jester

April 1, 2018

1 Creating 3 joke recommender systems on jester data and evaluating their results.

```
In [335]: import graphlab
```

2 Loading 1st set of data as jester1

```
In [336]: jester1 = graphlab.SFrame.read_csv('/Users/Sajjad/Downloads/jester1.csv', header = False)
```

```
Finished parsing file /Users/Sajjad/Downloads/jester1.csv


Parsing completed. Parsed 100 lines in 0.382521 secs.


------------------------------------------------------
Inferred types from first 100 line(s) of file as
column_type_hints=[int,float,float,float,float,float,float,float,float,float,float,float,float,float,flo
If parsing fails due to incorrect types, you can correct
the inferred type list above and pass it to read_csv in
the column_type_hints argument
------------------------------------------------------

Finished parsing file /Users/Sajjad/Downloads/jester1.csv


Parsing completed. Parsed 24983 lines in 0.48782 secs.
```

3 renaming first column as number

```
In [337]: jester1.rename({'X1':'number'})
```

```
Out[337]: Columns:
                  number        int
                  X2        float
                  X3        float
                  X4        float
                  X5        float
                  X6        float
                  X7        float
                  X8        float
```

```
X9        float
X10        float
X11        float
X12        float
X13        float
X14        float
X15        float
X16        float
X17        float
X18        float
X19        float
X20        float
X21        float
X22        float
X23        float
X24        float
X25        float
X26        float
X27        float
X28        float
X29        float
X30        float
X31        float
X32        float
X33        float
X34        float
X35        float
X36        float
X37        float
X38        float
X39        float
X40        float
X41        float
X42        float
X43        float
X44        float
X45        float
X46        float
X47        float
X48        float
X49        float
X50        float
X51        float
X52        float
X53        float
X54        float
X55        float
X56        float
X57        float
X58        float
X59        float
X60        float
X61        float
X62        float
```

```
        X63        float
        X64        float
        X65        float
        X66        float
        X67        float
        X68        float
        X69        float
        X70        float
        X71        float
        X72        float
        X73        float
        X74        float
        X75        float
        X76        float
        X77        float
        X78        float
        X79        float
        X80        float
        X81        float
        X82        float
        X83        float
        X84        float
        X85        float
        X86        float
        X87        float
        X88        float
        X89        float
        X90        float
        X91        float
        X92        float
        X93        float
        X94        float
        X95        float
        X96        float
        X97        float
        X98        float
        X99        float
        X100        float
        X101        float

Rows: 24983

Data:
```

number	X2	X3	X4	X5	X6	X7	X8	X9	X10
74	-7.82	8.79	-9.66	-8.16	-7.52	-8.5	-9.85	4.17	-8.98
100	4.08	-0.29	6.36	4.37	-2.38	-9.66	-0.73	-5.34	8.88
49	99.0	99.0	99.0	99.0	9.03	9.27	9.03	9.27	99.0
48	99.0	8.35	99.0	99.0	1.8	8.16	-2.82	6.21	99.0
91	8.5	4.61	-4.17	-5.39	1.36	1.6	7.04	4.61	-0.44
100	-6.17	-3.54	0.44	-8.5	-7.09	-4.32	-8.69	-0.87	-6.65
47	99.0	99.0	99.0	99.0	8.59	-9.85	7.72	8.79	99.0
100	6.84	3.16	9.17	-6.21	-8.16	-1.7	9.27	1.41	-5.19

```
|  100   | -3.79 | -3.54 | -9.42 | -6.89 | -8.74 | -0.29 | -5.29 | -8.93 | -7.86 |
|   72   |  3.01 |  5.15 |  5.15 |  3.01 |  6.41 |  5.15 |  8.93 |  2.52 |  3.01 |
+--------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-----+
|  X11  |  X12  |  X13  |  X14  |  X15  |  X16  |  X17  |  X18  |  X19  |  X20  | ... |
+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-----+
| -4.76 |  -8.5 | -6.75 | -7.18 |  8.45 | -7.18 | -7.52 | -7.43 | -9.81 | -9.85 | ... |
|  9.22 |  6.75 |  8.64 |  4.42 |  7.43 |  4.56 | -0.97 |  4.66 | -0.68 |  3.3  | ... |
|  99.0 |  7.33 |  7.57 |  9.37 |  6.17 | -6.36 | -6.89 | -7.86 |  9.03 |  9.03 | ... |
|  1.84 |  7.33 |  6.6  |  6.31 |  8.11 | -7.23 | -6.65 |  1.17 |  -6.6 | -3.64 | ... |
|  5.73 |  8.25 |  6.84 | -3.93 |  7.23 | -2.33 | -9.66 |  2.72 | -1.36 |  2.57 | ... |
|  -1.8 |  -6.8 | -5.73 |  -5.0 | -8.59 |  0.49 | -8.93 | -3.69 | -2.18 | -2.28 | ... |
|  99.0 |  4.27 |  7.62 | -6.26 |  2.96 |  6.07 |  -3.5 | -2.09 |  6.17 |  5.15 | ... |
| -4.42 |  8.2  | -7.86 | -6.94 | -7.96 |  0.29 |  -9.9 | -7.09 | -7.18 |  1.02 | ... |
|  -1.6 | -2.91 | -0.29 | -4.85 | -0.49 | -8.74 | -6.99 | -8.74 | -2.91 | -3.35 | ... |
|  8.16 |  5.53 |  6.02 |  4.47 |  5.44 | -4.66 | -0.97 | -0.44 |  1.55 |  0.49 | ... |
+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-----+
[24983 rows x 101 columns]
Note: Only the head of the SFrame is printed.
You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.
```

4 Loading second set of data as jester2

```
In [338]: jester2 = graphlab.SFrame.read_csv('/Users/Sajjad/Downloads/jester2.csv',header = False)

Finished parsing file /Users/Sajjad/Downloads/jester2.csv


Parsing completed. Parsed 100 lines in 0.253544 secs.


------------------------------------------------------
Inferred types from first 100 line(s) of file as
column_type_hints=[int,float,float,float,float,float,float,float,float,float,float,float,float,float,flo
If parsing fails due to incorrect types, you can correct
the inferred type list above and pass it to read_csv in
the column_type_hints argument
------------------------------------------------------

Finished parsing file /Users/Sajjad/Downloads/jester2.csv


Parsing completed. Parsed 23500 lines in 0.3849 secs.
```

