
Python for Collective Intelligence
and Collaborative Filtering

What is Collective Intelligence?

“A shared or group intelligence that emerges from the collaboration and competition of many individuals”

-Wikipedia


stackoverflow
99 designs
python
Linux
YAHOO! ANSWERS

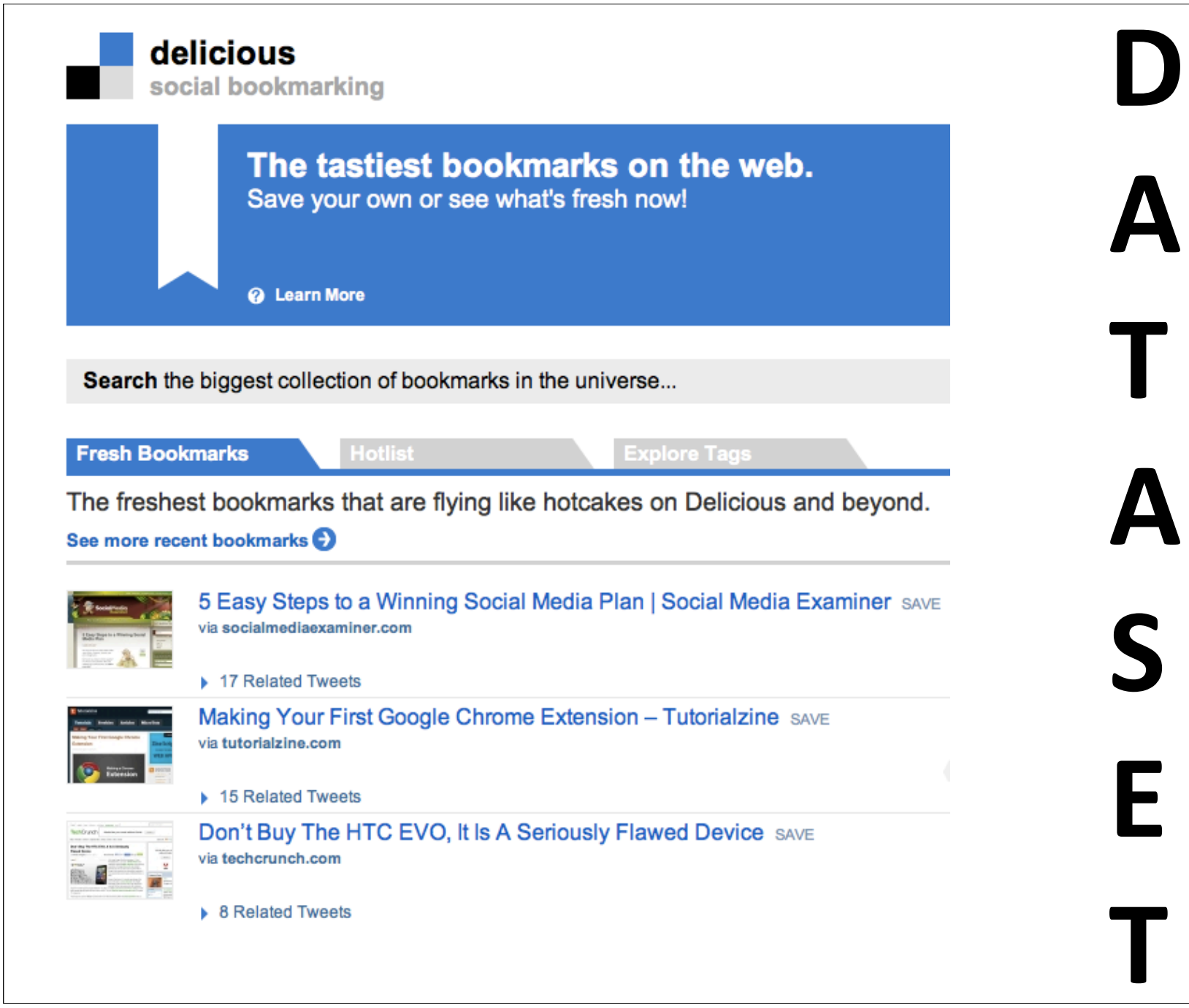
Collaborative Filtering

"The process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc."
-Wikipedia

Implementation

Concepts

* DataSet
* User Ratings
*“Similarity Score”


