Collaborative filtering on Google Analytics data

This notebook demonstrates how to implement a WALS matrix refactorization approach to do collaborative filtering.

In [1]:
```
import os
PROJECT = "qwiklabs-gcp-03-2787a45a1534" # REPLACE WITH YOUR PROJECT ID
BUCKET = "qwiklabs-gcp-03-2787a45a1534" # REPLACE WITH YOUR BUCKET NAME
REGION = "us-central1" # REPLACE WITH YOUR BUCKET REGION e.g. us-central1

# Do not change these
os.environ["PROJECT"] = PROJECT
os.environ["BUCKET"] = BUCKET
os.environ["REGION"] = REGION
os.environ["TFVERSION"] = "1.13"
```

In [2]:
```
%%bash
gcloud config set project $PROJECT
gcloud config set compute/region $REGION
```

```
Updated property [core/project].
Updated property [compute/region].
```

In [3]:
```
import tensorflow as tf
print(tf.__version__)
```

```
1.15.5
```

Create raw dataset

For collaborative filtering, we don't need to know anything about either the users or the content. Essentially, all we need to know is userId, itemId, and rating that the particular user gave the particular item.

In this case, we are working with newspaper articles. The company doesn't ask their users to rate the articles. However, we can use the time-spent on the page as a proxy for rating.

Normally, we would also add a time filter to this ("latest 7 days"), but our dataset is itself limited to a few days.

In [4]:
```
from google.cloud import bigquery
bq = bigquery.Client(project = PROJECT)

sql = """
WITH CTE_visitor_page_content AS (
    SELECT
        # Schema: https://support.google.com/analytics/answer/3437719?hl=en
        # For a completely unique visit-session ID, we combine combination of fu
        CONCAT(fullVisitorID,'-',CAST(visitNumber AS STRING)) AS visitorId,
        (SELECT MAX(IF(index=10, value, NULL)) FROM UNNEST(hits.customDimensions
```

```
        (LEAD(hits.time, 1) OVER (PARTITION BY fullVisitorId ORDER BY hits.time
    FROM
        `cloud-training-demos.GA360_test.ga_sessions_sample`,
        UNNEST(hits) AS hits
    WHERE
        # only include hits on pages
        hits.type = "PAGE"
GROUP BY
        fullVisitorId,
        visitNumber,
        latestContentId,
        hits.time )
-- Aggregate web stats
SELECT
    visitorId,
    latestContentId as contentId,
    SUM(session_duration) AS session_duration
FROM
    CTE_visitor_page_content
WHERE
    latestContentId IS NOT NULL
GROUP BY
    visitorId,
    latestContentId
HAVING
    session_duration > 0
"""

df = bq.query(sql).to_dataframe()
df.head()
```

Out[4]:

	visitorId	contentId	session_duration
0	1031539128969021923-1	299918857	46074
1	1655026264169370690-165	299170525	28438
2	1777072527276763113-113	299827911	6698
3	1818199630884742957-45	299936493	42307
4	2211768598185554204-465	255478055	155887

In [5]:

```
stats = df.describe()
stats
```

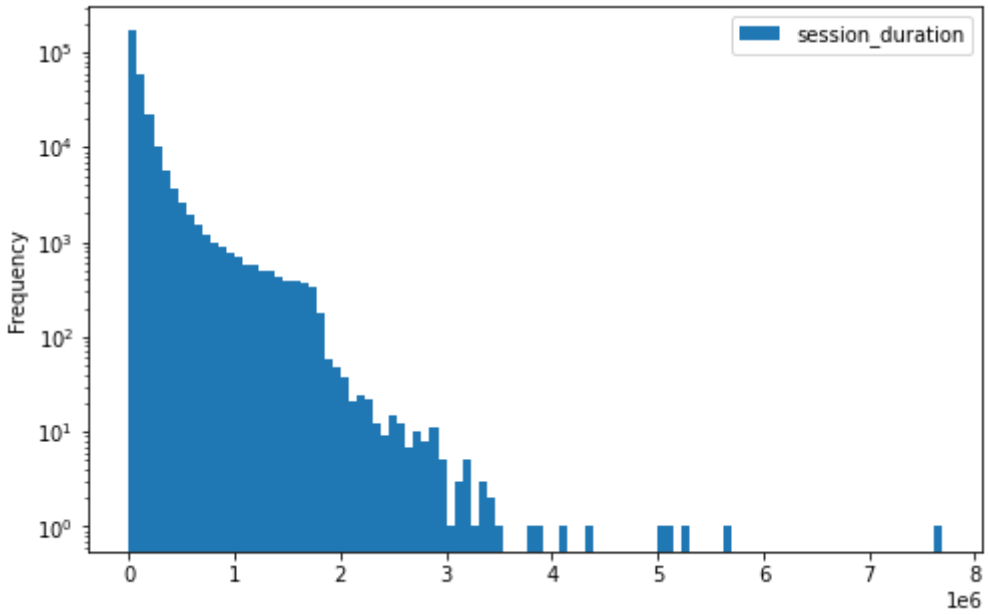
Out[5]:

	session_duration
count	2.843020e+05
mean	1.247962e+05
std	2.311864e+05
min	1.000000e+00
25%	1.607700e+04
50%	5.626050e+04
75%	1.271750e+05

	session_duration
max	7.690598e+06

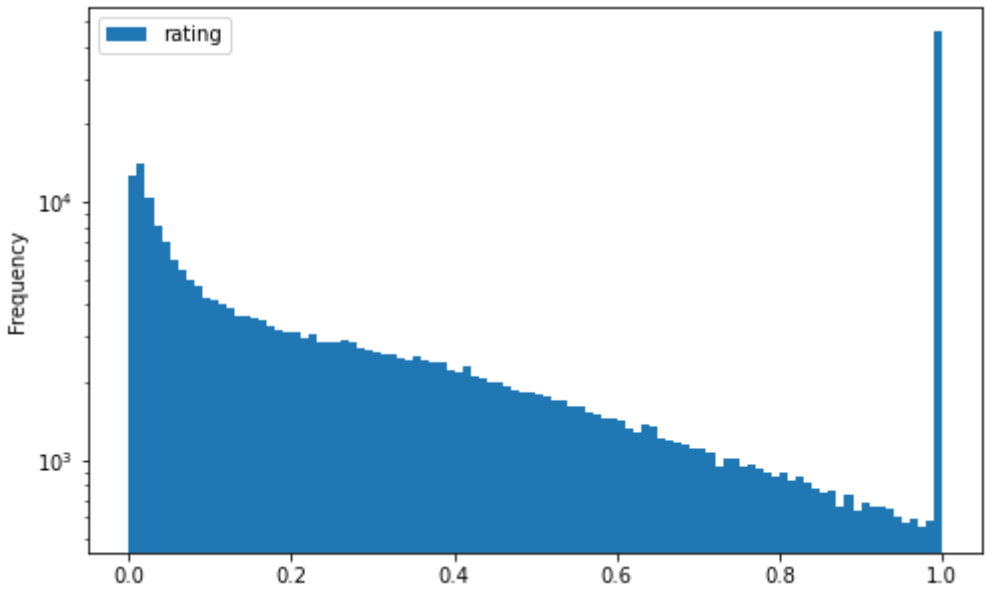
In [6]:
```
df[["session_duration"]].plot(kind="hist", logy=True, bins=100, figsize=[8,5])
```

Out[6]:
```
<AxesSubplot:ylabel='Frequency'>
```



In [7]:
```
# The rating is the session_duration scaled to be in the range 0-1.  This will h
median = stats.loc["50%", "session_duration"]
df["rating"] = 0.3 * df["session_duration"] / median
df.loc[df["rating"] > 1, "rating"] = 1
df[["rating"]].plot(kind="hist", logy=True, bins=100, figsize=[8,5])
```

Out[7]:
```
<AxesSubplot:ylabel='Frequency'>
```



In [8]:
```
del df["session_duration"]
```

In [9]:
```
%%bash
rm -rf data
mkdir data
```

In [10]:
```
df.to_csv(path_or_buf = "data/collab_raw.csv", index = False, header = False)
```

In [11]:
```
!head data/collab_raw.csv
```

```
1031539128969021923-1,299918857,0.24568213933399097
1655026264169370690-165,299170525,0.15164102700829177
1777072527276763113-113,299827911,0.03571599968005972
1818199630884742957-45,299936493,0.22559522222518463
2211768598185554204-465,255478055,0.831242168128616
2695873766870665370-5,299587923,0.20680406324152825
28411951843569 5096-1,299930675,1.0
2869452701170253857-741,299693260,0.04747913722771749
3010380941869544318-1441,299133651,0.027877462873596926
3074506062059024768-18,299945076,0.19248673580931558
```

Create dataset for WALS

The raw dataset (above) won't work for WALS:

1. The userId and itemId have to be 0,1,2 ... so we need to create a mapping from visitorId (in the raw data) to userId and contentId (in the raw data) to itemId.
2. We will need to save the above mapping to a file because at prediction time, we'll need to know how to map the contentId in the table above to the itemId.
3. We'll need two files: a "rows" dataset where all the items for a particular user are listed; and a "columns" dataset where all the users for a particular item are listed.