5 Renaming first column of jester2 as number

```
In [369]: jester2.rename({'X1':'number'})

---------------------------------------------------------------------------

ValueError                                Traceback (most recent call last)
```

```
    <ipython-input-369-f697663ef940> in <module>()
      1
----> 2 jester2.rename({'X1':'number'})


    /Applications/anaconda/envs/gl-env/lib/python2.7/site-packages/graphlab/data_structures/sframe.p
   3964         for k in names:
   3965             if not k in all_columns:
-> 3966                 raise ValueError('Cannot find column %s in the SFrame' % k)
   3967         with cython_context():
   3968             for k in names:


    ValueError: Cannot find column X1 in the SFrame
```

6 Loading third set of jester data as jester3

```
In [340]: jester3 = graphlab.SFrame.read_csv('/Users/Sajjad/Downloads/jester3.csv',header = False)
```

```
Finished parsing file /Users/Sajjad/Downloads/jester3.csv


Parsing completed. Parsed 100 lines in 0.233591 secs.


------------------------------------------------------
Inferred types from first 100 line(s) of file as
column_type_hints=[int,float,float,int,int,float,int,float,float,int,float,float,int,float,float,float,f
If parsing fails due to incorrect types, you can correct
the inferred type list above and pass it to read_csv in
the column_type_hints argument
------------------------------------------------------

Finished parsing file /Users/Sajjad/Downloads/jester3.csv


Parsing completed. Parsed 24938 lines in 0.35633 secs.
```

7 Renaiming first column of jester3 as number

```
In [341]: jester3.rename({'X1':'number'})
```

```
Out[341]: Columns:
                  number        int
                  X2        float
                  X3        float
                  X4        int
                  X5        int
                  X6        float
                  X7        int
                  X8        float
                  X9        float
```

X10	int
X11	float
X12	float
X13	int
X14	float
X15	float
X16	float
X17	float
X18	float
X19	float
X20	float
X21	float
X22	float
X23	float
X24	float
X25	float
X26	float
X27	float
X28	float
X29	float
X30	float
X31	int
X32	float
X33	float
X34	int
X35	float
X36	float
X37	float
X38	float
X39	float
X40	float
X41	float
X42	float
X43	float
X44	float
X45	float
X46	float
X47	float
X48	float
X49	float
X50	float
X51	float
X52	float
X53	float
X54	float
X55	float
X56	float
X57	float
X58	float
X59	float
X60	float
X61	float
X62	float
X63	float

```
        X64        float
        X65        float
        X66        float
        X67        float
        X68        float
        X69        float
        X70        float
        X71        float
        X72        float
        X73        float
        X74        float
        X75        float
        X76        float
        X77        float
        X78        float
        X79        float
        X80        float
        X81        float
        X82        float
        X83        float
        X84        float
        X85        float
        X86        float
        X87        float
        X88        float
        X89        float
        X90        float
        X91        float
        X92        float
        X93        float
        X94        float
        X95        float
        X96        float
        X97        float
        X98        float
        X99        float
        X100        float
        X101        float

Rows: 24938

Data:
```

number	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11
26	99.0	99.0	99	99	-1.65	99	-0.78	6.89	99	99.0
33	99.0	99.0	99	99	-9.27	99	-9.17	-8.59	99	99.0
16	99.0	99.0	99	99	-6.12	99	-7.48	-7.77	99	99.0
24	99.0	0.05	99	99	-2.82	99	-4.85	-0.87	99	99.0
22	99.0	99.0	99	99	-4.95	99	6.21	2.72	99	-8.59
26	99.0	99.0	99	99	3.11	99	4.42	1.41	99	99.0
17	99.0	99.0	99	99	-0.05	99	-8.11	-7.38	99	99.0
26	99.0	99.0	99	99	6.8	99	-2.38	-7.82	99	99.0
33	8.2	8.35	99	99	3.06	99	-6.89	-9.76	99	99.0

```
|   25   | 99.0 | 99.0 | 99 | 99 |  3.06 | 99 |  0.15 |  8.98 |  99 |  99.0 |
+--------+------+------+----+----+-------+----+-------+-------+-----+-------+
+------+-----+-------+------+-------+-------+-------+-------+-------+-----+
| X12  | X13 |  X14  | X15  |  X16  |  X17  |  X18  |  X19  |  X20  | ... |
+------+-----+-------+------+-------+-------+-------+-------+-------+-----+
| 99.0 |  99 | -2.57 | 99.0 | -1.31 | -0.19 | -5.97 |  2.96 | -0.29 | ... |
| 99.0 |  99 | -8.59 | 99.0 | -8.59 | -2.67 | -8.59 |  -1.6 | -6.41 | ... |
| 99.0 |  99 | -6.89 | 99.0 | -6.12 | -6.12 | -1.99 | -6.12 | -7.82 | ... |
| 99.0 |  99 |  2.77 | 99.0 | -3.69 |  1.46 |  0.39 |  3.4  | -4.17 | ... |
| 99.0 |  99 |  6.07 | 99.0 |  6.89 | -7.67 | -3.93 | -5.63 | -7.23 | ... |
| 99.0 |  99 |  1.5  | 99.0 |  6.94 |  5.83 |  2.23 |  0.87 |  0.05 | ... |
| 99.0 |  99 |  0.29 | 99.0 | -7.33 | -7.96 | -6.89 | -6.84 | -6.17 | ... |
| 99.0 |  99 |  2.43 | 99.0 | -0.19 | -3.98 | -7.62 |  6.55 |  7.38 | ... |
| 99.0 |  99 | -9.56 | 99.0 | -9.42 | -9.37 | -9.22 |  6.94 | -8.54 | ... |
| 99.0 |  99 | -0.73 | 99.0 |  8.59 |  8.59 |  1.46 |  1.55 | -2.62 | ... |
+------+-----+-------+------+-------+-------+-------+-------+-------+-----+
[24938 rows x 101 columns]
Note: Only the head of the SFrame is printed.
You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.
```

8 renaiming column[1,101] as joke_#colum for all three datasets

```
In [342]: col_name = jester1.column_names()
          for i in range (1,101):
              new_col_name = 'joke_'+str(i)
              jester1.rename({col_name[i]:new_col_name})
              jester2.rename({col_name[i]:new_col_name})
              jester3.rename({col_name[i]:new_col_name})
```

