# pa - play with w2v enwiki unlemmatized

June 2, 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_id_current = 99999
        #model_path_current = "models/enwiki-full-dict-"+str(model_id_current)+".model"
        #model_path_99999 = "models/enwiki-20190319-lemmatized-99999.model"
        model_path_current ="models/enwiki-20190409-unlemmatized.model"

        dictionary_full_wikien_lem_path = "dictionaries/enwiki-20190409-dict-unlemmatized.txt.

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

2019-04-23 12:55:31,926 : INFO : 'pattern' package found; tag filters are available for Englis
2019-04-23 12:55:31,935 : INFO : loading Word2Vec object from models/enwiki-20190409-unlemmati
2019-04-23 12:56:26,021 : INFO : loading wv recursively from models/enwiki-20190409-unlemmatiz
2019-04-23 12:56:26,024 : INFO : loading vectors from models/enwiki-20190409-unlemmatized.mode
2019-04-23 12:56:26,046 : INFO : setting ignored attribute vectors_norm to None
2019-04-23 12:56:26,049 : INFO : loading vocabulary recursively from models/enwiki-20190409-un
2019-04-23 12:56:26,051 : INFO : loading trainables recursively from models/enwiki-20190409-un
2019-04-23 12:56:26,053 : INFO : loading syn1neg from models/enwiki-20190409-unlemmatized.mode
2019-04-23 12:56:26,083 : INFO : setting ignored attribute cum_table to None
2019-04-23 12:56:26,085 : INFO : loaded models/enwiki-20190409-unlemmatized.model


In [ ]:

In [5]: # Custom lemmatizer function to play with word
        from gensim.utils import lemmatize
        #vocabulary = set(wv.index2word)
        def lem(word):
```

```
        try:
            return lemmatize(word)[0].decode("utf-8")
        except:
            pass

    print(lem("dog"))
    print(lem("that"))
```

```
dog/NN
None
```

```
In [5]: # Testing similarity
        print("Most similar to","woman")
        print(model.wv.most_similar("woman"))
```

```
2019-04-23 12:57:13,947 : INFO : precomputing L2-norms of word weight vectors


Most similar to woman
[('girl', 0.7156134247779846), ('man', 0.7035340070724487), ('person', 0.627473771572113), ('p
```

```
In [6]: print("Most similar to","doctor")
        print(model.wv.most_similar("doctor"))
```

```
Most similar to doctor
[('adric', 0.568127453327179), ('dentist', 0.5638059377670288), ('nardole', 0.5589247941970825
```

```
In [6]: print("Most similar to","doctor")
        print(model.wv.most_similar("doctor"))
```

```
Most similar to doctor
[('adric', 0.568127453327179), ('dentist', 0.5638059377670288), ('nardole', 0.5589247941970825
```

```
In [ ]:
```

```
In [7]: # Saving some ram by using the KeyedVectors instance
        wv = model.wv
        #del model
```

```
In [13]: # Testing similarity with KeyedVectors
         print("Most similar to","woman")
         print(wv.most_similar("woman"))
         print("\nMost similar to","man")
         print(wv.most_similar("man"))
         print("\nMost similar to","doctor")
         print(wv.most_similar("doctor"))
         print("\nMost similar to","doctor","cosmul")
         print(wv.most_similar_cosmul(positive=["doctor"]))
```

```
Most similar to woman
[('girl', 0.7156134247779846), ('man', 0.7035340070724487), ('person', 0.627473771572113), ('p

Most similar to man
[('woman', 0.7035340070724487), ('boy', 0.6310229301452637), ('girl', 0.6180729269981384), ('p

Most similar to doctor
[('adric', 0.568127453327179), ('dentist', 0.5638059377670288), ('nardole', 0.5589247941970825

Most similar to doctor cosmul
[('adric', 0.784062922000885), ('dentist', 0.7819021940231323), ('nardole', 0.779461681842804)
```

```
In [8]: # Testing similarity with KeyedVectors
        print("Most similar to","woman")
        print(wv.most_similar("woman"))
        print("\nMost similar to","man")
        print(wv.most_similar("man"))
        print("\nMost similar to","doctor")
        print(wv.most_similar("doctor"))
        print("\nMost similar to","doctor","cosmul")
        print(wv.most_similar_cosmul(positive=["doctor"]))
```

```
Most similar to woman
[('girl', 0.7156134247779846), ('man', 0.7035340070724487), ('person', 0.627473771572113), ('p

Most similar to man
[('woman', 0.7035340070724487), ('boy', 0.6310229301452637), ('girl', 0.6180729269981384), ('p

Most similar to doctor
[('adric', 0.568127453327179), ('dentist', 0.5638059377670288), ('nardole', 0.5589247941970825

Most similar to doctor cosmul
[('adric', 0.784062922000885), ('dentist', 0.7819021940231323), ('nardole', 0.779461681842804)
```

```
In [14]: print("similarity of doctor + woman - man")
         wv.most_similar(positive=["doctor","woman"], negative=["man"])
```

```
similarity of doctor + woman - man
```

```
Out[14]: [('nurse', 0.6044224500656128),
          ('midwife', 0.5872353911399841),
          ('gynecologist', 0.5799287557601929),
          ('physician', 0.5611956119537354),
          ('psychiatrist', 0.5463466644287109),
          ('pediatrician', 0.54508376121521),
          ('gynaecologist', 0.5450509786605835),
          ('obstetrician', 0.5407373905181885),
          ('dentist', 0.5338024497032166),
          ('pharmacist', 0.5186790227890015)]
```

```
In [9]: print("similarity of doctor + woman - man")
        wv.most_similar(positive=["doctor","woman"], negative=["man"])
```

```
similarity of doctor + woman - man
```

```
Out[9]: [('nurse', 0.6044224500656128),
         ('midwife', 0.5872353911399841),
         ('gynecologist', 0.5799287557601929),
         ('physician', 0.5611956119537354),
         ('psychiatrist', 0.5463466644287109),
         ('pediatrician', 0.54508376121521),
         ('gynaecologist', 0.5450509786605835),
         ('obstetrician', 0.5407373905181885),
         ('dentist', 0.5338024497032166),
         ('pharmacist', 0.5186790227890015)]
```

```
In [15]: # Get cosmul of logic
         print("cosmul of doctor + woman - man")
         wv.most_similar_cosmul(positive=["doctor","woman"], negative=["man"])
```

```
cosmul of doctor + woman - man
```

```
Out[15]: [('nurse', 0.9101355671882629),
          ('midwife', 0.9078396558761597),
          ('gynecologist', 0.901469886302948),
          ('physician', 0.8901388645172119),
          ('pediatrician', 0.8833072185516357),
          ('gynaecologist', 0.8768758773803711),
          ('obstetrician', 0.8726370930671692),
          ('psychiatrist', 0.8607419729232788),
          ('paediatrician', 0.8482840061187744),
          ('dentist', 0.8474707007408142)]
```

```
In [10]: # Get cosmul of logic
         print("cosmul of doctor + woman - man")
         wv.most_similar_cosmul(positive=["doctor","woman"], negative=["man"])
```

```
cosmul of doctor + woman - man
```

```
Out[10]: [('nurse', 0.9101355671882629),
          ('midwife', 0.9078396558761597),
          ('gynecologist', 0.901469886302948),
          ('physician', 0.8901388645172119),
          ('pediatrician', 0.8833072185516357),
          ('gynaecologist', 0.8768758773803711),
          ('obstetrician', 0.8726370930671692),
          ('psychiatrist', 0.8607419729232788),
          ('paediatrician', 0.8482840061187744),
          ('dentist', 0.8474707007408142)]
```

```
In [21]: # Ways to retrive word vector
         print("Get item dog")
         vec_dog = wv.__getitem__("dog")
         vec_dog = wv.get_vector("dog")
         vec_dog = wv.word_vec("dog")
         print("vec_dog", vec_dog.shape, vec_dog[:10])
```

```
Get item dog
vec_dog (300,) [-2.6634293   2.3062525   0.19561668  0.7163729  -0.52825516 -0.0074518
  1.9209532  -0.3408677  -1.0190932   0.07459767]
```

```
In [23]: # Get similar words to vector
         print("Similar by vector to dog vector at top 10")
         print(wv.similar_by_vector(vector=vec_dog, topn=10, restrict_vocab=None))
         print("Most similar to dog vector")
         print(wv.most_similar(positive=[vec_dog]))
         print("Similar to cat vector")
         vec_cat = wv.word_vec("cat")
         print(wv.most_similar(positive=[vec_cat]))
```

```
Similar by vector to dog vector at top 10
[('dog', 1.0), ('dogs', 0.7544571161270142), ('cat', 0.7354434132575989), ('puppy', 0.68804025
Most similar to dog vector
[('dog', 1.0), ('dogs', 0.7544571161270142), ('cat', 0.7354434132575989), ('puppy', 0.68804025
Similar to cat vector
[('cat', 1.0), ('dog', 0.7354434728622437), ('rabbit', 0.6862228512763977), ('kitten', 0.61512
```

```
In [24]: # closer to __ than __
         print("closer to dog than cat")
         print(wv.words_closer_than("dog", "cat"))
         print("\ncloser to cat than dog")
         print(wv.words_closer_than("cat", "dog"))
```

```
closer to dog than cat
['dogs']

closer to cat than dog
[]
```

```
In [26]: # Normalized Vector
         vec_king_norm = wv.word_vec("king", use_norm=True)
         print("vec_king_norm:",vec_king_norm.shape, vec_king_norm[:10])
         # Not normalized vectore
         vec_king_unnorm = wv.word_vec("king", use_norm=False)
         print("vec_king_unnorm:",vec_king_norm.shape, vec_king_unnorm[:10])

vec_king_norm: (300,) [ 0.01541833 -0.01279872 -0.01601281  0.02326567  0.08323052  0.02704921
  0.04290665  0.06306878  0.1487852   0.03694476]
vec_king_unnorm: (300,) [ 0.4366548  -0.36246604 -0.45349073  0.6588954   2.3571293   0.766047
  1.2151375   1.786139    4.2136703   1.0462939 ]


In [27]: wv.most_similar(positive=[vec_king_norm], negative=[vec_king_unnorm])

Out[27]: [('utilized', 0.3044779598712921),
          ('focused', 0.278587281703949),
          ('categorized', 0.275567889213562),
          ('saenchuanglek', 0.2754442095756531),
          ('braese', 0.2734081447124481),
          ('neshwille', 0.27287906408309937),
          ('knocknaskeharoe', 0.26975083351135254),
          ('structured', 0.2687339186668396),
          ('contributing', 0.265920490026474),
          ('individualistic', 0.2639663815498352)]

In [28]: # Generate random vector
         import numpy as np
         vec_random = np.random.rand(300,)
         vec_random_norm = vec_random / vec_random.max(axis=0)
         print("similar to random vector")
         print(wv.most_similar(positive=[vec_random]))
         print("\n similar to nomalized random vector")
         print(wv.most_similar(positive=[vec_random_norm]))

similar to random vector
[('vicarios', 0.27943286299705505), ('by', 0.2677918076515198), ('pyrophore', 0.263403713703155

 similar to nomalized random vector
[('vicarios', 0.27943286299705505), ('by', 0.2677918076515198), ('pyrophore', 0.263403713703155


In [29]: # Get similarity from a random vector and normilized king vector
         print("similarity from a normalized random vector to normalized vector of king")
         wv.most_similar(positive=[vec_random_norm,vec_king_norm])

similarity from a normalized random vector to normalized vector of king
```

```
Out[29]: [('vicarios', 0.28092366456985474),
          ('rarily', 0.2663201093673706),
          ('pyrophore', 0.26312553882598877),
          ('polymedicine', 0.26185768842697144),
          ('muihki', 0.2599378228187561),
          ('uaauk', 0.25813591480255127),
          ('krath', 0.2575136423110962),
          ('by', 0.25641798973083496),
          ('mosquital', 0.2561648488044739),
          ('', 0.25569337606430054)]
```

```
In [30]: # Get similarity from a random vector and unormalized king vector
         print("similarity from a random vector to unormalized vector of king")
         wv.most_similar(positive=[vec_random,vec_king_unnorm])
```

```
similarity from a random vector to unormalized vector of king
```

```
Out[30]: [('king', 0.9413427114486694),
          ('queen', 0.6534480452537537),
          ('prince', 0.6449348330497742),
          ('kings', 0.5767666697502136),
          ('emperor', 0.5365402102470398),
          ('monarch', 0.528211236000061),
          ('throne', 0.4935380220413208),
          ('crown', 0.49160435795783997),
          ('aethelred', 0.4883429706096649),
          ('princess', 0.48811477422714233)]
```

```
In [31]: # Get cosine similarities from a vector to an array of vectors
         print("cosine similarity from a random vector to unormalized vector of king")
         wv.cosine_similarities(vec_random, [vec_king_unnorm])
```

```
cosine similarity from a random vector to unormalized vector of king
```

```
Out[31]: array([-0.02642212])
```

```
In [ ]: # Tests analogies based on a text file
        analogy_scores = wv.accuracy('datasets/questions-words.txt')
        #print(analogy_scores)
```

```
In [34]: # The the distance of two words
         print("distance between dog and cat")
         wv.distance("dog","cat")
```

```
distance between dog and cat
```

```
Out[34]: 0.26455658819142336
```

```
In [35]: # Get the distance of a word for the list of word
         print("distance from dog to king and cat")
         wv.distances("dog",["king","cat"])

distance from dog to king and cat


Out[35]: array([0.8180392 , 0.26455647], dtype=float32)

In [ ]: # Evaluate pairs of words
        #wv.evaluate_word_pairs("datasets/SimLex-999.txt")

In [ ]: # Get sentence similarities

        from gensim.models import KeyedVectors
        from gensim.utils import simple_preprocess

        def tokemmized(sentence, vocabulary):
            tokens = [lem(word) for word in simple_preprocess(sentence)]
            return [word for word in tokens if word in vocabulary]

        def compute_sentence_similarity(sentence_1, sentence_2, model_wv):
            vocabulary = set(model_wv.index2word)
            tokens_1 = tokemmized(sentence_1, vocabulary)
            tokens_2 = tokemmized(sentence_2, vocabulary)
            del vocabulary
            print(tokens_1, tokens_2)
            return model_wv.n_similarity(tokens_1, tokens_2)

        similarity = compute_sentence_similarity('this is a sentence', 'this is also a sentenc
        print(similarity,"\n")

        similarity = compute_sentence_similarity('the cat is a mammal', 'the bird is a aves', 
        print(similarity,"\n")

        similarity = compute_sentence_similarity('the cat is a mammal', 'the dog is a mammal',
        print(similarity)

In [37]: # Analogy with not normalized vectors
         print("france is to paris as berlin is to ?")
         wv.most_similar([wv['france'] - wv['paris'] + wv['berlin']])

france is to france as berlin is to ?


Out[37]: [('germany', 0.8275506496429443),
          ('berlin', 0.6908831596374512),
          ('france', 0.6784806251525879),
          ('poland', 0.633112907409668),
          ('german', 0.588524341583252),
          ('russia', 0.5857172012329102),
          ('italy', 0.5818371772766113),
          ('europe', 0.5750494003295898),
          ('austria', 0.5719729065895081),
          ('netherlands', 0.5547516345977783)]
```

```
In [56]: # Analogy with normalized Vector
         vec_france_norm = wv.word_vec('france', use_norm=True)
         vec_paris_norm = wv.word_vec('paris', use_norm=True)
         vec_berlin_norm = wv.word_vec('berlin', use_norm=True)
         vec_germany_norm = wv.word_vec('germany', use_norm=True)
         vec_country_norm = wv.word_vec('country', use_norm=True)
         print("france is to paris as berlin is to ?")
         wv.most_similar([vec_france_norm - vec_paris_norm + vec_berlin_norm])
```

```
france is to paris as berlin is to ?
```

```
Out[56]: [('germany', 0.820073664188385),
          ('berlin', 0.6869513988494873),
          ('france', 0.6479591727256775),
          ('poland', 0.6227378845214844),
          ('german', 0.5865136384963989),
          ('russia', 0.5743523836135864),
          ('italy', 0.5623424053192139),
          ('austria', 0.5614084005355835),
          ('europe', 0.5605403184890747),
          ('netherlands', 0.5417272448539734)]
```

```
In [57]: # Cosine Similarities
         print("cosine_similarities of france and paris")
         print(wv.cosine_similarities(vec_france_norm, [vec_paris_norm]))
         print("cosine_similarities of france and berlin")
         print(wv.cosine_similarities(vec_france_norm, [vec_berlin_norm]))
         print("cosine_similarities of france and country")
         print(wv.cosine_similarities(vec_france_norm, [vec_country_norm]))
```

```
cosine_similarities of france and paris
[0.6420748]
cosine_similarities of france and berlin
[0.36795506]
cosine_similarities of france and country
[0.32212678]
```

```
In [52]: # Analogy
         print("Man is to Woman what King is to ?")
         wv.most_similar([wv['man'] - wv['woman'] + wv['king']])
```

```
Man is to Woman what King is to ?


Out[52]: [('king', 0.80283522605896),
          ('kings', 0.5250919461250305),
          ('prince', 0.5248726606369019),
          ('iii', 0.45816951990127563),
          ('lord', 0.45348936319351196),
          ('iv', 0.45300135016441345),
          ('vi', 0.44630226492881775),
          ('conqueror', 0.4445120692253113),
          ('ii', 0.41359907388687134),
          ('emperor', 0.4134453535079956)]
```

```
In [50]: # Analogy
         print("paris is to france as berlin is to ?")
         wv.most_similar([wv['paris'] - wv['france'] + wv['berlin']])

paris is to france as berlin is to ?


Out[50]: [('berlin', 0.8213962912559509),
          ('munich', 0.6599953770637512),
          ('paris', 0.6203441619873047),
          ('bonn', 0.6191271543502808),
          ('vienna', 0.6118754744529724),
          ('dresden', 0.605978786945343),
          ('leipzig', 0.6040723323822021),
          ('düsseldorf', 0.5992366075515747),
          ('charlottenburg', 0.5932153463363647),
          ('hamburg', 0.5931907892227173)]
```

```
In [58]: # Analogy
         print("cat is to mammal as sparrow is to ?")
         wv.most_similar([wv['cat'] - wv['mammal'] + wv['sparrow']])

cat is to mammal as sparrow is to ?


Out[58]: [('cat', 0.6686296463012695),
          ('sparrow', 0.661421537399292),
          ('kitty', 0.5254508852958679),
          ('ruby', 0.4817608892917633),
          ('kitten', 0.46180397272109985),
          ('rabbit', 0.4520624876022339),
          ('aka', 0.45059120655059814),
          ('dog', 0.44592469930648804),
          ('angel', 0.44519680738449097),
          ('granny', 0.4402121305465698)]
```

```
In [64]: # Analogy
         print("grass is to green as sky is to ?")
         wv.most_similar([wv['sky'] - wv['blue'] + wv['grass']])

grass is to green as sky is to ?


Out[64]: [('grass', 0.7242218255996704),
          ('sky', 0.5100921392440796),
          ('tussocks', 0.46009260416030884),
          ('grasses', 0.4372618794441223),
          ('bermudagrass', 0.42268460988998413),
          ('weeds', 0.4198673367500305),
          ('marram', 0.4184962511062622),
          ('tussac', 0.4171423316001892),
          ('bentgrass', 0.4109356999397278),
          ('skies', 0.4039731025695801)]

In [63]: # Analogy
         print("athens is to greece as baghdad is to ?")
         wv.most_similar([wv['athens'] - wv['greece'] + wv['iraq']])

athens is to greece as baghdad is to ?


Out[63]: [('iraq', 0.7558031678199768),
          ('baghdad', 0.6576923131942749),
          ('afghanistan', 0.6100637316703796),
          ('iraqi', 0.6068007946014404),
          ('tikrit', 0.5783915519714355),
          ('mosul', 0.5627672672271729),
          ('fallujah', 0.5387886762619019),
          ('damascus', 0.5282827615737915),
          ('kabul', 0.5265875458717346),
          ('kuwait', 0.5214947462081909)]

In [17]: wv.most_similar([wv["capital"]+wv["city"]])

Out[17]: [('city', 0.8364958763122559),
          ('capital', 0.8285309672355652),
          ('town', 0.5625525712966919),
          ('cities', 0.5351611971855164),
          ('downtown', 0.5260708928108215),
          ('municipal', 0.5095252990722656),
          ('territory', 0.4946771264076233),
          ('district', 0.48831993341445923),
          ('metropolis', 0.4836964011192322),
          ('region', 0.4836798906326294)]
```

```
In [13]: # Analogy
         print("capital + science")
         wv.most_similar([wv['capital'] + wv['science']])

2019-04-17 18:52:01,516 : INFO : precomputing L2-norms of word weight vectors


athens is to greece as baghdad is to ?


Out[13]: [('science', 0.7434406280517578),
          ('capital', 0.6892884969711304),
          ('sciences', 0.6157187819480896),
          ('biotechnology', 0.5573773384094238),
          ('technology', 0.5531710386276245),
          ('economics', 0.5489161610603333),
          ('humanities', 0.5242317318916321),
          ('informatics', 0.5220810770988464),
          ('institute', 0.5044348239898682),
          ('university', 0.49241507053375244)]

In [ ]:

In [12]: wv.cosine_similarities(wv["education"], [wv["natality"],wv["salubrity"],wv["economy"]]

         #wv.distance("education","natality")

         # education, natality, salubrity, economy

         #wv.most_similar_cosmul(positive=["doctor","woman"], negative=["man"])

Out[12]: array([0.07976063, 0.03727365, 0.31391722], dtype=float32)

In [ ]:
```