Mapping

In [12]:
```
import pandas as pd
import numpy as np
def create_mapping(values, filename):
    with open(filename, 'w') as ofp:
        value_to_id = {value:idx for idx, value in enumerate(values.unique())}
        for value, idx in value_to_id.items():
            ofp.write("{},{}\n".format(value, idx))
    return value_to_id

df = pd.read_csv(filepath_or_buffer = "data/collab_raw.csv",
                 header = None,
                 names = ["visitorId", "contentId", "rating"],
                dtype = {"visitorId": str, "contentId": str, "rating": np.float})
df.to_csv(path_or_buf = "data/collab_raw.csv", index = False, header = False)
user_mapping = create_mapping(df["visitorId"], "data/users.csv")
item_mapping = create_mapping(df["contentId"], "data/items.csv")
```

In [13]:
```
!head -3 data/*.csv
```

```
==> data/collab_raw.csv <==
1031539128969021923-1,299918857,0.2456821393339909
1655026264169370690-165,299170525,0.1516410270082917
1777072527276763113-113,299827911,0.0357159996800597

==> data/items.csv <==
299918857,0
299170525,1
299827911,2

==> data/users.csv <==
1031539128969021923-1,0
1655026264169370690-165,1
1777072527276763113-113,2
```

In [14]:
```
df["userId"] = df["visitorId"].map(user_mapping.get)
df["itemId"] = df["contentId"].map(item_mapping.get)
```

In [15]:
```
mapped_df = df[["userId", "itemId", "rating"]]
mapped_df.to_csv(path_or_buf = "data/collab_mapped.csv", index = False, header =
mapped_df.head()
```

Out[15]:

	userId	itemId	rating
0	0	0	0.245682
1	1	1	0.151641
2	2	2	0.035716
3	3	3	0.225595
4	4	4	0.831242

Creating rows and columns datasets

In [16]:
```
import pandas as pd
import numpy as np
mapped_df = pd.read_csv(filepath_or_buffer = "data/collab_mapped.csv", header =
mapped_df.head()
```

Out[16]:

	userId	itemId	rating
0	0	0	0.245682
1	1	1	0.151641
2	2	2	0.035716
3	3	3	0.225595
4	4	4	0.831242

In [17]:
```
NITEMS = np.max(mapped_df["itemId"]) + 1
NUSERS = np.max(mapped_df["userId"]) + 1
```

```
mapped_df["rating"] = np.round(mapped_df["rating"].values, 2)
print("{} items, {} users, {} interactions".format( NITEMS, NUSERS, len(mapped_d
```

```
5670 items, 120869 users, 284302 interactions
```