9 appending jester1 and jester2 datasets as jester

```
In [343]: jester = jester1.append(jester2)
          jester.column_types

Out[343]: <bound method SFrame.column_types of Columns:
                  number        int
                  joke_1        float
                  joke_2        float
                  joke_3        float
                  joke_4        float
                  joke_5        float
                  joke_6        float
                  joke_7        float
                  joke_8        float
                  joke_9        float
                  joke_10        float
                  joke_11        float
                  joke_12        float
                  joke_13        float
                  joke_14        float
                  joke_15        float
                  joke_16        float
```

```
joke_17         float
joke_18         float
joke_19         float
joke_20         float
joke_21         float
joke_22         float
joke_23         float
joke_24         float
joke_25         float
joke_26         float
joke_27         float
joke_28         float
joke_29         float
joke_30         float
joke_31         float
joke_32         float
joke_33         float
joke_34         float
joke_35         float
joke_36         float
joke_37         float
joke_38         float
joke_39         float
joke_40         float
joke_41         float
joke_42         float
joke_43         float
joke_44         float
joke_45         float
joke_46         float
joke_47         float
joke_48         float
joke_49         float
joke_50         float
joke_51         float
joke_52         float
joke_53         float
joke_54         float
joke_55         float
joke_56         float
joke_57         float
joke_58         float
joke_59         float
joke_60         float
joke_61         float
joke_62         float
joke_63         float
joke_64         float
joke_65         float
joke_66         float
joke_67         float
joke_68         float
joke_69         float
joke_70         float
```

```
        joke_71        float
        joke_72        float
        joke_73        float
        joke_74        float
        joke_75        float
        joke_76        float
        joke_77        float
        joke_78        float
        joke_79        float
        joke_80        float
        joke_81        float
        joke_82        float
        joke_83        float
        joke_84        float
        joke_85        float
        joke_86        float
        joke_87        float
        joke_88        float
        joke_89        float
        joke_90        float
        joke_91        float
        joke_92        float
        joke_93        float
        joke_94        float
        joke_95        float
        joke_96        float
        joke_97        float
        joke_98        float
        joke_99        float
        joke_100        float

Rows: 48483

Data:
+--------+--------+--------+--------+--------+--------+--------+--------+--------+
| number | joke_1 | joke_2 | joke_3 | joke_4 | joke_5 | joke_6 | joke_7 | joke_8 |
+--------+--------+--------+--------+--------+--------+--------+--------+--------+
|   74   | -7.82  |  8.79  | -9.66  | -8.16  | -7.52  |  -8.5  | -9.85  |  4.17  |
|  100   |  4.08  | -0.29  |  6.36  |  4.37  | -2.38  | -9.66  | -0.73  | -5.34  |
|   49   |  99.0  |  99.0  |  99.0  |  99.0  |  9.03  |  9.27  |  9.03  |  9.27  |
|   48   |  99.0  |  8.35  |  99.0  |  99.0  |  1.8   |  8.16  | -2.82  |  6.21  |
|   91   |  8.5   |  4.61  | -4.17  | -5.39  |  1.36  |  1.6   |  7.04  |  4.61  |
|  100   | -6.17  | -3.54  |  0.44  |  -8.5  | -7.09  | -4.32  | -8.69  | -0.87  |
|   47   |  99.0  |  99.0  |  99.0  |  99.0  |  8.59  | -9.85  |  7.72  |  8.79  |
|  100   |  6.84  |  3.16  |  9.17  | -6.21  | -8.16  |  -1.7  |  9.27  |  1.41  |
|  100   | -3.79  | -3.54  | -9.42  | -6.89  | -8.74  | -0.29  | -5.29  | -8.93  |
|   72   |  3.01  |  5.15  |  5.15  |  3.01  |  6.41  |  5.15  |  8.93  |  2.52  |
+--------+--------+--------+--------+--------+--------+--------+--------+--------+
+--------+---------+---------+---------+---------+---------+---------+---------+
| joke_9 | joke_10 | joke_11 | joke_12 | joke_13 | joke_14 | joke_15 | joke_16 |
+--------+---------+---------+---------+---------+---------+---------+---------+
| -8.98  |  -4.76  |   -8.5  |  -6.75  |  -7.18  |   8.45  |  -7.18  |  -7.52  |
|  8.88  |   9.22  |   6.75  |   8.64  |   4.42  |   7.43  |   4.56  |  -0.97  |
|  99.0  |   99.0  |   7.33  |   7.57  |   9.37  |   6.17  |  -6.36  |  -6.89  |
```

```
|  99.0  |   1.84  |   7.33  |   6.6   |   6.31  |   8.11  |  -7.23  |  -6.65  |
| -0.44  |   5.73  |   8.25  |   6.84  |  -3.93  |   7.23  |  -2.33  |  -9.66  |
| -6.65  |   -1.8  |   -6.8  |  -5.73  |   -5.0  |  -8.59  |   0.49  |  -8.93  |
|  99.0  |   99.0  |   4.27  |   7.62  |  -6.26  |   2.96  |   6.07  |   -3.5  |
| -5.19  |  -4.42  |   8.2   |  -7.86  |  -6.94  |  -7.96  |   0.29  |   -9.9  |
| -7.86  |   -1.6  |  -2.91  |  -0.29  |  -4.85  |  -0.49  |  -8.74  |  -6.99  |
|  3.01  |   8.16  |   5.53  |   6.02  |   4.47  |   5.44  |  -4.66  |  -0.97  |
+--------+---------+---------+---------+---------+---------+---------+---------+
+---------+---------+---------+-----+
| joke_17 | joke_18 | joke_19 | ... |
+---------+---------+---------+-----+
|  -7.43  |  -9.81  |  -9.85  | ... |
|   4.66  |  -0.68  |   3.3   | ... |
|  -7.86  |   9.03  |   9.03  | ... |
|   1.17  |   -6.6  |  -3.64  | ... |
|   2.72  |  -1.36  |   2.57  | ... |
|  -3.69  |  -2.18  |  -2.28  | ... |
|  -2.09  |   6.17  |   5.15  | ... |
|  -7.09  |  -7.18  |   1.02  | ... |
|  -8.74  |  -2.91  |  -3.35  | ... |
|  -0.44  |   1.55  |   0.49  | ... |
+---------+---------+---------+-----+
[48483 rows x 101 columns]
Note: Only the head of the SFrame is printed.
You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.>
```