delicious
social bookmarking
The tastiest bookmarks on the web.
Save your own or see what's fresh now!
Learn More
Search the biggest collection of bookmarks in the universe...
Fresh Bookmarks
Hotlist
Explore Tags
The freshest bookmarks that are flying like hotcakes on Delicious and beyond.
See more recent bookmarks
5 Easy Steps to a Winning Social Media Plan | Social Media Examiner SAVE
via socialmediaexaminer.com
17 Related Tweets
Making Your First Google Chrome Extension – Tutorialzine SAVE
via tutorialzine.com
15 Related Tweets
Don't Buy The HTC EVO, It Is A Seriously Flawed Device SAVE
via techcrunch.com
8 Related Tweets

DATASET

http://code.google.com/p/pydelicious/

Owners:
gregpin...@gmail.com,
berend.van.berkum

Committers:
nelson.minar, dfdeshom, alexcoppe,
aman.coe

```
from pydelicious import get_popular, get_userposts, get_urlposts
```

Delicious API

```
def initializeUserDict(tag,count=5):
  user_dict={}
  #get the top count popular posts
  for p1 in get_popular(tag=tag)[0:count]:
    #find all users who posted this
    for p2 in get_urlposts(p1['url']):
      user=p2['user']
      user_dict[user]={}
  return user_dict
```

Get list of recent posters

```
def fillItems(user_dict):
  all_items={}
  # Find links posted by all users
  for user in user_dict:
    for i in range(3):
      try:
        posts=get_userposts(user)
        break
      except:
        print "Failed user "+user+", retrying"
        tim.sleep(4)
    for post in posts:
      url=post['url']
      user_dict[user][url]=1.0
      all_items[url]=1
```

get recent items posted by recent posters

```
In [11]: pythonistas = initializeUserDict('python')

In [12]: pythonistas
Out[12]:
{u'3rdparty': {},
 u'AlexTreppass': {},
 u'BLConger': {},
 u'Chaseology': {},
 u'HiwayBK': {},
 u'Passy89': {},
 u'Pgawlik': {},
 u'antoni.aloy': {},
 u'asenchi': {},
 u'bandini': {},
 u'binary_core': {},
 u'catalinc': {},
 u'coffeebucket': {},
 u'contro': {},
 u'crowesesse': {},
 u'deduce': {},
 u'deltasig': {},
 u'eheder': {},
 u'ericbrothers': {},
 u'guatepapi': {},
 u'gwendoux': {},
 u'hergo': {},
```

User Ratings

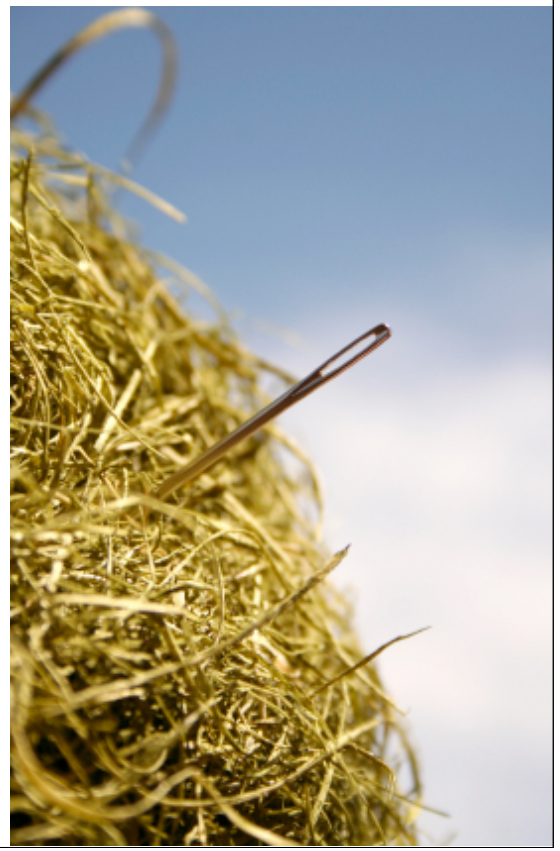
```
def fillItems(user_dict):
  all_items={}
  # Find links posted by all users
  for user in user_dict:
    for i in range(3):
      try:
        posts=get_userposts(user)
        break
      except:
        print "Fialed user "+user+", retrying"
        time.sleep(4)
    for post in posts:
      url=post['url']
      user_dict[user][url]=1.0
      all_items[url]=1
```

