Matrix Factorization

MovieLens DataSet -> https://grouplens.org/datasets/movielens/ (https://grouplens.org/datasets/movielens/)

In [1]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
import numpy as np
import pandas as pd
```

In [9]:
```
!pwd
```

```
/tf/notebooks/recommendation/Strata-NY
```

Get the Data

In [ ]:
```
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 [15]:
```
# ratings = pd.read_csv("/tf/notebooks/data/data/ratings.csv")
# users = pd.read_csv("/tf/notebooks/data/data/users.csv")
# items = pd.read_csv("/tf/notebooks/data/data/items.csv")
```

About the Data

In [21]:
```
ratings.columns.tolist()
```

Out[21]:
```
['user_id', 'movie_id', 'rating', 'unix_timestamp']
```

In [16]:
```
ratings.head()
```

Out[16]:

	user_id	movie_id	rating	unix_timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

In [27]:
```
#items.head()
```

In [28]:
```
#users.head()
```

Let us build a basic model

In [29]:
```
from recoflow.datasets import SampleData
```

In [30]:
```
sample_users, sample_items, sample_ratings = SampleData(users, items, ratings)
```

Preprocessing

Encoding

- Label Encode: Index the User and Items => They can be string, unique values,

Train and Test Split

- Random Split
- Stratified Split
- Chronological Split

In [33]:
```
from recoflow.preprocessing import EncodeUserItem
```

In [35]:
```
interaction, n_users, n_items, user_encoder, item_encoder = EncodeUserItem(sample_ratings,
                                                                           "user_id",
                                                                           "movie_id",
                                                                           "rating",
                                                                           "unix_timestamp")
```

```
Number of users:  10
Number of items:  6
```

In [36]:
```
interaction.head()
```

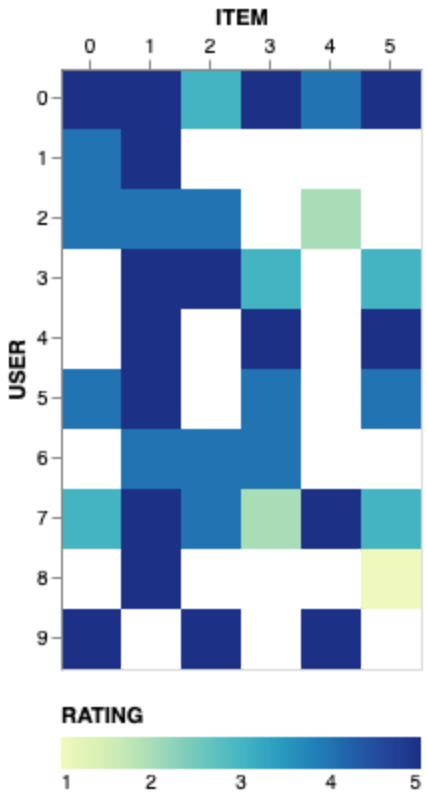
Out[36]:

	user_id	movie_id	RATING	TIMESTAMP	USER	ITEM
1052	2	50	5	888552084	1	1
1090	8	50	5	879362124	4	1
3672	6	95	2	883602133	2	4
4280	1	82	5	878542589	0	3
4596	12	82	4	879959610	6	3

In [37]:
```
from recoflow.vis import InteractionVis
```

In [38]:
```
InteractionVis(interaction)
```

Out[38]:



In [39]:
```
interaction[interaction.USER == 9]
```

Out[39]:

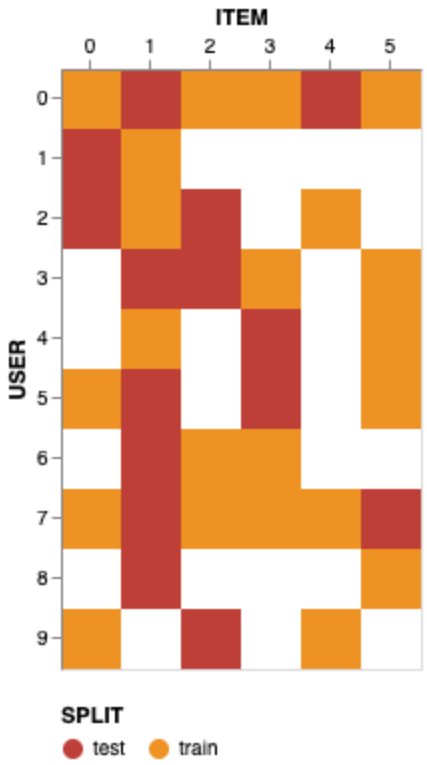
	user_id	movie_id	RATING	TIMESTAMP	USER	ITEM
24076	16	95	5	877728417	9	4
38429	16	1	5	877717833	9	0
54505	16	71	5	877721071	9	2

In [42]:
```
from recoflow.preprocessing import RandomSplit, StratifiedSplit, ChronoSplit
from recoflow.vis import TrainTestVis
```

In [44]:
```
#train, test = RandomSplit(interaction, [0.6, 0.4])
train, test = StratifiedSplit(interaction, [0.6, 0.4])
```

In [45]:
```
TrainTestVis(train, test)
```

Out[45]:



Build the Learning Architecture

Explicit Matrix Factorisation -> Splits into user and item representation / embeddings with latent factors / embedding dimensions

In [57]:
```
from keras.models import Model
from keras.layers import Embedding, Dot, Input, Flatten
from keras.regularizers import l2
```

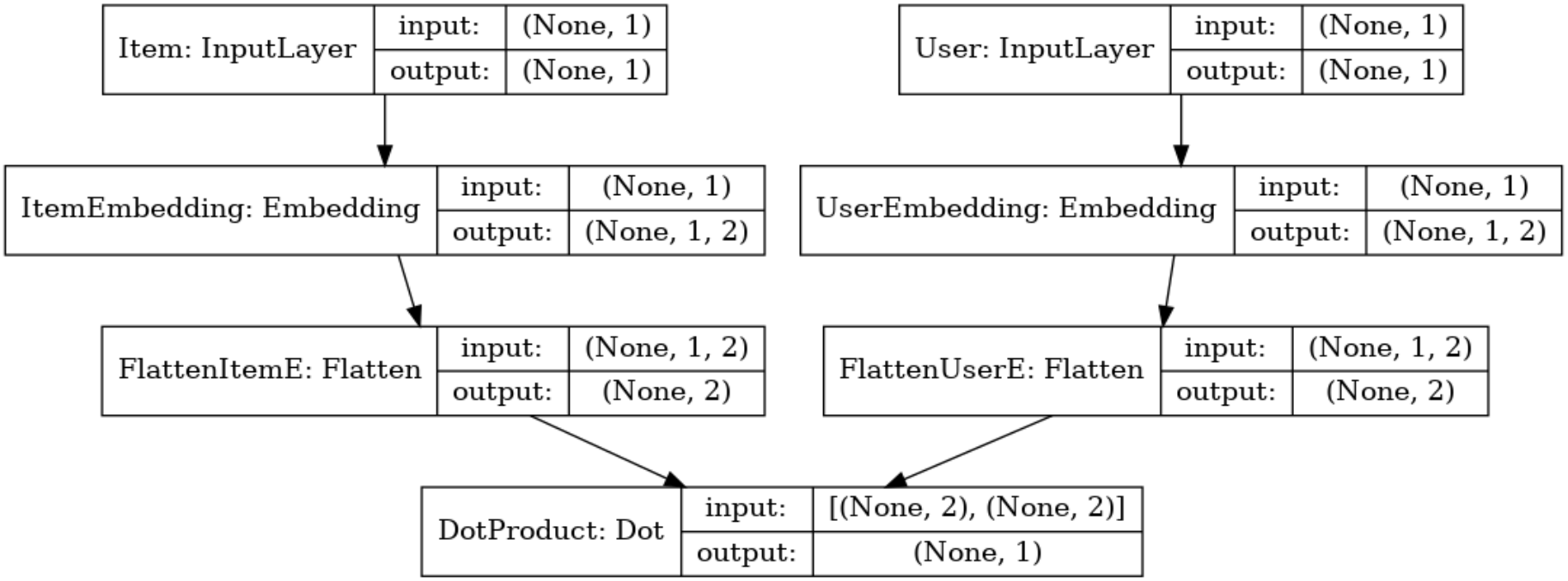
In [69]:
```
def ExplicitMF(n_users, n_items, n_factors):

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

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

    # Dot Product of Item and User
    rating = Dot(axes=1, name="DotProduct")([item_vec, user_vec])

    # Create the Model
    model = Model([user_input, item_input], rating, name="ExplicitMF")

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

    return model
```

In [70]:
```
n_factors= 2
model = ExplicitMF(n_users, n_items, n_factors)
```

In [71]:
```
from keras.utils import plot_model
```

In [72]:
```
plot_model(model, show_layer_names=True, show_shapes=True)
```

Out[72]:



In [73]:
```
model.summary()
```

```
Model: "ExplicitMF"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
Item (InputLayer)               (None, 1)            0
__________________________________________________________________________________________________
User (InputLayer)               (None, 1)            0
__________________________________________________________________________________________________
ItemEmbedding (Embedding)       (None, 1, 2)         12          Item[0][0]
__________________________________________________________________________________________________
UserEmbedding (Embedding)       (None, 1, 2)         20          User[0][0]
__________________________________________________________________________________________________
FlattenItemE (Flatten)          (None, 2)            0           ItemEmbedding[0][0]
__________________________________________________________________________________________________
FlattenUserE (Flatten)          (None, 2)            0           UserEmbedding[0][0]
__________________________________________________________________________________________________
DotProduct (Dot)                (None, 1)            0           FlattenItemE[0][0]
                                                                 FlattenUserE[0][0]
==================================================================================================
Total params: 32
Trainable params: 32
Non-trainable params: 0
__________________________________________________________________________________________________
```

Let us learn

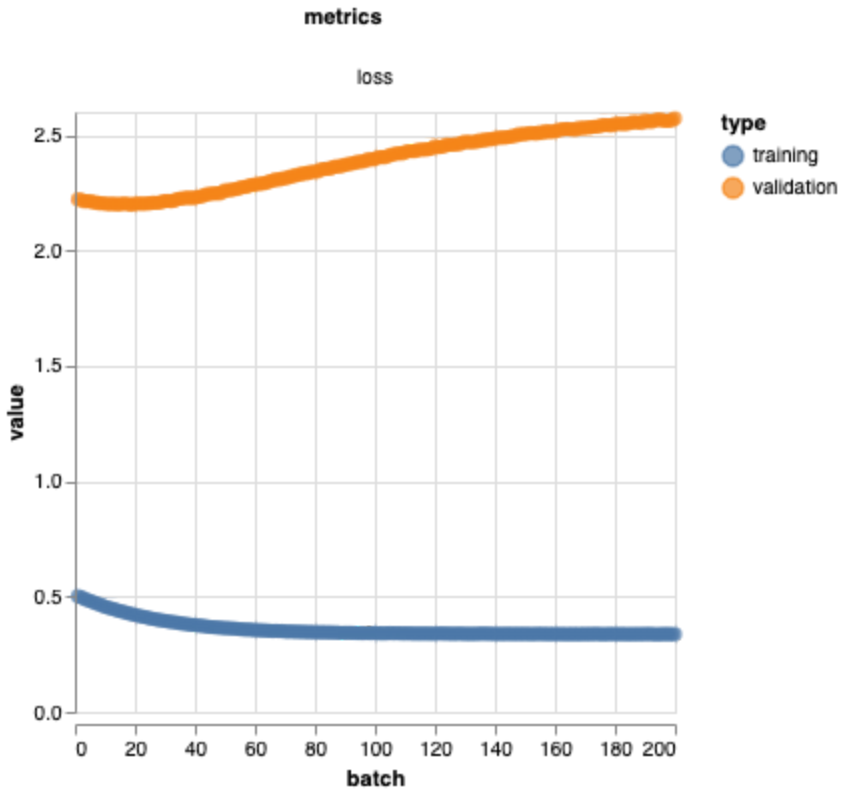
In [80]:
```
%%time
output = model.fit([train.USER, train.ITEM], train.RATING, batch_size=1, epochs=200, verbose=0,
                   validation_data=([test.USER, test.ITEM], test.RATING))
```

```
CPU times: user 4.97 s, sys: 299 ms, total: 5.27 s
Wall time: 3.92 s
```

In [81]:
```
from recoflow.vis import MetricsVis
```

In [82]:
```
MetricsVis(output.history)
```

Out[82]:



Let us get the Embedding

In [94]:
```
from recoflow.recommend import UserEmbedding, ItemEmbedding, _GetEmbedding
```

In [97]:
```
model.get_layer(name = "ItemEmbedding").get_weights()[0]
```

Out[97]:
```
array([[1.6080673, 1.4615829],
       [1.9636486, 1.9338186],
       [1.2449423, 1.2702792],
       [1.3422577, 1.3380105],
       [1.7207041, 1.63891  ],
       [1.6555494, 1.6383454]], dtype=float32)
```

In [88]:
```
item_embedding = ItemEmbedding(model, "ItemEmbedding")
user_embedding = UserEmbedding(model, "UserEmbedding")
```

In [92]:
```
item_embedding.shape, user_embedding.shape
```

Out[92]:
```
((6, 2), (10, 2))
```

In [89]:
```
item_embedding
```

Out[89]:
```
array([[1.6080673, 1.4615829],
       [1.9636486, 1.9338186],
       [1.2449423, 1.2702792],
       [1.3422577, 1.3380105],
       [1.7207041, 1.63891  ],
       [1.6555494, 1.6383454]], dtype=float32)
```

In [90]:
```
user_embedding
```

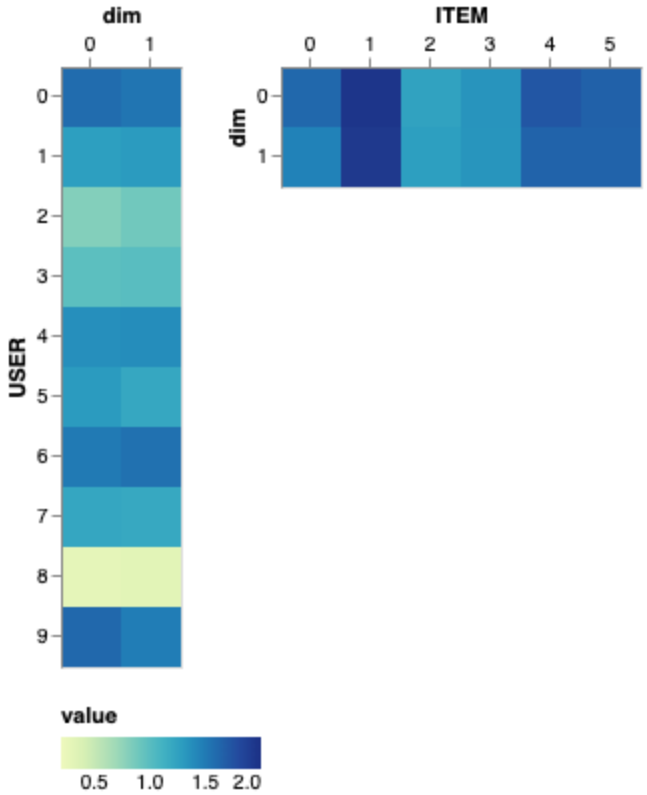
Out[90]:
```
array([[1.5905198 , 1.532982  ],
       [1.271329  , 1.2975943 ],
       [0.8092552 , 0.8855725 ],
       [0.9873716 , 1.0008088 ],
       [1.3755066 , 1.3917732 ],
       [1.3019269 , 1.2066387 ],
       [1.5087961 , 1.5602189 ],
       [1.215981  , 1.1977849 ],
       [0.28792676, 0.31721923],
       [1.610516  , 1.4901564 ]], dtype=float32)
```

In [104]:
```
from recoflow.vis import EmbeddingVis
import altair as alt
```

In [105]:
```
def EmbeddingVis(embedding, n_factors, name):
    embedding_df_wide = pd.DataFrame(embedding)
    embedding_df_wide[name]= embedding_df_wide.index
    embedding_df = pd.melt(embedding_df_wide, id_vars=[name], value_vars=np.arange(n_factors).tolist(),
       var_name='dim', value_name='value')


    if name == "ITEM":
        vis = alt.Chart(embedding_df).mark_rect().encode(
            alt.X(field=name, type="nominal", axis=alt.Axis(orient="top", labelAngle=0)),
            alt.Y(field="dim", type="nominal", axis=alt.Axis(orient="left")),
            alt.Color(field="value", type="quantitative",
                  scale=alt.Scale(type="bin-ordinal", scheme='yellowgreenblue', nice=True),
                  legend=alt.Legend(titleOrient='top', orient="bottom",
                                    direction= "horizontal", tickCount=5))
        ).properties(
            width=180,
            height=30*n_factors
        )

    else:
        vis = alt.Chart(embedding_df).mark_rect().encode(
            alt.X(field="dim", type="nominal", axis=alt.Axis(orient="top", labelAngle=0)),
            alt.Y(field=name, type="nominal", axis=alt.Axis(orient="left")),
            alt.Color(field="value", type="quantitative",
                  scale=alt.Scale(type="bin-ordinal", scheme='yellowgreenblue', nice=True),
                  legend=alt.Legend(titleOrient='top', orient="bottom",
                                    direction= "horizontal", tickCount=5))
        ).properties(
            width=30*n_factors,
            height=300
        )


    return vis
```

In [107]:
```
EmbeddingVis(user_embedding, n_factors, "USER") | EmbeddingVis(item_embedding, n_factors, "ITEM")
```

Out[107]:

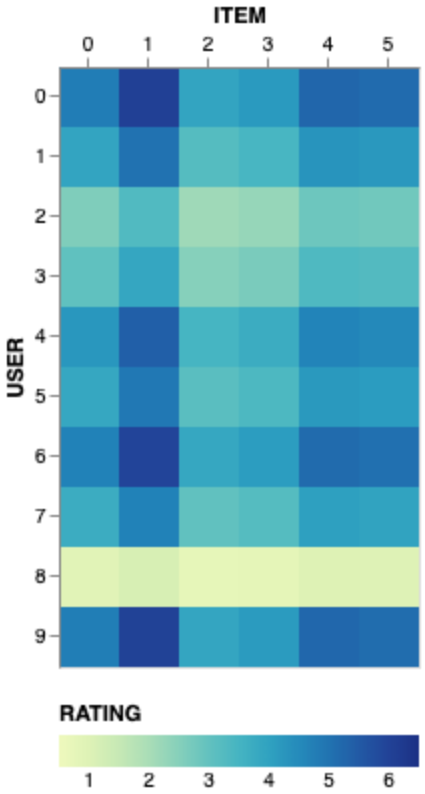


In [108]:
```
from recoflow.recommend import GetPredictions
```

In [109]:
```
predictions = GetPredictions(model, interaction)
```

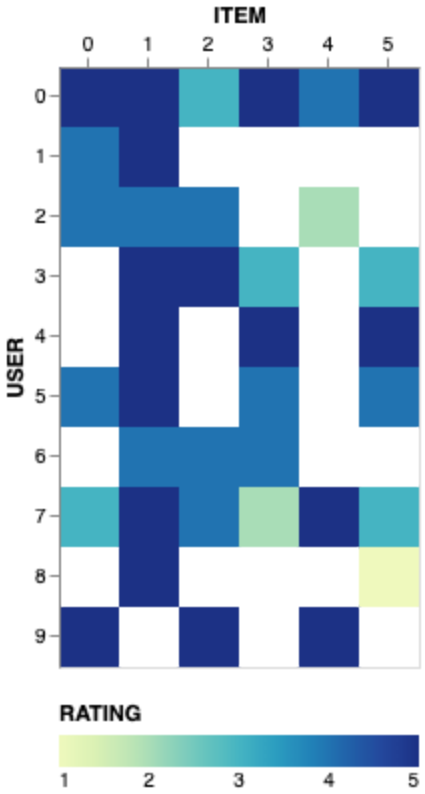
In [112]:
```
InteractionVis(predictions)
```