In [344]:
```
jester3.column_types
```

Out[344]:
```
<bound method SFrame.column_types of Columns:
        number          int
        joke_1          float
        joke_2          float
        joke_3          int
        joke_4          int
        joke_5          float
        joke_6          int
        joke_7          float
        joke_8          float
        joke_9          int
        joke_10          float
        joke_11          float
        joke_12          int
        joke_13          float
        joke_14          float
        joke_15          float
        joke_16          float
        joke_17          float
        joke_18          float
        joke_19          float
        joke_20          float
        joke_21          float
        joke_22          float
        joke_23          float
        joke_24          float
        joke_25          float
```

```
joke_26         float
joke_27         float
joke_28         float
joke_29         float
joke_30         int
joke_31         float
joke_32         float
joke_33         int
joke_34         float
joke_35         float
joke_36         float
joke_37         float
joke_38         float
joke_39         float
joke_40         float
joke_41         float
joke_42         float
joke_43         float
joke_44         float
joke_45         float
joke_46         float
joke_47         float
joke_48         float
joke_49         float
joke_50         float
joke_51         float
joke_52         float
joke_53         float
joke_54         float
joke_55         float
joke_56         float
joke_57         float
joke_58         float
joke_59         float
joke_60         float
joke_61         float
joke_62         float
joke_63         float
joke_64         float
joke_65         float
joke_66         float
joke_67         float
joke_68         float
joke_69         float
joke_70         float
joke_71         float
joke_72         float
joke_73         float
joke_74         float
joke_75         float
joke_76         float
joke_77         float
joke_78         float
joke_79         float
```

```
        joke_80        float
        joke_81        float
        joke_82        float
        joke_83        float
        joke_84        float
        joke_85        float
        joke_86        float
        joke_87        float
        joke_88        float
        joke_89        float
        joke_90        float
        joke_91        float
        joke_92        float
        joke_93        float
        joke_94        float
        joke_95        float
        joke_96        float
        joke_97        float
        joke_98        float
        joke_99        float
        joke_100        float

Rows: 24938

Data:
```

number	joke_1	joke_2	joke_3	joke_4	joke_5	joke_6	joke_7	joke_8
26	99.0	99.0	99	99	-1.65	99	-0.78	6.89
33	99.0	99.0	99	99	-9.27	99	-9.17	-8.59
16	99.0	99.0	99	99	-6.12	99	-7.48	-7.77
24	99.0	0.05	99	99	-2.82	99	-4.85	-0.87
22	99.0	99.0	99	99	-4.95	99	6.21	2.72
26	99.0	99.0	99	99	3.11	99	4.42	1.41
17	99.0	99.0	99	99	-0.05	99	-8.11	-7.38
26	99.0	99.0	99	99	6.8	99	-2.38	-7.82
33	8.2	8.35	99	99	3.06	99	-6.89	-9.76
25	99.0	99.0	99	99	3.06	99	0.15	8.98

joke_9	joke_10	joke_11	joke_12	joke_13	joke_14	joke_15	joke_16
99	99.0	99.0	99	-2.57	99.0	-1.31	-0.19
99	99.0	99.0	99	-8.59	99.0	-8.59	-2.67
99	99.0	99.0	99	-6.89	99.0	-6.12	-6.12
99	99.0	99.0	99	2.77	99.0	-3.69	1.46
99	-8.59	99.0	99	6.07	99.0	6.89	-7.67
99	99.0	99.0	99	1.5	99.0	6.94	5.83
99	99.0	99.0	99	0.29	99.0	-7.33	-7.96
99	99.0	99.0	99	2.43	99.0	-0.19	-3.98
99	99.0	99.0	99	-9.56	99.0	-9.42	-9.37
99	99.0	99.0	99	-0.73	99.0	8.59	8.59

```
| joke_17 | joke_18 | joke_19 | ... |
+---------+---------+---------+-----+
|  -5.97  |   2.96  |  -0.29  | ... |
|  -8.59  |   -1.6  |  -6.41  | ... |
|  -1.99  |  -6.12  |  -7.82  | ... |
|   0.39  |   3.4   |  -4.17  | ... |
|  -3.93  |  -5.63  |  -7.23  | ... |
|   2.23  |   0.87  |   0.05  | ... |
|  -6.89  |  -6.84  |  -6.17  | ... |
|  -7.62  |   6.55  |   7.38  | ... |
|  -9.22  |   6.94  |  -8.54  | ... |
|   1.46  |   1.55  |  -2.62  | ... |
+---------+---------+---------+-----+
[24938 rows x 101 columns]
Note: Only the head of the SFrame is printed.
You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.>
```

10 changing jester3 columns as data type float for column[1,101]

In [345]:
```
clm = jester3.column_names()
for i in range (1,len(clm)):
    jester3[clm[i]] = jester3[clm[i]].astype (float)
```

11 appending jester and jester3 in one SFrame as jester_data

In [346]:
```
jester_data = jester.append(jester3)
```

In [ ]:

12 extracting all data for a new dataframe with 3 columns. Those columns are

13 user_id, joke_id, review

In [370]:
```
user_id = []
joke_id = []
review = []
for i in range(0,len(jester_data)):
    for j in range(0,len(clm_name)):
        if (jester_data[i][clm_name[j]]!= 99.0):
            user_id.append('user_'+str(i))
            joke_id.append('joke_'+str(j))
            review.append(jester_data[i][clm_name[j]])
```

14 creating new dataframe with 3 columns

In [384]:
```
df_11 = graphlab.SFrame({'user_id':user_id,'joke_id':joke_id,'review':review})
```

deal with missing data

In [385]:
```
df_11
```

Out[385]: Columns:
joke_id str
review float
user_id str

Rows: 4136360

Data:

joke_id	review	user_id
joke_0	-7.82	user_0
joke_1	8.79	user_0
joke_2	-9.66	user_0
joke_3	-8.16	user_0
joke_4	-7.52	user_0
joke_5	-8.5	user_0
joke_6	-9.85	user_0
joke_7	4.17	user_0
joke_8	-8.98	user_0
joke_9	-4.76	user_0

[4136360 rows x 3 columns]
Note: Only the head of the SFrame is printed.
You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.

