dist-packages (from scikit-learn<0.22.0,>=0.21.3->reflow) (1.3.1)
Requirement already satisfied, skipping upgrade: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-learn<0.22.0,>=0.21.3->reflow) (0.13.2)
Requirement already satisfied, skipping upgrade: pyyaml in /usr/local/lib/python3.6/dist-packages (from keras<3.0,>=2.3->reflow) (5.1.2)
Requirement already satisfied, skipping upgrade: keras-applications>=1.0.6 in /usr/local/lib/python3.6/dist-packages (from keras<3.0,>=2.3->reflow) (1.0.8)
Requirement already satisfied, skipping upgrade: h5py in /usr/local/lib/python3.6/dist-packages (from keras<3.0,>=2.3->reflow) (2.9.0)
Requirement already satisfied, skipping upgrade: keras-preprocessing>=1.0.5 in /usr/local/lib/python3.6/dist-packages (from keras<3.0,>=2.3->reflow) (1.1.0)
Requirement already satisfied, skipping upgrade: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests<3.0,>=2.22->reflow) (2019.6.16)

Requirement already satisfied, skipping upgrade: idna<2.9,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests<3.0,>=2.22->recoflow) (2.8)
Requirement already satisfied, skipping upgrade: urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests<3.0,>=2.22->recoflow) (1.24.2)
Requirement already satisfied, skipping upgrade: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests<3.0,>=2.22->recoflow) (3.0.4)
Requirement already satisfied, skipping upgrade: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas<0.26.0,>=0.25.1->recoflow) (2019.2)
Requirement already satisfied, skipping upgrade: setuptools in /usr/local/lib/python3.6/dist-packages (from kiwisolver>=1.0.1->matplotlib<4.0,>=3.1->recoflow) (41.0.1)
Requirement already satisfied, skipping upgrade: pyparsing>=2.2.0 in /usr/local/lib/python3.6/dist-packages (from jsonschema->altair<4.0,>=3.2->recoflow) (2.4.6)
Requirement already satisfied, skipping upgrade: attrs>=17.4.0 in /usr/local/lib/python3.6/dist-packages (from jsonschema->altair<4.0,>=3.2->recoflow) (19.1.0)
Requirement already satisfied, skipping upgrade: MarkupSafe>=0.23 in /usr/local/lib/python3.6/dist-packages (from jinja2->altair<4.0,>=3.2->recoflow) (1.1.1)
WARNING: You are using pip version 19.1.1, however version 19.2.3 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.

In [128]:

```
# !zip -r stuffs.zip /data/posters
!tar -czf data.gz /tf/data/
```

tar: Removing leading `/' from member names

Data Preprocessing

In [129]:

```
import numpy as np
import pandas as pd
```

In [130]:

```
ratings = pd.read_csv("/tf/data/ratings.csv")
users = pd.read_csv("/tf/data/users.csv")
items = pd.read_csv("/tf/data/items.csv")
```

In [6]:

```
from recoflow.preprocessing import EncodeUserItem, StratifiedSplit
```

In [8]:

```
# Encoding the data
interaction, n_users, n_items, user_encoder, item_encoder = EncodeUserItem(ratings,
                                                                           "user_id",
                                                                           "movie_id",
                                                                           "rating",
                                                                           "unix_timestamp")
```

```
Number of users:  943
Number of items: 1682
```

In [18]:

```
train, test = StratifiedSplit(interaction, [0.8, 0.2])
train.shape, test.shape
```

Out[18]:

```
((80000, 7), (20000, 7))
```

In [46]:

```
min_rating = interaction.RATING.min()
max_rating = interaction.RATING.max()
min_rating, max_rating
```

Out[46]:

```
(1, 5)
```

Model - Deep Facorization

In [47]:

```
from keras.models import Model
from keras.layers import Embedding, Input, Dot, Add, Activation, Lambda, Concatenate, Dense, Flatten, Dropout
from keras.regularizers import l2
from keras.optimizers import Adam
```

In [48]:

```
Embedding?
```

Init signature:

```
Embedding(  
    input_dim,  
    output_dim,  
    embeddings_initializer='uniform',  
    embeddings_regularizer=None,  
    activity_regularizer=None,  
    embeddings_constraint=None,  
    mask_zero=False,  
    input_length=None,  
    **kwargs,  
)
```

Docstring:

Turns positive integers (indexes) into dense vectors of fixed size.
eg. [[4], [20]] -> [[0.25, 0.1], [0.6, -0.2]]

This layer can only be used as the first layer in a model.