In [18]:

```
grouped_by_items = mapped_df.groupby("itemId")
iter = 0
for item, grouped in grouped_by_items:
    print(item, grouped["userId"].values, grouped["rating"].values)
    iter = iter + 1
    if iter > 5:
        break
```

```
0 [    0   362   588 ... 11687 58106 14393] [0.25 1.   0.3  ... 0.2  0.6  0.36]
1 [    1    30   249   351   800  1096  1285  1464  1472  1475  1750  1753
  1959  2014  2018  2162  2229  2261  2274  2379  2467  2479  2522  2543
  2699  2702  3255  3315  3393  3934  4134  4316  4621  4738  1963  5187
  5431  5433  2299  5537  5548  5676  5780  5985  6132  6277  6386  6610
  7075  7439  7508  7610  7631  7742  7749  7785  7827  8028  2009  8253
  8311  8430  8435  8481  8563  8670  8710  8881  9063  9123  9521 10014
 10069  1142 10569  7794 10783 11098 11216 11246  8547 11535 11629 11833
 12099 12120 12199 12242 12473 12500 12522 12911 13119 13304  5034 13447
 13738 13915 14251 14608 14609 14697 14709 15057 15200 15530 15531 15712
 15823 15863 10753 15947 16076 16322 16505 16593  8923 16839 16934 17140
 17214   492 17657 17779 17784 17811 17904  7870 18429 18510 18749 18755
 18774 18908 19057 19193 19200 11721  9042  3042 19285 19300 19409 19494
 19683 12394 19732 19814 19830 19843 19877   854  7080 20002 20148 20169
 20390 20675 20801 20994 21160 21186 11434 21327 21354 21561 21630 21829
 21866 21962 22003 22569 22810 22825 22831  5090 23014 23060 23131 23251
 23411 11407 23490 16817 23777    65 23960  9431 24320 24324  6991 24435
 24453 24515 24572 24787  7673 20530  5296 25230 20944 25504 25524 25614
 25628 25810  6224 14559 19539  3793 26304 24275 19995  1082 26642 26676
 26688 26712 26795 26823 18147 23013 27559 11637 16914 27937 27964  3360
  3384  6800 28377 28427 28430 28520 28552 28632  4389 28808 28847 28886
  4675 22693 29007 29053 29143 18555 29308 29340 29409  5527 29523 29668
 14117 29796 19410 30039 23957 30192 30245   450  3761 30371 30394   841
 30868 30871  7698 30958 30987 24965  1760  8017  8163 29514  5723 31600
 31608 31699 31715 31726 31746 31779 14359 31869 31953 30075 32014 32022
 32087 32183 30305 32254 32416 32470 32667 32718 28885 32816 32889 33007
 33204  5472 33279 33319 33353 33364 33535 33629 33638 33794 33801 33829
 33837 33865 34180 34440 34650 10526 28955 34774 34861 34925 34976 35278
 35338 35398 35406 35446 31681 35529 35605   137 35740 32138 35904 17438
 15129 36046 36054 36089 36106 32561 36274 36276 36385 36389 36450 24997
 36717 36750 36760  8236 36885 11255 29630  6019 37321 37355 37357 37381
   121 14668 37583 37639 37651 37692 37749 37822 37901 37995 38470  5243
 38587 38612 38617 20962 38834 21286 38969 38999 39057 14409 35622 39149
  3486 39391 39433 17368  3879 39521 32334 39532 12611 19900 39702 39713
 39926 39928 39957 40181 40871  6502 40974 41091 41097 24206 41116 41268
 41290 41543 10392 41580 41588 41616  4800 34736 41743 27047 41773 41838
 41856 41960 27498  8496  2841 42380 23893  9416 42897 42920 42976 10159
 38063 10366  4643 43244 43286 43315  1767 43547 43556 43864 43926 43990
 19226 44024 14478 44095 44109 21681 44303 34249 44462 44762 22822 45007
 45052 45146  5449 45258 31540 45398 45506 45543 29846 21437 45644 45899
 19769 39499 24535 15350 44645 46280 46342 46582 46722 46835 11179  8464
 46894 47052  6021 25868 47232 19569 33988  9760  9794 47644 47651 42973
 22345 34502 47856 26818 47904 47944 28968  4988 48109 48141 48154 48248
 48264 48320 48334  8615 37160 48588 48619 42367 42370 48732 48762 48796
 48988 49089  1324 24816 10715 16198 49786 46973 14072 49983 50094 25957
 30097 50279 50286  6700 14958   584 37687 50644 38082 44772 50859 38570
 37005 51391 51435 51446 51714 17196 51764 30413 41206  9893  4505 52213
```

```
 52320 10649  7998 52505 20862 52571  5570  8987 52896 52943 33773 25999
 28225 53150 53334   964 53370 53413  1232 34594  7588 46387 53657  2686
 35362 54263 35648 37411 54359 54424 54586 54728 39731 54781 15447 30751
 54822  1295 47901 38343 51086 53939 55272 45362 14028 55466 55472 55477
 42336 55492 55506 21490  3075  9152 29973 55622 26020 55693 55774 55796
 55889 39555 56077 56224 36673 40220 56414 18712  5569 56753 56838 23790
 39212 56993 19624  6692 57228  7048 57241 57258  4088 43128  1737 15877
 57677 20883 33291 57889] [0.15 0.2  0.11 0.8  0.29 0.03 0.04 0.35 0.29 1.   1.
0.68 1.   0.21
 1.   0.49 0.02 0.2  0.8  0.13 1.   0.07 0.11 0.89 1.   0.48 0.14 0.72
 0.63 0.84 1.   1.   0.7  1.   0.75 1.   0.59 0.04 0.44 0.03 0.01 1.
 0.92 1.   0.28 0.05 0.64 0.05 1.   0.36 0.61 0.91 0.78 0.95 0.8  0.06
 0.03 0.42 1.   0.04 1.   1.   0.21 1.   0.38 0.73 1.   0.49 0.   1.
 0.68 0.82 0.48 0.05 1.   1.   0.   0.23 0.72 0.7  0.01 1.   0.02 0.32
 1.   0.81 1.   0.46 1.   0.65 0.04 0.   1.   1.   0.41 0.   0.54 0.36
 0.01 0.9  0.57 0.22 0.05 0.79 1.   1.   0.61 0.   0.5  0.39 0.05 0.99
 0.46 0.97 0.42 0.46 1.   0.44 1.   1.   0.76 0.05 1.   1.   1.   0.49
 1.   0.07 0.   1.   0.   0.58 0.26 0.07 1.   0.13 0.69 0.65 0.14 0.1
 1.   0.32 1.   0.46 0.17 0.63 1.   0.75 1.   0.08 0.63 0.56 0.57 0.68
 0.17 1.   1.   1.   0.82 0.4  0.73 0.9  0.08 0.47 1.   1.   0.42 1.
 1.   1.   0.06 1.   0.5  0.01 0.12 0.19 0.5  0.9  0.88 0.26 0.51 0.7
 0.97 0.21 0.11 0.49 1.   0.01 0.01 0.85 1.   1.   0.06 1.   0.23 1.
 0.01 0.05 0.21 1.   0.28 0.01 0.   0.6  0.46 0.93 0.28 0.06 0.73 0.69
 1.   0.01 0.01 0.97 0.51 0.05 0.69 1.   0.73 0.87 1.   0.1  0.85 0.95
 0.94 1.   0.41 0.02 0.3  0.47 0.   1.   0.   0.02 0.2  1.   0.59 0.
 0.53 0.11 0.99 0.13 0.56 0.02 0.75 0.62 0.83 1.   0.69 1.   0.49 0.78
 1.   0.04 0.1  0.08 0.5  0.88 0.79 0.43 0.36 1.   0.15 1.   0.04 1.
 0.57 0.01 1.   0.42 0.42 0.01 0.   1.   0.45 1.   0.44 0.11 0.   1.
 0.28 0.1  0.56 1.   0.03 0.04 1.   0.8  1.   0.41 0.93 1.   0.05 0.44
 0.32 0.8  0.02 0.09 0.11 0.54 0.05 1.   0.   0.81 0.01 1.   1.   0.35
 0.87 0.82 0.41 0.12 0.67 1.   0.51 0.14 0.73 1.   0.93 0.   1.   0.49
 0.04 0.41 0.14 1.   0.   0.82 1.   0.02 0.44 0.9  0.58 0.02 0.19 0.77
 0.47 0.02 1.   0.3  0.56 0.67 0.18 0.79 1.   0.1  0.29 0.13 0.26 0.22
 1.   0.09 0.29 0.36 0.41 0.36 0.03 0.66 0.44 0.45 0.82 0.05 1.   0.66
 1.   1.   0.2  0.64 0.63 0.   0.11 0.58 0.03 0.   0.36 0.44 0.61 0.8
 0.69 0.86 0.84 0.92 0.01 0.04 1.   0.49 1.   0.56 0.96 0.04 0.35 0.47
 0.04 0.04 0.13 0.45 1.   1.   0.44 0.04 0.1  0.83 1.   0.01 0.73 0.09
 1.   0.57 1.   0.46 1.   0.01 1.   0.69 1.   1.   0.03 0.01 0.29 0.78
 0.98 1.   0.8  0.1  0.88 0.72 0.04 0.7  1.   0.08 0.99 0.18 0.46 1.
 1.   0.   0.14 1.   1.   0.6  1.   0.53 1.   0.74 0.61 0.06 1.   0.98
 0.75 0.08 0.7  0.01 1.   0.05 0.09 0.26 1.   0.38 0.72 0.42 1.   0.03
 1.   1.   0.83 1.   0.06 0.27 0.83 0.63 0.67 1.   0.51 0.82 0.63 0.
 0.22 0.03 0.89 0.08 1.   1.   1.   0.14 0.18 0.14 0.39 0.08 1.   0.12
 0.07 1.   1.   0.38 0.44 0.55 0.58 0.04 0.37 1.   0.97 0.2  1.   1.
 1.   0.34 0.58 0.33 0.73 1.   0.04 0.05 1.   1.   0.38 0.08 0.14 0.
 0.87 0.27 1.   0.21 1.   0.85 0.04 0.03 0.72 0.69 1.   0.12 0.63 0.03
 0.02 0.69 0.03 0.65 0.08 0.02 1.   0.62 1.   0.72 0.12 0.2  0.79 0.52
 0.29 0.05 0.65 0.62 0.01 1.   0.74 0.16 0.98 0.56 0.47 0.85 0.87 0.05
 0.19 0.11 0.02 0.04 0.32 0.47 0.32 1.   0.98 1.   0.12 1.   0.2  1.
 0.24 0.27 1.   0.75 0.26 0.04 0.01 0.   0.27 1.   0.24 0.04 0.49 0.47
 0.04 0.33 0.63 0.14 0.51 0.91 0.09 1.   0.49 0.52 0.64 0.44 0.7  0.02
 1.   0.1  0.03 0.27 1.   1.   1.   0.59 1.   0.7  0.03 0.09 0.54 1.
 0.65 1.   0.12 0.   0.59 0.22 0.13 0.9  1.   0.74 1.   0.13 0.34 0.34
 0.73 0.08 0.54 0.8  0.71 0.46 1.   0.02 0.25 0.49]
2 [    2    45    51 ... 58072 31797 45666] [0.04 0.04 1.   ... 0.43 0.04 0.04]
3 [    3   101   154   396   529   687   982  1059  1692  1693  1733  1909
  2333  2383  2887  3072  3156  3414  3849  4005  4046  4210  4694  4736
  5125  5326  5883  5992  6507   264  6735  7534  7934  7974  7982  7995
  5154  8221  8367  8529  8589  8737  8806  8966  9364  9798 10253 10409
 10552 10602 10790 10983 11010 11024 11219 11245 11278 11327 11441 11963
  6352 12789  1039 12843 13329 13515 13604 13981 14061 14100 14210 14466
```

```
 14578 14883   719 15238 16071 13838  9147 16976  3364 17311 17358   896
  7480  7588 18234 18250 18264 18340 18353 18478 18519 19100 19178  6176
   445 19736 19776 19818 19820 19935 20135  7516 20322 20450 13393  5133
 20972 21024 21153 21170  5789  8707  8961  9115 21636 17624 10410 22598
 20555 22927 22952 23256 11255  8861 23577 17310 24566 12922 23370  2584
 25618 14165 14497 26046  9851 26501 12822 20152 26960 24893 27251 27276
 27363 23163 27684 27791 27875 21783  6771  6961 22139  4096 26566  4452
 24790 29019 29025 29107 18560 29386 29461 13944 29581 25538 29665  5808
 11514 29727 29869 29924  6519   766 30539 30800  1595 18209 31198 31297
 31485 31563 11913 31939 17075 32052 32444 32448 17776 32678 32765 32826
 15779  7795 33005  1917 10920 20951 33274 33469 11821 33967 34106 17631
 34519 34581 34598  4685 34649 16038 35057 35097 18995 14277 19237 35607
 31943 35785 36095  4265 20182  4622 10607 36499 13374 31098 29496  5656
 37046 23746 37374 37673 37713 36132 24605 15768 38306 38427 18590 35012
  2384 21082 38737  5793 38931  6111 19361 39277 12220 39553 22134 39691
 39987 40013 40144 40150 40153 13499 40239  2266 40399 40624 40632  6511
 41246 12794 20130 41412 41464  1295 41529 41532 41552 41606 41658  7781
 36531 31142 41964 42146 42211  5848 16642 42260 42300 42332 42521 42869
 43017 26759 43199 26835 43261 43322 31100 43496 10902 43878 19093 43911
 11699 43994 44015 19294 11889 44143 32269 15140 44464 44466  7382 44732
  7614 44905 29203 31193 45089 45139 16362 45322 45324 29553 29579  5743
 46183 10485 46623 36662  8108 46955 11563  8974 47196  3874 47566  1569
 48075 27080 34832 33049 16043 22982 23083 48480 48486 48759 26073 17200
 49065 49085 49099 49314 27003  2061 38532 49777 49919 45653  9607 10039
 36111 42992 18649 51246 51281 51291 51372 45449 33496 51471 51767 19631
 21935 51927 52036  1084 52289 52350 27318 52534 52623 52643 52648 45419
 52750  2854 52978 53197 52205 26853 36409 53828 53850 48254 23110 21074
 36946  2728 35443 52822 19581 54413 39439 54642 54645 54923 54945 55070
 33058  8082 43580 33229 11328 33601 54385 37501  9604 42748 38287 13294
 56306  5015 20929 56496 35115 56519 25396 33338 56630 54069 19224 33845
 21931 36240 49308 26791 49351 18227 51059 38525 57715 40267 57753 57783
 57799 36969 57945 21252 58004 33583 23829] [0.23 0.85 0.22 0.14 0.33 0.18 0.5
0.74 0.03 0.33 0.34 0.01 0.05 0.06
 0.03 0.01 1.   0.26 0.46 0.64 0.29 0.12 0.02 0.37 0.23 1.   0.14 0.19
 0.28 0.23 0.01 0.05 0.21 0.12 1.   0.09 0.01 0.3  0.14 0.02 0.94 0.27
 0.6  0.22 0.3  0.06 0.96 0.05 0.15 0.12 0.94 0.77 0.79 0.22 0.32 0.39
 0.12 0.46 0.   0.5  0.33 0.2  0.29 0.77 0.39 0.21 0.2  0.51 0.46 0.46
 0.25 0.41 0.32 0.04 0.4  1.   0.16 0.32 0.03 0.18 0.46 0.12 0.12 0.24
 0.37 0.18 0.08 0.02 0.11 0.29 0.01 0.22 0.01 0.14 0.5  0.   0.23 0.43
 0.34 0.04 0.22 0.45 0.35 0.11 0.17 0.26 0.08 0.11 0.16 0.07 0.07 0.21
 0.29 0.04 0.25 0.28 0.16 0.03 1.   0.02 0.11 1.   0.01 0.37 0.25 0.04
 0.16 0.89 1.   0.01 0.58 0.01 0.16 0.35 0.04 0.04 0.04 0.17 0.39 0.21
 0.17 0.07 0.04 0.03 0.44 0.81 0.29 0.01 0.49 0.2  0.14 0.22 0.08 0.27
 0.41 0.23 0.06 0.19 0.14 0.34 0.75 0.56 0.01 0.05 0.17 0.5  0.15 0.02
 0.2  0.35 0.31 0.21 0.35 0.23 0.19 0.33 0.02 0.6  0.3  0.35 0.   0.05
 0.16 0.12 0.12 0.31 0.13 0.12 0.21 0.2  0.25 0.1  0.12 0.01 0.02 0.06
 0.2  0.48 0.02 0.31 0.39 1.   1.   0.37 0.26 0.71 1.   0.64 0.25 0.33
 0.44 0.2  0.05 0.04 0.08 0.11 0.44 0.31 0.28 0.02 0.13 0.42 0.15 1.
 0.01 0.14 0.14 0.58 0.38 0.02 0.13 0.29 0.34 0.59 0.44 0.54 1.   0.24
 0.09 0.67 0.15 0.33 0.63 0.24 0.06 0.33 0.04 0.1  0.38 0.46 0.06 0.25
 0.13 0.42 0.21 0.32 0.35 0.27 0.29 0.12 0.22 0.4  1.   0.19 0.09 0.45
 0.34 0.27 0.2  0.25 0.   0.25 0.19 0.01 0.36 0.12 0.04 1.   0.21 0.21
 0.16 0.02 0.38 0.74 1.   1.   0.48 0.41 0.24 0.46 0.02 0.35 0.08 0.23
 0.23 0.09 0.4  0.28 0.13 0.32 0.59 0.26 0.04 1.   0.34 0.61 0.68 0.21
 0.24 0.45 0.68 0.39 0.32 1.   1.   0.13 1.   1.   0.62 1.   0.5  0.29
 0.48 0.81 0.33 0.02 0.14 0.02 0.26 0.35 0.24 0.01 0.4  0.15 0.03 0.22
 0.36 0.44 0.57 0.04 0.06 0.2  0.36 0.02 0.05 0.21 0.28 0.17 0.36 0.18
 0.29 0.12 0.   0.36 0.89 0.02 0.11 1.   0.29 0.26 0.01 0.49 0.08 0.71
 0.11 0.23 0.18 0.97 0.21 0.25 0.02 0.04 0.36 0.22 1.   0.35 0.02 0.03
 1.   0.12 0.31 0.25 0.02 0.2  0.47 0.2  1.   1.   0.65 0.4  0.02 0.23
 0.66 0.38 0.11 0.41 0.24 0.34 0.23 0.53 0.29 0.01 1.   0.24 0.3  0.14
```

```
 0.54 0.59 0.49 0.07 0.59 0.15 0.32 1.   0.42 0.24 1.   0.01 1.   0.06
 0.19 1.   0.03 0.25 0.38 0.67 0.1  0.21 0.19 0.23 0.03 0.52 0.08 1.
 0.15 0.27 0.22 0.2  0.32 1.   0.06 0.18 0.03 0.03 0.23 0.41 1.   0.02
 1.   0.69 0.43]
4 [    4   742  3749  6584  7292  9301  7948 16278 18835 21612 27372 23267
 32547  7575 34536 37748 18439 39951 41049 21244 45746  2083 18565 23386
 47871 51257 19147  7920 52827 39576 15336 30929 11394] [0.83 0.37 0.41 1.   0.1
7 1.   0.74 0.28 0.14 0.41 0.43 0.03 0.71 1.
 1.   1.   0.36 0.29 0.4  0.04 0.02 0.31 0.55 0.2  0.31 0.1  1.   1.
 1.   0.04 0.21 1.   0.21]
5 [    5  6437  6675  8790  9685 13124 14040 17498 29182 29584 30280 25783
 32808 36938 36705 18344 47303 10529 50815 40573 10799 19449  4909 12740
 43213] [0.21 0.05 0.69 0.44 0.82 0.01 1.   0.02 0.71 0.   0.06 0.04 0.01 0.25
 0.28 0.5  0.03 0.22 0.   0.02 0.76 0.12 0.04 0.21 0.01]
```