In [ ]:

15 splitting data into train and test data

In [372]:
```
train_data,test_data = df_11.random_split(0.8,seed = 1)
```

Build popularity model

16 Building popularity based recommend model

In [386]:
```
popularity_model = graphlab.popularity_recommender.create(train_data,
                                                          user_id = 'user_id',
                                                          item_id = 'joke_id',
                                                          target = 'review')
```

```
Recsys training: model = popularity


Preparing data set.


    Data has 3309475 observations with 73421 users and 100 items.


    Data prepared in: 4.38888s


3309475 observations to process; with 100 unique items.
```

Creating first recommender system

```
In [374]: users = df_11['user_id'].unique()

In [375]: users[1]

Out[375]: 'user_34332'
```

17 recommending for user[2]

```
In [376]: popularity_model.recommend(users= [users[2]])

Out[376]: Columns:
                user_id        str
                joke_id        str
                score         float
                rank          int

          Rows: 10

          Data:
          +------------+---------+---------------+------+
          |  user_id   | joke_id |     score     | rank |
          +------------+---------+---------------+------+
          | user_70080 | joke_88 | 3.34239015175 |  1   |
          | user_70080 | joke_31 | 2.97628068454 |  2   |
          | user_70080 | joke_26 | 2.92080915354 |  3   |
          | user_70080 | joke_71 | 2.63586814492 |  4   |
          | user_70080 | joke_48 | 2.49938281524 |  5   |
          | user_70080 | joke_65 | 2.44040476828 |  6   |
          | user_70080 | joke_75 |  2.3584540408 |  7   |
          | user_70080 | joke_68 | 2.28665911971 |  8   |
          | user_70080 | joke_92 | 2.24987329207 |  9   |
          | user_70080 | joke_60 | 2.10284269083 |  10  |
          +------------+---------+---------------+------+
          [10 rows x 4 columns]
```

Personalized model

18 building personalized model

```
In [387]: personalized_model = graphlab.item_similarity_recommender.create(train_data,
                                                                           user_id = 'user_id',
                                                                           item_id = 'joke_id',
                                                                           target = 'review')

Recsys training: model = item_similarity


Preparing data set.


    Data has 3309475 observations with 73421 users and 100 items.
```

```
    Data prepared in: 4.00863s
Training model from provided data.
Gathering per-item and per-user statistics.
+--------------------------------+------------+
| Elapsed Time (Item Statistics) | % Complete |
+--------------------------------+------------+
| 5.028ms                        | 1.25       |
| 210.641ms                      | 100        |
+--------------------------------+------------+
Setting up lookup tables.
Processing data in one pass using dense lookup tables.
+-------------------------------------+------------------+-----------------+
| Elapsed Time (Constructing Lookups) | Total % Complete | Items Processed |
+-------------------------------------+------------------+-----------------+
| 213.078ms                           | 0                | 0               |
| 779.706ms                           | 100              | 100             |
+-------------------------------------+------------------+-----------------+
Finalizing lookup tables.
Generating candidate set for working with new users.
Finished training in 1.09941s
```

```
In [388]: popularity_model.evaluate(test_data)
```

```
recommendations finished on 1000/73227 queries. users per second: 3308.63

recommendations finished on 2000/73227 queries. users per second: 3062.97

recommendations finished on 3000/73227 queries. users per second: 3046.07

recommendations finished on 4000/73227 queries. users per second: 2856.92

recommendations finished on 5000/73227 queries. users per second: 2669.43

recommendations finished on 6000/73227 queries. users per second: 2608.12

recommendations finished on 7000/73227 queries. users per second: 2434.96

recommendations finished on 8000/73227 queries. users per second: 2454.42

recommendations finished on 9000/73227 queries. users per second: 2495.87

recommendations finished on 10000/73227 queries. users per second: 2503.72

recommendations finished on 11000/73227 queries. users per second: 2534.26

recommendations finished on 12000/73227 queries. users per second: 2577.14

recommendations finished on 13000/73227 queries. users per second: 2618.46

recommendations finished on 14000/73227 queries. users per second: 2650.58

recommendations finished on 15000/73227 queries. users per second: 2678.36

recommendations finished on 16000/73227 queries. users per second: 2708.18

recommendations finished on 17000/73227 queries. users per second: 2732.78

recommendations finished on 18000/73227 queries. users per second: 2754.12

recommendations finished on 19000/73227 queries. users per second: 2768.94
```

```
recommendations finished on 20000/73227 queries. users per second: 2790.59

recommendations finished on 21000/73227 queries. users per second: 2808.22

recommendations finished on 22000/73227 queries. users per second: 2819.67

recommendations finished on 23000/73227 queries. users per second: 2796.85

recommendations finished on 24000/73227 queries. users per second: 2786.14

recommendations finished on 25000/73227 queries. users per second: 2790.17

recommendations finished on 26000/73227 queries. users per second: 2797.46

recommendations finished on 27000/73227 queries. users per second: 2767.33

recommendations finished on 28000/73227 queries. users per second: 2763.02

recommendations finished on 29000/73227 queries. users per second: 2742.32

recommendations finished on 30000/73227 queries. users per second: 2739.2

recommendations finished on 31000/73227 queries. users per second: 2741.12

recommendations finished on 32000/73227 queries. users per second: 2741.93

recommendations finished on 33000/73227 queries. users per second: 2705.02

recommendations finished on 34000/73227 queries. users per second: 2685.25

recommendations finished on 35000/73227 queries. users per second: 2682.81

recommendations finished on 36000/73227 queries. users per second: 2686.87

recommendations finished on 37000/73227 queries. users per second: 2687.43

recommendations finished on 38000/73227 queries. users per second: 2690.29

recommendations finished on 39000/73227 queries. users per second: 2678.78
```

```
recommendations finished on 40000/73227 queries. users per second: 2688.28
recommendations finished on 41000/73227 queries. users per second: 2677.37
recommendations finished on 42000/73227 queries. users per second: 2688.73
recommendations finished on 43000/73227 queries. users per second: 2696.21
recommendations finished on 44000/73227 queries. users per second: 2698.92
recommendations finished on 45000/73227 queries. users per second: 2698.77
recommendations finished on 46000/73227 queries. users per second: 2703.15
recommendations finished on 47000/73227 queries. users per second: 2711.73
recommendations finished on 48000/73227 queries. users per second: 2716.21
recommendations finished on 49000/73227 queries. users per second: 2723.99
recommendations finished on 50000/73227 queries. users per second: 2726.94
recommendations finished on 51000/73227 queries. users per second: 2717.13
recommendations finished on 52000/73227 queries. users per second: 2714.31
recommendations finished on 53000/73227 queries. users per second: 2717.87
recommendations finished on 54000/73227 queries. users per second: 2715.23
recommendations finished on 55000/73227 queries. users per second: 2704.61
recommendations finished on 56000/73227 queries. users per second: 2695.7
recommendations finished on 57000/73227 queries. users per second: 2691.26
recommendations finished on 58000/73227 queries. users per second: 2695.24
recommendations finished on 59000/73227 queries. users per second: 2698.47
```

```
recommendations finished on 60000/73227 queries. users per second: 2701.65

recommendations finished on 61000/73227 queries. users per second: 2708.62

recommendations finished on 62000/73227 queries. users per second: 2710.28

recommendations finished on 63000/73227 queries. users per second: 2711.82

recommendations finished on 64000/73227 queries. users per second: 2711.29

recommendations finished on 65000/73227 queries. users per second: 2716.49

recommendations finished on 66000/73227 queries. users per second: 2718.78

recommendations finished on 67000/73227 queries. users per second: 2719.26

recommendations finished on 68000/73227 queries. users per second: 2719.08

recommendations finished on 69000/73227 queries. users per second: 2719.13

recommendations finished on 70000/73227 queries. users per second: 2728.67

recommendations finished on 71000/73227 queries. users per second: 2733.27

recommendations finished on 72000/73227 queries. users per second: 2737.82

recommendations finished on 73000/73227 queries. users per second: 2735.72

Precision and recall summary statistics by cutoff
+--------+----------------+-----------------+
| cutoff | mean_precision |   mean_recall   |
+--------+----------------+-----------------+
|   1    | 0.392628402092 | 0.0350818895863 |
|   2    | 0.481755363459 | 0.0920615370151 |
|   3    | 0.507995684652 |  0.146108926035 |
|   4    | 0.511211711527 |  0.193897603679 |
|   5    | 0.504081827741 |  0.235101455663 |
|   6    | 0.490681943363 |  0.270010116192 |
|   7    | 0.474776087665 |  0.299527635016 |
|   8    | 0.458710584894 |  0.325276331194 |
|   9    | 0.443673629794 |  0.348655914674 |
|   10   | 0.430483291682 |  0.370468672755 |
+--------+----------------+-----------------+
```

```
[10 rows x 3 columns]

('\nOverall RMSE: ', 5.046616644117791)

Per User RMSE (best)
+------------+-------+-----------------+
|  user_id   | count |       rmse      |
+------------+-------+-----------------+
| user_70357 |   1   | 0.0098379060879 |
+------------+-------+-----------------+
[1 rows x 3 columns]


Per User RMSE (worst)
+------------+-------+---------------+
|  user_id   | count |      rmse     |
+------------+-------+---------------+
| user_61500 |   2   | 12.4015141955 |
+------------+-------+---------------+
[1 rows x 3 columns]


Per Item RMSE (best)
+---------+-------+---------------+
| joke_id | count |      rmse     |
+---------+-------+---------------+
| joke_49 | 13980 | 4.43347347107 |
+---------+-------+---------------+
[1 rows x 3 columns]


Per Item RMSE (worst)
+---------+-------+---------------+
| joke_id | count |      rmse     |
+---------+-------+---------------+
| joke_70 |  3752 | 5.73695367039 |
+---------+-------+---------------+
[1 rows x 3 columns]

Out[388]: {'precision_recall_by_user': Columns:
                  user_id        str
                  cutoff        int
                  precision        float
                  recall        float
                  count        int

           Rows: 1318086

           Data:
           +---------+--------+----------------+-----------------+-------+
           | user_id | cutoff |   precision    |      recall     | count |
           +---------+--------+----------------+-----------------+-------+
           |  user_0 |   1    |      0.0       |       0.0       |   12  |
           |  user_0 |   2    |      0.5       | 0.0833333333333 |   12  |
```

```
|  user_0 |   3    | 0.333333333333 | 0.0833333333333 |   12  |
|  user_0 |   4    |      0.5       |  0.166666666667 |   12  |
|  user_0 |   5    |      0.4       |  0.166666666667 |   12  |
|  user_0 |   6    | 0.333333333333 |  0.166666666667 |   12  |
|  user_0 |   7    | 0.428571428571 |       0.25      |   12  |
|  user_0 |   8    |     0.375      |       0.25      |   12  |
|  user_0 |   9    | 0.333333333333 |       0.25      |   12  |
|  user_0 |   10   |      0.3       |       0.25      |   12  |
+---------+--------+----------------+-----------------+-------+
[1318086 rows x 5 columns]
Note: Only the head of the SFrame is printed.
You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.,
'precision_recall_overall': Columns:
        cutoff        int
        precision        float
        recall        float

Rows: 18

Data:
+--------+----------------+-----------------+
| cutoff |   precision    |      recall     |
+--------+----------------+-----------------+
|   1    | 0.392628402092 | 0.0350818895863 |
|   2    | 0.481755363459 | 0.0920615370151 |
|   3    | 0.507995684652 |  0.146108926035 |
|   4    | 0.511211711527 |  0.193897603679 |
|   5    | 0.504081827741 |  0.235101455663 |
|   6    | 0.490681943363 |  0.270010116192 |
|   7    | 0.474776087665 |  0.299527635016 |
|   8    | 0.458710584894 |  0.325276331194 |
|   9    | 0.443673629794 |  0.348655914674 |
|   10   | 0.430483291682 |  0.370468672755 |
+--------+----------------+-----------------+
[18 rows x 3 columns]
Note: Only the head of the SFrame is printed.
You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.,
'rmse_by_item': Columns:
        joke_id        str
        count        int
        rmse        float

Rows: 100

Data:
+---------+-------+---------------+
| joke_id | count |      rmse     |
+---------+-------+---------------+
| joke_78 |  3915 |  5.2706092644 |
| joke_57 |  6432 | 5.06888091275 |
| joke_99 |  3879 | 5.30913045448 |
| joke_32 |  6321 | 5.27330861522 |
| joke_16 | 14847 | 4.44703714777 |
| joke_94 |  4456 | 5.27025442705 |
```

```
| joke_71 |  3815 | 4.65792146235 |
| joke_56 |  6533 |  5.435632758  |
| joke_65 | 11911 | 4.77237117336 |
| joke_93 |  4404 |  5.1595079663 |
+---------+-------+---------------+
[100 rows x 3 columns]
Note: Only the head of the SFrame is printed.
You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.,
'rmse_by_user': Columns:
        user_id        str
        count         int
        rmse         float

Rows: 73227

Data:
+------------+-------+---------------+
|  user_id   | count |      rmse     |
+------------+-------+---------------+
| user_32879 |   14  | 5.21729715327 |
| user_34332 |   15  | 5.14576468901 |
| user_70080 |   5   | 3.75966012684 |
| user_1475  |   14  | 5.22835560505 |
| user_41387 |   17  | 2.41622834582 |
| user_63187 |   4   | 5.98411085532 |
|  user_960  |   7   | 4.14671027835 |
| user_27414 |   9   | 3.71588111606 |
| user_52473 |   9   | 3.47944717945 |
| user_32483 |   7   | 6.10349552005 |
+------------+-------+---------------+
[73227 rows x 3 columns]
Note: Only the head of the SFrame is printed.
You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.,
'rmse_overall': 5.046616644117791}
```