Example

```
```python  
model = Sequential()
model.add(Embedding(1000, 64, input_length=10))
the model will take as input an integer matrix of size (batch, input_length).
the largest integer (i.e. word index) in the input should be
no larger than 999 (vocabulary size).
now model.output_shape == (None, 10, 64), where None is the batch dimension.

input_array = np.random.randint(1000, size=(32, 10))

model.compile('rmsprop', 'mse')
output_array = model.predict(input_array)
assert output_array.shape == (32, 10, 64)
```
```

Arguments

`input_dim`: int > 0. Size of the vocabulary,
i.e. maximum integer index + 1.
`output_dim`: int >= 0. Dimension of the dense embedding.
`embeddings_initializer`: Initializer for the `embeddings` matrix
(see [initializers](../initializers.md)).
`embeddings_regularizer`: Regularizer function applied to
the `embeddings` matrix
(see [regularizer](../regularizers.md)).
`activity_regularizer`: Regularizer function applied to
the output of the layer (its "activation").
(see [regularizer](../regularizers.md)).

```
embeddings_constraint: Constraint function applied to
    the `embeddings` matrix
    (see [constraints](../constraints.md)).
mask_zero: Whether or not the input value 0 is a special "padding"
    value that should be masked out.
    This is useful when using [recurrent layers](recurrent.md)
    which may take variable length input.
    If this is `True` then all subsequent layers
    in the model need to support masking or an exception will be raised.
    If mask_zero is set to True, as a consequence, index 0 cannot be
    used in the vocabulary (input_dim should equal size of
    vocabulary + 1).
input_length: Length of input sequences, when it is constant.
    This argument is required if you are going to connect
    `Flatten` then `Dense` layers upstream
    (without it, the shape of the dense outputs cannot be computed).

# Input shape
    2D tensor with shape: `(batch_size, sequence_length)`.

# Output shape
    3D tensor with shape: `(batch_size, sequence_length, output_dim)`.

# References
    - [A Theoretically Grounded Application of Dropout in
      Recurrent Neural Networks](http://arxiv.org/abs/1512.05287)
File:      /usr/local/lib/python3.6/dist-packages/keras/layers/embeddings.py
Type:      type
Subclasses:
```


In [49]:

```
def DeepMF(n_users, n_items, n_factors, min_rating, max_rating):

    # Item Layer
    item_input = Input(shape=[1], name='Item')
    item_embedding = Embedding(n_items, n_factors, embeddings_regularizer=l2(1e-6),
                               embeddings_initializer="glorot_normal",
                               name='ItemEmbedding')(item_input)
    item_vec = Flatten(name='FlattenItemE')(item_embedding)

    # Item Bias
    item_bias = Embedding(n_items, 1, embeddings_regularizer=l2(1e-6),
                          embeddings_initializer="glorot_normal",
                          name='ItemBias')(item_input)
    item_bias_vec = Flatten(name='FlattenItemBiasE')(item_bias)

    # User Layer
    user_input = Input(shape=[1], name='User')
    user_embedding = Embedding(n_users, n_factors, embeddings_regularizer=l2(1e-6),
                               embeddings_initializer="glorot_normal",
                               name='UserEmbedding')(user_input)
    user_vec = Flatten(name='FlattenUserE')(user_embedding)

    # User Bias
    user_bias = Embedding(n_users, 1, embeddings_regularizer=l2(1e-6),
                          embeddings_initializer="glorot_normal",
                          name='UserBias')(user_input)
    user_bias_vec = Flatten(name='FlattenUserBiasE')(user_bias)

    # Concatenation of Item and User & then Add Bias
    Concat = Concatenate(name="Concat")([item_vec, user_vec])
    ConcatDrop = Dropout(0.5)(Concat)

    # Use the Dense layer for non-linear interaction learning
    Dense_1 = Dense(32, name="Dense1", activation="relu")(ConcatDrop)
    Dense_1_Drop = Dropout(0.5)(Dense_1)
    Dense_2 = Dense(1, name="Dense2")(Dense_1_Drop)

    # Add the Bias
    AddBias = Add(name="AddBias")([Dense_2, item_bias_vec, user_bias_vec])

    # Scaling for each user
    y = Activation('sigmoid')(AddBias)
    rating_output = Lambda(lambda x: x * (max_rating - min_rating) + min_rating)(y)

    # Model Creation
```

```
model = Model([user_input, item_input], rating_output, name="DeepFM")

# Compile Model
model.compile(loss='mean_squared_error', optimizer=Adam(lr=0.001))

return model
```

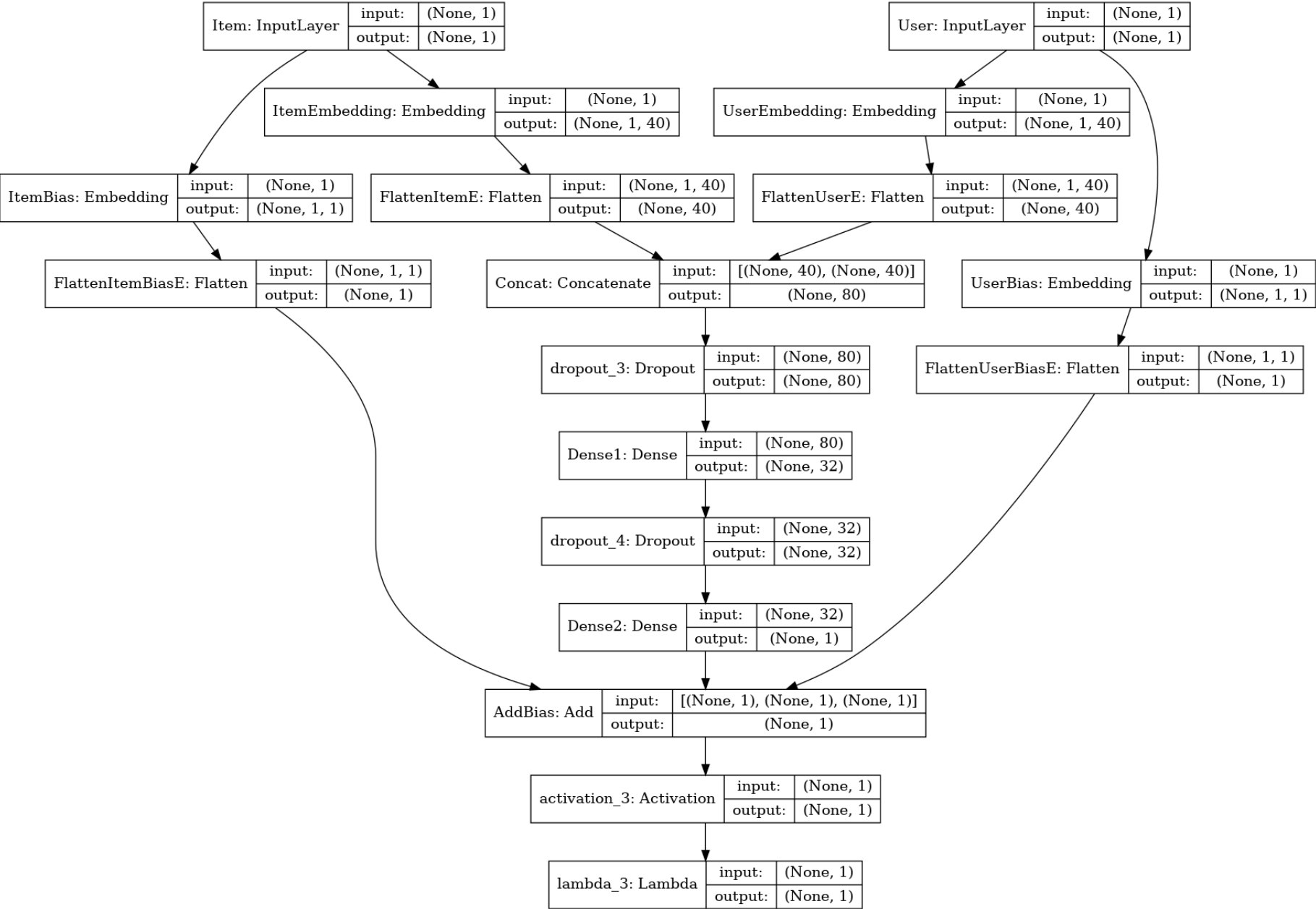
In [50]:

```
n_factors = 40
model = DeepMF(n_users, n_items, n_factors, min_rating, max_rating)
```

In [51]:

```
from keras.utils import plot_model  
plot_model(model, show_layer_names=True, show_shapes=True)
```

Out[51]:



In [52]:

```
model.summary()
```

Model: "DeepFM"

| Layer (type) | Output Shape | Param # | Connected to |
|----------------------------|---------------|---------|--|
| ===== | | | |
| Item (InputLayer) | (None, 1) | 0 | |
| User (InputLayer) | (None, 1) | 0 | |
| ItemEmbedding (Embedding) | (None, 1, 40) | 67280 | Item[0][0] |
| UserEmbedding (Embedding) | (None, 1, 40) | 37720 | User[0][0] |
| FlattenItemE (Flatten) | (None, 40) | 0 | ItemEmbedding[0][0] |
| FlattenUserE (Flatten) | (None, 40) | 0 | UserEmbedding[0][0] |
| Concat (Concatenate) | (None, 80) | 0 | FlattenItemE[0][0]
FlattenUserE[0][0] |
| dropout_3 (Dropout) | (None, 80) | 0 | Concat[0][0] |
| Dense1 (Dense) | (None, 32) | 2592 | dropout_3[0][0] |
| dropout_4 (Dropout) | (None, 32) | 0 | Dense1[0][0] |
| ItemBias (Embedding) | (None, 1, 1) | 1682 | Item[0][0] |
| UserBias (Embedding) | (None, 1, 1) | 943 | User[0][0] |
| Dense2 (Dense) | (None, 1) | 33 | dropout_4[0][0] |
| FlattenItemBiasE (Flatten) | (None, 1) | 0 | ItemBias[0][0] |
| FlattenUserBiasE (Flatten) | (None, 1) | 0 | UserBias[0][0] |
| AddBias (Add) | (None, 1) | 0 | Dense2[0][0]
FlattenItemBiasE[0][0]
FlattenUserBiasE[0][0] |
| activation_3 (Activation) | (None, 1) | 0 | AddBias[0][0] |
| lambda_3 (Lambda) | (None, 1) | 0 | activation_3[0][0] |
| ===== | | | |
| Total params: 110,250 | | | |
| Trainable params: 110,250 | | | |
| Non-trainable params: 0 | | | |

In [61]:

```
output = model.fit([train.USER, train.ITEM], train.RATING,  
                    batch_size=32, epochs=5, verbose=1,  
                    validation_data=([test.USER, test.ITEM], test.RATING))
```

Train on 80000 samples, validate on 20000 samples

Epoch 1/5

80000/80000 [=====] - 5s 60us/step - loss: 0.8432 - val_loss: 0.8693

Epoch 2/5

80000/80000 [=====] - 5s 60us/step - loss: 0.8378 - val_loss: 0.8677

Epoch 3/5

80000/80000 [=====] - 5s 60us/step - loss: 0.8318 - val_loss: 0.8652

Epoch 4/5

80000/80000 [=====] - 5s 60us/step - loss: 0.8262 - val_loss: 0.8635

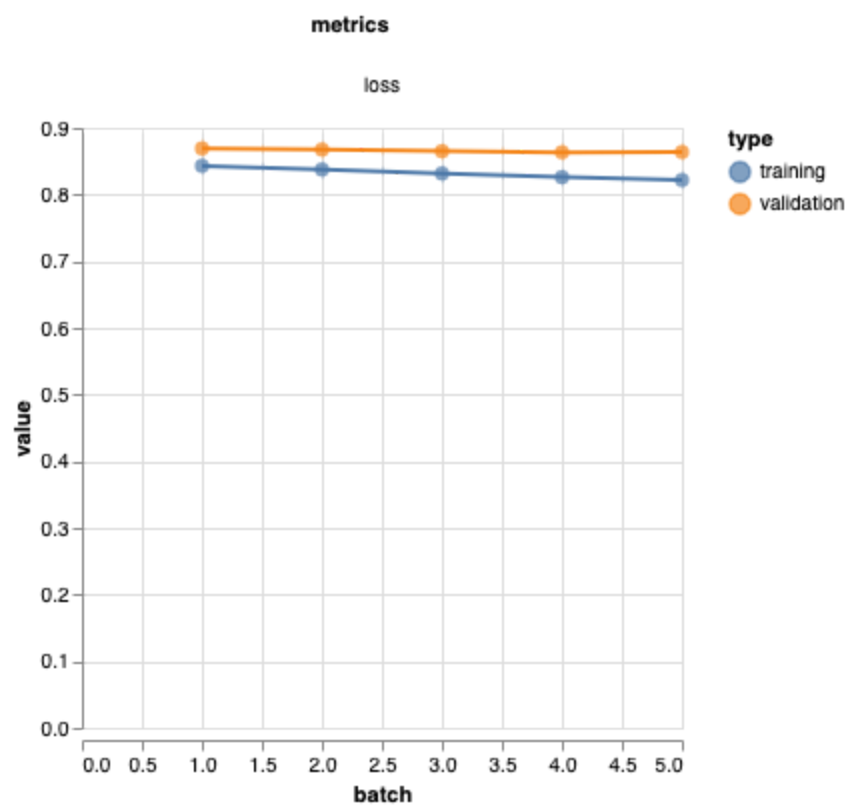
Epoch 5/5

80000/80000 [=====] - 5s 60us/step - loss: 0.8219 - val_loss: 0.8640

In [62]:

```
from recoflow.vis import MetricsVis  
MetricsVis(output.history)
```

Out[62]:



In [63]:

```
from recoflow.recommend import GetPredictions

predictions = GetPredictions(model, interaction)
predictions.head()
```

Out[63]:

| | USER | ITEM | RATING |
|---|------|------|----------|
| 0 | 195 | 241 | 4.077020 |
| 1 | 195 | 301 | 4.219788 |
| 2 | 195 | 376 | 2.237257 |
| 3 | 195 | 50 | 3.569890 |
| 4 | 195 | 345 | 3.739755 |

In [64]:

GetPredictions?

Signature: GetPredictions(model, data)

Docstring:

Get predictions for all user-item combinations

Params:

data (pandas.DataFrame): DataFrame of entire rating data

model (Keras.model): Trained keras model

Returns:

pd.DataFrame: DataFrame of rating predictions for each user and each item

File: /usr/local/lib/python3.6/dist-packages/recoflow/recommend.py

Type: function

In [65]:

```
from recoflow.metrics import RatingMetrics
```

In [68]:

```
RatingMetrics(test, predictions)
```

Out[68]:

| | MSE | RMSE | MAE |
|---|--------|--------|--------|
| 0 | 0.8619 | 0.9284 | 0.7355 |

Recommendations

In [75]:

```
from recoflow.recommend import ItemEmbedding, UserEmbedding
item_embedding = ItemEmbedding(model, "ItemEmbedding")
user_embedding = UserEmbedding(model, "UserEmbedding")
```

Getting Recommendation

- Given an Item, what are the similar Items
- Given a User, what items should I recommend

In [76]:

In [80]:

```
from recoflow.recommend import GetSimilar, ShowSimilarItems
similar_items = GetSimilar(item_embedding, k=5, metric="cosine")
similar_items
```

Out[80]:

```
array([[ 0, 1518, 1638, 1514, 1622, 1428],
       [ 1, 1678, 1509, 1204, 1363, 1435],
       [ 2,  902, 1322, 1435, 1362, 1353],
       ...,
       [1679, 1602, 1431, 1556, 1454, 1551],
       [1680, 1599, 1610, 1631, 1155, 1235],
       [1681, 1121, 1351, 1658, 1640, 1593]])
```

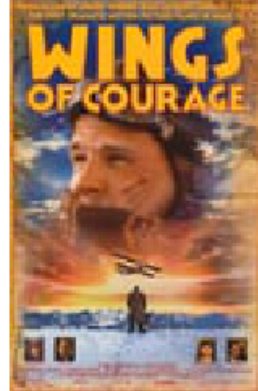
In [85]:

```
ShowSimilarItems(0, similar_items, item_encoder, items, image_path='/tf/data/posters/')
```

Toy Story (1995)



New Jersey Drive (1995) Bitter Sugar (Azucar Amargo) (1996) Wings of Courage (1995)



Cérémonie, La (1995)



Sliding Doors (1998)



In [91]:

```
recoflow.__version__
```

Out[91]:

'0.0.6'

In [93]:

```
from recoflow.metrics import RatingMetrics
```

For a user I want to find relevant

In [102]:

```
## For a user I want to find relevant
from recoflow.recommend import GetRankingTopK
GetRankingTopK
ranking_topk = GetRankingTopK
```

Ranking Top K

- Get all the predictions for all users
- Remove all the items the user has already seen (train)
- Sort the remaining data by predicted ratings
- Cut off at K

In [107]:

```
ranking_topk = GetRankingTopK(model, interaction, train, k=5)
```

In [108]:

```
ranking_topk.head()
```

Out[108]:

| | USER | ITEM | RATING | rank |
|---|------|------|----------|------|
| 0 | 0 | 1448 | 4.681684 | 1 |
| 1 | 0 | 407 | 4.654933 | 2 |
| 2 | 0 | 482 | 4.587464 | 3 |
| 3 | 0 | 602 | 4.564211 | 4 |
| 4 | 0 | 317 | 4.539346 | 5 |

In [109]:

```
n_users * 5
```

Out[109]:

4715

In [111]:

```
ranking_topk.shape
```

Out[111]:

(4715, 4)

Model - Neural Collaborative Filtering

In [116]:

```
from recoflow.models import NeuralCollaborativeFiltering
```

In [118]:

```
NeuralCollaborativeFiltering??
```


Signature:

```
NeuralCollaborativeFiltering(  
    n_users,  
    n_items,  
    n_factors,  
    min_rating,  
    max_rating,  
)
```

Docstring: <no docstring>

Source:

```
def NeuralCollaborativeFiltering(n_users, n_items, n_factors, min_rating, max_rating):
```

```
    # Item Layer
```

```
    item_input = Input(shape=[1], name='Item')
```

```
    # Item Embedding MF
```

```
    item_embedding_mf = Embedding(n_items, n_factors, embeddings_regularizer=l2(1e-6),  
                                  embeddings_initializer='he_normal',  
                                  name='ItemEmbeddingMF')(item_input)
```

```
    item_vec_mf = Flatten(name='FlattenItemMF')(item_embedding_mf)
```

```
    # Item embedding MLP
```

```
    item_embedding_mlp = Embedding(n_items, n_factors, embeddings_regularizer=l2(1e-6),  
                                   embeddings_initializer='he_normal',  
                                   name='ItemEmbeddingMLP')(item_input)
```

```
    item_vec_mlp = Flatten(name='FlattenItemMLP')(item_embedding_mlp)
```

```
    # User Layer
```

```
    user_input = Input(shape=[1], name='User')
```

```
    # User Embedding MF
```

```
    user_embedding_mf = Embedding(n_users, n_factors, embeddings_regularizer=l2(1e-6),  
                                  embeddings_initializer='he_normal',  
                                  name='UserEmbeddingMF')(user_input)
```

```
    user_vec_mf = Flatten(name='FlattenUserMF')(user_embedding_mf)
```

```
    # User Embedding MF
```

```
    user_embedding_mlp = Embedding(n_users, n_factors, embeddings_regularizer=l2(1e-6),  
                                   embeddings_initializer='he_normal',  
                                   name='UserEmbeddingMLP')(user_input)
```

```
    user_vec_mlp = Flatten(name='FlattenUserMLP')(user_embedding_mlp)
```

```
    # Multiply MF paths
```

```
    DotProductMF = Dot(axes=1, name='DotProductMF')([item_vec_mf, user_vec_mf])
```

```

# Concat MLP paths
ConcatMLP = Concatenate(name='ConcatMLP')([item_vec_mlp, user_vec_mlp])

# Use Dense to learn non-linear dense representation
Dense_1 = Dense(50, name="Dense1")(ConcatMLP)
Dense_2 = Dense(20, name="Dense2")(Dense_1)

# Concatenate MF and MLP paths
Concat = Concatenate(name="ConcatAll")([DotProductMF, Dense_2])

# Use Dense to learn non-linear dense representation
Pred = Dense(1, name="Pred")(Concat)

# Item Bias
item_bias = Embedding(n_items, 1, embeddings_regularizer=l2(1e-5), name='ItemBias')(item_input)
item_bias_vec = Flatten(name='FlattenItemBiasE')(item_bias)

# User Bias
user_bias = Embedding(n_users, 1, embeddings_regularizer=l2(1e-5), name='UserBias')(user_input)
user_bias_vec = Flatten(name='FlattenUserBiasE')(user_bias)

# Pred with bias added
PredAddBias = Add(name="AddBias")([Pred, item_bias_vec, user_bias_vec])

# Scaling for each user
y = Activation('sigmoid')(PredAddBias)
rating_output = Lambda(lambda x: x * (max_rating - min_rating) + min_rating)(y)

# Model Creation
model = Model([user_input, item_input], rating_output, name="NeuralCF")

# Compile Model
model.compile(loss='mean_squared_error', optimizer="adam")

return model

```

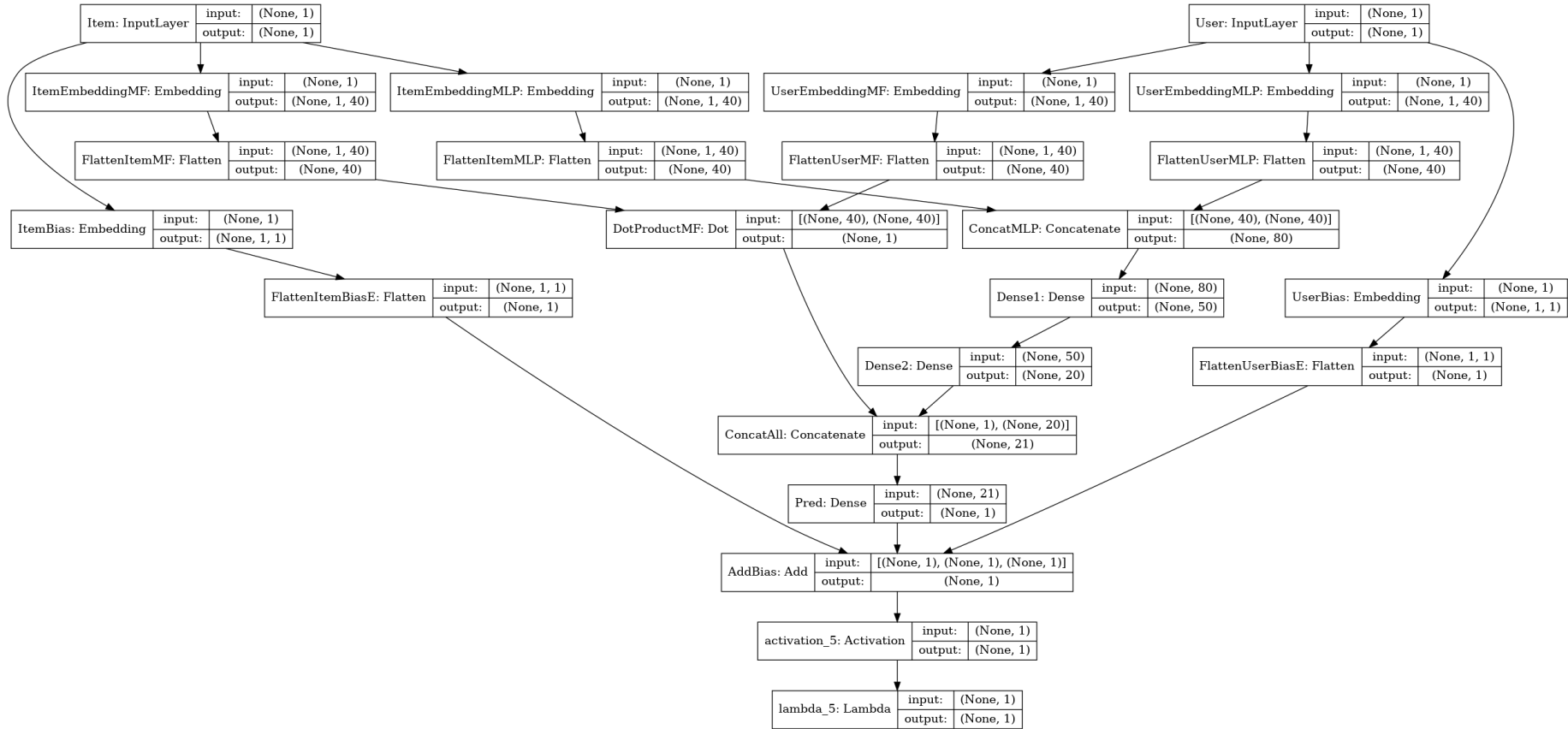
File: /usr/local/lib/python3.6/dist-packages/recoflow/models.py

Type: function

In [120]:

```
from recoflow.models import NeuralCollaborativeFiltering
ncf = NeuralCollaborativeFiltering(n_users, n_items, n_factors, min_rating, max_rating)
plot_model(ncf, show_layer_names=True, show_shapes=True)
```

Out[120]:



In [121]:

```
ncf.summary()
```

Model: "NeuralCF"

| Layer (type) | Output Shape | Param # | Connected to |
|------------------------------|---------------|---------|--|
| ===== | | | |
| Item (InputLayer) | (None, 1) | 0 | |
| User (InputLayer) | (None, 1) | 0 | |
| ItemEmbeddingMLP (Embedding) | (None, 1, 40) | 67280 | Item[0][0] |
| UserEmbeddingMLP (Embedding) | (None, 1, 40) | 37720 | User[0][0] |
| FlattenItemMLP (Flatten) | (None, 40) | 0 | ItemEmbeddingMLP[0][0] |
| FlattenUserMLP (Flatten) | (None, 40) | 0 | UserEmbeddingMLP[0][0] |
| ItemEmbeddingMF (Embedding) | (None, 1, 40) | 67280 | Item[0][0] |
| UserEmbeddingMF (Embedding) | (None, 1, 40) | 37720 | User[0][0] |
| ConcatMLP (Concatenate) | (None, 80) | 0 | FlattenItemMLP[0][0]
FlattenUserMLP[0][0] |
| FlattenItemMF (Flatten) | (None, 40) | 0 | ItemEmbeddingMF[0][0] |
| FlattenUserMF (Flatten) | (None, 40) | 0 | UserEmbeddingMF[0][0] |
| Dense1 (Dense) | (None, 50) | 4050 | ConcatMLP[0][0] |
| DotProductMF (Dot) | (None, 1) | 0 | FlattenItemMF[0][0]
FlattenUserMF[0][0] |
| Dense2 (Dense) | (None, 20) | 1020 | Dense1[0][0] |
| ConcatAll (Concatenate) | (None, 21) | 0 | DotProductMF[0][0]
Dense2[0][0] |
| ItemBias (Embedding) | (None, 1, 1) | 1682 | Item[0][0] |
| UserBias (Embedding) | (None, 1, 1) | 943 | User[0][0] |
| Pred (Dense) | (None, 1) | 22 | ConcatAll[0][0] |
| FlattenItemBiasE (Flatten) | (None, 1) | 0 | ItemBias[0][0] |
| FlattenUserBiasE (Flatten) | (None, 1) | 0 | UserBias[0][0] |

| | | | |
|---------------------------|-----------|---|--|
| AddBias (Add) | (None, 1) | 0 | Pred[0][0]
FlattenItemBiasE[0][0]
FlattenUserBiasE[0][0] |
| activation_5 (Activation) | (None, 1) | 0 | AddBias[0][0] |
| lambda_5 (Lambda) | (None, 1) | 0 | activation_5[0][0] |
| ===== | | | |
| Total params: 217,717 | | | |
| Trainable params: 217,717 | | | |
| Non-trainable params: 0 | | | |