Wikipedia miner interface

October 2, 2014

1 Wikipedia miner interface

This notebook describes the process used to extract 'key' words from a novel (using wikipedia miner) and extracting relative interest from particular countries (namely, the Netherlands and Canada).

```
In [1]: import pycurl
        import json
        from StringIO import StringIO
        from urllib import urlencode
```

```
In [2]: chunk = "We want to test our team play before going up against the boys of " + \
                "Keyport High, that's a fact; and Scranton can put up a hard fighting " + \
                "bunch of irregulars.  There are some mighty clever hockey players " + \
                "in and out of the high school, who are not on our Seven.  I guess " + \
                "there ought to be a pretty lively game on Saturday; and there will " + \
                "be if several fellows I could mention line up against us."
```

```
In [3]: wikifyURL = 'http://wikipedia-miner.cms.waikato.ac.nz/services/wikify'

        c = pycurl.Curl()
        c.setopt(c.URL, wikifyURL)
        buffer = StringIO()

        post_data = {'responseFormat': 'json',
                     'source': chunk}

        # Form data must be provided already urlencoded.
        postfields = urlencode(post_data)
        # Sets request method to POST,
        # Content-Type header to application/x-www-form-urlencoded
        # and data to send in request body.
        c.setopt(c.POSTFIELDS, postfields)
        c.setopt(c.WRITEFUNCTION, buffer.write)
        c.perform()
        c.close()

        body = buffer.getvalue()
        response = json.loads(body)
```

```
In [4]: for topic in response['detectedTopics']:
            print topic['title'], '%4.3f '%topic['weight']
```

```
Scranton, Pennsylvania 0.764
High school 0.706
```

```
In [81]: def wikify(chunk, prob=0.5):
             wikifyURL = 'http://wikipedia-miner.cms.waikato.ac.nz/services/wikify'

             c = pycurl.Curl()
             c.setopt(c.URL, wikifyURL)
             buffer = StringIO()

             post_data = {'responseFormat': 'json',
                          'minProbability': prob,
                          'source': chunk}

             # Form data must be provided already urlencoded.
             postfields = urlencode(post_data)
             # Sets request method to POST,
             # Content-Type header to application/x-www-form-urlencoded
             # and data to send in request body.
             c.setopt(c.POSTFIELDS, postfields)
             c.setopt(c.WRITEFUNCTION, buffer.write)
             c.perform()
             c.close()

             body = buffer.getvalue()
             response = json.loads(body)
             return response
```

```
In [6]: chunk = '"Well," commented Thad, "we all know that Nick is a boss skater, even ' + \
            'on the old runners he sports, and which mebbe his dad used before ' + \
            "him, they're that ancient.  He can hold his own with the next one " + \
            "whenever there's any ice worth using.  And as to hockey, why, if Nick " + \
            'would only play fair, which he never will, it seems because his ' + \
            'nature must be warped and crooked, he could have a leading place on ' + \
            'our Seven.  As it is, the boys refused to stand for him in any game, ' + \
            'and so he had to herd with the scratch players.  Even then Mr. ' + \
            'Leonard, our efficient coach and trainer, has to call him down good ' + \
            "and hard for cheating, or playing off-side purposely.  It's anything " + \
            'to win, with Nick."'

        response = wikify(chunk, 0.01)

        for topic in response['detectedTopics']:
            print topic['title'], '%4.3f '%topic['weight']
```

```
Coach (sport) 0.444
Herd 0.340
Sport 0.332
Horse trainer 0.274
Ice 0.260
Fair 0.223
Hockey 0.182
Offside (association football) 0.170
Boss (video gaming) 0.131
Cheating 0.093
Shilling 0.048
```

```
In [68]: chapter = '../wikipedia-miner-1.2.0/data/books/pg13250_ch1.txt'
In [82]: def processChunk(chunk, words):
             try:
                 response = wikify(chunk, 0.005)
                 for topic in response['detectedTopics']:
                     words.append((topic['title'], topic['weight']))
             except Exception as e:
                 print 'Cannot wikify! Problematic chunk: ',chunk
             return words

In [83]: def makeParagraphChunks(filename):
             with open(filename,'r') as fin:
                 chunk = ''
                 for line in fin:
                     if len(line.strip())==0:
                         yield chunk
                         chunk = ''
                     else:
                         chunk += ' ' + line.strip()

         def makeSentenceChunks(filename):
             for para in makeParagraphChunks(filename):
                 para = para.replace('!', '.')
                 para = para.replace('?', '.')
                 para = para.replace('"', '')
                 sentences = para.split('.')
                 for s in sentences:
                     s = s.strip()
                     if len(s)>0:
                         yield s

         def processText(inFile, chunker):
             for chunk in chunker(inFile):
                 for word in processChunk(chunk, []):
                     yield word

         def saveWords(filename, words):
             with open(filename,'w') as fout:
                 for word,prob in words:
                     if ',' in word:
                         word = '"' + word + '"'
                     fout.write('%s,%.2f%%\n'%(word,prob*100))

In [112]: wordsPara = []
          for chunk in makeParagraphChunks(chapter):
              wordsPara = processChunk(chunk, wordsPara)

In [113]: len(wordsPara)

Out[113]: 182

In [115]: wordsSent = []
          for chunk in makeSentenceChunks(chapter):
              wordsSent = processChunk(chunk, wordsSent)

In [116]: len(wordsSent)
```

```
Out[116]: 203

In [5]: import glob

In [19]: chunkers = {
             'paragraph': makeParagraphChunks,
             'sentence': makeSentenceChunks
         }

         for inFile in glob.glob('../wikipedia-miner-1.2.0/data/books/pg13250_*.txt'):
             print 'Processing ' + inFile + '...'
             for chunkType in chunkers:
                 outFile = inFile
                 outFile = outFile.replace('books/', 'books/pyAnnotations/')
                 outFile = outFile.replace('.txt', '_' + chunkType + '.txt')
                 wordGenerator = processText(inFile, chunkers[chunkType])
                 saveWords(outFile, wordGenerator)

Processing ../wikipedia-miner-1.2.0/data/books/pg13250_ch2.txt...
Cannot wikify! Problematic chunk:  exclaimed Thad, you're referring to his _Les Miserables_, I guess
Processing ../wikipedia-miner-1.2.0/data/books/pg13250_ch5.txt...
Processing ../wikipedia-miner-1.2.0/data/books/pg13250_ch3.txt...
Cannot wikify! Problematic chunk:  Hugh at first flush felt indignant
Processing ../wikipedia-miner-1.2.0/data/books/pg13250_ch7.txt...
Cannot wikify! Problematic chunk:   "Well, the old man went through a bitter experience many years ago,
Cannot wikify! Problematic chunk:  Well, the old man went through a bitter experience many years ago, Tl
Processing ../wikipedia-miner-1.2.0/data/books/pg13250_ch6.txt...
Processing ../wikipedia-miner-1.2.0/data/books/pg13250_ch1.txt...
Processing ../wikipedia-miner-1.2.0/data/books/pg13250_ch4.txt...

In []:

In [5]: import pandas as pd
        import numpy as np

In [9]: # dataFile = '../wikipedia-miner-1.2.0/data/books/pyAnnotations/pg13250_ch1_paragraph.txt'
        dataFile = 'data/books/pyAnnotations/pg13250_ch1_paragraph.txt'

        data = pd.read_csv(dataFile,names = ['Word','Probability'], quotechar='"')
        data['Probability'] = data['Probability'].apply(lambda x: np.double(x.replace('%', '')))

In [11]: data[:10]

Out[11]:                       Word  Probability
         0       Mercury (element)        88.04
         1             Thermometer        75.48
         2              Coming out        23.63
         3               Cold wave        42.25
         4                    Wave        26.85
         5  Scranton, Pennsylvania        24.08
         6                  Second        16.08
         7                   Feels         4.80
         8                    Pond        49.14
         9                Baseball        39.97

In [132]: byWord = data.groupby('Word')
          words = data['Word'].unique()
        data2 = pd.DataFrame(words, index=words, columns=['Word'])
        data2['Probability'] = byWord.mean()
        data2['Count'] = byWord.size()
        data2[:10]
```

```
Out[132]:                                              Word  Probability  Count
        Mercury (element)           Mercury (element)      88.0400      1
        Thermometer                       Thermometer      39.5300      2
        Coming out                         Coming out      23.6300      1
        Cold wave                           Cold wave      42.2500      1
        Wave                                     Wave      26.8500      1
        Scranton, Pennsylvania  Scranton, Pennsylvania      40.5080     10
        Second                                 Second      12.1900      6
        Feels                                   Feels       4.8000      1
        Pond                                     Pond      58.3650      2
        Baseball                             Baseball      40.4575      4
```

```
In [2]: import btb.utils.tools as btbtools
        import btb.utils.wikiquery as wq

        import mwclient
        from __future__ import division

        wiki = mwclient.Site('en.wikipedia.org')
        bots = wq.getAllBots(wiki)
```

```
{'augroup': 'bot'}
```

```
In [3]: def wikiCountryInterest(pageTitle):
            ips, usrs, nrevs = wq.getContributionsForPage(wiki, pageTitle)
            knwRevs, conf, nIP, nUsr, nBot, nUnkn = btbtools.prepareData(ips, usrs, bots)
            expEdits = wq.getTotalContributions()

            cmpEdits = btbtools.compareEdits(expEdits, knwRevs)

            nl_e, nl_o, nl_m = cmpEdits['NL']
            ca_e, ca_o, ca_m = cmpEdits['CA']
            return (nl_o, ca_o, conf)
```

```
In [157]: data2['TEMP'] = data2['Word'].apply(lambda x: wikiCountryInterest(x))
          data2['NLO'] = data2['TEMP'].apply(lambda x: x[0])
          data2['CAO'] = data2['TEMP'].apply(lambda x: x[1])
          data2['Conf'] = data2['TEMP'].apply(lambda x: x[2])
          del data2['TEMP']
```

```
In [186]: data2
```

```
Out[186]: <class 'pandas.core.frame.DataFrame'>
          Index: 134 entries, Mercury (element) to Morgan Motor Company
          Data columns (total 6 columns):
          Word           134  non-null values
          Probability    134  non-null values
          Count          134  non-null values
          NLO            134  non-null values
          CAO            134  non-null values
          Conf           134  non-null values
          dtypes: float64(4), int64(1), object(1)
```

```
In [158]: data2.to_pickle('data2.pkl')

In [6]: data2 = pd.read_pickle('data2.pkl')

In [7]: data2.ix[:,['NLO', 'CAO', 'Conf']][:10]

Out[7]:                              NLO       CAO      Conf
        Mercury (element)       0.002453  0.089348  0.568526
        Thermometer             0.008873  0.157054  0.606566
        Coming out              0.007092  0.065603  0.436533
        Cold wave               0.000000  0.085227  0.553459
        Wave                    0.001875  0.042500  0.551154
        Scranton, Pennsylvania  0.003470  0.027065  0.484370
        Second                  0.007895  0.092105  0.385005
        Feels                   0.009091  0.136364  0.495495
        Pond                    0.000000  0.085427  0.546203
        Baseball                0.010437  0.076078  0.468537

In [8]: expEdits = wq.getTotalContributions()
        NLE = expEdits['NL']
        CAE = expEdits['CA']

        data2['Interest-NL'] = data2.apply(lambda x: btbtools.relativeInterest(NLE, x['NLO']), axis=1)
        data2['Interest-CA'] = data2.apply(lambda x: btbtools.relativeInterest(CAE, x['CAO']), axis=1)

In [9]: data2.ix[:,['NLO', 'CAO', 'Interest-NL', 'Interest-CA']][:10]

Out[9]:                              NLO       CAO  Interest-NL  Interest-CA
        Mercury (element)       0.002453  0.089348    -0.693413     0.395624
        Thermometer             0.008873  0.157054     0.098400     0.656169
        Coming out              0.007092  0.065603    -0.113475     0.176865
        Cold wave               0.000000  0.085227    -1.000000     0.366400
        Wave                    0.001875  0.042500    -0.765625    -0.212963
        Scranton, Pennsylvania  0.003470  0.027065    -0.566273    -0.498805
        Second                  0.007895  0.092105    -0.013158     0.413714
        Feels                   0.009091  0.136364     0.120000     0.604000
        Pond                    0.000000  0.085427    -1.000000     0.367882
        Baseball                0.010437  0.076078     0.233474     0.290202

In [10]: nItems = data2['Count'].sum()
         data2['Total-NL'] = data2.apply(lambda x: x['Probability'] * x['Count'] / nItems * x['Conf'] *
         data2['Total-CA'] = data2.apply(lambda x: x['Probability'] * x['Count'] / nItems * x['Conf'] *

In [11]: data2.ix[:,['Total-NL', 'Total-CA']][:10]

Out[11]:                         Total-NL  Total-CA
         Mercury (element)      -0.190700  0.108803
         Thermometer             0.025927  0.172894
         Coming out             -0.006431  0.010024
         Cold wave              -0.128482  0.047076
         Wave                   -0.062253 -0.017316
         Scranton, Pennsylvania -0.610482 -0.537746
         Second                 -0.002036  0.064010
         Feels                   0.001568  0.007893
         Pond                   -0.350320  0.128877
         Baseball                0.097268  0.120902
```

```
In [12]: %pylab inline
```

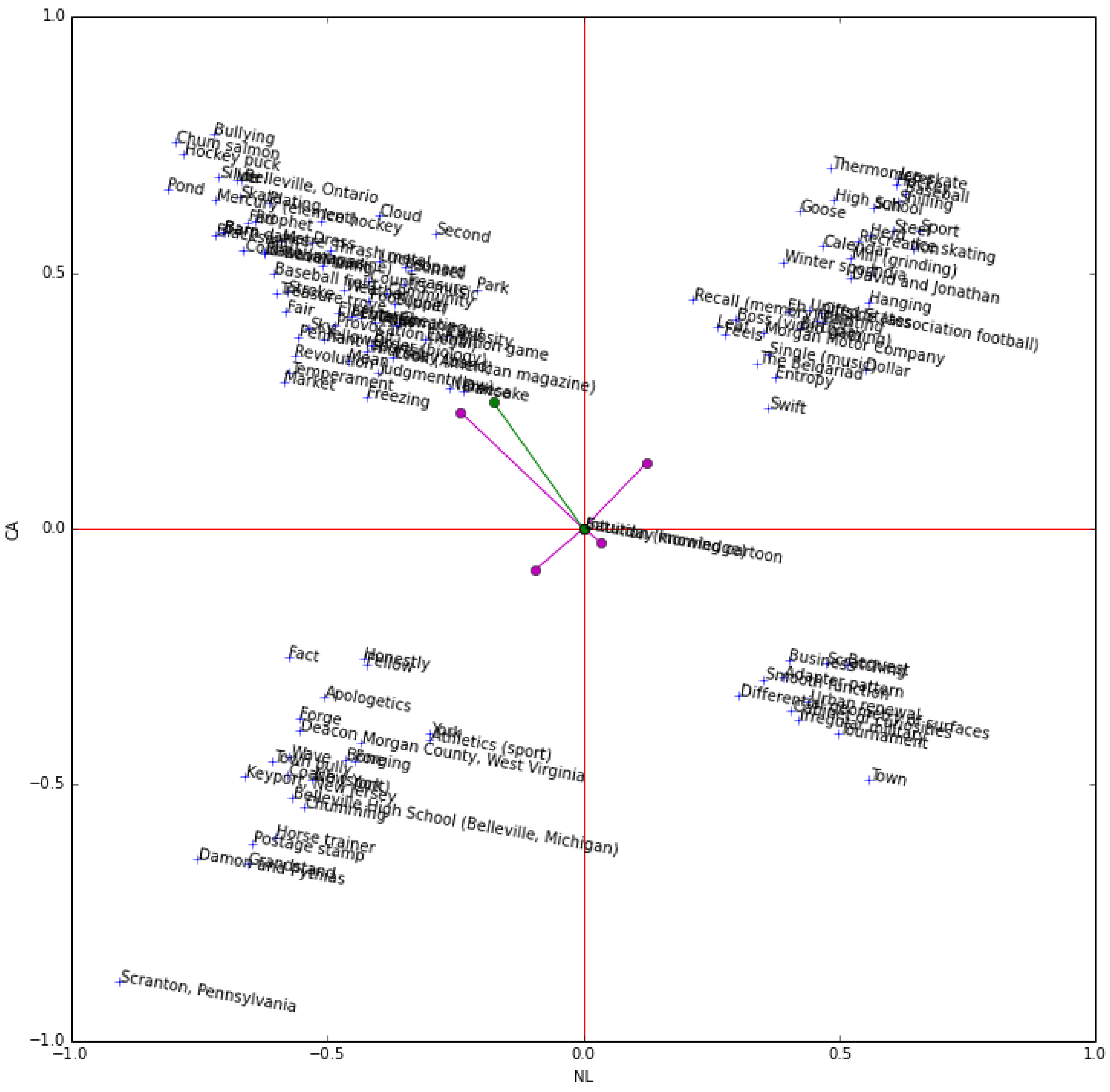
```
Populating the interactive namespace from numpy and matplotlib
```

```
In [13]: def explode(x, level):
             exp = 1/level
             if x<0:
                 return -((-x)**exp)
             else:
                 return x**exp
```

```
In [14]: data3 = data2.copy()
         data3['NL'] = data3['Total-NL'].apply(lambda x: explode(x, 5))
         data3['CA'] = data3['Total-CA'].apply(lambda x: explode(x, 5))

         del data3['Probability']
         del data3['Count']
         del data3['NL0']
         del data3['CA0']
         del data3['Conf']
         del data3['Interest-NL']
         del data3['Interest-CA']
         del data3['Total-NL']
         del data3['Total-CA']
```

```
In [21]: figure(figsize=(12,12))

         plot(data3['NL'], data3['CA'],'+')

         for nl,ca,txt in zip(data3['NL'], data3['CA'], data3['Word']):
             annotate(txt, (nl,ca), rotation=-10)

         plot([-1, 1], [0, 0] , 'r-')
         plot([0, 0] , [-1, 1],'r-')

         x = len(data3)

         data4 = data3[(data3['NL']>0) & (data3['CA']>0)]
         plot([0, data4['NL'].sum()/x], [0, data4['CA'].sum()/x], 'm-o')
         data4 = data3[(data3['NL']>0) & (data3['CA']<0)]
         plot([0, data4['NL'].sum()/x], [0, data4['CA'].sum()/x], 'm-o')
         data4 = data3[(data3['NL']<0) & (data3['CA']>0)]
         plot([0, data4['NL'].sum()/x], [0, data4['CA'].sum()/x], 'm-o')
         data4 = data3[(data3['NL']<0) & (data3['CA']<0)]
         plot([0, data4['NL'].sum()/x], [0, data4['CA'].sum()/x], 'm-o')
         plot([0, data3['NL'].sum()/x], [0, data3['CA'].sum()/x], 'g-o')


         xlabel('NL')
         ylabel('CA')
```

```
Out[21]: <matplotlib.text.Text at 0x7f5d84f9e390>
```



2 All Chapters

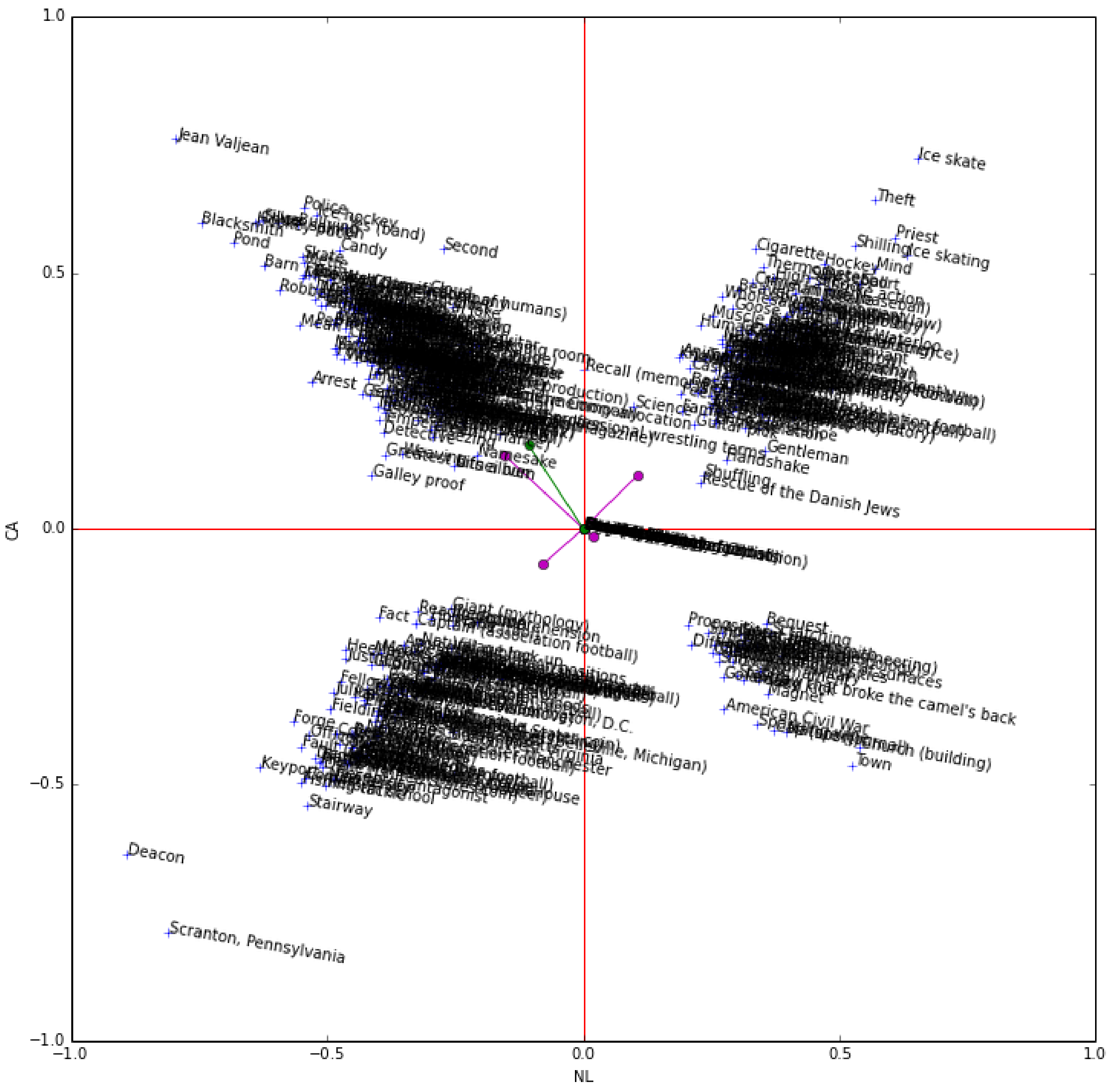
```
In [45]: data = pd.DataFrame()
         for dataFile in glob.glob('../wikipedia-miner-1.2.0/data/books/pyAnnotations/pg13250_ch*_parag
             print 'Processing ' + dataFile + '...'
             dataTmp = pd.read_csv(dataFile,names = ['Word','Probability'], quotechar='"')
             dataTmp['Probability'] = dataTmp['Probability'].apply(lambda x: np.double(x.replace('%', '

             data = data.append(dataTmp)

         byWord = data.groupby('Word')
         words = data['Word'].unique()
         data2 = pd.DataFrame(words, index=words, columns=['Word'])
         data2['Probability'] = byWord.mean()
         data2['Count'] = byWord.size()
         data2[:10]

Processing ../wikipedia-miner-1.2.0/data/books/pyAnnotations/pg13250_ch6_paragraph.txt...
Processing ../wikipedia-miner-1.2.0/data/books/pyAnnotations/pg13250_ch5_paragraph.txt...
Processing ../wikipedia-miner-1.2.0/data/books/pyAnnotations/pg13250_ch3_paragraph.txt...
Processing ../wikipedia-miner-1.2.0/data/books/pyAnnotations/pg13250_ch4_paragraph.txt...
Processing ../wikipedia-miner-1.2.0/data/books/pyAnnotations/pg13250_ch2_paragraph.txt...
Processing ../wikipedia-miner-1.2.0/data/books/pyAnnotations/pg13250_ch7_paragraph.txt...
Processing ../wikipedia-miner-1.2.0/data/books/pyAnnotations/pg13250_ch1_paragraph.txt...

Out[45]:                                            Word  Probability  Count
         Pet                                         Pet    43.840000      2
         Chum salmon                         Chum salmon    31.910000      7
         Evil                                       Evil    41.180000      1
         Scranton, Pennsylvania  Scranton, Pennsylvania    40.192222     36
         Light                                     Light    20.360000      2
         Metre                                     Metre    23.315000     12
         Mind                                       Mind    24.059444     18
         Leaf                                       Leaf    18.625000      2
         Second                                   Second    12.530000     28
         Asset forfeiture               Asset forfeiture     4.800000      1

In [71]: data2['TEMP'] = data2['Word'].apply(lambda x: wikiCountryInterest(x))
         data2['NLO'] = data2['TEMP'].apply(lambda x: x[0])
         data2['CAO'] = data2['TEMP'].apply(lambda x: x[1])
         data2['Conf'] = data2['TEMP'].apply(lambda x: x[2])
         del data2['TEMP']

         data2.to_pickle('data2_scranton.pkl')

In [77]: data2[:5]

Out[77]:                                            Word  Probability  Count       NLO  \
         Pet                                         Pet    43.840000      2  0.000702
         Chum salmon                         Chum salmon    31.910000      7  0.000000
         Evil                                       Evil    41.180000      1  0.006086
         Scranton, Pennsylvania  Scranton, Pennsylvania    40.192222     36  0.003470
         Light                                     Light    20.360000      2  0.010356

                                      CAO      Conf
         Pet                     0.070877  0.551471
         Chum salmon             0.224359  0.521739
         Evil                    0.090967  0.548200
         Scranton, Pennsylvania  0.027065  0.483882
         Light                   0.079244  0.523944

In [74]: data2 = pd.read_pickle('data2_scranton.pkl')

In [78]: expEdits = wq.getTotalContributions()
         NLE = expEdits['NL']
         CAE = expEdits['CA']

         data2['Interest-NL'] = data2.apply(lambda x: btbtools.relativeInterest(NLE, x['NLO']), axis=1)
         data2['Interest-CA'] = data2.apply(lambda x: btbtools.relativeInterest(CAE, x['CAO']), axis=1)

         nItems = data2['Count'].sum()
```

```
        data2['Total-NL'] = data2.apply(lambda x: x['Probability'] * x['Count'] / nItems * x['Conf'] *
        data2['Total-CA'] = data2.apply(lambda x: x['Probability'] * x['Count'] / nItems * x['Conf'] *
```

```
In [79]: data3 = data2.copy()
         data3['NL'] = data3['Total-NL'].apply(lambda x: explode(x, 5))
         data3['CA'] = data3['Total-CA'].apply(lambda x: explode(x, 5))

         del data3['Probability']
         del data3['Count']
         del data3['NLO']
         del data3['CAO']
         del data3['Conf']
         del data3['Interest-NL']
         del data3['Interest-CA']
         del data3['Total-NL']
         del data3['Total-CA']
```

```
In [80]: figure(figsize=(12,12))

         plot(data3['NL'], data3['CA'],'+')

         for nl,ca,txt in zip(data3['NL'], data3['CA'], data3['Word']):
             annotate(txt, (nl,ca), rotation=-10)

         plot([-1, 1], [0, 0] , 'r-')
         plot([0, 0] , [-1, 1],'r-')

         x = len(data3)

         data4 = data3[(data3['NL']>0) & (data3['CA']>0)]
         plot([0, data4['NL'].sum()/x], [0, data4['CA'].sum()/x], 'm-o')
         data4 = data3[(data3['NL']>0) & (data3['CA']<0)]
         plot([0, data4['NL'].sum()/x], [0, data4['CA'].sum()/x], 'm-o')
         data4 = data3[(data3['NL']<0) & (data3['CA']>0)]
         plot([0, data4['NL'].sum()/x], [0, data4['CA'].sum()/x], 'm-o')
         data4 = data3[(data3['NL']<0) & (data3['CA']<0)]
         plot([0, data4['NL'].sum()/x], [0, data4['CA'].sum()/x], 'm-o')
         plot([0, data3['NL'].sum()/x], [0, data3['CA'].sum()/x], 'g-o')


         xlabel('NL')
         ylabel('CA')
```

```
Out[80]: <matplotlib.text.Text at 0x7f5d7c7b0950>
```



3 Other book

```
In [84]: chunkers = {
             'paragraph': makeParagraphChunks,
             'sentence': makeSentenceChunks
         }

         inFile = '../wikipedia-miner-1.2.0/data/books/pg31127_core.txt'
         print 'Processing ' + inFile + '...'
         for chunkType in chunkers:
             outFile = inFile
             outFile = outFile.replace('books/', 'books/pyAnnotations/')
             outFile = outFile.replace('.txt', '_' + chunkType + '.txt')
             wordGenerator = processText(inFile, chunkers[chunkType])
             saveWords(outFile, wordGenerator)
```

```
Processing ../wikipedia-miner-1.2.0/data/books/pg31127_core.txt...
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
Cannot wikify! Problematic chunk:
```

```
In [85]: dataFile = '../wikipedia-miner-1.2.0/data/books/pyAnnotations/pg31127_core_paragraph.txt'
         data = pd.read_csv(dataFile, names = ['Word','Probability'], quotechar='"')
         data['Probability'] = data['Probability'].apply(lambda x: np.double(x.replace('%', '')))

         byWord = data.groupby('Word')
         words = data['Word'].unique()
         data2 = pd.DataFrame(words, index=words, columns=['Word'])
         data2['Probability'] = byWord.mean()
         data2['Count'] = byWord.size()
         data2[:10]
```

```
Out[85]:                           Word  Probability  Count
         Amsterdam             Amsterdam    54.407500     64
         Rembrandt             Rembrandt    79.401224     49
         Etching                 Etching    36.155714     14
         Sovereignty         Sovereignty    73.860000      1
         Mast (sailing)  Mast (sailing)    57.010000      1
         Harbor                   Harbor    49.556667      3
         Forest                   Forest    54.490000      1
         Commerce               Commerce    32.788333      6
```

```
        City                      City    25.922308     13
        Merchant              Merchant    38.964000      5
```

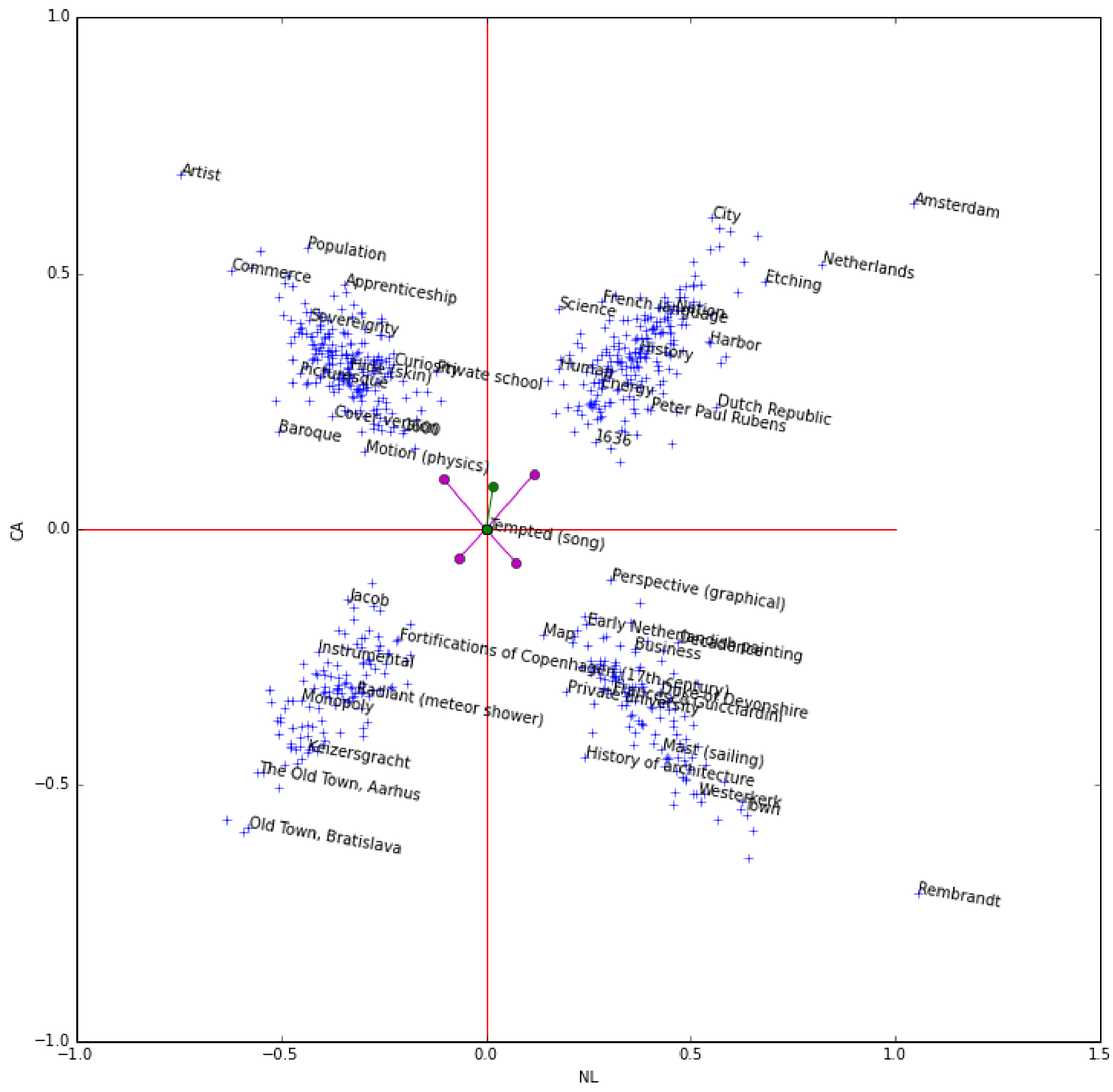
```
In [86]: data2['TEMP'] = data2['Word'].apply(lambda x: wikiCountryInterest(x))
         data2['NLO'] = data2['TEMP'].apply(lambda x: x[0])
         data2['CAO'] = data2['TEMP'].apply(lambda x: x[1])
         data2['Conf'] = data2['TEMP'].apply(lambda x: x[2])
         del data2['TEMP']

         data2.to_pickle('data2_rembrant.pkl')
```

```
In [87]: expEdits = wq.getTotalContributions()
         NLE = expEdits['NL']
         CAE = expEdits['CA']

         data2['Interest-NL'] = data2.apply(lambda x: btbtools.relativeInterest(NLE, x['NLO']), axis=1)
         data2['Interest-CA'] = data2.apply(lambda x: btbtools.relativeInterest(CAE, x['CAO']), axis=1)

         nItems = data2['Count'].sum()
         data2['Total-NL'] = data2.apply(lambda x: x['Probability'] * x['Count'] / nItems * x['Conf'] *
         data2['Total-CA'] = data2.apply(lambda x: x['Probability'] * x['Count'] / nItems * x['Conf'] *

         data3 = data2.copy()
         data3['NL'] = data3['Total-NL'].apply(lambda x: explode(x, 5))
         data3['CA'] = data3['Total-CA'].apply(lambda x: explode(x, 5))

         del data3['Probability']
         del data3['Count']
         del data3['NLO']
         del data3['CAO']
         del data3['Conf']
         del data3['Interest-NL']
         del data3['Interest-CA']
         del data3['Total-NL']
         del data3['Total-CA']
```

```
In [145]: figure(figsize=(12,12))

          plot(data3['NL'], data3['CA'],'+')

          points = []
          for nl,ca,txt in zip(data3['NL'], data3['CA'], data3['Word']):
              point = np.array([nl, ca])
              distances = [ dist(point, x) for x in points ]
              if len(distances)==0 or np.min(distances)>0.1:
                  annotate(txt, (nl,ca), rotation=-10)
                  points.append(point)

          plot([-1, 1], [0, 0] , 'r-')
          plot([0, 0] , [-1, 1],'r-')

          x = len(data3)

          data4 = data3[(data3['NL']>0) & (data3['CA']>0)]
          plot([0, data4['NL'].sum()/x], [0, data4['CA'].sum()/x], 'm-o')
```

```
data4 = data3[(data3['NL']>0) & (data3['CA']<0)]
plot([0, data4['NL'].sum()/x], [0, data4['CA'].sum()/x], 'm-o')
data4 = data3[(data3['NL']<0) & (data3['CA']>0)]
plot([0, data4['NL'].sum()/x], [0, data4['CA'].sum()/x], 'm-o')
data4 = data3[(data3['NL']<0) & (data3['CA']<0)]
plot([0, data4['NL'].sum()/x], [0, data4['CA'].sum()/x], 'm-o')
plot([0, data3['NL'].sum()/x], [0, data3['CA'].sum()/x], 'g-o')

xlabel('NL')
ylabel('CA')
```

```
Out[145]: <matplotlib.text.Text at 0x7f5d7c0a3c90>
```


Artist
Amsterdam
City
Netherlands
Etching
Population
Commerce
Apprenticeship
Science
French language
Nation
Sovereignty
Harbor
History
Human
Curiosity
Private school
Picturesque
Energy
Dutch Republic
Peter Paul Rubens
Conversation
Baroque
Motion (physics)
Tempted (song)
Perspective (graphical)
Jacob
Map
Early Netherlandish painting
Business
Fortifications of Copenhagen (17th century)
Instrumental
Private university
Duke of Devonshire
Guicciardini
Radiant (meteor shower)
Monopoly
Mast (sailing)
Keizersgracht
History of architecture
The Old Town, Aarhus
Westerkerk
Town
Old Town, Bratislava
Rembrandt
CA
NL