Practicum Update for Attain, LLC

By Weihang Wen, Mengchen Xiao, Xuan Yang, Weichao Zhu, Yinian Lyu

Master of Science in Business Analytics, Sep 2018,

George Washington University School of Business

An Update submitted to

The Faculty of

The School of Business

Of The George Washington University

In partial fulfillment of the requirements

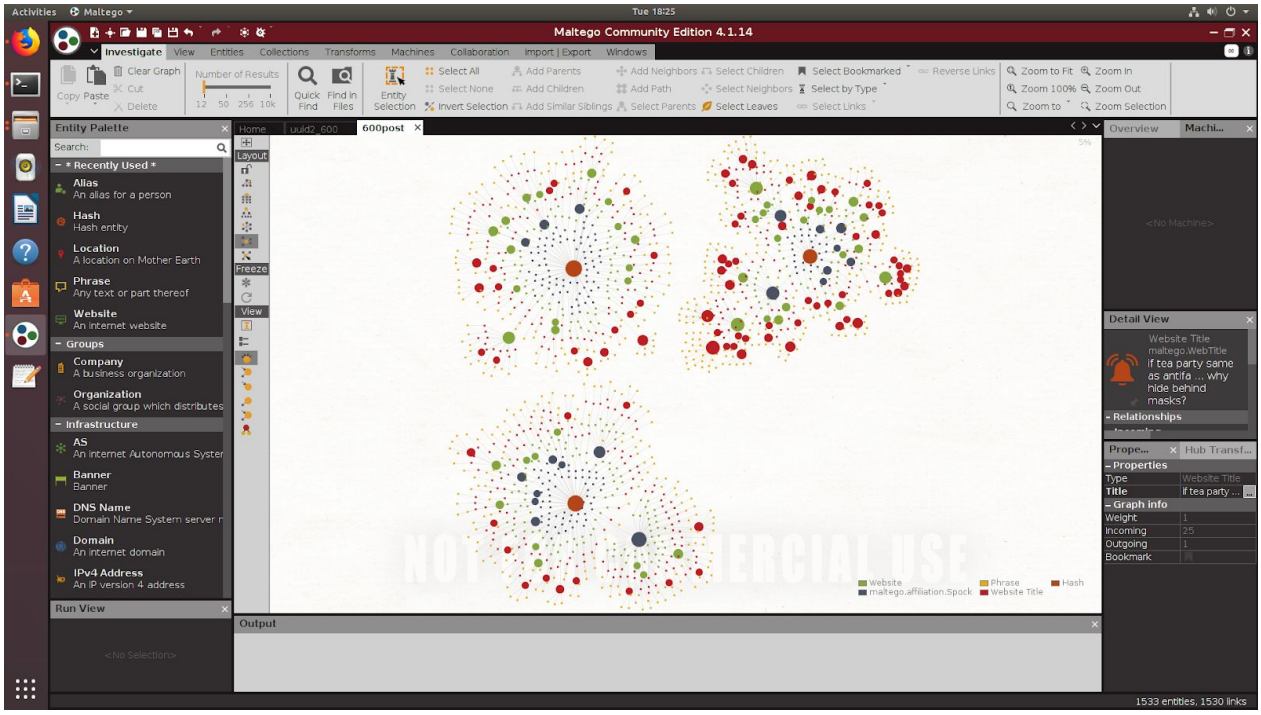
For the degree of Master of Science

In Business Analytics

Oct 16, 2018

1. Anti-semitism & Antifa trends across the sites

For the social network analysis platform part, we found a new and easy way to achieve the result which is the visualization of the json data from the Webhose in Maltego. Basically, what we are doing now is using the Local Transforms in Maltego which can transform python scripts to the configuration package that Maltego can use and then visualizing relationships between entities in the way we want. For now, we've visualized over 1000 entities and below screenshoot is our newest outcome. In short, three centers of the network are three site types, blogs, news and discussions. Outer nodes are sites like reddit.com, 4chan.org, freerepublic.com, etc. Next level of nodes are section titles under different sites for instance, /pol/ - Politically Incorrect - 4chan, The Donald - America First!, etc. The inferior nodes are different threads under section titles like "Trump Rally", "MSM blatantly lies about Antifa violently attacking Proud Boys. Skips over the fact that the three arrested were ANTIFA, not Proud Boys.", etc. The outest nodes are posts under different threads.

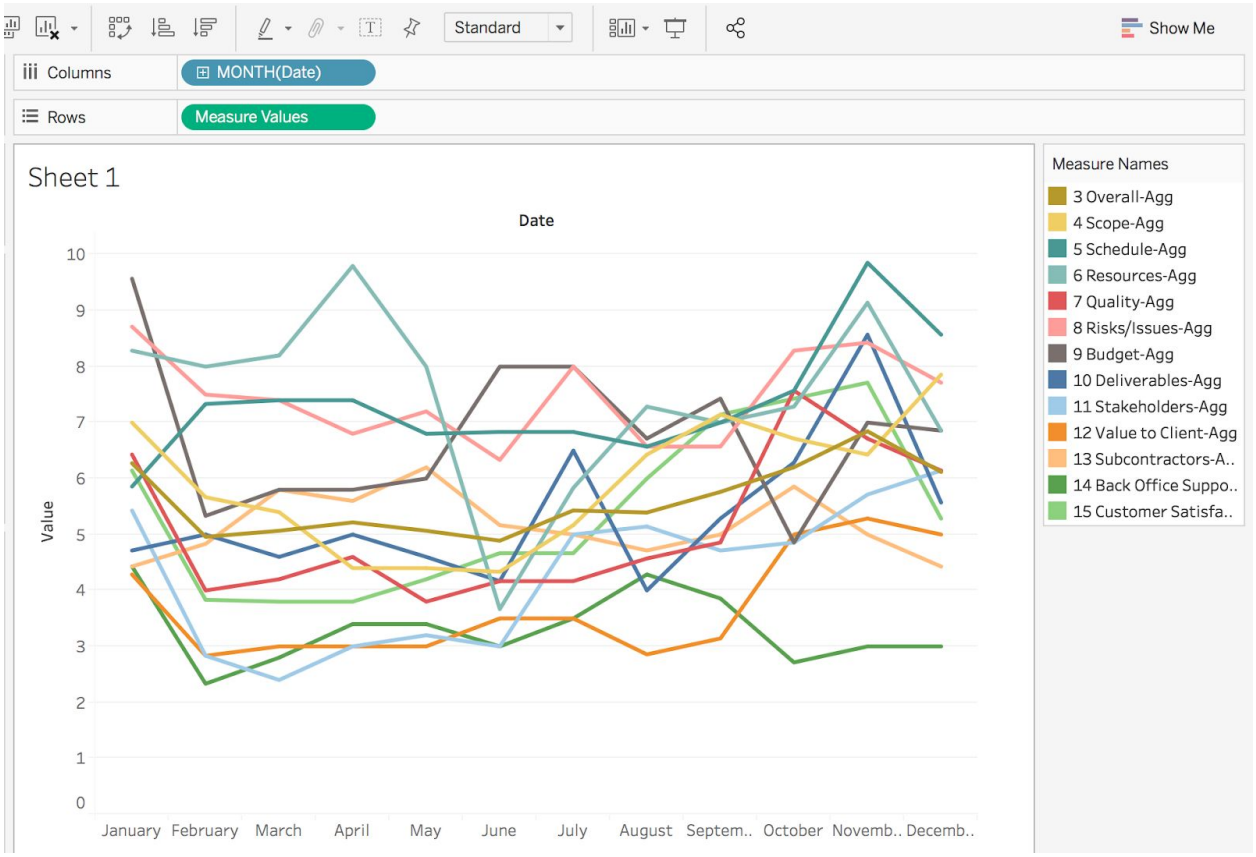


2. Risk data

Speaking of the Risk Data Analysis, we got 8 seperated project dataset and each has 13 measure factors such as schedule, quality, budget, etc., indicating potential dangers criticality level from 0-25 over time.

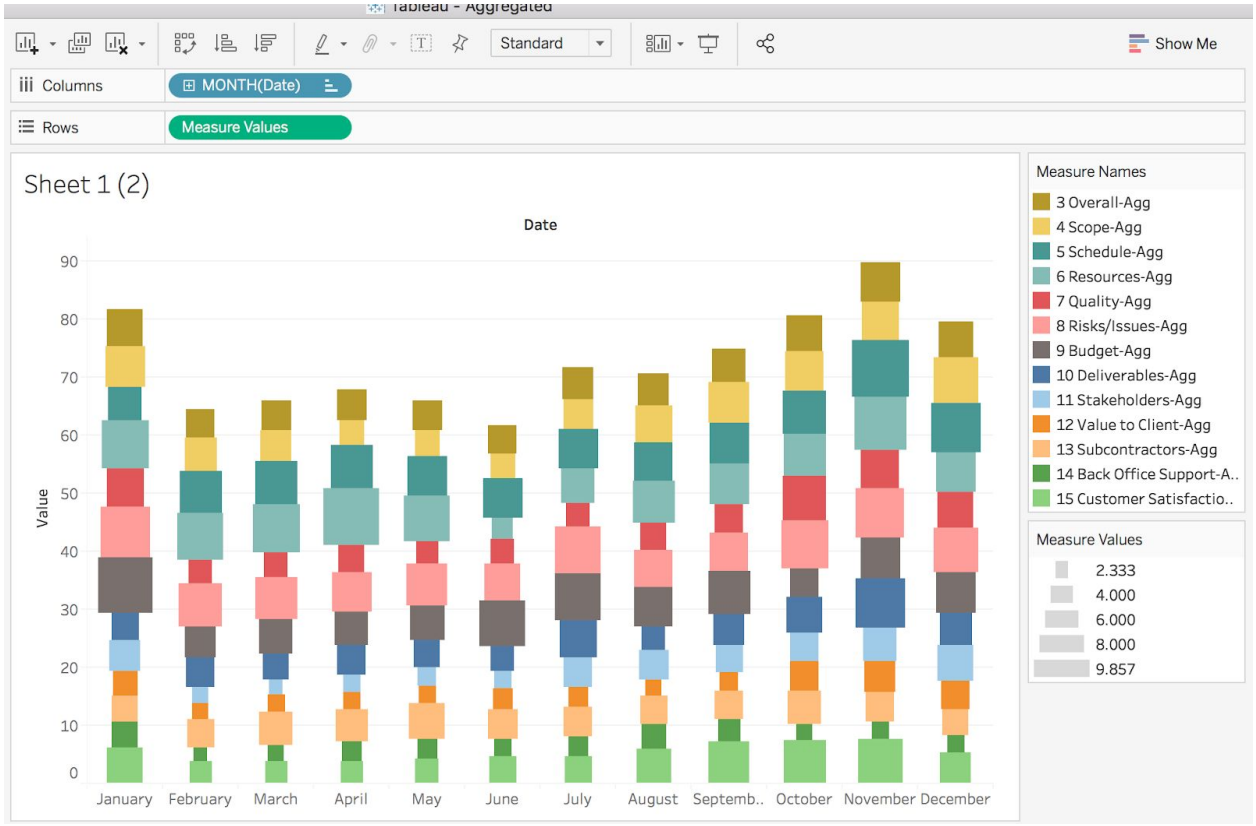
2.1 Descriptive analysis

For descriptive analysis, we utilize aggregated values of each factor for plotting trending lines across all months. Specifically, we preprocess average of each factor for every same month across all projects. Then we could see in general the trending fluctuation of each factor overtime. According to the result, we found, in general, project risk getting worse in winter focusing in November, December and January. On the contrary, May, June and Febrary are generally likely to have smooth project going. See Graph 2.1.1



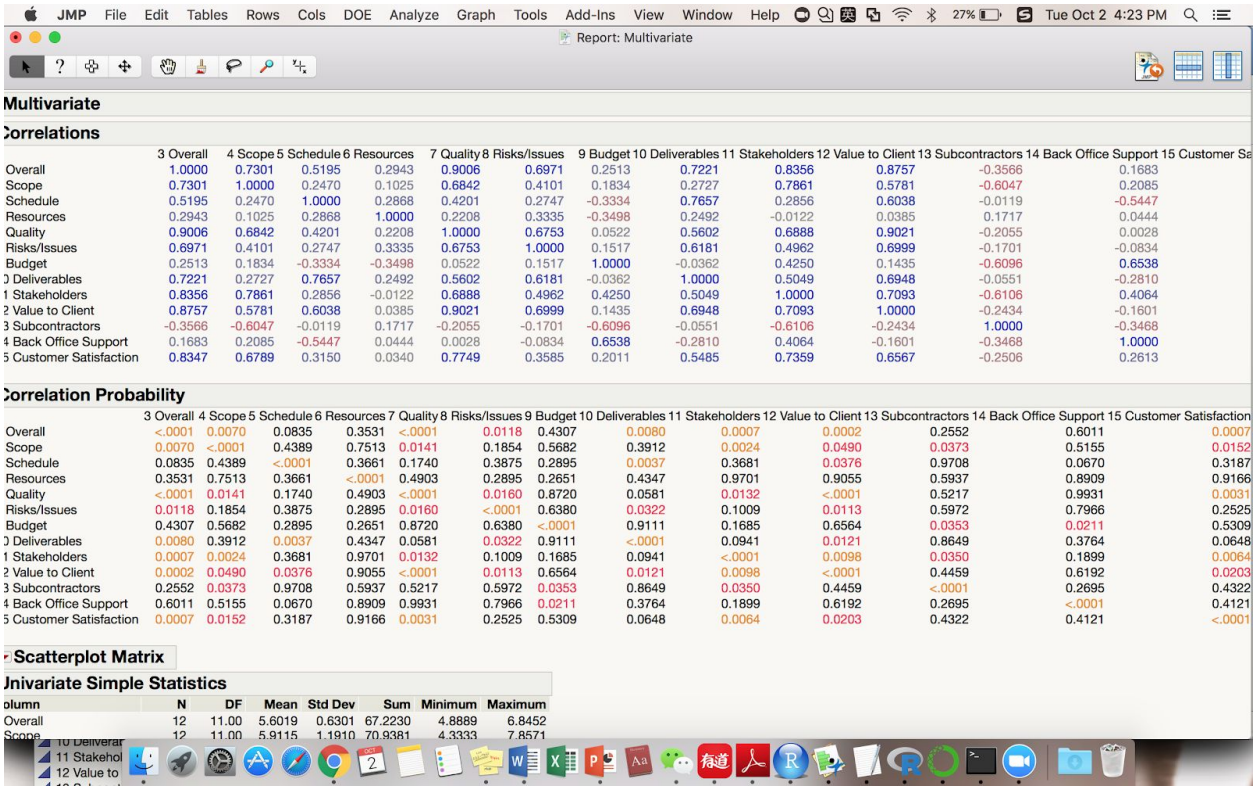
Graph 2.1.1

Also we would like to show the percentage of each factor accounting for total in every month. So we also visualized aggregated bar plot over time. According to the plot shows, the highest level of criticality is 9.857 and the lowest level of criticality is 2.333. Besides, Budget and Resource are generally the most dangerous factors which we should pay more attention to. See Graph 2.1.2.



Graph 2.1.2

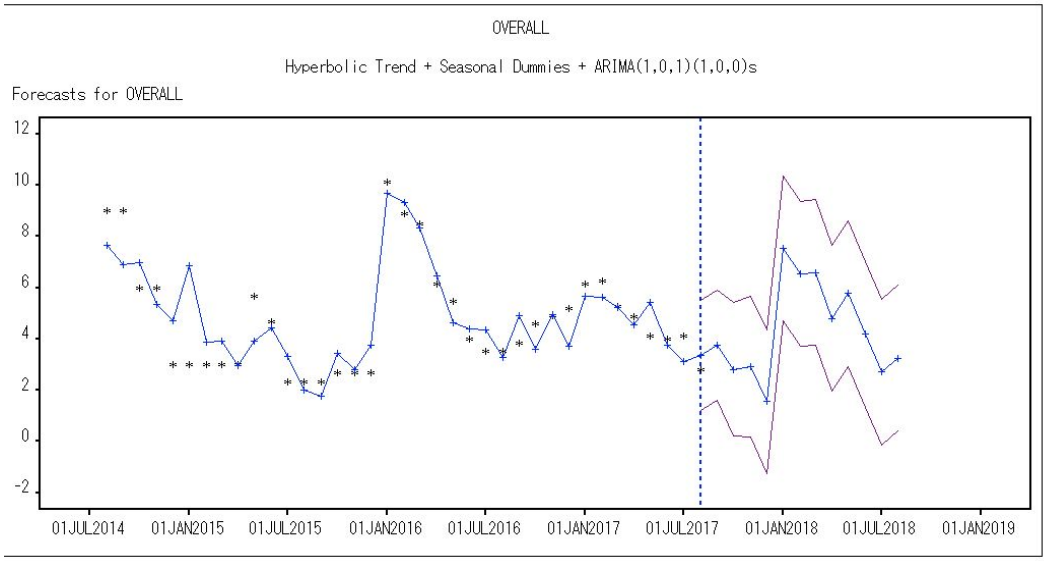
In addition, we are also curious about the multicollinearity of these measure factors. So we also conduct correlation analysis by binary table as well as statistical probability to see how strong of their positive or negative relationships. See Graph 2.1.3.



Graph 2.1.3

2.2 Time Series Predictive Analysis

For the risk data, we has the criticality for 8 different projects overtime and each project has the same 13 risk factors such as resources and quality. Last time, we wanted to see if there was a pattern in the risk data and those risks could be predicted by a model. But the time series models for the overall risk and risks for different projects we got had a low R square. Therefore, we made some improvements for the models such as adding seasonal dummies, different kind of trends and ARMA model for seasonals. As we expected, the R square increased a lot, which is more than 0.7. Here are the results from SAS for the overall risk and the resources risk. For example, the resources risk factor, the R Square is even more than 0.8. The R square increased and the error decreased significantly after we use the hyperbolic trend.

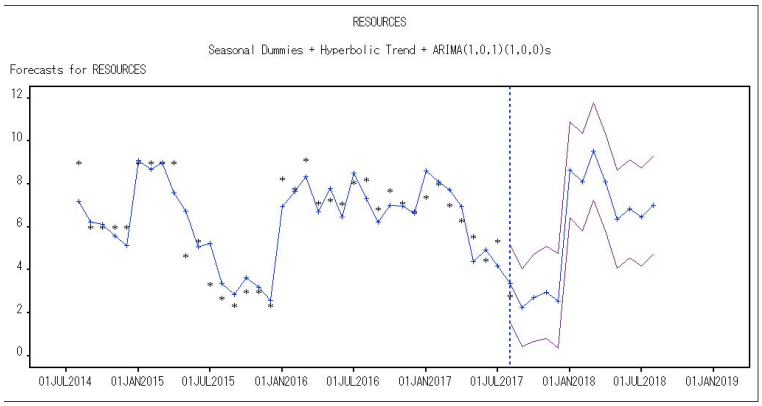


Graph 2.2.1

Statistics of Fit
OVERALL
Hyperbolic Trend + Seasonal Dummies + ARIMA(1,0,1)(1,0,0)s

Statistic of Fit	Value
Mean Square Error	1.18436
Root Mean Square Error	1.08829
Mean Absolute Percent Error	20.56570
Mean Absolute Error	0.81479
R-Square	0.731

Graph 2.2.2



Graph 2.2.3

Statistics of Fit
RESOURCES
Seasonal Dummies + Hyperbolic Trend + ARIMA(1,0,1)(1,0,0)s

Statistic of Fit	Value
Mean Square Error	0.71181
Root Mean Square Error	0.84369
Mean Absolute Percent Error	12.36031
Mean Absolute Error	0.66160
R-Square	0.839

Graph 2.2.4

3. Resume Text Mining

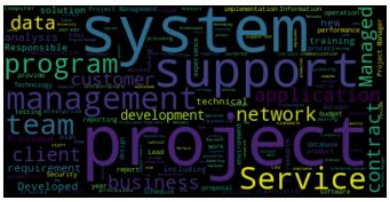
For the resume files from Attain, we did some basic text mining exploration like word count and word cloud visualization, and text transformation using TF-IDF on the given resumes and have been trying to working on the topic clustering based on LDA model.

```
In [26]: %pylab inline
         import matplotlib.pyplot as plt
         plt.imshow(wordcloud, interpolation='bilinear')
         plt.axis("off")

         Populating the interactive namespace from numpy and matplotlib

         /anaconda3/lib/python3.6/site-packages/IPython/core/magics/pylab.py:160: UserWarning: pylab import has clobbered these variables: ['f']
         `%matplotlib` prevents importing * from pylab and numpy
           "\n`%matplotlib` prevents importing * from pylab and numpy"

Out[26]: (-0.5, 399.5, 199.5, -0.5)
```



Graph 3.1

See Graph 3.1. Here is the word cloud generated from all the resume files, showing the frequency of the words in the resumes of Attain.

We then did some basic text preprocessing like tokenizer, stemmer, removing stopwords and common punctuations so that the text files are much cleaner for further analysis. See Graph 3.2 and Graph 3.3

```
In [9]: # Clean non-alphabetic characters and convert to lowercase
        clean_line = ''
        with open('Attain Resumes ao .txt', 'r') as in_file:
            with open('resumes_clean.txt', 'w') as out_file:
                for j, line in enumerate(in_file):
                    for character in line:
                        if character.isalpha() or character.isspace():
                            clean_line += character
                        clean_line = clean_line.lower()
                    out_file.write(clean_line)
                    clean_line = ''
                    print ('Line %i/%i cleaned ...' % (j+1, nlines))
        print ('Done.')

        Line 1/10459 cleaned ...
        Line 2/10459 cleaned ...
        Line 3/10459 cleaned ...
        Line 4/10459 cleaned ...
        Line 5/10459 cleaned ...
        Line 6/10459 cleaned ...
        Line 7/10459 cleaned ...
        Line 8/10459 cleaned ...
        Line 9/10459 cleaned ...
        Line 10/10459 cleaned ...
        Line 11/10459 cleaned ...
        Line 12/10459 cleaned ...
        Line 13/10459 cleaned ...
        Line 14/10459 cleaned ...
        Line 15/10459 cleaned ...
        Line 16/10459 cleaned ...
        Line 17/10459 cleaned ...
```

Graph 3.2

```
In [27]: #insert a file including stopwords we indicate
         file = open("stopwords.csv",'r')
         stop_words = file.read()

In [32]: print(stop_words)

         above
         after
         again
         against
         all
         am
         an
         and
         any
         are
         aren't
         as
         at
         be
         because
         been
         before
         being
         below

In [42]: #removing stopwords
         text_data= [w for w in words if not w in stop_words]
```

Graph 3.3

Here are the 20 most common words from the resume files. See Graph 3.4.

```
In [36]: from collections import Counter
         words_counter=Counter(words_stem1)
         words_counter.most_common(20)

Out[36]: [('manag', 1893),
          ('project', 1047),
          ('develop', 900),
          ('system', 809),
          ('support', 782),
          ('team', 528),
          ('D', 528),
          ('servic', 524),
          ('provid', 488),
          ('busi', 485),
          ('program', 443),
          ('implement', 429),
          ('applic', 387),
          ('contract', 377),
          ('includ', 376),
          ('&', 374),
          ('respons', 365),
          ('secur', 354),
          ('perform', 352),
          ('train', 345)]
```

Graph 3.4

Here are some common collocations from resume files. See Graph 3.5.

```
In [38]: word_text1.collocations(num=50, window_size=2)

project manag; profession experi; life cycl; public honor; honor
award; best practic; help desk; educ train; public award; award
achiev; qualiti assur; activ directori; problem solv; program manag;
attain llc; task order; marybeth peter; elizabeth morin-kensicki;
inform technolog; inform system; earn valu; end user; team member;
subject matter; educ certiic; west virginia; train associ; unit state;
kansa citi; strateg plan; resolv issu; educ certif; univers maryland;
saurabh rohatgi; north carolina; abbey mcmanu; key accomplish; mr.
patterson; window server; bachelor scienc; year experi; eric
schonbachl; certiic train; intern extern; susan martin; deliveri
manag; kent gring; wilson ndeh; michel muse; air forc
```

Graph 3.5

Here is another exploratory analysis on the resume files that could be useful when we want to analyze some specific words. See Graph 3.6.

```
In [41]: word_text1.concordance("skill",lines=40)

Displaying 40 of 97 matches:
gaynor success util conflict resolut skill improv team 's moral confid trust li
m tdwr jaw aswon wsd develop coordin skill set train document qar team charter
continu grow dedic local team highli skill technic manag consult alreadi activ
custom interperson team build mentor skill compet includ equip servic manag pro
it 2 million aciv duti famili member skill project/transiion/integraion manag -
si owner 15 year strong technic code skill overal busi sens work histori tricar
ommun of-sit toxicologist supervisor skill geospati experienc spatial non-spati
30 student order reinforc test- take skill well grade test quizz essay creat di
ganiz object possess broad-bas manag skill with strong plan commun and decision
team member guest except interperson skill achiev compani rank 7 450 also achie
year self-motiv excel technic commun skill achiev qualiti result lead work mult
 templat perform team profici requir skill set necessari alm perform test tool
L & P test methodolog profici requir skill set necessari alm perform test tool
viron provid access inform technolog skill network machin center optim output h
activ sixty-two unit auto worker uaw skill tradesmen fortun 500 compani maintai
ley associ profession develop commun skill certif learn tree intern profession
 experi editor I possess excel write skill special technic propos rfp analysi q
expand capabl hire profession divers skill streamlin public process appli six s

In [33]: word_text1.common_contexts(["develop"],num=10)

applic_methodolog agil_softwar enhanc_mission increment_continu test-
driven_within softwar_qualiti manag_new applic_horac schedul_custom
progress_effort

In [34]: word_text1.common_contexts(["skill"],num=10)

resolut_improv coordin_set highli_technic mentor_compet
member_project/transiion/integraion code_overal supervisor_geospati
take_well manag_with interperson_achiev
```

Graph 3.6

After resume data clean and data exploration, we next step would like to implement text transformation. We referred to TF-IDF theory. This theory could be divided into two concept: one is importance and the other is frequency. TF stands for term frequency, which equals to count(word) / len(document); IDF stand for inverse document frequency namely term importance, which formulated by log(total number of document / count(document_containing_term)).
Then we define TF-IDF function in python and apply to resume data. See Graph 3.7. In the graph, we could find the most feature word of each resume order by value of TF-IDF from high to low.

```
In [21]: sortedtfidf1 = sorted(tfidf1.items(), key=itemgetter(1), reverse = True)
         sortedtfidf1

Out[21]: [('clarity', 0.0021072099696478686),
          ('morinkensicki', 0.0010536049848239343),
          ('elizabeth', 0.0006020599913279624),
          ('martin', 0.0004214419939295737),
          ('susan', 0.0004214419939295737),
          ('kensicki', 0.0003612359947967774),
          ('tricare', 0.0003311329952303793),
          ('siemens', 0.0003311329952303793),
          ('shellie', 0.0003010299956639812),
          ('scientiic', 0.0003010299956639812),
          ('biology', 0.0003010299956639812),
          ('agee', 0.00027092699609758305),
          ('attain&', 0.00027092699609758305),
          ('ism', 0.00027092699609758305),
          ('pms', 0.00027092699609758305),
          ('gaynor', 0.00024082399653118497),
          ('horace', 0.00024082399653118497),
          ('division', 0.00024082399653118497),
          ('locations', 0.00021072099696478684),

In [22]: sortedtfidf2 = sorted(tfidf2.items(), key=itemgetter(1), reverse = True)
         sortedtfidf2

Out[22]: [('nortel', 0.0006923689900271568),
          ('salesforce', 0.0006020599913279624),
          ('fundraising', 0.0004515449934959718),
          ('database', 0.00039133899436317554),
          ('jafer', 0.0003612359947967774),
          ('telecommunications', 0.0003311329952303793),
```

Graph 3.7