In [19]:

```
import tensorflow as tf
grouped_by_items = mapped_df.groupby("itemId")
with tf.python_io.TFRecordWriter("data/users_for_item") as ofp:
    for item, grouped in grouped_by_items:
        example = tf.train.Example(features = tf.train.Features(feature = {
            "key": tf.train.Feature(int64_list = tf.train.Int64List(value = [ite
            "indices": tf.train.Feature(int64_list = tf.train.Int64List(value =
            "values": tf.train.Feature(float_list = tf.train.FloatList(value = g
        }))
        ofp.write(example.SerializeToString())
```

```
WARNING:tensorflow:From /tmp/ipykernel_25848/1646577164.py:3: The name tf.python
_io.TFRecordWriter is deprecated. Please use tf.io.TFRecordWriter instead.
```

In [20]:

```
grouped_by_users = mapped_df.groupby("userId")
with tf.python_io.TFRecordWriter("data/items_for_user") as ofp:
    for user, grouped in grouped_by_users:
        example = tf.train.Example(features = tf.train.Features(feature = {
            "key": tf.train.Feature(int64_list = tf.train.Int64List(value = [use
            "indices": tf.train.Feature(int64_list = tf.train.Int64List(value =
            "values": tf.train.Feature(float_list = tf.train.FloatList(value = g
        }))
        ofp.write(example.SerializeToString())
```

In [21]:

```
!ls -lrt data
```

```
total 36552
-rw-r--r-- 1 jupyter jupyter 14121616 Nov  8 15:03 collab_raw.csv
-rw-r--r-- 1 jupyter jupyter  3525697 Nov  8 15:03 users.csv
-rw-r--r-- 1 jupyter jupyter    82217 Nov  8 15:03 items.csv
-rw-r--r-- 1 jupyter jupyter  7661290 Nov  8 15:03 collab_mapped.csv
-rw-r--r-- 1 jupyter jupyter  2296744 Nov  8 15:03 users_for_item
-rw-r--r-- 1 jupyter jupyter  9728585 Nov  8 15:03 items_for_user
```

To summarize, we created the following data files from collab_raw.csv:

1. ```collab_mapped.csv``` is essentially the same data as in ```collab_raw.csv``` except that ```visitorId``` and ```contentId``` which are business-specific have been mapped to ```userId``` and ```itemId``` which are enumerated in 0,1,2,.... The mappings themselves are stored in ```items.csv``` and ```users.csv``` so that they can be used during inference.

2. ```users_for_item``` contains all the users/ratings for each item in TFExample format
3. ```items_for_user``` contains all the items/ratings for each user in TFExample format

Train with WALS

Once you have the dataset, do matrix factorization with WALS using the WALSMatrixFactorization in the contrib directory. This is an estimator model, so it should be relatively familiar.

As usual, we write an input_fn to provide the data to the model, and then create the Estimator to do train_and_evaluate. Because it is in contrib and hasn't moved over to tf.estimator yet, we use tf.contrib.learn.Experiment to handle the training loop.

In [24]:

```
import os
import tensorflow as tf
from tensorflow.python.lib.io import file_io
from tensorflow.contrib.factorization import WALSMatrixFactorization

def read_dataset(mode, args):
    def decode_example(protos, vocab_size):
        features = {
            "key": tf.FixedLenFeature(shape = [1], dtype = tf.int64),
            "indices": tf.VarLenFeature(dtype = tf.int64),
            "values": tf.VarLenFeature(dtype = tf.float32)}
        parsed_features = tf.parse_single_example(serialized = protos, features
        values = tf.sparse_merge(sp_ids = parsed_features["indices"], sp_values
        # Save key to remap after batching
        # This is a temporary workaround to assign correct row numbers in each b
        # You can ignore details of this part and remap_keys().
        key = parsed_features["key"]
        decoded_sparse_tensor = tf.SparseTensor(indices = tf.concat(values = [va
                                                values = tf.concat(values = [val
                                                dense_shape = values.dense_shape
        return decoded_sparse_tensor


    def remap_keys(sparse_tensor):
        # Current indices of our SparseTensor that we need to fix
        bad_indices = sparse_tensor.indices # shape = (current_batch_size * (num
        # Current values of our SparseTensor that we need to fix
        bad_values = sparse_tensor.values # shape = (current_batch_size * (numbe

        # Since batch is ordered, the last value for a batch index is the user
        # Find where the batch index chages to extract the user rows
        # 1 where user, else 0
        user_mask = tf.concat(values = [bad_indices[1:,0] - bad_indices[:-1,0],

        # Mask out the user rows from the values
        good_values = tf.boolean_mask(tensor = bad_values, mask = tf.equal(x = u
        item_indices = tf.boolean_mask(tensor = bad_indices, mask = tf.equal(x =
        user_indices = tf.boolean_mask(tensor = bad_indices, mask = tf.equal(x =

        good_user_indices = tf.gather(params = user_indices, indices = item_indi

        # User and item indices are rank 1, need to make rank 1 to concat
        good_user_indices_expanded = tf.expand_dims(input = good_user_indices, a
```

```
        good_item_indices_expanded = tf.expand_dims(input = item_indices[:, 1],
        good_indices = tf.concat(values = [good_user_indices_expanded, good_item

        remapped_sparse_tensor = tf.SparseTensor(indices = good_indices, values
        return remapped_sparse_tensor


    def parse_tfrecords(filename, vocab_size):
        if mode == tf.estimator.ModeKeys.TRAIN:
            num_epochs = None # indefinitely
        else:
            num_epochs = 1 # end-of-input after this

        files = tf.gfile.Glob(filename = os.path.join(args["input_path"], filena

        # Create dataset from file list
        dataset = tf.data.TFRecordDataset(files)
        dataset = dataset.map(map_func = lambda x: decode_example(x, vocab_size)
        dataset = dataset.repeat(count = num_epochs)
        dataset = dataset.batch(batch_size = args["batch_size"])
        dataset = dataset.map(map_func = lambda x: remap_keys(x))
        return dataset.make_one_shot_iterator().get_next()

    def _input_fn():
        features = {
            WALSMatrixFactorization.INPUT_ROWS: parse_tfrecords("items_for_user"
            WALSMatrixFactorization.INPUT_COLS: parse_tfrecords("users_for_item"
            WALSMatrixFactorization.PROJECT_ROW: tf.constant(True)
        }
        return features, None

    return _input_fn
```

This code is helpful in developing the input function. You don't need it in production.

In [25]:

```
def try_out():
    with tf.Session() as sess:
        fn = read_dataset(
            mode = tf.estimator.ModeKeys.EVAL,
            args = {"input_path": "data", "batch_size": 4, "nitems": NITEMS, "nu
        feats, _ = fn()

        print(feats["input_rows"].eval())
        print(feats["input_rows"].eval())

try_out()
```

```
SparseTensorValue(indices=array([[   0,    0],
       [   0,   95],
       [   0, 5626],
       [   0, 5632],
       [   0, 5644],
       [   0, 5647],
       [   0, 5653],
       [   1,    1],
       [   1,  100],
       [   1,  144],
       [   1,  978],
       [   1, 5635],
```

```
       [   2,    2],
       [   2,   32],
       [   2,  893],
       [   2, 1614],
       [   2, 2649],
       [   2, 5630],
       [   2, 5632],
       [   3,    3],
       [   3,   58],
       [   3, 5664],
       [   3, 5669]]), values=array([0.25, 0.36, 0.04, 1.  , 0.07, 0.32, 0.03,
0.15, 0.06, 0.48, 0.13,
       0.83, 0.04, 0.03, 0.05, 0.02, 0.02, 0.02, 0.13, 0.23, 0.28, 0.13,
       0.76], dtype=float32), dense_shape=array([   4, 5670]))
SparseTensorValue(indices=array([[   4,    4],
       [   4,  136],
       [   4,  168],
       [   4,  469],
       [   4, 5620],
       [   4, 5629],
       [   4, 5650],
       [   4, 5656],
       [   5,    5],
       [   5,   91],
       [   5,  178],
       [   5, 5624],
       [   5, 5628],
       [   5, 5631],
       [   5, 5633],
       [   5, 5637],
       [   5, 5642],
       [   5, 5643],
       [   5, 5645],
       [   5, 5646],
       [   5, 5648],
       [   5, 5654],
       [   6,    6],
       [   6,   44],
       [   6,  155],
       [   6,  179],
       [   6,  423],
       [   6,  678],
       [   6, 1091],
       [   6, 1654],
       [   6, 2722],
       [   6, 5645],
       [   6, 5646],
       [   6, 5652],
       [   6, 5655],
       [   7,    7],
       [   7, 5634]]), values=array([0.83, 1.  , 0.82, 1.  , 0.27, 0.64, 0.28,
0.99, 0.21, 0.23, 0.33,
       0.23, 1.  , 0.41, 0.87, 0.92, 0.42, 0.24, 0.29, 1.  , 0.51, 0.69,
       1.  , 1.  , 0.6 , 0.32, 1.  , 1.  , 0.35, 0.25, 1.  , 0.2 , 1.  ,
       0.37, 0.71, 0.05, 0.  ], dtype=float32), dense_shape=array([   4, 5670]))
```

In [26]:

```
def find_top_k(user, item_factors, k):
    all_items = tf.matmul(a = tf.expand_dims(input = user, axis = 0), b = tf.tra
    topk = tf.nn.top_k(input = all_items, k = k)
    return tf.cast(x = topk.indices, dtype = tf.int64)
```

```
def batch_predict(args):
    import numpy as np
    with tf.Session() as sess:
        estimator = tf.contrib.factorization.WALSMatrixFactorization(
            num_rows = args["nusers"],
            num_cols = args["nitems"],
            embedding_dimension = args["n_embeds"],
            model_dir = args["output_dir"])

        # This is how you would get the row factors for out-of-vocab user data
        # row_factors = list(estimator.get_projections(input_fn=read_dataset(tf.
        # user_factors = tf.convert_to_tensor(np.array(row_factors))

        # But for in-vocab data, the row factors are already in the checkpoint
        user_factors = tf.convert_to_tensor(value = estimator.get_row_factors()[
        # In either case, we have to assume catalog doesn"t change, so col_facto
        item_factors = tf.convert_to_tensor(value = estimator.get_col_factors()[

        # For each user, find the top K items
        topk = tf.squeeze(input = tf.map_fn(fn = lambda user: find_top_k(user, i
        with file_io.FileIO(os.path.join(args["output_dir"], "batch_pred.txt"),
            for best_items_for_user in topk.eval():
                f.write(",".join(str(x) for x in best_items_for_user) + '\n')

def train_and_evaluate(args):
    train_steps = int(0.5 + (1.0 * args["num_epochs"] * args["nusers"]) / args["
    steps_in_epoch = int(0.5 + args["nusers"] / args["batch_size"])
    print("Will train for {} steps, evaluating once every {} steps".format(train
    def experiment_fn(output_dir):
        return tf.contrib.learn.Experiment(
            tf.contrib.factorization.WALSMatrixFactorization(
                num_rows = args["nusers"],
                num_cols = args["nitems"],
                embedding_dimension = args["n_embeds"],
                model_dir = args["output_dir"]),
            train_input_fn = read_dataset(tf.estimator.ModeKeys.TRAIN, args),
            eval_input_fn = read_dataset(tf.estimator.ModeKeys.EVAL, args),
            train_steps = train_steps,
            eval_steps = 1,
            min_eval_frequency = steps_in_epoch
        )

    from tensorflow.contrib.learn.python.learn import learn_runner
    learn_runner.run(experiment_fn = experiment_fn, output_dir = args["output_di

    batch_predict(args)
```

In [27]:

```
import shutil
shutil.rmtree(path = "wals_trained", ignore_errors=True)
train_and_evaluate({
    "output_dir": "wals_trained",
    "input_path": "data/",
    "num_epochs": 0.05,
    "nitems": NITEMS,
    "nusers": NUSERS,

    "batch_size": 512,
    "n_embeds": 10,
```

```
    "topk": 3
  })
```

```
Will train for 12 steps, evaluating once every 236 steps
WARNING:tensorflow:From /tmp/ipykernel_25848/1819520735.py:49: run (from tensorf
low.contrib.learn.python.learn.learn_runner) is deprecated and will be removed i
n a future version.
Instructions for updating:
Use tf.estimator.train_and_evaluate.
WARNING:tensorflow:
The TensorFlow contrib module will not be included in TensorFlow 2.0.
For more information, please see:
  * https://github.com/tensorflow/community/blob/master/rfcs/20180907-contrib-su
nset.md
  * https://github.com/tensorflow/addons
  * https://github.com/tensorflow/io (for I/O related ops)
If you depend on functionality not listed there, please file an issue.

WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/c
ontrib/learn/python/learn/estimators/estimator.py:1180: BaseEstimator.__init__
(from tensorflow.contrib.learn.python.learn.estimators.estimator) is deprecated
and will be removed in a future version.
Instructions for updating:
Please replace uses of any Estimator from tf.contrib.learn with an Estimator fro
m tf.estimator.*
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/c
ontrib/learn/python/learn/estimators/estimator.py:427: RunConfig.__init__ (from
tensorflow.contrib.learn.python.learn.estimators.run_config) is deprecated and w
ill be removed in a future version.
Instructions for updating:
When switching to tf.estimator.Estimator, use tf.estimator.RunConfig instead.
INFO:tensorflow:Using default config.
INFO:tensorflow:Using config: {'_task_type': None, '_task_id': 0, '_cluster_spe
c': <tensorflow.python.training.server_lib.ClusterSpec object at 0x7fef980e4a90
>, '_master': '', '_num_ps_replicas': 0, '_num_worker_replicas': 0, '_environmen
t': 'local', '_is_chief': True, '_evaluation_master': '', '_train_distribute': N
one, '_eval_distribute': None, '_experimental_max_worker_delay_secs': None, '_de
vice_fn': None, '_tf_config': gpu_options {
  per_process_gpu_memory_fraction: 1.0
}
, '_tf_random_seed': None, '_save_summary_steps': 100, '_save_checkpoints_secs':
600, '_log_step_count_steps': 100, '_protocol': None, '_session_config': None,
'_save_checkpoints_steps': None, '_keep_checkpoint_max': 5, '_keep_checkpoint_ev
ery_n_hours': 10000, '_model_dir': 'wals_trained', '_session_creation_timeout_se
cs': 7200}
WARNING:tensorflow:From /tmp/ipykernel_25848/1819520735.py:45: Experiment.__init
__ (from tensorflow.contrib.learn.python.learn.experiment) is deprecated and wil
l be removed in a future version.
Instructions for updating:
Please switch to tf.estimator.train_and_evaluate. You will also have to convert
to a tf.estimator.Estimator.
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/c
ontrib/learn/python/learn/monitors.py:279: BaseMonitor.__init__ (from tensorflo
w.contrib.learn.python.learn.monitors) is deprecated and will be removed after 2
016-12-05.
Instructions for updating:
Monitors are deprecated. Please use tf.train.SessionRunHook.
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/p
ython/training/training_util.py:236: Variable.initialized_value (from tensorflo
w.python.ops.variables) is deprecated and will be removed in a future version.
Instructions for updating:
```

```
Use Variable.read_value. Variables in 2.X are initialized automatically both in
eager and graph (inside tf.defun) contexts.
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/c
ontrib/factorization/python/ops/wals.py:315: ModelFnOps.__new__ (from tensorflo
w.contrib.learn.python.learn.estimators.model_fn) is deprecated and will be remo
ved in a future version.
Instructions for updating:
When switching to tf.estimator.Estimator, use tf.estimator.EstimatorSpec. You ca
n use the `estimator_spec` method to create an equivalent one.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Saving checkpoints for 0 into wals_trained/model.ckpt.
INFO:tensorflow:SweepHook running init op.
INFO:tensorflow:SweepHook running prep ops for the row sweep.
INFO:tensorflow:Next fit step starting.
INFO:tensorflow:loss = 188015.6, step = 1
INFO:tensorflow:Next fit step starting.
INFO:tensorflow:Next fit step starting.
INFO:tensorflow:Next fit step starting.
INFO:tensorflow:Next fit step starting.
INFO:tensorflow:Next fit step starting.
INFO:tensorflow:Next fit step starting.
INFO:tensorflow:Next fit step starting.
INFO:tensorflow:Next fit step starting.
INFO:tensorflow:Next fit step starting.
INFO:tensorflow:Next fit step starting.
INFO:tensorflow:Next fit step starting.
INFO:tensorflow:Saving checkpoints for 12 into wals_trained/model.ckpt.
INFO:tensorflow:Loss for final step: 171757.6.
INFO:tensorflow:Starting evaluation at 2021-11-08T15:05:26Z
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from wals_trained/model.ckpt-12
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Evaluation [1/1]
INFO:tensorflow:Finished evaluation at 2021-11-08-15:05:27
INFO:tensorflow:Saving dict for global step 12: global_step = 12, loss = 188015.
6
INFO:tensorflow:Using default config.
INFO:tensorflow:Using config: {'_task_type': None, '_task_id': 0, '_cluster_spe
c': <tensorflow.python.training.server_lib.ClusterSpec object at 0x7fef89b21fd0
>, '_master': '', '_num_ps_replicas': 0, '_num_worker_replicas': 0, '_environmen
t': 'local', '_is_chief': True, '_evaluation_master': '', '_train_distribute': N
one, '_eval_distribute': None, '_experimental_max_worker_delay_secs': None, '_de
vice_fn': None, '_tf_config': gpu_options {
  per_process_gpu_memory_fraction: 1.0
}
, '_tf_random_seed': None, '_save_summary_steps': 100, '_save_checkpoints_secs':
600, '_log_step_count_steps': 100, '_protocol': None, '_session_config': None,
'_save_checkpoints_steps': None, '_keep_checkpoint_max': 5, '_keep_checkpoint_ev
ery_n_hours': 10000, '_model_dir': 'wals_trained', '_session_creation_timeout_se
cs': 7200}
```

In [28]:

```
!ls wals_trained
```

```
batch_pred.txt
checkpoint
eval
```

```
events.out.tfevents.1636383923.tensorflow-1-15-20211108-104507
graph.pbtxt
model.ckpt-0.data-00000-of-00001
model.ckpt-0.index
model.ckpt-0.meta
model.ckpt-12.data-00000-of-00001
model.ckpt-12.index
model.ckpt-12.meta
```

In [29]:

```
!head wals_trained/batch_pred.txt
```

```
632,1458,3023
3785,4847,1061
2838,5179,3959
3378,1412,447
4065,1948,1882
1038,5581,5447
2764,2005,4180
574,5607,725
3023,3225,388
2246,3364,574
```

Run as a Python module

Let's run it as Python module for just a few steps.

In [30]:

```
os.environ["NITEMS"] = str(NITEMS)
os.environ["NUSERS"] = str(NUSERS)
```

In [31]:

```
%%bash
rm -rf wals.tar.gz wals_trained
gcloud ai-platform local train \
    --module-name=walsmodel.task \
    --package-path=${PWD}/walsmodel \
    -- \
    --output_dir=${PWD}/wals_trained \
    --input_path=${PWD}/data \
    --num_epochs=0.01 --nitems=${NITEMS} --nusers=${NUSERS} \
    --job-dir=./tmp
```

```
Will train for 2 steps, evaluating once every 236 steps
WARNING:tensorflow:From /home/jupyter/training-data-analyst/courses/machine_lear
ning/deepdive/10_recommend/walsmodel/model.py:27: The name tf.logging.set_verbos
ity is deprecated. Please use tf.compat.v1.logging.set_verbosity instead.

WARNING:tensorflow:From /home/jupyter/training-data-analyst/courses/machine_lear
ning/deepdive/10_recommend/walsmodel/model.py:27: The name tf.logging.INFO is de
precated. Please use tf.compat.v1.logging.INFO instead.

WARNING:tensorflow:From /home/jupyter/training-data-analyst/courses/machine_lear
ning/deepdive/10_recommend/walsmodel/model.py:163: run (from tensorflow.contrib.
learn.python.learn.learn_runner) is deprecated and will be removed in a future v
ersion.
Instructions for updating:
Use tf.estimator.train_and_evaluate.
WARNING:tensorflow:
```

```
The TensorFlow contrib module will not be included in TensorFlow 2.0.
For more information, please see:
  * https://github.com/tensorflow/community/blob/master/rfcs/20180907-contrib-su
nset.md
  * https://github.com/tensorflow/addons
  * https://github.com/tensorflow/io (for I/O related ops)
If you depend on functionality not listed there, please file an issue.

WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/c
ontrib/learn/python/learn/estimators/estimator.py:1180: BaseEstimator.__init__
(from tensorflow.contrib.learn.python.learn.estimators.estimator) is deprecated
and will be removed in a future version.
Instructions for updating:
Please replace uses of any Estimator from tf.contrib.learn with an Estimator fro
m tf.estimator.*
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/c
ontrib/learn/python/learn/estimators/estimator.py:427: RunConfig.__init__ (from
tensorflow.contrib.learn.python.learn.estimators.run_config) is deprecated and w
ill be removed in a future version.
Instructions for updating:
When switching to tf.estimator.Estimator, use tf.estimator.RunConfig instead.
INFO:tensorflow:Using default config.
INFO:tensorflow:Using config: {'_task_type': None, '_task_id': 0, '_cluster_spe
c': <tensorflow.python.training.server_lib.ClusterSpec object at 0x7f4d8fbcb910
>, '_master': '', '_num_ps_replicas': 0, '_num_worker_replicas': 0, '_environmen
t': 'cloud', '_is_chief': True, '_evaluation_master': '', '_train_distribute': N
one, '_eval_distribute': None, '_experimental_max_worker_delay_secs': None, '_de
vice_fn': None, '_tf_config': gpu_options {
  per_process_gpu_memory_fraction: 1.0
}
, '_tf_random_seed': None, '_save_summary_steps': 100, '_save_checkpoints_secs':
600, '_log_step_count_steps': 100, '_protocol': None, '_session_config': None,
'_save_checkpoints_steps': None, '_keep_checkpoint_max': 5, '_keep_checkpoint_ev
ery_n_hours': 10000, '_model_dir': '/home/jupyter/training-data-analyst/courses/
machine_learning/deepdive/10_recommend/wals_trained/', '_session_creation_timeou
t_secs': 7200}
WARNING:tensorflow:From /home/jupyter/training-data-analyst/courses/machine_lear
ning/deepdive/10_recommend/walsmodel/model.py:159: Experiment.__init__ (from ten
sorflow.contrib.learn.python.learn.experiment) is deprecated and will be removed
in a future version.
Instructions for updating:
Please switch to tf.estimator.train_and_evaluate. You will also have to convert
to a tf.estimator.Estimator.
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/c
ontrib/learn/python/learn/monitors.py:279: BaseMonitor.__init__ (from tensorflo
w.contrib.learn.python.learn.monitors) is deprecated and will be removed after 2
016-12-05.
Instructions for updating:
Monitors are deprecated. Please use tf.train.SessionRunHook.
WARNING:tensorflow:From /home/jupyter/training-data-analyst/courses/machine_lear
ning/deepdive/10_recommend/walsmodel/model.py:90: The name tf.gfile.Glob is depr
ecated. Please use tf.io.gfile.glob instead.

WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/p
ython/autograph/converters/directives.py:119: The name tf.FixedLenFeature is dep
recated. Please use tf.io.FixedLenFeature instead.

WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/p
ython/autograph/converters/directives.py:119: The name tf.VarLenFeature is depre
cated. Please use tf.io.VarLenFeature instead.
```

```
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/p
ython/autograph/converters/directives.py:119: The name tf.parse_single_example i
s deprecated. Please use tf.io.parse_single_example instead.

WARNING:tensorflow:From /home/jupyter/training-data-analyst/courses/machine_lear
ning/deepdive/10_recommend/walsmodel/model.py:46: sparse_merge (from tensorflow.
python.ops.sparse_ops) is deprecated and will be removed in a future version.
Instructions for updating:
No similar op available at this time.
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/p
ython/ops/array_ops.py:1475: where (from tensorflow.python.ops.array_ops) is dep
recated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /home/jupyter/training-data-analyst/courses/machine_lear
ning/deepdive/10_recommend/walsmodel/model.py:98: DatasetV1.make_one_shot_iterat
or (from tensorflow.python.data.ops.dataset_ops) is deprecated and will be remov
ed in a future version.
Instructions for updating:
Use `for ... in dataset:` to iterate over a dataset. If using `tf.estimator`, re
turn the `Dataset` object directly from your input function. As a last resort, y
ou can use `tf.compat.v1.data.make_one_shot_iterator(dataset)`.
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/p
ython/training/training_util.py:236: Variable.initialized_value (from tensorflo
w.python.ops.variables) is deprecated and will be removed in a future version.
Instructions for updating:
Use Variable.read_value. Variables in 2.X are initialized automatically both in
eager and graph (inside tf.defun) contexts.
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/c
ontrib/factorization/python/ops/wals.py:315: ModelFnOps.__new__ (from tensorflo
w.contrib.learn.python.learn.estimators.model_fn) is deprecated and will be remo
ved in a future version.
Instructions for updating:
When switching to tf.estimator.Estimator, use tf.estimator.EstimatorSpec. You ca
n use the `estimator_spec` method to create an equivalent one.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Graph was finalized.
2021-11-08 15:05:47.219643: I tensorflow/core/platform/profile_utils/cpu_utils.c
c:94] CPU Frequency: 2200160000 Hz
2021-11-08 15:05:47.220080: I tensorflow/compiler/xla/service/service.cc:168] XL
A service 0x55e8f74da580 initialized for platform Host (this does not guarantee
that XLA will be used). Devices:
2021-11-08 15:05:47.220145: I tensorflow/compiler/xla/service/service.cc:176]
StreamExecutor device (0): Host, Default Version
2021-11-08 15:05:47.220300: I tensorflow/core/common_runtime/process_util.cc:13
6] Creating new thread pool with default inter op setting: 2. Tune using inter_o
p_parallelism_threads for best performance.
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Saving checkpoints for 0 into /home/jupyter/training-data-analys
t/courses/machine_learning/deepdive/10_recommend/wals_trained/model.ckpt.
INFO:tensorflow:SweepHook running init op.
INFO:tensorflow:SweepHook running prep ops for the row sweep.
INFO:tensorflow:Next fit step starting.
INFO:tensorflow:loss = 187902.4, step = 1
INFO:tensorflow:Next fit step starting.
INFO:tensorflow:Saving checkpoints for 2 into /home/jupyter/training-data-analys
t/courses/machine_learning/deepdive/10_recommend/wals_trained/model.ckpt.
INFO:tensorflow:Loss for final step: 181397.06.
INFO:tensorflow:Starting evaluation at 2021-11-08T15:05:49Z
INFO:tensorflow:Graph was finalized.
```

```
INFO:tensorflow:Restoring parameters from /home/jupyter/training-data-analyst/co
urses/machine_learning/deepdive/10_recommend/wals_trained/model.ckpt-2
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Evaluation [1/1]
INFO:tensorflow:Finished evaluation at 2021-11-08-15:05:50
INFO:tensorflow:Saving dict for global step 2: global_step = 2, loss = 187902.4
WARNING:tensorflow:From /home/jupyter/training-data-analyst/courses/machine_lear
ning/deepdive/10_recommend/walsmodel/model.py:122: The name tf.Session is deprec
ated. Please use tf.compat.v1.Session instead.

INFO:tensorflow:Using default config.
INFO:tensorflow:Using config: {'_task_type': None, '_task_id': 0, '_cluster_spe
c': <tensorflow.python.training.server_lib.ClusterSpec object at 0x7f4dd8db5410
>, '_master': '', '_num_ps_replicas': 0, '_num_worker_replicas': 0, '_environmen
t': 'cloud', '_is_chief': True, '_evaluation_master': '', '_train_distribute': N
one, '_eval_distribute': None, '_experimental_max_worker_delay_secs': None, '_de
vice_fn': None, '_tf_config': gpu_options {
  per_process_gpu_memory_fraction: 1.0
}
, '_tf_random_seed': None, '_save_summary_steps': 100, '_save_checkpoints_secs':
600, '_log_step_count_steps': 100, '_protocol': None, '_session_config': None,
'_save_checkpoints_steps': None, '_keep_checkpoint_max': 5, '_keep_checkpoint_ev
ery_n_hours': 10000, '_model_dir': '/home/jupyter/training-data-analyst/courses/
machine_learning/deepdive/10_recommend/wals_trained/', '_session_creation_timeou
t_secs': 7200}
```

Run on Cloud

In [32]:

```
%%bash
gsutil -m cp data/* gs://${BUCKET}/wals/data
```

```
Copying file://data/collab_mapped.csv [Content-Type=text/csv]...
Copying file://data/collab_raw.csv [Content-Type=text/csv]...
Copying file://data/items.csv [Content-Type=text/csv]...
Copying file://data/users_for_item [Content-Type=application/octet-stream]...
Copying file://data/users.csv [Content-Type=text/csv]...
Copying file://data/items_for_user [Content-Type=application/octet-stream]...
- [6/6 files][ 35.7 MiB/ 35.7 MiB] 100% Done
Operation completed over 6 objects/35.7 MiB.
```

In [33]:

```
%%bash
OUTDIR=gs://${BUCKET}/wals/model_trained
JOBNAME=wals_$(date -u +%y%m%d_%H%M%S)
echo $OUTDIR $REGION $JOBNAME
gsutil -m rm -rf $OUTDIR
gcloud ai-platform jobs submit training $JOBNAME \
    --region=$REGION \
    --module-name=walsmodel.task \
    --package-path=${PWD}/walsmodel \
    --job-dir=$OUTDIR \
    --staging-bucket=gs://$BUCKET \
    --scale-tier=BASIC_GPU \
    --runtime-version=$TFVERSION \
    -- \
    --output_dir=$OUTDIR \
    --input_path=gs://${BUCKET}/wals/data \
    --num_epochs=10 --nitems=${NITEMS} --nusers=${NUSERS}
```

```
gs://qwiklabs-gcp-03-2787a45a1534/wals/model_trained us-central1 wals_211108_150
600
CommandException: 1 files/objects could not be removed.
ERROR: (gcloud.ai-platform.jobs.submit.training) FAILED_PRECONDITION: Field: job
_dir Error: The provided GCS path gs://qwiklabs-gcp-03-2787a45a1534/wals/model_t
rained cannot be written by service account service-243818477912@cloud-ml.googl
e.com.iam.gserviceaccount.com.
- '@type': type.googleapis.com/google.rpc.BadRequest
  fieldViolations:
  - description: The provided GCS path gs://qwiklabs-gcp-03-2787a45a1534/wals/mo
del_trained
      cannot be written by service account service-243818477912@cloud-ml.google.
com.iam.gserviceaccount.com.
    field: job_dir
```

```
CalledProcessError                        Traceback (most recent call last)
/tmp/ipykernel_25848/158902532.py in <module>
----> 1 get_ipython().run_cell_magic('bash', '', 'OUTDIR=gs://${BUCKET}/wals/mod
el_trained\nJOBNAME=wals_$(date -u +%y%m%d_%H%M%S)\necho $OUTDIR $REGION $JOBNAM
E\ngsutil -m rm -rf $OUTDIR\ngcloud ai-platform jobs submit training $JOBNAME
 \\\n    --region=$REGION \\\n    --module-name=walsmodel.task \\\n    --package
-path=${PWD}/walsmodel \\\n    --job-dir=$OUTDIR \\\n    --staging-bucket=gs://
$BUCKET \\\n    --scale-tier=BASIC_GPU \\\n    --runtime-version=$TFVERSION \\\n
-- \\\n    --output_dir=$OUTDIR \\\n    --input_path=gs://${BUCKET}/wals/data
 \\\n    --num_epochs=10 --nitems=${NITEMS} --nusers=${NUSERS} \n')

/opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.py in run_c
ell_magic(self, magic_name, line, cell)
   2404             with self.builtin_trap:
   2405                 args = (magic_arg_s, cell)
-> 2406                 result = fn(*args, **kwargs)
   2407             return result
   2408

/opt/conda/lib/python3.7/site-packages/IPython/core/magics/script.py in named_sc
ript_magic(line, cell)
    140             else:
    141                 line = script
--> 142             return self.shebang(line, cell)
    143
    144         # write a basic docstring:

/opt/conda/lib/python3.7/site-packages/decorator.py in fun(*args, **kw)
    230             if not kwsyntax:
    231                 args, kw = fix(args, kw, sig)
--> 232             return caller(func, *(extras + args), **kw)
    233     fun.__name__ = func.__name__
    234     fun.__doc__ = func.__doc__

/opt/conda/lib/python3.7/site-packages/IPython/core/magic.py in <lambda>(f, *a,
 **k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):

/opt/conda/lib/python3.7/site-packages/IPython/core/magics/script.py in shebang
(self, line, cell)
    243             sys.stderr.flush()
```

```
    244         if args.raise_error and p.returncode!=0:
--> 245             raise CalledProcessError(p.returncode, cell, output=out, std
err=err)
    246
    247     def _run_script(self, p, cell, to_close):

CalledProcessError: Command 'b'OUTDIR=gs://${BUCKET}/wals/model_trained\nJOBNAME
=wals_$(date -u +%y%m%d_%H%M%S)\necho $OUTDIR $REGION $JOBNAME\ngsutil -m rm -rf
$OUTDIR\ngcloud ai-platform jobs submit training $JOBNAME \\\n    --region=$REGI
ON \\\n    --module-name=walsmodel.task \\\n    --package-path=${PWD}/walsmodel
 \\\n    --job-dir=$OUTDIR \\\n    --staging-bucket=gs://$BUCKET \\\n    --scale
-tier=BASIC_GPU \\\n    --runtime-version=$TFVERSION \\\n    -- \\\n    --output
_dir=$OUTDIR \\\n    --input_path=gs://${BUCKET}/wals/data \\\n    --num_epochs=
10 --nitems=${NITEMS} --nusers=${NUSERS} \n'' returned non-zero exit status 1.
```

This took **10 minutes** for me.

Get row and column factors

Once you have a trained WALS model, you can get row and column factors (user and item embeddings) from the checkpoint file. We'll look at how to use these in the section on building a recommendation system using deep neural networks.

In [ ]:
```
def get_factors(args):
    with tf.Session() as sess:
        estimator = tf.contrib.factorization.WALSMatrixFactorization(
            num_rows = args["nusers"],
            num_cols = args["nitems"],
            embedding_dimension = args["n_embeds"],
            model_dir = args["output_dir"])

        row_factors = estimator.get_row_factors()[0]
        col_factors = estimator.get_col_factors()[0]
    return row_factors, col_factors
```

In [ ]:
```
args = {
    "output_dir": "gs://{}/wals/model_trained".format(BUCKET),
    "nitems": NITEMS,
    "nusers": NUSERS,
    "n_embeds": 10
  }

user_embeddings, item_embeddings = get_factors(args)
print(user_embeddings[:3])
print(item_embeddings[:3])
```

You can visualize the embedding vectors using dimensional reduction techniques such as PCA.

In [ ]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.decomposition import PCA

pca = PCA(n_components = 3)
pca.fit(user_embeddings)
user_embeddings_pca = pca.transform(user_embeddings)
```

```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(111, projection = "3d")
xs, ys, zs = user_embeddings_pca[::150].T
ax.scatter(xs, ys, zs)
```

```
# Copyright 2018 Google Inc. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing,
software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
# See the License for the specific language governing permissions
and
# limitations under the License.
```