19 evaluating popularity model

```
In [362]: popularity_model.evaluate_rmse(test_data,target='review')
```

```
Out[362]: {'rmse_by_item': Columns:
        joki_id        str
        count         int
        rmse         float

Rows: 100

Data:
+---------+-------+---------------+
| joki_id | count |      rmse     |
+---------+-------+---------------+
| joke_78 |  3915 |  5.2706092644 |
| joke_57 |  6432 | 5.06888091275 |
| joke_99 |  3879 | 5.30913045448 |
| joke_32 |  6321 | 5.27330861522 |
```

```
| joke_16 | 14847 | 4.44703714777 |
| joke_94 |  4456 | 5.27025442705 |
| joke_71 |  3815 | 4.65792146235 |
| joke_56 |  6533 |  5.435632758  |
| joke_65 | 11911 | 4.77237117336 |
| joke_93 |  4404 |  5.1595079663 |
+---------+-------+---------------+
[100 rows x 3 columns]
Note: Only the head of the SFrame is printed.
You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.,
'rmse_by_user': Columns:
        user_id        str
        count        int
        rmse        float

Rows: 73227

Data:
+------------+-------+---------------+
|  user_id   | count |      rmse     |
+------------+-------+---------------+
| user_32879 |   14  | 5.21729715327 |
| user_34332 |   15  | 5.14576468901 |
| user_70080 |   5   | 3.75966012684 |
| user_1475  |   14  | 5.22835560505 |
| user_41387 |   17  | 2.41622834582 |
| user_63187 |   4   | 5.98411085532 |
|  user_960  |   7   | 4.14671027835 |
| user_27414 |   9   | 3.71588111606 |
| user_52473 |   9   | 3.47944717945 |
| user_32483 |   7   | 6.10349552005 |
+------------+-------+---------------+
[73227 rows x 3 columns]
Note: Only the head of the SFrame is printed.
You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.,
'rmse_overall': 5.046616644117791}
```

20 Building general recommender

```
In [379]: general_recommender = graphlab.recommender.create(train_data,user_id='user_id', item_id='joke
```

```
Recsys training: model = ranking_factorization_recommender


Preparing data set.


    Data has 3309475 observations with 73421 users and 100 items.


    Data prepared in: 3.71343s


Training ranking_factorization_recommender for recommendations.
```

```
+--------------------------------+-----------------------------------------------------+----------+
| Parameter                      | Description                                         | Value    |
+--------------------------------+-----------------------------------------------------+----------+
| num_factors                    | Factor Dimension                                    | 32       |
| regularization                 | L2 Regularization on Factors                        | 1e-09    |
| solver                         | Solver used for training                            | sgd      |
| linear_regularization          | L2 Regularization on Linear Coefficients            | 1e-09    |
| ranking_regularization         | Rank-based Regularization Weight                    | 0.25     |
| max_iterations                 | Maximum Number of Iterations                        | 25       |
+--------------------------------+-----------------------------------------------------+----------+

  Optimizing model using SGD; tuning step size.

  Using 413684 / 3309475 points for tuning the step size.

+---------+------------------+-------------------------------------------+
| Attempt | Initial Step Size | Estimated Objective Value                |
+---------+------------------+-------------------------------------------+
| 0       | 25               | Not Viable                                |
| 1       | 6.25             | Not Viable                                |
| 2       | 1.5625           | Not Viable                                |
| 3       | 0.390625         | Not Viable                                |
| 4       | 0.0976562        | Not Viable                                |
```

```
| 5       | 0.0244141         | 42.3368                                    |

| 6       | 0.012207          | 27.0128                                    |

| 7       | 0.00610352        | 43.3131                                    |

| 8       | 0.00305176        | 48.2564                                    |

| 9       | 0.00152588        | 48.5417                                    |

+---------+-------------------+--------------------------------------------+

| Final   | 0.012207          | 27.0128                                    |

+---------+-------------------+--------------------------------------------+

Starting Optimization.

+---------+--------------+-------------------+----------------------+-------------+

| Iter.   | Elapsed Time | Approx. Objective | Approx. Training RMSE | Step Size   |

+---------+--------------+-------------------+----------------------+-------------+

| Initial | 84us         | 54.9809           | 5.29576              |             |

+---------+--------------+-------------------+----------------------+-------------+

| 1       | 2.12s        | 33.9055           | 4.88738              | 0.012207    |

| 2       | 4.65s        | 29.1659           | 4.47454              | 0.00725834  |

| 3       | 7.28s        | 26.4201           | 4.20878              | 0.00535512  |

| 4       | 9.30s        | 24.3381           | 3.99986              | 0.00431584  |

| 5       | 11.37s       | 23.2614           | 3.88677              | 0.00365075  |

| 6       | 13.53s       | 22.0154           | 3.75433              | 0.00318417  |
```

7	15.64s	20.7576	3.61711	0.00283652
8	17.67s	19.8428	3.51879	0.00256621
9	19.76s	19.236	3.45242	0.00234924
10	21.84s	18.5359	3.37494	0.00217075
11	24.01s	18.005	3.31699	0.002021
12	26.13s	17.5978	3.2731	0.00189332
13	28.17s	17.2357	3.23346	0.001783
14	30.22s	16.9378	3.20172	0.00168661
15	32.36s	16.6666	3.17147	0.00160155
16	34.49s	16.4456	3.14745	0.00152588
17	36.93s	16.2828	3.13064	0.00145805
18	39.06s	16.1079	3.11229	0.00139687
19	41.13s	15.9598	3.09671	0.00134136
20	43.24s	15.8274	3.08234	0.00129074
21	45.35s	15.704	3.06858	0.00124436
22	47.40s	15.5933	3.05806	0.00120169
23	49.40s	15.5075	3.04793	0.00116229
24	51.44s	15.4159	3.03903	0.00112578
25	53.53s	15.3387	3.03054	0.00109183

```
Optimization Complete: Maximum number of passes through the data reached.

Computing final objective value and training RMSE.

       Final objective value: 32.137

       Final training RMSE: 2.90386
```

21 evaluating general recommender

```
In [364]: general_recommender.evaluate_rmse(test_data,target='review')

Out[364]: {'rmse_by_item': Columns:
                  joki_id        str
                  count         int
                  rmse         float

           Rows: 100

           Data:
           +---------+-------+---------------+
           | joki_id | count |      rmse     |
           +---------+-------+---------------+
           | joke_78 |  3915 | 6.12883505924 |
           | joke_57 |  6432 | 4.82362908967 |
           | joke_99 |  3879 | 5.70530880192 |
           | joke_32 |  6321 | 4.62443314465 |
           | joke_16 | 14847 |  4.7395895361 |
           | joke_94 |  4456 | 10.1155154598 |
           | joke_71 |  3815 |  9.7602384386 |
           | joke_56 |  6533 | 5.47871344112 |
           | joke_65 | 11911 | 6.06577206843 |
           | joke_93 |  4404 | 7.82481055382 |
           +---------+-------+---------------+
           [100 rows x 3 columns]
           Note: Only the head of the SFrame is printed.
           You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.,
           'rmse_by_user': Columns:
                  user_id        str
                  count         int
                  rmse         float

           Rows: 73227

           Data:
           +------------+-------+---------------+
           |  user_id   | count |      rmse     |
           +------------+-------+---------------+
           | user_32879 |   14  | 4.77668717563 |
           | user_34332 |   15  |  7.0989893879 |
```

```
| user_70080 |   5   | 7.27333790121 |
| user_1475  |   14  | 7.15723705609 |
| user_41387 |   17  | 2.53174570595 |
| user_63187 |   4   | 5.02444636001 |
|  user_960  |   7   | 9.08041728172 |
| user_27414 |   9   | 6.94757390208 |
| user_52473 |   9   | 11.5083839901 |
| user_32483 |   7   | 3.46832290559 |
+------------+-------+---------------+
[73227 rows x 3 columns]
Note: Only the head of the SFrame is printed.
You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.,
'rmse_overall': 6.272985790588099}
```

22 evaluating personalized recommender

```
In [380]: personalized_model.evaluate_rmse(test_data,target = 'review')
```

```
Out[380]: {'rmse_by_item': Columns:
        joke_id        str
        count        int
        rmse        float

Rows: 100

Data:
+---------+-------+---------------+
| joke_id | count |      rmse     |
+---------+-------+---------------+
| joke_78 |  3915 | 5.27463781338 |
| joke_57 |  6432 | 6.53650605181 |
| joke_99 |  3879 |  5.3861877125 |
| joke_32 |  6321 | 5.56070174459 |
| joke_16 | 14847 | 4.75290014345 |
| joke_94 |  4456 | 5.33362442618 |
| joke_71 |  3815 | 5.28987317219 |
| joke_56 |  6533 | 5.91247126564 |
| joke_65 | 11911 | 5.17207504741 |
| joke_93 |  4404 | 5.25955290418 |
+---------+-------+---------------+
[100 rows x 3 columns]
Note: Only the head of the SFrame is printed.
You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.,
'rmse_by_user': Columns:
        user_id        str
        count        int
        rmse        float

Rows: 73227

Data:
+------------+-------+---------------+
|  user_id   | count |      rmse     |
+------------+-------+---------------+
```

```
| user_32879 |   14   | 5.63581533765 |
| user_34332 |   15   | 5.31692305374 |
| user_70080 |   5    | 4.76225440368 |
| user_1475  |   14   | 6.22835840175 |
| user_41387 |   17   | 2.72639588879 |
| user_63187 |   4    | 5.84379164485 |
|  user_960  |   7    | 5.90023980565 |
| user_27414 |   9    | 4.25081974718 |
| user_52473 |   9    |  3.7167997157 |
| user_32483 |   7    | 5.47128396104 |
+------------+-------+---------------+
[73227 rows x 3 columns]
Note: Only the head of the SFrame is printed.
You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.,
'rmse_overall': 5.30675182934206}
```

In [381]:
```
popularity_model.recommend(users=[users[0]])
```

Out[381]:
```
Columns:
        user_id        str
        joke_id        str
        score        float
        rank         int

Rows: 10

Data:
+------------+---------+---------------+------+
|  user_id   | joke_id |     score     | rank |
+------------+---------+---------------+------+
| user_32879 | joke_49 | 3.36345782642 |  1   |
| user_32879 | joke_88 | 3.34239015175 |  2   |
| user_32879 | joke_71 | 2.63586814492 |  3   |
| user_32879 | joke_75 |  2.3584540408 |  4   |
| user_32879 | joke_92 | 2.24987329207 |  5   |
| user_32879 | joke_82 | 1.99036453758 |  6   |
| user_32879 | joke_87 | 1.96596530196 |  7   |
| user_32879 | joke_90 | 1.90584991337 |  8   |
| user_32879 | joke_86 | 1.86409602955 |  9   |
| user_32879 | joke_10 | 1.75043701272 |  10  |
+------------+---------+---------------+------+
[10 rows x 4 columns]
```

In [382]:
```
personalized_model.recommend(users=[users[0]])
```

Out[382]:
```
Columns:
        user_id        str
        joke_id        str
        score        float
        rank         int

Rows: 10

Data:
+------------+---------+---------------+------+
```

```
|  user_id   | joke_id |     score      | rank |
+------------+---------+----------------+------+
| user_32879 | joke_25 | 0.509686829751 |  1   |
| user_32879 | joke_21 | 0.509445395386 |  2   |
| user_32879 | joke_46 |  0.5044700953  |  3   |
| user_32879 | joke_10 | 0.487552360484 |  4   |
| user_32879 | joke_40 | 0.479456583659 |  5   |
| user_32879 | joke_49 | 0.476574662485 |  6   |
| user_32879 | joke_58 | 0.47456200081  |  7   |
| user_32879 | joke_66 | 0.467159702067 |  8   |
| user_32879 | joke_43 | 0.444288187905 |  9   |
| user_32879 | joke_41 | 0.434963774263 |  10  |
+------------+---------+----------------+------+
[10 rows x 4 columns]
```

In [383]:
```
general_recommender.recommend(users=[users[0]])
```

Out[383]:
```
Columns:
        user_id        str
        joke_id        str
        score        float
        rank         int

Rows: 10

Data:
+------------+---------+----------------+------+
|  user_id   | joke_id |     score      | rank |
+------------+---------+----------------+------+
| user_32879 | joke_46 | 3.02015872232  |  1   |
| user_32879 | joke_10 | 0.196886255661 |  2   |
| user_32879 | joke_25 | -1.75251393088 |  3   |
| user_32879 | joke_49 | -2.19586615332 |  4   |
| user_32879 | joke_41 | -2.44015841254 |  5   |
| user_32879 | joke_58 | -2.9478497959  |  6   |
| user_32879 | joke_76 | -3.11662869223 |  7   |
| user_32879 | joke_79 | -4.50840334662 |  8   |
| user_32879 | joke_21 | -4.57616429099 |  9   |
| user_32879 | joke_15 | -4.78444723853 |  10  |
+------------+---------+----------------+------+
[10 rows x 4 columns]
```

In [ ]: