pa - w2v explore vector influence

June 3, 2019

```
In [1]: # Turn on Auto-Complete
        %config IPCompleter.greedy=True
```

```
In [2]: # Start logging process at root level
        import logging
        logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.I
        logging.root.setLevel(level=logging.INFO)
```

```
In [3]: # Load model and dictionary
        model_path ="models/wiki-en-190409-s300-w5-mc1-bw10000-cbow-i5-c10-unlem.model"
        dictionary_path = "dictionaries/enwiki-20190409-dict-unlemmatized.txt.bz2"
        is_lemmatized = False
```

```
In [4]: # Load word2vec unlemmatized model
        from gensim.models import Word2Vec
        model = Word2Vec.load(model_path, mmap='r')
```

```
2019-05-09 21:43:53,243 : INFO : 'pattern' package found; tag filters are available for Englisl
2019-05-09 21:43:53,254 : INFO : loading Word2Vec object from models/wiki-en-190409-s300-w5-mc1
2019-05-09 21:45:13,816 : INFO : loading wv recursively from models/wiki-en-190409-s300-w5-mc1-
2019-05-09 21:45:13,818 : INFO : loading vectors from models/wiki-en-190409-s300-w5-mc1-bw1000
2019-05-09 21:45:13,822 : INFO : setting ignored attribute vectors_norm to None
2019-05-09 21:45:13,825 : INFO : loading vocabulary recursively from models/wiki-en-190409-s30
2019-05-09 21:45:13,827 : INFO : loading trainables recursively from models/wiki-en-190409-s30
2019-05-09 21:45:13,828 : INFO : loading syn1neg from models/wiki-en-190409-s300-w5-mc1-bw1000
2019-05-09 21:45:13,834 : INFO : setting ignored attribute cum_table to None
2019-05-09 21:45:13,836 : INFO : loaded models/wiki-en-190409-s300-w5-mc1-bw10000-cbow-i5-c10-
```

```
In [ ]:
```

```
In [5]: # Saving some ram by using the KeyedVectors instance
        wv = model.wv
        #del model
```

```
In [6]: # Translate a string
        vocabulary = set(model.wv.index2word)
        #del vocabulary
```

In [37]:

```
2019-05-09 22:31:58,746 : INFO : precomputing L2-norms of word weight vectors
```

```
        ---------------------------------------------------------------------------

        ValueError                                Traceback (most recent call last)

        <ipython-input-37-ac9683c895ed> in <module>
    ----> 1 test_wv = model.wv.init_sims(replace=True)


        ~/anaconda3/envs/py36/lib/python3.6/site-packages/gensim/models/keyedvectors.py in ini
       1043             if replace:
       1044                 for i in xrange(self.vectors.shape[0]):
    -> 1045                     self.vectors[i, :] /= sqrt((self.vectors[i, :] ** 2).sum(-1))
       1046                 self.vectors_norm = self.vectors
       1047             else:


        ValueError: output array is read-only
```

In [50]:

```python
word = "federer"
word_vector_normed = model.wv.word_vec(word, use_norm=False)
#word = wv['federer']
word_vector_normed
```

Out[50]:

```
memmap([-6.18925393e-01, -2.63230252e+00, -4.08494830e-01,
        -2.00294518e+00,  7.08984435e-01,  2.05416489e+00,
        -5.16878188e-01, -7.01288223e-01, -9.48285997e-01,
         2.88235843e-01,  1.15889668e+00,  2.01796699e+00,
        -5.30680895e-01,  2.73732972e+00, -1.80597043e+00,
        -1.14714420e+00, -1.43301988e+00, -4.15284443e+00,
         1.98107028e+00,  7.92585015e-01, -2.36859381e-01,
         2.46196198e+00, -1.18992245e+00, -6.58327639e-01,
        -9.92180824e-01, -1.46043614e-01, -1.19345474e+00,
        -2.84575129e+00,  8.00136179e-02,  2.64145803e+00,
        -1.99233806e+00, -6.28322303e-01,  8.87283027e-01,
         1.82593536e+00,  1.51064873e+00, -1.52190673e+00,
         2.14218330e+00, -4.90742713e-01, -9.56747010e-02,
        -9.96588886e-01,  1.51458633e+00,  2.87261534e+00,
        -1.51088393e+00, -1.92478740e+00, -4.52478361e+00,
         6.67510509e-01, -1.02351856e+00, -7.61030197e-01,
         1.53777480e+00, -9.06242132e-01,  8.44932735e-01,
         4.98783064e+00,  9.08310831e-01, -1.08353543e+00,
        -3.35844135e+00, -2.54110432e+00, -1.05841696e+00,
```

```
 2.40799975e+00, -1.18265355e+00,  9.74503636e-01,
-4.45132303e+00,  1.99527219e-01, -3.38648152e+00,
-1.97579634e+00, -6.63368165e-01, -4.70089293e+00,
-2.56522489e+00, -3.63663721e+00, -1.44106495e+00,
 1.25158298e+00, -1.89410496e+00,  1.36002004e+00,
 5.55473268e-01,  2.52934623e+00,  1.66235483e+00,
 1.76116788e+00, -3.12326938e-01, -1.76909256e+00,
-4.18765688e+00, -2.51366711e+00, -3.40094733e+00,
-1.52807784e+00, -4.23344564e+00, -1.07804620e+00,
-4.48759906e-02,  2.52516770e+00,  1.05409253e+00,
 3.17115378e+00, -5.04737496e-01, -1.96719003e+00,
 4.22375835e-02,  3.16077769e-01,  8.09300303e-01,
-1.40376353e+00,  6.33677483e-01,  8.32299054e-01,
 7.38000929e-01, -8.74754012e-01, -2.24206543e+00,
-6.08895969e+00,  2.60793597e-01,  2.79832888e+00,
 3.90673846e-01,  1.66055894e+00, -1.63100636e+00,
 9.69337583e-01, -2.99261063e-01,  1.27206898e+00,
 5.25174570e+00, -6.63333237e-01, -1.05876565e-01,
 2.13192374e-01, -1.27140418e-01, -1.55355072e+00,
-2.04498410e+00,  4.44159061e-01, -2.09208179e+00,
-1.44336104e+00,  1.03373086e+00,  1.73766565e+00,
 3.18745637e+00, -2.35295117e-01, -3.29425406e+00,
-7.71311462e-01, -7.68274069e-01,  1.28418076e+00,
 3.15255737e+00, -5.52293360e-01, -4.24576104e-01,
-2.57788032e-01, -3.17244411e+00, -9.27932501e-01,
 1.95427942e+00, -7.75950730e-01, -3.13804954e-01,
-8.16547930e-01,  2.63076329e+00, -1.18856478e+00,
-4.18826056e+00,  2.04658508e-01, -1.74126804e+00,
-3.82765710e-01, -2.64287996e+00, -1.93172550e+00,
 6.31775856e-01, -1.79389215e+00,  2.69437361e+00,
 8.37376237e-01,  1.27131772e+00, -3.62222105e-01,
-2.89487720e+00,  1.18296909e+00,  2.62431359e+00,
 1.58916938e+00, -3.43252826e+00,  1.64197966e-01,
-1.01217651e+00,  1.16142583e+00, -7.21167803e-01,
 5.39030015e-01, -9.61165011e-01, -1.96345890e+00,
 1.79012728e+00, -1.64320397e+00,  9.73478913e-01,
 1.73690128e+00,  1.13848954e-01,  1.99949205e+00,
-1.28283992e-01,  1.74980319e+00, -1.73674583e+00,
-6.14000916e-01,  1.94625347e-03,  1.69094133e+00,
 6.51710868e-01, -5.06711006e-01, -3.58566213e+00,
-1.66409647e+00,  2.47298980e+00,  2.07342148e+00,
 1.51900792e+00, -1.76749361e+00,  6.59212530e-01,
 9.57130134e-01, -7.87378848e-01,  5.64545870e-01,
-9.49935913e-01,  3.33191663e-01,  8.78647387e-01,
 7.51346946e-01, -8.02423060e-01, -5.11909008e+00,
-1.57033825e+00,  7.94730112e-02, -9.08883274e-01,
-1.48905694e+00, -8.77789974e-01, -2.78020430e+00,
 1.22793995e-01, -1.71156728e+00, -1.38483191e+00,
```

```
       -1.39555082e-01,  2.68837571e+00, -3.17743373e+00,
       -4.62927192e-01, -2.24102592e+00, -3.27516943e-01,
        1.49417245e+00,  1.21285546e+00, -2.73910952e+00,
       -1.07029760e+00,  7.42883921e-01,  1.19110036e+00,
        1.89043730e-01, -1.68961257e-01, -2.51898193e+00,
       -1.15615535e+00,  6.18629646e+00, -4.49218927e-03,
        2.39551091e+00, -2.66092747e-01, -1.94369614e+00,
       -9.13581371e-01, -1.84409130e+00,  3.01744270e+00,
        2.27388215e+00, -2.16824436e+00,  1.45145297e+00,
       -1.44702017e+00,  2.00416493e+00,  6.52281761e-01,
        2.26616907e+00, -4.36778498e+00, -5.23083985e-01,
        1.66123497e+00, -2.05925250e+00,  3.96204782e+00,
        2.28493381e+00,  3.28405476e+00,  3.46280307e-01,
       -6.57519758e-01, -3.10387278e+00,  1.28713202e+00,
        1.07400548e+00,  2.30289721e+00,  1.71653950e+00,
        1.46952152e+00, -2.87493110e+00, -3.21868062e+00,
       -7.78048098e-01,  2.76273459e-01,  6.49546087e-01,
        4.84754711e-01,  6.74558580e-01,  5.46981335e-01,
       -8.39316189e-01, -6.43987358e-01,  5.67382097e-01,
        1.11942184e+00,  2.60270655e-01, -3.74251246e+00,
        2.47075081e+00, -1.67907107e+00,  8.53723824e-01,
        1.05925667e+00,  1.56830299e+00, -4.73578334e-01,
        7.48232543e-01,  2.59435558e+00,  9.72280085e-01,
       -1.31866884e+00,  3.73632336e+00,  1.99710679e+00,
        2.58648729e+00, -1.48084119e-01, -3.63610208e-01,
       -8.28449011e-01, -3.32384318e-01,  4.36410379e+00,
        4.98680305e+00,  3.31934166e+00, -1.27479351e+00,
       -1.37494612e+00, -2.59322906e+00, -2.07643414e+00,
        1.26520061e+00,  2.91685915e+00,  2.87692398e-01,
       -6.45411849e-01, -2.93601775e+00,  2.72036672e+00,
       -8.58729601e-01, -3.41265678e-01, -3.36974096e+00,
        7.64959693e-01,  3.13690495e+00,  3.29867649e+00,
       -3.04303694e+00,  9.68316913e-01,  4.81662393e-01], dtype=float32)
```

In [40]:
```
word_vector_normed.max()
```

Out[40]:
```
6.1862965
```

In [41]:
```
word_vector_normed.min()
```

Out[41]:
```
-6.0889597
```

In [98]:
```
import numpy as np
word = "federer"
word_vector= model.wv.word_vec(word, use_norm=False)

max_value = word_vector.max()
min_value = word_vector.min()
```

```python
    higher_vectors = []
    lower_vectors = []
    for i in range(word_vector.shape[0]):
        tmp = np.array(word_vector, dtype="float32")
        tmp[i] = max_value
        higher_vectors.append(tmp)

        tmp = np.array(word_vector, dtype="float32")
        tmp[i] = min_value
        lower_vectors.append(tmp)

    #lower_vectors = np.array(lower_vectors[:], dtype="float32")
    #print(lower_vectors)
```

In [121]:
```python
top = 5
similar_word_vector = wv.most_similar(positive=[word_vector], topn=top)
print(similar_word_vector,"\n")

for v in range(len(higher_vectors)):
    similar_vector = wv.most_similar(positive=[higher_vectors[v]], topn=top)
    for i in range(len(similar_word_vector)):
        if (similar_word_vector[i][0] != similar_vector[i][0]):
            #print(i,": similar_vector :",similar_vector[i])
            print(v,":",similar_vector,"-> d:",round(np.linalg.norm(word_vector-vecto
            break
```

```
[('federer', 1.0000001192092896), ('djokovic', 0.7750359773635864), ('nadal', 0.76740032434463

1 : [('federer', 0.9674688577651978), ('nadal', 0.7478001117706299), ('djokovic', 0.74649113416
2 : [('federer', 0.981936514377594), ('nadal', 0.7617033123970032), ('djokovic', 0.757794141769
3 : [('federer', 0.9720087051391602), ('nadal', 0.7415935397148132), ('djokovic', 0.73969566822
4 : [('federer', 0.9875750541687012), ('djokovic', 0.7640445828437805), ('nadal', 0.76097118854
6 : [('federer', 0.9813322424888611), ('nadal', 0.745172917842865), ('djokovic', 0.74319565296
22 : [('federer', 0.9773505926132202), ('djokovic', 0.7645877599716187), ('nadal', 0.754730463
26 : [('federer', 0.9773287177085876), ('djokovic', 0.7515519261360168), ('nadal', 0.745600044
27 : [('federer', 0.9658478498458862), ('nadal', 0.7535587549209595), ('djokovic', 0.747471332
38 : [('federer', 0.9836235046386719), ('djokovic', 0.7491652965545654), ('nadal', 0.748436987
44 : [('federer', 0.9516366720199585), ('djokovic', 0.7459554076194763), ('nadal', 0.720818996
45 : [('federer', 0.9873849749565125), ('nadal', 0.7569801211357117), ('djokovic', 0.754308640
54 : [('federer', 0.9617849588394165), ('djokovic', 0.7550451755523682), ('nadal', 0.747102022
62 : [('federer', 0.9615557789802551), ('djokovic', 0.7405887842178345), ('nadal', 0.736121177
63 : [('federer', 0.972196638584137), ('djokovic', 0.7520910501480103), ('nadal', 0.7483494877
66 : [('federer', 0.9679697155952454), ('nadal', 0.7509782314300537), ('djokovic', 0.748198628
76 : [('federer', 0.9824643731117249), ('nadal', 0.7614733576774597), ('djokovic', 0.756993353
80 : [('federer', 0.9614373445510864), ('djokovic', 0.7506026029586792), ('nadal', 0.742543697
89 : [('federer', 0.9722560048103333), ('nadal', 0.7581894993782043), ('djokovic', 0.747678875
93 : [('federer', 0.9760022759437561), ('nadal', 0.7520438432693481), ('djokovic', 0.750524878
```

```
96 : [('federer', 0.9877071976661682), ('nadal', 0.76472008228302), ('djokovic', 0.762816071510
97 : [('federer', 0.9792643785476685), ('djokovic', 0.7504821419715881), ('nadal', 0.7479398250
99 : [('federer', 0.9359966516494751), ('nadal', 0.7525387406349182), ('djokovic', 0.7373512387
109 : [('federer', 0.9804996848106384), ('nadal', 0.7527401447296143), ('djokovic', 0.750105023
110 : [('federer', 0.983569860458374), ('nadal', 0.7643581032752991), ('djokovic', 0.7642256617
111 : [('federer', 0.9852062463760376), ('nadal', 0.751778244972229), ('djokovic', 0.7496097087
112 : [('federer', 0.9834575653076172), ('djokovic', 0.7717949151992798), ('nadal', 0.759754180
114 : [('federer', 0.971716582775116), ('nadal', 0.7597029209136963), ('djokovic', 0.7522677183
118 : [('federer', 0.9890128374099731), ('nadal', 0.7692031860351562), ('djokovic', 0.761624574
121 : [('federer', 0.9828811883926392), ('nadal', 0.7608731985092163), ('djokovic', 0.756851255
122 : [('federer', 0.9623067378997803), ('nadal', 0.7562921047210693), ('djokovic', 0.754964768
127 : [('federer', 0.9811326265335083), ('djokovic', 0.7630206346511841), ('nadal', 0.757006824
140 : [('federer', 0.9737919569015503), ('nadal', 0.7469925284385681), ('djokovic', 0.738647818
141 : [('federer', 0.9820785522460938), ('djokovic', 0.7541807889938354), ('nadal', 0.744783937
143 : [('federer', 0.9725000858306885), ('djokovic', 0.7532610297203064), ('nadal', 0.745602369
149 : [('federer', 0.9821915626525879), ('nadal', 0.7587418556213379), ('djokovic', 0.755823194
153 : [('federer', 0.9912641048431396), ('nadal', 0.7679131031036377), ('djokovic', 0.765571594
154 : [('federer', 0.9611778855323792), ('nadal', 0.7640374302864075), ('djokovic', 0.746991395
159 : [('federer', 0.9867867231369019), ('nadal', 0.761846125125885), ('djokovic', 0.7588425874
161 : [('federer', 0.972281813621521), ('djokovic', 0.7496495842933655), ('nadal', 0.7331033945
168 : [('federer', 0.9834516048431396), ('nadal', 0.7455811500549316), ('djokovic', 0.745033740
169 : [('federer', 0.9918667078018188), ('nadal', 0.7636914253234863), ('djokovic', 0.763045430
170 : [('federer', 0.9738223552703857), ('nadal', 0.7491594552993774), ('djokovic', 0.747865730
177 : [('federer', 0.9743062257766724), ('nadal', 0.7564150094985962), ('djokovic', 0.750969767
183 : [('federer', 0.9886819124221802), ('nadal', 0.7658869028091431), ('djokovic', 0.752552986
185 : [('federer', 0.9869067668914795), ('nadal', 0.7604039907455444), ('djokovic', 0.755593776
188 : [('federer', 0.988337516784668), ('nadal', 0.7713549137115479), ('djokovic', 0.7697630524
190 : [('federer', 0.9796915054321289), ('nadal', 0.75789475440979), ('djokovic', 0.75641536712
191 : [('federer', 0.945972740650177), ('djokovic', 0.7349231243133545), ('nadal', 0.7236721515
192 : [('federer', 0.9749240279197693), ('djokovic', 0.7552589178085327), ('nadal', 0.743764817
194 : [('federer', 0.9790613651275635), ('nadal', 0.7561460137367249), ('djokovic', 0.753683567
205 : [('federer', 0.9703332185745239), ('nadal', 0.7397903800010681), ('djokovic', 0.739217758
209 : [('federer', 0.9666629433631897), ('djokovic', 0.7642092108726501), ('nadal', 0.752450585
214 : [('federer', 0.9832359552383423), ('djokovic', 0.7674907445907593), ('nadal', 0.75379753
223 : [('federer', 0.973098635673523), ('djokovic', 0.757429301738739), ('nadal', 0.7523142099
226 : [('federer', 0.9708508253097534), ('nadal', 0.7517833113670349), ('djokovic', 0.748589515
234 : [('federer', 0.9915372133255005), ('nadal', 0.7670965790748596), ('djokovic', 0.764221906
249 : [('federer', 0.979834258556366), ('nadal', 0.7571359872817993), ('djokovic', 0.7570855617
250 : [('federer', 0.9855191707611084), ('nadal', 0.7594801783561707), ('djokovic', 0.755433142
254 : [('federer', 0.9868241548538208), ('nadal', 0.7586868405342102), ('djokovic', 0.757548809
256 : [('federer', 0.980610728263855), ('nadal', 0.7567037343978882), ('djokovic', 0.7537702322
283 : [('federer', 0.9677611589431763), ('djokovic', 0.7655210494995117), ('nadal', 0.754936575
289 : [('federer', 0.9651498794555664), ('nadal', 0.7487481832504272), ('djokovic', 0.748298346
291 : [('federer', 0.9793593883514404), ('nadal', 0.7567390203475952), ('djokovic', 0.754786014
297 : [('federer', 0.9643127918243408), ('nadal', 0.7438105344772339), ('djokovic', 0.743334233
```

```
In [101]: higher_vectors[0][:10]
```

```
Out[101]: array([ 6.1862965 , -2.6323025 , -0.40849483, -2.0029452 ,  0.70898443,
                  2.054165  , -0.5168782 , -0.7012882 , -0.948286  ,  0.28823584],
                dtype=float32)
```

```
In [102]: higher_vectors[1][:10]
```

```
Out[102]: array([-0.6189254 ,  6.1862965 , -0.40849483, -2.0029452 ,  0.70898443,
                  2.054165  , -0.5168782 , -0.7012882 , -0.948286  ,  0.28823584],
                dtype=float32)
```

```
In [91]: wv.most_similar(positive=[word_vector])
```

```
Out[91]: [('federer', 1.0000001192092896),
          ('djokovic', 0.7750359773635864),
          ('nadal', 0.767400324344635),
          ('wawrinka', 0.6816220879554749),
          ('vasselin', 0.6664406657218933),
          ('berdych', 0.645709753036499),
          ('mahut', 0.6398465633392334),
          ('sampras', 0.6329268217086792),
          ('verdasco', 0.6302427649497986),
          ('roddick', 0.622452974319458)]
```

```
In [92]: wv.most_similar(positive=[lower_vectors])
```

```
Out[92]: [('federer', 1.0000001192092896),
          ('djokovic', 0.7750359773635864),
          ('nadal', 0.767400324344635),
          ('wawrinka', 0.6816220879554749),
          ('vasselin', 0.6664406657218933),
          ('berdych', 0.645709753036499),
          ('mahut', 0.6398465633392334),
          ('sampras', 0.6329268217086792),
          ('verdasco', 0.6302427649497986),
          ('roddick', 0.622452974319458)]
```

```
In [103]: wv.most_similar(positive=[higher_vectors[0]])
```

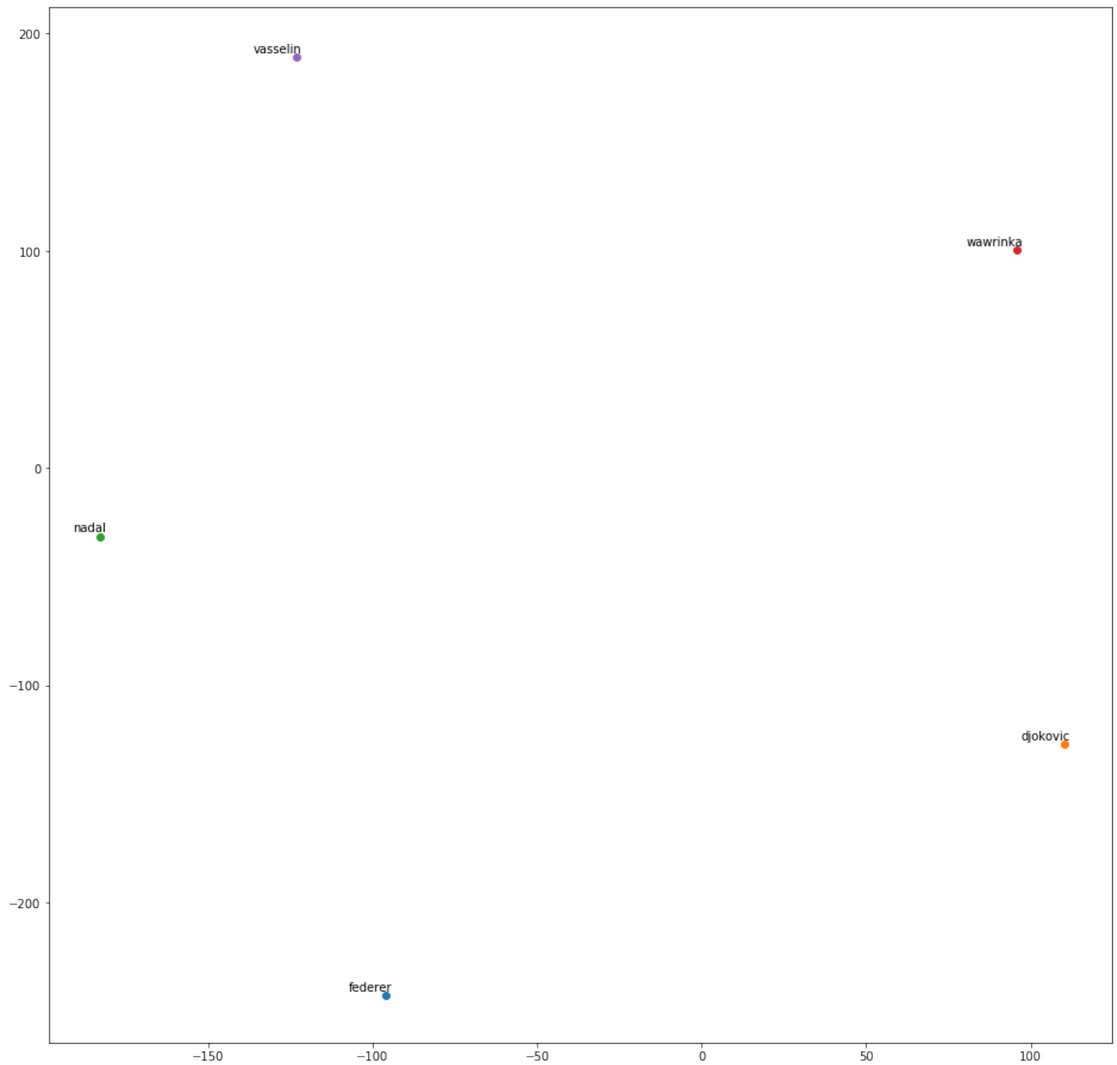
```
Out[103]: [('federer', 0.980754017829895),
           ('djokovic', 0.7609372735023499),
           ('nadal', 0.759452223777771),
           ('wawrinka', 0.6838314533233643),
           ('vasselin', 0.6651499271392822),
           ('berdych', 0.6336572170257568),
           ('mahut', 0.6318328380584717),
           ('verdasco', 0.6219873428344727),
           ('roddick', 0.6185396909713745),
           ('sampras', 0.6147887706756592)]
```

```
In [ ]:
```

```
In [125]: top = 10
          similar_vectors = wv.most_similar(positive=[word_vector], topn=top)

In [126]: my_vocabulary = []
          for vector in similar_vectors:
              my_vocabulary.append(vector[0])

          #print(my_vocabulary)
```

```
['federer', 'djokovic', 'nadal', 'wawrinka', 'vasselin']
```

```
In [ ]: labels = []
        tokens = []
        for word in my_vocabulary:
            tokens.append(model[word])
            labels.append(word)

        #print(tokens)
        #print(labels)

In [132]: from sklearn.manifold import TSNE
          tsne_model = TSNE(perplexity=40, n_components=2, init='pca', n_iter=2500, random_sta
          new_values = tsne_model.fit_transform(tokens)

In [135]: import matplotlib.pyplot as plt
          %matplotlib inline

          x = []
          y = []
          for value in new_values:
              x.append(value[0])
              y.append(value[1])

          plt.figure(figsize=(16, 16))
          for i in range(len(x)):
              plt.scatter(x[i],y[i])
              plt.annotate(labels[i],
                           xy=(x[i], y[i]),
                           xytext=(5, 2),
                           textcoords='offset points',
                           ha='right',
                           va='bottom')
          plt.show()
```



In [ ]:

In [149]:
```
top = 10
similar_vectors = wv.most_similar(positive=[word_vector], topn=top)

my_vocabulary = []
for vector in similar_vectors:
    my_vocabulary.append(vector[0])

print(my_vocabulary)
```

```
['federer', 'djokovic', 'nadal', 'wawrinka', 'vasselin', 'berdych', 'mahut', 'sampras', 'verdas
```

In [152]:
```
for v in range(len(higher_vectors)):
    similar_vector = wv.most_similar(positive=[higher_vectors[v]], topn=top)
```

```
    for i in range(len(similar_word_vector)):
        if (similar_word_vector[i][0] not in my_vocabulary):
            my_vocabulary.append(similar_word_vector[i][0])
            if (i%5==0): print(i)
print(my_vocabulary)
```

```
['federer', 'djokovic', 'nadal', 'wawrinka', 'vasselin', 'berdych', 'mahut', 'sampras', 'verdas
```

In [168]:
```
import time

top = 10
similar_vector = wv.most_similar(positive=[word_vector], topn=top)
print(len(similar_vector))
print(similar_vector,"\n")

custom_vocabulary = []
for vector in similar_vectors:
    custom_vocabulary.append(vector[0])

vector_name = similar_vector[0][0]

print("intial vocab for \""+vector_name+"\"->",custom_vocabulary,"\n")

vector_len = len(higher_vectors)
start_time = time.time()
for v in range(vector_len):
    similar_word_vector = wv.most_similar(positive=[higher_vectors[v]], topn=top)
    for i in range(len(similar_word_vector)):
        if (similar_word_vector[i][0] not in custom_vocabulary):
            print(v,":",i,"->", similar_word_vector[i][0])
            custom_vocabulary.append(similar_word_vector[i][0])
    if (v!=0 and v%5==0): print(v,"/",vector_len, "iterations so far")
        #print(i, similar_word_vector[i][0])
    #print("\n")
end_time = time.time()
print("\nRunning time is {}s".format(end_time-start_time))
print("\nfinal vocab for: \"",vector_name,"\"",custom_vocabulary)
```

```
10
[('federer', 1.0000001192092896), ('djokovic', 0.7750359773635864), ('nadal', 0.76740032434463

intial vocab for "federer"-> ['federer', 'djokovic', 'nadal', 'wawrinka', 'vasselin', 'berdych

2 : 7 -> raonic
5 / 300 iterations so far
7 : 9 -> davydenko
10 / 300 iterations so far
```

```
15 / 300 iterations so far
20 / 300 iterations so far
25 / 300 iterations so far
30 / 300 iterations so far
35 / 300 iterations so far
40 / 300 iterations so far
45 / 300 iterations so far
50 / 300 iterations so far
55 : 9 -> monfils
55 / 300 iterations so far
60 / 300 iterations so far
65 / 300 iterations so far
70 / 300 iterations so far
75 / 300 iterations so far
80 / 300 iterations so far
85 / 300 iterations so far
90 / 300 iterations so far
95 / 300 iterations so far
100 / 300 iterations so far
105 / 300 iterations so far
110 / 300 iterations so far
115 / 300 iterations so far
120 / 300 iterations so far
125 / 300 iterations so far
130 / 300 iterations so far
135 / 300 iterations so far
140 / 300 iterations so far
145 / 300 iterations so far
150 / 300 iterations so far
155 / 300 iterations so far
160 / 300 iterations so far
165 / 300 iterations so far
170 / 300 iterations so far
175 / 300 iterations so far
180 / 300 iterations so far
185 / 300 iterations so far
190 / 300 iterations so far
191 : 9 -> henin
195 / 300 iterations so far
200 / 300 iterations so far
205 / 300 iterations so far
210 / 300 iterations so far
215 / 300 iterations so far
220 / 300 iterations so far
225 / 300 iterations so far
230 / 300 iterations so far
235 / 300 iterations so far
240 / 300 iterations so far
```

```
245 / 300 iterations so far
250 / 300 iterations so far
255 / 300 iterations so far
260 / 300 iterations so far
265 / 300 iterations so far
270 / 300 iterations so far
275 / 300 iterations so far
280 / 300 iterations so far
285 / 300 iterations so far
290 / 300 iterations so far
295 / 300 iterations so far
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-168-91ce9bccf5ac> in <module>
 25         #print(i, similar_word_vector[i][0])
 26     #print("\n")
---> 27 print("\nRunning time is {}s".format(end_time-start_time))
 28 print("\nfinal vocab for: \"",vector_name,"\"")


NameError: name 'end_time' is not defined
```

In [176]:
```
print("\nfinal vocab for: \"",vector_name,"\"",custom_vocabulary)
```

```
final vocab for: " federer " ['federer', 'djokovic', 'nadal', 'wawrinka', 'vasselin', 'berdych
```

In [190]:
```
labels = []
tokens = []

label_count = 0
for vector in higher_vectors:
    tokens.append(vector)
    #label = "federer_"+str(label_count)
    label = str(label_count)
    labels.append(label)
    label_count += 1

for word in custom_vocabulary:
    tokens.append(model[word])
    labels.append(word)
```

```
          new_values = tsne_model.fit_transform(tokens)
```

```
/home/rclaret/anaconda3/envs/py36/lib/python3.6/site-packages/ipykernel_launcher.py:13: Depreca
  del sys.path[0]
```

```
In [192]: #import matplotlib
          import matplotlib.pyplot as plt
          #matplotlib.use('nbagg')
          %matplotlib notebook

          x = []
          y = []
          for value in new_values:
              x.append(value[0])
              y.append(value[1])

          plt.figure(figsize=(15, 15))
          for i in range(len(x)):
              plt.scatter(x[i],y[i])
              plt.annotate(labels[i],
                           xy=(x[i], y[i]),
                           xytext=(5, 2),
                           textcoords='offset points',
                           ha='right',
                           va='bottom')
          plt.show()
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
In [ ]:

In [ ]:

In [175]: import time

          top = 50
          similar_vector = wv.most_similar(positive=[word_vector], topn=top)
          print(len(similar_vector))
          print(similar_vector,"\n")

          custom_vocabulary_50 = []
          for vector in similar_vectors:
              custom_vocabulary_50.append(vector[0])
```

```
        vector_name = similar_vector[0][0]

        print("intial vocab for \""+vector_name+"\"->",custom_vocabulary_50,"\n")

        vector_len = len(higher_vectors)
        start_time = time.time()
        for v in range(vector_len):
            similar_word_vector = wv.most_similar(positive=[higher_vectors[v]], topn=top)
            for i in range(len(similar_word_vector)):
                if (similar_word_vector[i][0] not in custom_vocabulary_50):
                    print(v,":",i,"->", similar_word_vector[i][0])
                    custom_vocabulary_50.append(similar_word_vector[i][0])
            if (v!=0 and v%5==0): print(v,"/",vector_len, "iterations so far")
                #print(i, similar_word_vector[i][0])
            #print("\n")
        end_time = time.time()
        print("\nRunning time is {}s".format(end_time-start_time))
        print("\nfinal vocab for: \"",vector_name,"\"",custom_vocabulary_50)
```

```
50
[('federer', 1.0000001192092896), ('djokovic', 0.7750359773635864), ('nadal', 0.76740032434463

intial vocab for "federer"-> ['federer', 'djokovic', 'nadal', 'wawrinka', 'vasselin', 'berdych

0 : 10 -> raonic
0 : 11 -> davydenko
0 : 12 -> henin
0 : 13 -> monfils
0 : 14 -> kvitová
0 : 15 -> youzhny
0 : 16 -> sharapova
0 : 17 -> llodra
0 : 18 -> stosur
0 : 19 -> radwaska
0 : 20 -> isner
0 : 21 -> gasquet
0 : 22 -> clijsters
0 : 23 -> söderling
0 : 24 -> benneteau
0 : 25 -> agassi
0 : 26 -> kuerten
0 : 27 -> potro
0 : 28 -> kuznetsova
0 : 29 -> safin
0 : 30 -> wozniacki
0 : 31 -> hingis
0 : 32 -> fognini
```

```
0 : 33 -> dementieva
0 : 34 -> federerwomen
0 : 35 -> halep
0 : 36 -> hantuchová
0 : 37 -> mauresmo
0 : 38 -> lendl
0 : 39 -> moyá
0 : 40 -> rezaï
0 : 41 -> nishikori
0 : 42 -> ivanovic
0 : 43 -> baghdatis
0 : 44 -> zimonji
0 : 45 -> tipsarevi
0 : 46 -> thiem
0 : 47 -> muguruza
0 : 48 -> kafelnikov
0 : 49 -> zvonareva
1 : 36 -> henman
1 : 43 -> rosewall
1 : 45 -> karlovi
1 : 49 -> twose
2 : 42 -> seppi
2 : 48 -> querrey
3 : 42 -> tiebreak
4 : 48 -> svitolina
5 / 300 iterations so far
9 : 48 -> dodig
10 / 300 iterations so far
12 : 38 -> philippoussis
15 : 48 -> tpánek
15 / 300 iterations so far
18 : 49 -> mcenroe
19 : 41 -> kohlschreiber
20 / 300 iterations so far
130 / 300 iterations so far
135 / 300 iterations so far
140 / 300 iterations so far
145 / 300 iterations so far
150 / 300 iterations so far
155 / 300 iterations so far
160 / 300 iterations so far
165 / 300 iterations so far
170 / 300 iterations so far
175 / 300 iterations so far
180 / 300 iterations so far
185 / 300 iterations so far
190 / 300 iterations so far
195 / 300 iterations so far
```

```
200 / 300 iterations so far
204 : 43 -> haitengi
205 / 300 iterations so far
210 / 300 iterations so far
215 / 300 iterations so far
220 / 300 iterations so far
225 / 300 iterations so far
230 / 300 iterations so far
235 / 300 iterations so far
240 / 300 iterations so far
245 / 300 iterations so far
250 / 300 iterations so far
255 / 300 iterations so far
260 : 45 -> tsonga
260 / 300 iterations so far
265 / 300 iterations so far
270 / 300 iterations so far
275 / 300 iterations so far
280 / 300 iterations so far
285 / 300 iterations so far
290 / 300 iterations so far
295 / 300 iterations so far

Running time is 1787.7045834064484s

final vocab for: " federer " ['federer', 'djokovic', 'nadal', 'wawrinka', 'vasselin', 'berdych
```

```
In [193]: labels = []
          tokens = []

          label_count = 0
          for vector in higher_vectors:
              tokens.append(vector)
              #label = "federer_"+str(label_count)
              label = str(label_count)
              labels.append(label)
              label_count += 1

          for word in custom_vocabulary_50:
              tokens.append(model[word])
              labels.append(word)

          new_values = tsne_model.fit_transform(tokens)
```

```
/home/rclaret/anaconda3/envs/py36/lib/python3.6/site-packages/ipykernel_launcher.py:13: Depreca
  del sys.path[0]
```

In [194]:
```
#import matplotlib
import matplotlib.pyplot as plt
#matplotlib.use('nbagg')
%matplotlib notebook

x = []
y = []
for value in new_values:
    x.append(value[0])
    y.append(value[1])

plt.figure(figsize=(15, 15))
for i in range(len(x)):
    plt.scatter(x[i],y[i])
    plt.annotate(labels[i],
                 xy=(x[i], y[i]),
                 xytext=(5, 2),
                 textcoords='offset points',
                 ha='right',
                 va='bottom')
plt.show()
```

```
<IPython.core.display.Javascript object>


<IPython.core.display.HTML object>
```

In [ ]:

In [185]:
```
word_vector_federer = model.wv.word_vec("federer", use_norm=False)
word_vector_tiebreak = model.wv.word_vec("tiebreak", use_norm=False)
print("dist federer<->tiebreak:",round(np.linalg.norm(word_vector_federer-word_vecto
print("cos federer<->tiebreak:",wv.cosine_similarities(word_vector_federer, [word_ve
```

```
dist federer<->tiebreak: 30.303486
cos federer<->tiebreak: [0.5283575]
```

In [187]:
```
word_vector_tennis = model.wv.word_vec("tennis", use_norm=False)
print("dist federer<->tennis:",round(np.linalg.norm(word_vector_federer-word_vector_
print("cos federer<->tennis:",wv.cosine_similarities(word_vector_federer, [word_vect
```

```
dist federer<->tennis: 40.23253
cos federer<->tennis: [0.28765106]
```

In [188]:
```
print("dist tennis<->tiebreak:",round(np.linalg.norm(word_vector_tiebreak-word_vecto
print("cos tennis<->tiebreak:",wv.cosine_similarities(word_vector_tiebreak, [word_ve
```

```
dist tennis<->tiebreak: 39.396652
cos tennis<->tiebreak: [0.13990684]
```

```
In [ ]:
```