User Ratings:
if user posted the link = 1
else = 0

```
In [16]: fillItems(pythonistas)

In [17]: pythonistas
Out[17]:
{u'3rdparty': {u'http://$http://blogs.msdn.com/jonbox/archive/2010/02/20/ie8-building-a-web-slice-does
               u'http://$http://blogs.msdn.com/jonbox/archive/2010/02/28/ie8-fundamentals-of-visual-se
               u'http://0xfe.blogspot.com/2010/05/music-notation-with-html5-canvas.html': 0.0,
               u'http://4sysops.com/archives/microsoft-vdi-resources/?utm_source=feedburner&utm_medium
               u'http://4sysops.com/archives/scan-your-website-for-malware-with-free-tools/?utm_source
               u'http://abcnews.go.com/print?id=3951187': 0.0,
               u'http://abovethelaw.com/2010/05/a-massachusetts-lawyer-you-dont-want-to-work-for-and-d
               u'http://activemq.apache.org/': 0.0,
               u'http://aencre.org/blog/2010/bol-directors-cut/': 0.0,
               u'http://aht.seriouseats.com/archives/2010/05/the-burger-lab-how-to-make-perfect-mcdond
               u'http://aja-aja.com/': 0.0,
               u'http://ajaxian.com/archives/the-vml-changes-in-ie-8': 0.0,
               u'http://alestic.com/2010/05/ec2-move-ebs-boot-instance?utm_source=feedburner&utm_mediu
               u'http://alexpogosyan.com/color-theme-creator/': 0.0,
               u'http://alicebob.cryptoland.net/closure-gotcha-with-python/': 0.0,
               u'http://alistrol.com/': 0.0,
               u'http://alt.pluralsight.com/wiki/default.aspx/Keith.GuideBook/HomePage.html': 0.0,
               u'http://anchor.com.au/hosting/dedicated/Tuning_PostgreSQL_on_your_Dedicated_Server': 0
               u'http://andyman3000.posterous.com/the-coke-zero-and-mentos-rocket-car-25': 0.0,
               u'http://answers.oreilly.com/topic/1463-create-3d-photographs-with-processing/': 0.0,
               u'http://apirocks.com/html5/html5.html#slide1': 0.0,
               u'http://arcademicskillbuilders.com/': 0.0,
               u'http://arstechnica.com/web/news/2009/12/commonjs-effort-sets-javascript-on-path-for-w
               u'http://arsvox.com/': 0.0,
               u'http://article.yeeyan.org/view/127113/109664': 0.0,
               u'http://article.yeeyan.org/view/135134/109783': 0.0,
               u'http://article.yeeyan.org/view/149012/109781': 0.0,
               u'http://article.yeeyan.org/view/151032/109762': 0.0,
```

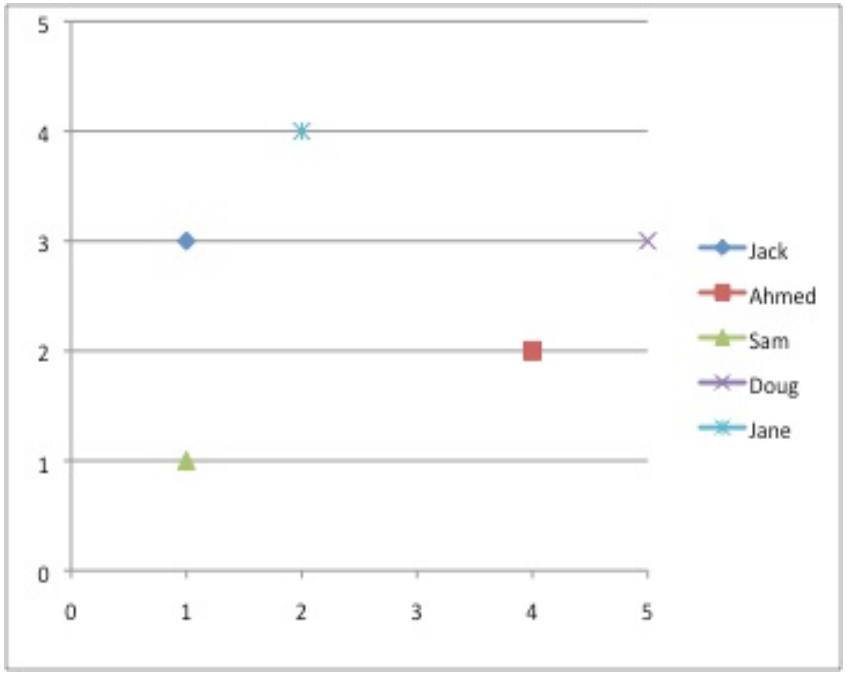
“Similarity Scores”

i.e.



Filtering Techniques

Euclidean Distance



“In mathematics the Euclidean distance or Euclidean metric is the "ordinary" distance between two points that one would measure with a ruler, and is given by the Pythagorean Formula.” -Wikipedia

the formula:

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}.$$

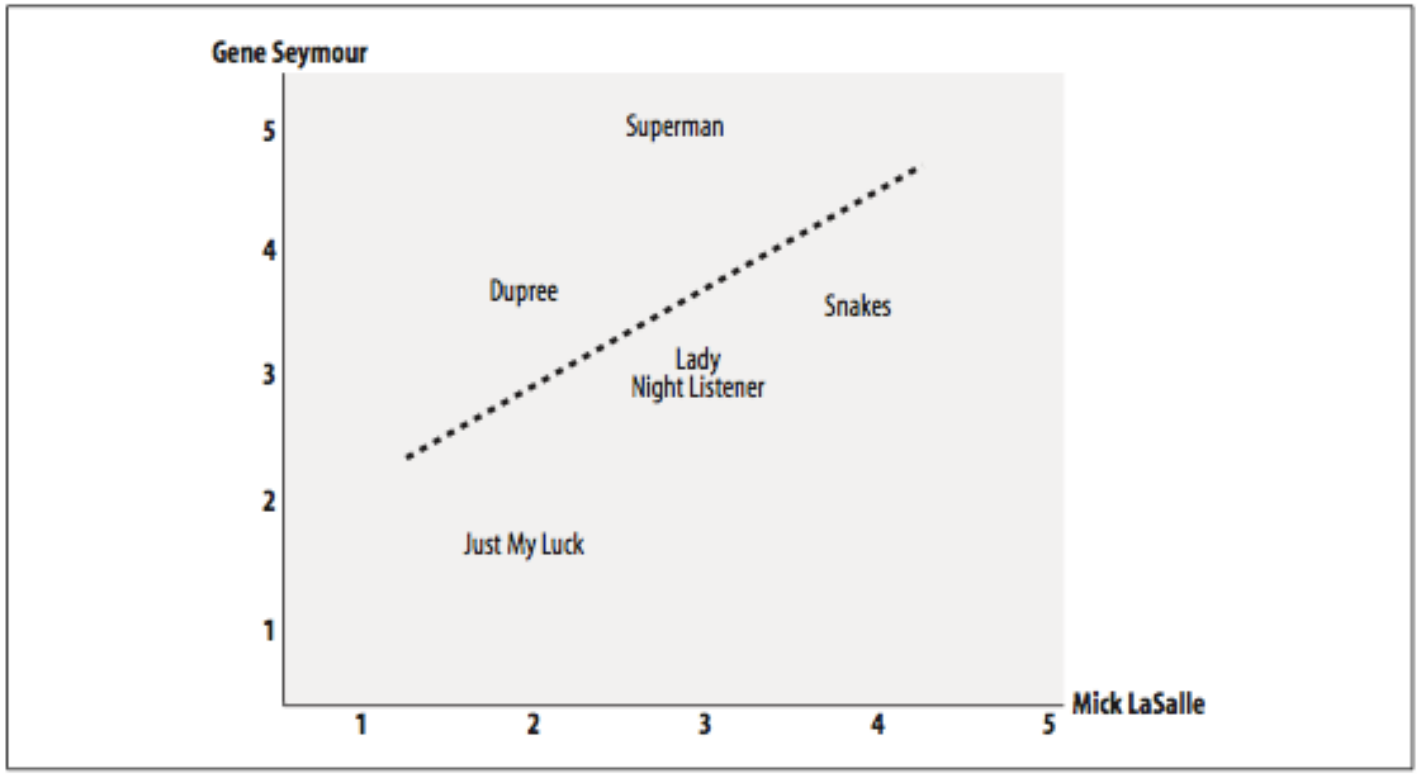
the code:

```
from math import sqrt
def euclidean_distance(prefs,p1,p2):
    '''Returns a distance-based similarity score for person1 and person2
    this is an example of the Euclidean distance formula for making recommendations
    between two people'''
    si={}
    for item in prefs[p1]:
        if item in prefs[p2]:
            si[item]=1

    #if they have no ratings in common, return 0
    if len(si)==0: return 0

    # Add up the squares of all the differences
    sum_of_squares=sum([pow(prefs[p1][item]-prefs[p2][item],2)
        for item in prefs[p1] if item in prefs[p2]])

    return 1/(1+sum_of_squares)
```

```
In [46]:
In [46]: euclidean_distance(pythonistas, '3rdparty','binary_core')
Out[46]: 0.032258064516129031
```

```
In [53]: for item in pythonistas :
   ....:     euclidean_distance(pythonistas, 'binary_core', item)
   ....:
Out[53]: 0.032258064516129031
Out[53]: 0.032258064516129031
Out[53]: 0.040000000000000001
Out[53]: 0.032258064516129031
Out[53]: 0.032258064516129031
Out[53]: 0.032258064516129031
Out[53]: 0.032258064516129031
Out[53]: 0.032258064516129031
Out[53]: 0.032258064516129031
Out[53]: 0.032258064516129031
Out[53]: 0.032258064516129031
Out[53]: 1.0
Out[53]: 0.032258064516129031
Out[53]: 0.03125
Out[53]: 0.034482758620689655
Out[53]: 0.028571428571428571
```

Pearson's Correlation Coefficient



For the case where users have similar preferences but user A consistently gives higher ratings than user B the Euclidian distance score will not produce accurate results. However the Pearson Correlation Coefficient will factor out that bias because it focuses on measuring the data's fit to a straight line. It will return a number between 1 and -1 with a value of 1 indicating two users have rated all items the same.

the formula: $P_{a,u} = \frac{\sum_{i=1}^{m} (r_{a,i} - \overline{r}_a) \times (r_{u,i} - \overline{r}_u)}{\sqrt{\sum_{i=1}^{m} (r_{a,i} - \overline{r}_a)^2 \times \sum_{i=1}^{m} (r_{u,i} - \overline{r}_u)^2}}$

the code:

```
def pearson_correlation(prefsMapping, p1, p2):
    '''returns the pearson Correlation Coefficient, which is a measure of how similar p1 and p2s
    to one another.  Value is between 1 and 0 with 1 meaning p1 and p2 have identical preferences
    itemRatings={}
    for item in prefsMapping[p1]:
        if item in prefsMapping[p2]: itemRatings[item]=1 #both parties rated the items

    n=len(itemRatings)
    if n==0: return 0

    # Add up all the preferences
    sum1=sum([prefsMapping[p1][it] for it in itemRatings])
    sum2=sum([prefsMapping[p2][it] for it in itemRatings])
    #sum up the squares
    sum1Sq=sum([pow(prefsMapping[p1][it],2) for it in itemRatings])
    sum2Sq=sum([pow(prefsMapping[p2][it],2) for it in itemRatings])
    #sum up the products
    pSum=sum([prefsMapping[p1][it]*prefsMapping[p2][it] for it in itemRatings])
    #Calculate Pearson score
    num=pSum-(sum1*sum2/n)
    den=sqrt((sum1Sq-pow(sum1,2)/n)*(sum2Sq-pow(sum2,2)/n))
    if den==0: return 0

    r=num/den
    return r
```

```
In [58]: pearson_correlation(pythonistas, '3rdparty', 'binary_core')
Out[58]: -0.015822784810126583

In [59]: for item in pythonistas :
   ....:     pearson_correlation(pythonistas, 'binary_core', item)
   ....:
Out[59]: -0.015822784810126583
Out[59]: -0.015822784810126583
Out[59]: -0.015822784810126583
Out[59]: -0.015822784810126583
Out[59]: -0.015822784810126583
Out[59]: -0.015822784810126583
Out[59]: -0.015822784810126583
Out[59]: -0.015822784810126583
Out[59]: 1.0
Out[59]: -0.015822784810126583
Out[59]: -0.016350327748670335
Out[59]: 0.051898734177215189
```

Which Similarity Metric to Use?

- For del.ici.ous the user ranking score is only 1 or 0
- Euclidean is the most straight forward
- Pearson's Correlation is perhaps better for 1 to 10 user rankings
- There are also many others:
  - Tanimoto Coefficient
  - Manhattan Distance

Choose Similar People

```
def findPeopleWithSimilarInterests(prefs, subject, scoringAlgorithm=euclidean_distance, numResults=5):
    '''call the scoring Algorighm for everybody in the prefs dataset and compare
    that person to subject.  The return the dataset ordered by similarity to subject'''

    scores=[(scoringAlgorithm(prefs,subject,otherPerson),otherPerson)
            for otherPerson in prefs if otherPerson != subject]

    #sort by similarity with most similar on top
    scores.sort()
    scores.reverse()
    return scores[0:numResults]
```

Euclidean Distance

```
In [69]: from recommendations import findPeopleWithSimilarInterests

In [70]: findPeopleWithSimilarInterests(pythonistas, 'j1z0')
Out[70]:
[(0.14285714285714285, u'lixipeng'),
 (0.0625, u'sixh'),
 (0.052631578947368418, u'zecharia'),
 (0.052631578947368418, u'zbskii'),
 (0.052631578947368418, u'virkang')]
```

Pearson's Correlation

```
In [71]: findPeopleWithSimilarInterests(pythonistas, 'j1z0', scoringAlgorithm=pearson_correlation)
Out[71]:
[(-0.0031120331950207471, u'lixipeng'),
 (-0.0062533256777233589, u'sixh'),
 (-0.0070024378952024437, u'zecharia'),
 (-0.0070024378952024437, u'zbskii'),
 (-0.0070024378952024437, u'virkang')]
```


Making Recommendations
Use similarity score
or better yet
use weighted rankings

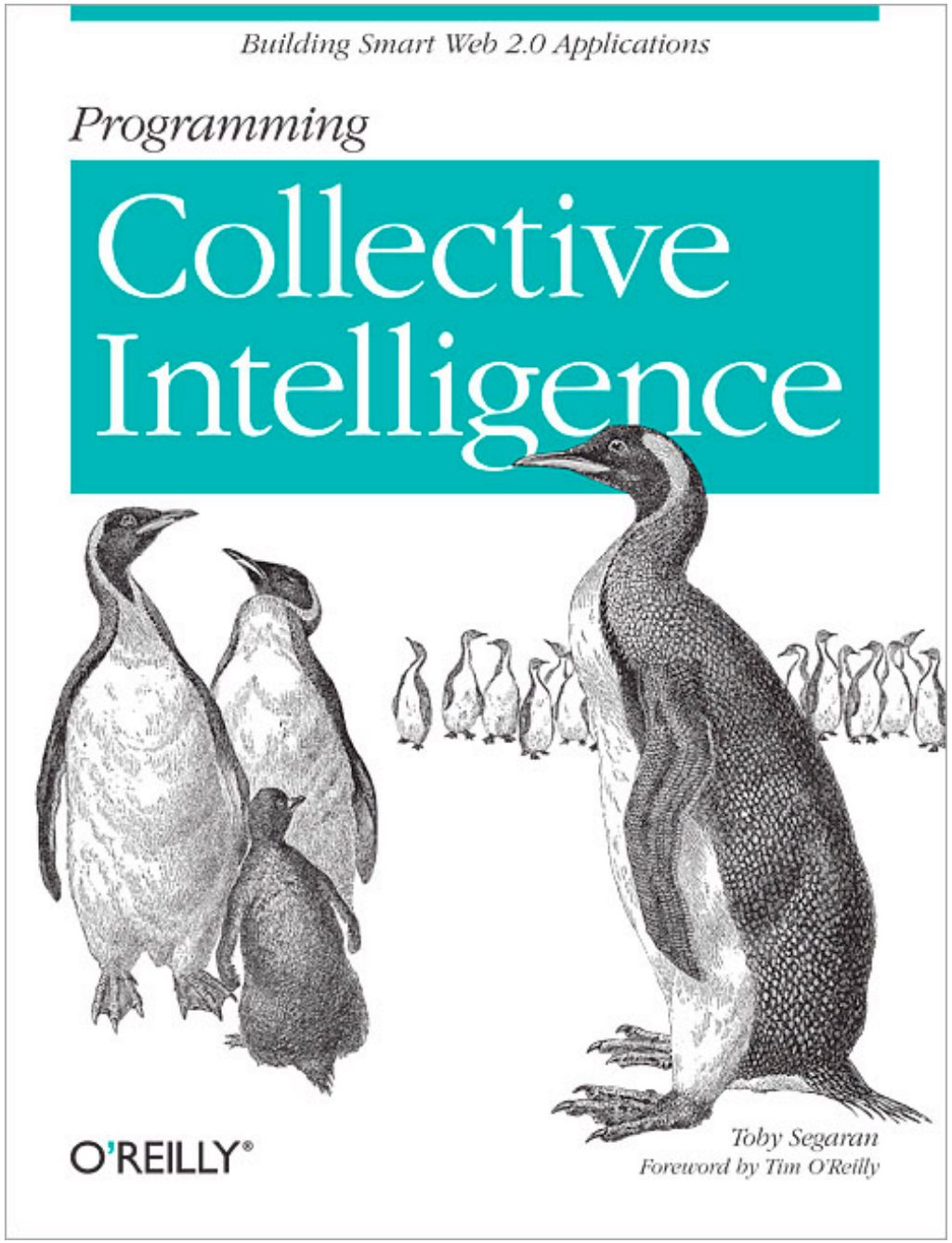
Recommendations = similarity score * user rankings

```
def getRecommendations(prefs, subject, scoringAlgorithm=euclidean_distance):
    #could use defaultdict(int) here
    totals={}
    weights={}
    for otherPerson in prefs:
        if otherPerson==subject: continue
        score=scoringAlgorithm(prefs,subject,otherPerson)
        #ignore zero scores
        if score<=0: continue
        for item in prefs[otherPerson]:
            #only score items I haven't rated (i.e. not viewed)
            if item not in prefs[subject] or prefs[subject][item] == 0:
                #store the weighted score for the item in totals
                totals.setdefault(item,0)
                totals[item]+=prefs[otherPerson][item]*score
                weights.setdefault(item,0)
                weights[item]+=score

    #take the average weights so popular sites are rated higher
    rankings=[(total/weights[item],item) for item,total in totals.items()]

    #return sorted list
    rankings.sort()
    rankings.reverse()
    return rankings
```

```
In [72]: getRecommendations(pythonistas,'j1z0')[:5]
Out[72]:
[(0.15574531950510848, u'http://stackp.online.fr/?p=28'),
 (0.11236420311327462,
  u'http://pythonconquerstheuniverse.wordpress.com/2009/09/10/debugging-in-python/'),
 (0.10972272998402947, u'https://wiki.ubuntu.com/Xpresser'),
 (0.07307229031575296 4,
  u'http://tech.blog.aknin.name/2010/05/19/guidos-python-objects-102/'),
 (0.07040541840641891, u'http://cb.vu/unixtoolbox.xhtml')]
```

End to End Code

```
#build delicious dataset
pythonistas = initializeUserDict('python')
#fill data set with user rankings
fillItems(pythonistas)
#find other people with the same interests as you
findPeopleWithSimilarInterests(pythonistas, 'j1z0', scoringAlgorithm=euclidean_distance)
#get a list of recommended links
getRecommendations(pythonistas, 'j1z0')[:10]
```


Building Smart Web 2.0 Applications
Programming
Collective Intelligence
O'REILLY®
Toby Segaran
Foreword by Tim O'Reilly

Thanks Toby


Q & A


Jeremy Johnson
Python Geek
Email: jeremy@python.my
python
MALAYSIA