Michael Yung

Data Science   
Final Report

2014- April



# Hypothesis

Nowadays it’s very easy to connect to any people in Social Networks, no matter you know them or not. You can connect to colleagues, friends, relatives, ex-colleagues, and many others in different types of Social Networks such as Facebook, Twitter, Foursquare and Linkedin. Unlike other social networks, Linkedin let you connect with like-minded professionals in your industry.

In fact, Linkedin has also become a very good recruitment tool. You can search, filter, and contact any members in the Linkedin and of course in your own network. And you can do more like checking their job history, skills and expert areas to see if any one of them matches the vacancy’s job description.

Out of all these information, the member’s skills and expert areas are very important as those tell a story of what he or she is really good at. And the way those information are collected makes these “background check” authentic because both skills and experts areas are “endorsed” by their connections, but not entered by themselves.

Finding the right people to join your company is always a tough task. Therefore, it will be great if we can analyze the endorsed skills of candidates to see:

1. Whether he or her is really as good as claimed, say by matching the skills with other skills of a group of professionals;
2. Whether we can further develop the candidate through trainings, after we identified gaps between the candidate’s skill sets with others with similar group of skills.

Machine learning algorithms like Clustering, looks like a good solution.

# Data Set

Linkedin data is used to collect the skill sets data for analysis. One can easily assess his or her profile as well as endorsed skill sets through the Linkedin development API. Other information like the connections and the basic profile of the connections (like First name, Last name, Job titles etc.) are also available through the API.

However, the crucial information – the skill sets of other people in the network, are not available. Therefore, some website scraping tools are needed to “retrieve” all those skill sets. The pseudo codes are:

From the starting profile (likely your own Linkedin profile)

Retrieve the URLs of all the connections’ profile

For each URL {

Visit the webpage

Find the skill section with BeautifulSoup library

Extract the skill description

Store into output file

}

One good thing about using Linkedin skill keywords, compare to other textual content analysis is “Skill” keywords are pretty much free from Less-important (e.g. most, best, largest etc.), Stemming (e.g. Development and Developing) and Stop words (e.g. a, an, the etc.). In other words, we can pretty safe quite a lot of works to use the collected skill sets.

I used my Linkedin profile as a starting point to collect skill sets and what I got was:

1. More than 500 Connections (from my profile)
2. More than 3,600 Skills (all skills of those connections)
3. More than 400 unique Skills (after removing those duplicated skill sets)

# Statistical Model

Three models and techniques are used in this project:

1. Bag-of-word
2. Euclidean Distance, and
3. KMeans Clustering

**Bag-of-word approach** uses simple word counts as its basis. For each word in the skill set file, its occurrence is counted and noted in a vector (i.e. vectorization). This vector is huge as it contains as many elements as the words that occur in the whole dataset.

The vectorized skill set is than used to calculate the **Similarity Measurement (Euclidean Distance)** with a new skill sets (for example, from a candidate). In this exercise, I used my own skill set to try – **“IT Strategy Information Technology Cloud Computing IT Management Project Management Security IT Audit PKI Databases Internet Technologies”**.

The pseudo code is:

Function dist\_norm(v1, v2):

v1\_normalized = v1/sp.linalg.norm(v1.toarray())

v2\_normalized = v2/sp.linalg.norm(v2.toarray())

delta = v1\_normalized - v2\_normalized

return sp.linalg.norm(delta.toarray())

Input new skill set and vectorize

For each skill vector {

Call dist\_norm(skill vector, new skill vector)

Find the shortest distance

Find the farthest distance

}

Lastly, we have our skill vectors captured and it’s time to group them together in clusters. I initially set it to **a maximum of 10 clusters and then run thru the KMeans algorithm**. Following that is to again input the new skill sets as test data to predict which group the new skill set will fall in. Lastly, the distances of different skill sets to the centroid in the same cluster are calculated. Recall that the new skill set is:

**“IT Strategy Information Technology Cloud Computing IT Management Project Management Security IT Audit PKI Databases Internet Technologies”**

The pseudo code is:

Set # of cluster to 10

Run KMeans to find group the skill sets to defined no. of clusters

Input new skill sets and vectorize

Predict which cluster the new skill sets will fall into

Find out all skill sets in the same cluster

Find out the similarity distance of the following skill sets to the centroid:

1. The closest one
2. The middle one
3. The farthest one

# Challenges

There was couple of challenges in working on this final project. First and foremost, it’s not easy to decide what project to do and what data science knowledge to apply. Once I decided to use Linkedin Skill set as data and then the Text clustering as the method, the challenges were mainly related to technology and programming, namely:

1. Being a rookie to Python, handling all sorts of input, output and looping logic were a challenge. Still I think it’s an excellent experience.
2. Linkedin APIs let you retrieve your own skills, and your connections’ URLs but not your connections’ skills – that’s why I needed to use site scraping.
3. Linkedin block scraping tools like Kimono, Scrape – that’s why I used BeautifulSoup. And of course, wasted quite some time to research and to try out all these tools before settled on BeautifulSoup.
4. Scraping performance is rather poor – it took about 20 to 30 minutes to scape 500+ connections’ profile sites.
5. It took me some time to figure out why the generated results are different every time I ran KMeans and good to know that it generated random results since the initialization method is random.
6. Lastly, unlike data sets with numbers, I didn’t know how to plot the data to illustrate the work. Thanks to the eye-opening presentation by Jane who introduced visualization tool - ManyEyes.com. That helps a lot in text visualization. However, I soon found out the ManyEyes.com website has SSL certificate problem, requires Java and requires the PC to relax security. Anyway, it finished the job nicely.

# Visualizations

Three word related visualization charts were prepared, using ManyEyes.com – Word Cloud, Phrase Net and Word Tree.

**Word Cloud** – It shows the most popular words in the skill sets.

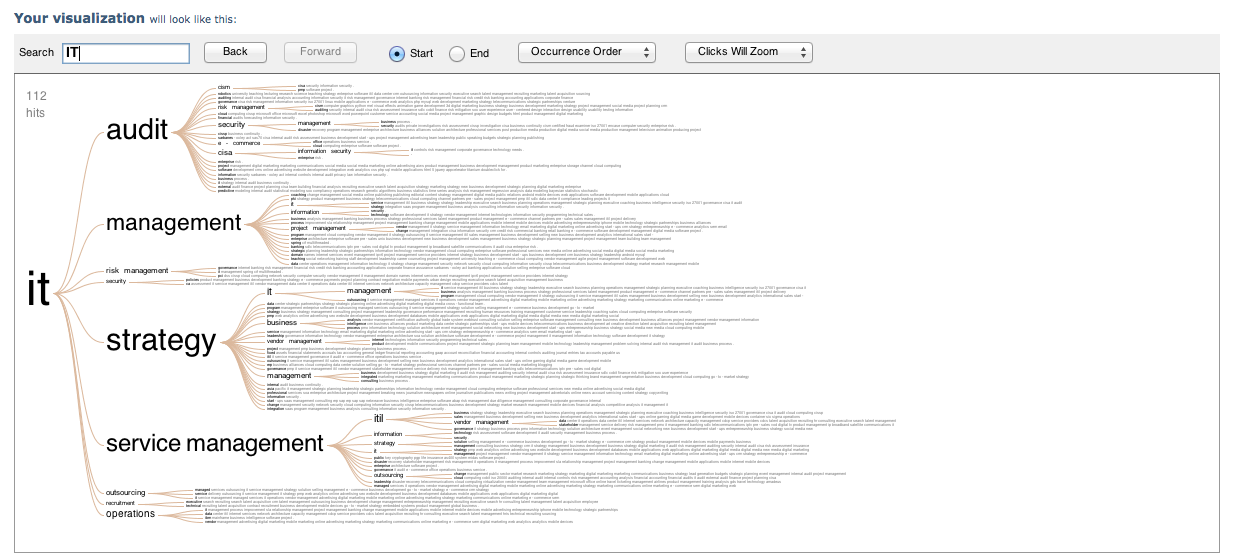


**Phrase Net** – Other than the words, and it also shows the strength of connections.

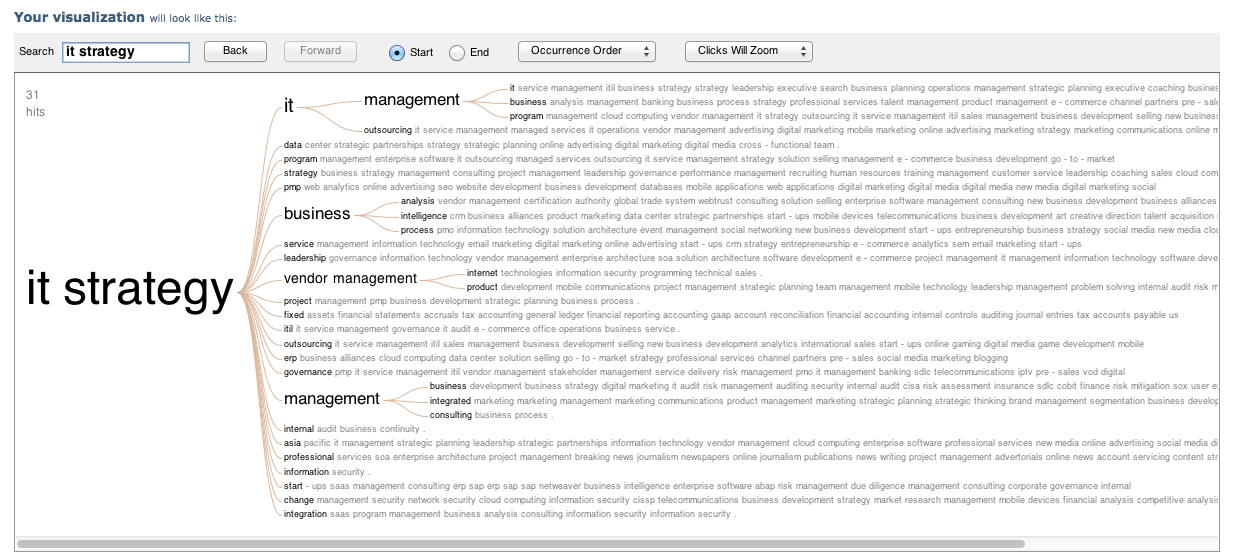


**Word Tree**

Word tree with “IT” as the stem:



A drill down of “IT Strategy”:



# Findings

The closest and the farthest connections, from the input skill sets, are:

|  |  |
| --- | --- |
| Input Skill | IT Strategy Information Technology Cloud Computing IT Management Project Management Security IT Audit PKI Databases Internet Technologies |
| Connection # with closest skill | 69 |
| With Euclidean Distance : | 0.810780965067 |
| Connection's corresponding URL : | <http://www.linkedin.com/pub/andrew-cheng/34/887/848> |
| Connection's domain knowledge : | Information Security Cloud Computing E-commerce Enterprise Software Product Management IT Management PKI Strategy |

|  |  |
| --- | --- |
| Input Skill | IT Strategy Information Technology Cloud Computing IT Management Project Management Security IT Audit PKI Databases Internet Technologies |
| Connection # and farthest | 183 |
| With Euclidean Distance : | 1.41421356237 |
| Connection's corresponding URL : | <http://www.linkedin.com/pub/bryan-hsiao/51/754/215> |
| Connection's domain knowledge : | Strategic Planning Budgets PowerPoint Customer Service Microsoft Word English Research Outlook Public Speaking Negotiation Microsoft Office Microsoft Excel Windows Teaching |

The similarity distance of the other skill sets in the same cluster:

|  |  |  |
| --- | --- | --- |
| **Skills** | **Similarity** | **Cluster Text** |
| 1 | 3.7416 | Information Security Cloud Computing E-commerce Enterprise Software Product Management IT Management PKI Strategy |
| 12 | 4.8989 | Solution Selling Cloud Computing Enterprise Software Product Management Account Management Information Security Project Management E-commerce |
| 24 | 8.5440 | Testing Software Engineering PMP CMMI Software Project... Process Improvement Quality Assurance Software Quality... SDLC Project Management Linux Scrum Security Requirements Analysis Process Development Software Development... Agile Process Engineering Quality Management |

Note that the closest skills to the centroid is in fact very similar to the input skill sets:

**“IT Strategy Information Technology Cloud Computing IT Management Project Management Security IT Audit PKI Databases Internet Technologies”**

But the skill set that farthest from the cluster centroid looks very different to the test skill sets.

# Summary

By using the Linkedin API and scraping tool BeautifulSoup, it’s not difficult to scrape people’s information. And by calculating the distance between the different skill sets, we can tell whether two people’s skill sets are similar.

In addition, we can group skill sets to clusters using KMeans algorithm, and then we can tell what cluster the skill set belongs to, as well as what is one people’s skill set gap (i.e. the distance between the skill set to the centroid, compare to other skill sets).

# Appendix (Codes and data have also been uploaded to GitHub)

**Sample URLs (the first 20 lines)**

http://www.linkedin.com/in/moniquearnoux

http://www.linkedin.com/in/cioasia

http://www.linkedin.com/pub/richard-au/0/357/b73

http://www.linkedin.com/in/bakalis

http://www.linkedin.com/in/garrybarnes

http://www.linkedin.com/in/adamcbell

http://www.linkedin.com/pub/yan-benjamin-fcca-cpa-cfe-cams/20/253/392

http://www.linkedin.com/pub/robert-bergquist-cae/5/33/156

http://www.linkedin.com/in/napoleonbiggs

http://www.linkedin.com/pub/anita-billings/19/138/aa3

http://www.linkedin.com/pub/trish-blomfield/0/a09/87a

http://www.linkedin.com/in/allanboardman

http://www.linkedin.com/pub/robin-bradbeer/1/61/186

http://www.linkedin.com/pub/randy-brooks/6/327/998

http://www.linkedin.com/in/fredchankh

http://www.linkedin.com/in/brantburke

http://www.linkedin.com/pub/aurelie-busollo/13/538/12b

http://www.linkedin.com/in/hkgheadhunter

http://www.linkedin.com/in/geraldcai

<http://www.linkedin.com/in/pascalcaloc>

**Training Skills data (first 20 lines)**

E-commerce Online Travel Tourism Hotels Leisure Revenue Analysis Tour Operators Travel Management New Business Development Leisure Travel Hospitality Industry Hotel Management Contract Negotiation Travel Technology Negotiation Hospitality Management Hospitality Yield Management Business Travel Business Strategy Business Intelligence &... Business Needs Analysis C-Level Advisory Complex Mergers &...

Solution Selling Enterprise Software Security Major Accounts CRM VMware

E-commerce Online Marketing Start-ups Marketing Management Strategy Online Advertising

Information Security IT Audit CISM CISA Security Information Security...

Start-ups Entrepreneurship Mobile Applications Cloud Computing Mobile Internet Strategic Partnerships Mobile Marketing Integration Managed Services Wireless International Business Strategic Planning

Internal Audit Fraud Forensic Accounting Internal Controls Auditing FCPA Due Diligence Sarbanes-Oxley Act Consolidation Investigation Litigation Support Restructuring Risk Assessment

Program Management Governance Leadership Project Management Information Security Security

Digital Marketing Digital Media Social Media Mobile Marketing

Executive Coaching Workshop Facilitation Training Delivery Leadership Development Management Consulting Change Management Strategy

CISA Governance Information Security Risk Management CISM IT Audit

Robotics University Teaching Lecturing Research Science Teaching

Strategy Enterprise Software ITIL Data Center CRM Outsourcing Information Security

Strategy Recruiting Management Consulting Consulting

Executive Search Talent Management Recruiting Marketing Talent Acquisition Sourcing

Video Games Mobile Games Xbox 360 Game Development Social Games Online Gaming Photoshop

Executive Search Recruiting Business Development Networking Technical Recruiting Human Resources Telecommunications Information Technology Marketing Management

Business Development Product Development Sales Management Strategic Planning Market Development

Business Development Team Leadership Change Management Project Management Cloud Computing Executive Search Leadership Interviews Analytics

**Test Skills data**

IT Strategy Information Technology Cloud Computing IT Management Project Management Security IT Audit PKI Databases Internet Technologies **Program\_1.py - Connect to Linkedin and get the connections’ URLs**

"""

### GA Data Science Class Final Project

###

### Michael Yung

###

### program\_1.py

###

### First Linkedin program to output my connections' URL and my skills

###

### Input - n/a

### Output -

### Connections' URL - output\_connections\_url.csv

### Public URL - output\_public\_url.csv

### My Skills - output\_my\_skills.csv

"""

from linkedin import linkedin # pip install python-linkedin

from prettytable import PrettyTable # pip install prettytable

import json

"""

### Define CONSUMER\_KEY, CONSUMER\_SECRET,

### USER\_TOKEN, and USER\_SECRET from the credentials provided in LinkedIn application

"""

CONSUMER\_KEY = ' '

CONSUMER\_SECRET = ' '

USER\_TOKEN = ' '

USER\_SECRET = ' '

RETURN\_URL = '' # Not required for developer authentication

# Instantiate the developer authentication class

auth = linkedin.LinkedInDeveloperAuthentication(CONSUMER\_KEY, CONSUMER\_SECRET,

USER\_TOKEN, USER\_SECRET,

RETURN\_URL,

permissions=linkedin.PERMISSIONS.enums.values())

"""

### The meat starts here ...

"""

print "Connect to Linkedin ..."

app = linkedin.LinkedInApplication(auth)

print "Get own profile ..."

profile = app.get\_profile()

print "Get my connections ..."

connections = app.get\_connections()

"""

### Get the connections' URL

"""

print "Get the Public URL of all the connections and output to csv file ..."

public\_url = '/Users/michaelyung/DS\_HK\_1/finals/mikeyung/output\_public\_urls.csv'

f = open(public\_url, 'w')

public = app.get\_connections(selectors=['publicProfileUrl'])

for u in public["values"]:

if u.has\_key('publicProfileUrl'):

f.write(u['publicProfileUrl'] + '\n')

f.close()

"""

### My own data - skills using selectors

"""

print "Get my own skills and output to csv file ..."

my\_skills = app.get\_profile(selectors=['skills:(skill:(name))'])

result = my\_skills["skills"]["values"]

my\_skills\_data = '/Users/michaelyung/DS\_HK\_1/finals/mikeyung/output\_my\_skills.csv'

f = open(my\_skills\_data, 'w')

for rs in result:

f.write(rs["skill"]["name"] + '\n')

f.close()

**Program\_2.py – use BeautifulSoup to scrape each connections’ skill**

"""

### GA Data Science Class Final Project

###

### Michael Yung

###

### program\_2.py

###

### Python program to output my connections' skills by scraping their Linkedin public profile

### with BeautifulSoup

###

### Input - n/a

### Output -

### Connections' skills - output\_connections\_skills.csv

###

"""

import cookielib

import os

import urllib

import urllib2

import re

import string

from bs4 import BeautifulSoup

username = " "

password = " "

cookie\_filename = "parser.cookies.txt"

class LinkedInParser(object):

def \_\_init\_\_(self, login, password):

""" Start up... """

self.login = login

self.password = password

# Simulate browser with cookies enabled

self.cj = cookielib.MozillaCookieJar(cookie\_filename)

if os.access(cookie\_filename, os.F\_OK):

self.cj.load()

self.opener = urllib2.build\_opener(

urllib2.HTTPRedirectHandler(),

urllib2.HTTPHandler(debuglevel=0),

urllib2.HTTPSHandler(debuglevel=0),

urllib2.HTTPCookieProcessor(self.cj)

)

self.opener.addheaders = [

('User-agent', ('Mozilla/4.0 (compatible; MSIE 6.0; '

'Windows NT 5.2; .NET CLR 1.1.4322)'))

]

# Login

# self.loginPage()

skill = self.loadSkill()

self.cj.save()

def loadPage(self, url, data=None):

try:

if data is not None:

response = self.opener.open(url, data)

else:

response = self.opener.open(url)

return ''.join(response.readlines())

except:

# If URL doesn't load for ANY reason, try again...

# Quick and dirty solution for 404 returns because of network problems

# However, this could infinite loop if there's an actual problem

return self.loadPage(url, data)

def loginPage(self):

"""

Handle login. This should populate our cookie jar.

"""

login\_data = urllib.urlencode({

'session\_key': self.login,

'session\_password': self.password,

})

html = self.loadPage("https://www.linkedin.com/uas/login-submit", login\_data)

return

def loadSkill(self):

f\_url = open('/Users/michaelyung/DS\_HK\_1/finals/mikeyung/output\_public\_urls.csv', 'r')

f\_skill = open('/Users/michaelyung/DS\_HK\_1/finals/mikeyung/output\_connections\_skills.csv', 'w')

i = 1

for line in f\_url:

print i, line

html = self.loadPage(line)

soup = BeautifulSoup(html)

for s in soup.find\_all("span", "endorse-item-name-text"):

f\_skill.write(s.string + ' ')

f\_skill.write('\n')

i = i + 1

f\_skill.close()

f\_url.close()

return

parser = LinkedInParser(username, password)

**Program\_3.py – use Euclidean Norm to calculate shortest / farthest distances**

"""

### GA Data Science Class Final Project

###

### Michael Yung

###

### program\_3.py

###

### Python program to calculate Euclidean distance - cloest and farthest

###

### Input - Skill files and a new skill set (from my own profile)

### Output -

###

"""

#!/usr/bin/env python

"""

Tag cloud of the skill keywords (Javascript page)

"""

import scipy as sp

import sys

import numpy as np

import pandas as pd

from StringIO import StringIO

from sklearn.feature\_extraction.text import CountVectorizer

rootpath = '/Users/michaelyung/DS\_HK\_1/finals/mikeyung/'

skillpath = rootpath + 'output\_connections\_skills.csv'

urlpath = rootpath + 'output\_public\_urls.csv'

print "=== Read in public URLs for future referencing ..."

urlfile = open(urlpath, "r" )

allurls = []

for line in urlfile:

allurls.append( line.strip() )

urlfile.close()

print "=== Read in skill samples ..."

skillfile = open(skillpath, "r" )

allskills = []

for line in skillfile:

allskills.append( line.strip() )

skillfile.close()

print "=== Vectorize the skills ..."

vectorizer = CountVectorizer(min\_df=1)

X = vectorizer.fit\_transform(allskills)

num\_samples, num\_features = X.shape

print "=== Number of samples : ", num\_samples

print "=== Number of features : ", num\_features

my\_skills = "IT Strategy Information Technology Cloud Computing IT Management Project Management Security IT Audit PKI Databases Internet Technologies"

my\_skills\_vec = vectorizer.transform([my\_skills])

print "=== Read in my skills ..."

print " My skills : ", my\_skills

def dist(v1, v2):

delta = v1 - v2

return sp.linalg.norm(delta.toarray())

def dist\_norm(v1, v2):

v1\_normalized = v1/sp.linalg.norm(v1.toarray())

v2\_normalized = v2/sp.linalg.norm(v2.toarray())

delta = v1\_normalized - v2\_normalized

return sp.linalg.norm(delta.toarray())

print "=== Normalize the vector ..."

print "=== Calculate the Euclidean Distance ..."

best\_doc = None

best\_dist = sys.maxint

best\_i = None

worst\_dist = 0

worst\_i = None

for i in range(0, num\_samples):

skill = allskills[i]

if skill == my\_skills:

continue

skill\_vec = X.getrow(i)

d = dist\_norm(skill\_vec, my\_skills\_vec)

if d < best\_dist:

best\_dist = d

best\_i = i

if d > worst\_dist:

worst\_dist = d

worst\_i = i

print "==========="

print "Best skill match is connection # : ", best\_i

print " With Euclidean distance : ", best\_dist

print " Connection's domain knowledge : ", allurls[best\_i]

print " Connection's corresponding URL : ", allskills[best\_i]

print "==========="

print "Worst skill match is connection # : ", worst\_i

print " With Euclidean distance : ", worst\_dist

print " Connection's domain knowledge : ", allurls[worst\_i]

print " Connection's corresponding URL : ", allskills[worst\_i]

# print X.getrow(best\_i).toarray()

**Program\_4.py – Use KMeans to find clusters and see where the new skill falls in**

"""

### GA Data Science Class Final Project

###

### Michael Yung

###

### program\_4.py

###

### Python program to do clustering

###

### Input - Skill files and a new skill set (from my own profile)

### Output -

###

"""

#!/usr/bin/env python

import scipy as sp

import sys

import numpy as np

import pandas as pd

from StringIO import StringIO

from sklearn.feature\_extraction.text import CountVectorizer

import sklearn.datasets

from sklearn.cluster import KMeans

from sklearn.metrics.pairwise import euclidean\_distances

rootpath = '/Users/michaelyung/DS\_HK\_1/finals/mikeyung/'

skillpath = rootpath + 'output\_connections\_skills.csv'

"""

Read in all the skills

Learn the vocabulary dictionary and return the count vectors

"""

print "=== Reading in skill samples ..."

skillfile = open(skillpath, "r" )

allskills = []

for line in skillfile:

allskills.append( line.strip() )

skillfile.close()

#print allskills

#raw\_input("Press Enter to continue...")

"""

Vectorize the skills

X is array of word label and the count for each connection's skills

"""

print "=== Vectorizing the skills ..."

vectorizer = CountVectorizer(min\_df=1)

X = vectorizer.fit\_transform(allskills)

print "=== Skill Vector"

print X.toarray()

raw\_input("Press Enter to continue...")

num\_samples, num\_features = X.shape

print "=== Skill Vector Shape"

print "=== Number of samples : ", num\_samples

print "=== Number of features : ", num\_features

"""

Set number of clusters and start KMeans

Form clusters ...

"""

num\_clusters = 10

print "=== Grouping to clusters"

print "Number of clusters :", num\_clusters

km = KMeans(n\_clusters=num\_clusters, init='k-means++', n\_init=1, verbose=1)

km.fit(X)

raw\_input("Press Enter to continue...")

print "=== km.labels\_ : "

print km.labels\_

print "=== km.labels\_.shape : ", km.labels\_.shape

centroids = km.cluster\_centers\_

print "=== centroids : "

print centroids

print centroids.shape

"""

Input my skills to give check what cluster it belongs to

similar\_indices is the similar post in the same cluster

"""

my\_skills = "IT Strategy Information Technology Cloud Computing IT Management Project Management Security IT Audit PKI Databases Internet Technologies"

my\_skills\_vec = vectorizer.transform([my\_skills])

my\_skill\_label = km.predict(my\_skills\_vec)[0]

similar\_indices = (km.labels\_==my\_skill\_label).nonzero()[0]

print "=== Read in my skills ..."

print " My skills : ", my\_skills

zz = euclidean\_distances(centroids, my\_skills\_vec)

print "=== Distance to the 10 centroids"

print zz

print "=== Closet to cluster # :", my\_skill\_label

print "=== Skill sets in the same cluster :", similar\_indices

print "=== No. of skill sets in the same cluster :", similar\_indices.shape

"""

Similar skill sets in the same cluster

"""

similar = []

for i in similar\_indices:

dist = sp.linalg.norm((my\_skills\_vec - X[i]).toarray())

similar.append((dist, allskills[i]))

similar = sorted(similar)

show\_at\_1 = similar[0]

show\_at\_2 = similar[len(similar)/2]

show\_at\_3 = similar[-1]

print "=== Within the same cluster, the skillsets and distance"

print show\_at\_1

print show\_at\_2

print show\_at\_3