Out[112]:



In [113]:
```
InteractionVis(interaction)
```

Out[113]:

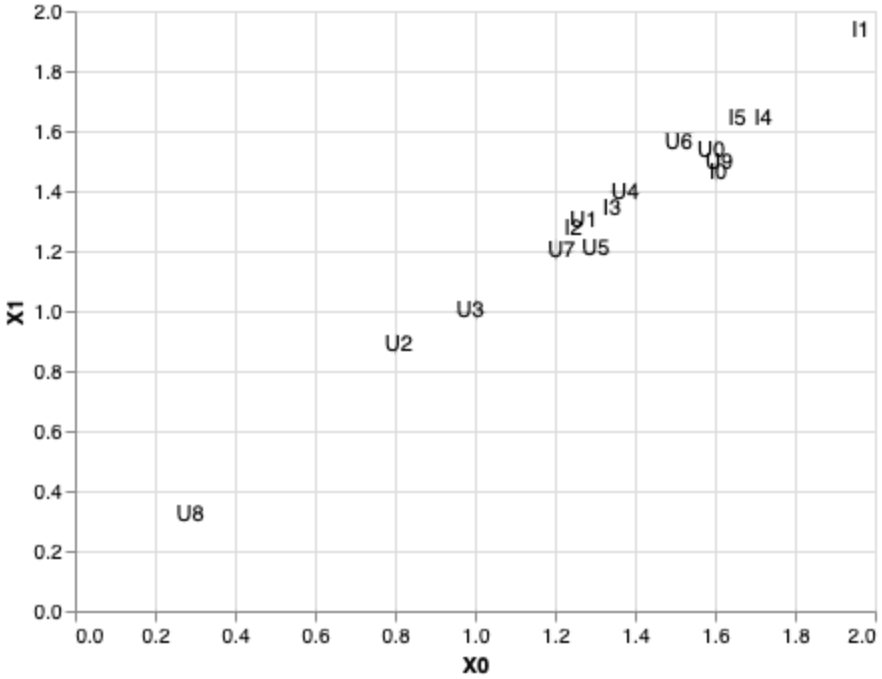


In [114]:
```
from recoflow.vis import SimilarityVis
```

In [119]:
```
def SimilarityVis(item_embedding, user_embedding):

    item_embedding_df_wide = pd.DataFrame(item_embedding)
    user_embedding_df_wide = pd.DataFrame(user_embedding)

    item_embedding_df_wide.reset_index(inplace=True)
    item_embedding_df_wide["idx"] = item_embedding_df_wide["index"].apply(lambda x: "I" + str(x))
    item_embedding_df_wide.columns = ["index", "X0", "X1", "idx" ]

    user_embedding_df_wide.reset_index(inplace=True)
    user_embedding_df_wide["idx"] = user_embedding_df_wide["index"].apply(lambda x: "U" + str(x))
    user_embedding_df_wide.columns = ["index", "X0", "X1", "idx" ]

    embedding_df_wide = pd.concat([item_embedding_df_wide, user_embedding_df_wide])

    base = alt.Chart(embedding_df_wide).encode(
        alt.X("X0:Q", axis = alt.Axis(bandPosition = 0.5)),
        alt.Y("X1:Q", axis = alt.Axis(bandPosition = 0.5))
    )

    vis = base.mark_point(size=0) + base.mark_text().encode(text="idx")

    return vis
```

In [120]:
```
SimilarityVis(item_embedding, user_embedding)
```

Out[120]:



In [ ]: