pa - play with w2v enwiki lemmatized

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-lemmatized.model"

dictionary_full_wikien_lem_path = "dictionaries/enwiki-20190409-dict-lemmatized.txt.bz2
```

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

```
2019-04-23 12:51:02,012 : INFO : 'pattern' package found; tag filters are available for Englisl
2019-04-23 12:51:02,021 : INFO : loading Word2Vec object from models/enwiki-20190409-lemmatize
2019-04-23 12:52:19,106 : INFO : loading wv recursively from models/enwiki-20190409-lemmatized
2019-04-23 12:52:19,109 : INFO : loading vectors from models/enwiki-20190409-lemmatized.model.
2019-04-23 12:52:19,130 : INFO : setting ignored attribute vectors_norm to None
2019-04-23 12:52:19,135 : INFO : loading vocabulary recursively from models/enwiki-20190409-lem
2019-04-23 12:52:19,137 : INFO : loading trainables recursively from models/enwiki-20190409-lem
2019-04-23 12:52:19,139 : INFO : loading syn1neg from models/enwiki-20190409-lemmatized.model.
2019-04-23 12:52:19,143 : INFO : loading vectors_lockf from models/enwiki-20190409-lemmatized.r
2019-04-23 12:52:19,146 : INFO : setting ignored attribute cum_table to None
2019-04-23 12:52:19,147 : INFO : loaded models/enwiki-20190409-lemmatized.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 [7]: # Testing similarity
        print("Most similar to","woman")
        print(model.wv.most_similar(lem("woman")))
```

```
Most similar to woman
[('man/NN', 0.6361385583877563), ('individual/NN', 0.5763572454452515), ('person/NN', 0.556853
```

```
In [9]: print("Most similar to","doctor")
        print(model.wv.most_similar(lem("doctor")))
```

```
Most similar to doctor
[('dentist/NN', 0.5610849261283875), ('nardole/NN', 0.5584279894828796), ('nurse/NN', 0.556573
```

```
In [ ]:
```

```
In [11]: # Saving some ram by using the KeyedVectors instance
         wv = model.wv
         #del model
```

```
In [12]: # Testing similarity with KeyedVectors
         print("Most similar to","woman")
         print(wv.most_similar(lem("woman")))
         print("\nMost similar to","man")
         print(wv.most_similar(lem("man")))
         print("\nMost similar to","doctor")
         print(wv.most_similar(lem("doctor")))
         print("\nMost similar to","doctor","cosmul")
         print(wv.most_similar_cosmul(positive=[lem("doctor")]))
```

```
Most similar to woman
[('man/NN', 0.6361385583877563), ('individual/NN', 0.5763572454452515), ('person/NN', 0.556853

Most similar to man
[('woman/NN', 0.6361386179924011), ('boy/NN', 0.5653619170188904), ('person/NN', 0.53528153896
```

```
Most similar to doctor
[('dentist/NN', 0.5610849261283875), ('nardole/NN', 0.5584279894828796), ('nurse/NN', 0.556573

Most similar to doctor cosmul
[('dentist/NN', 0.7805417776107788), ('nardole/NN', 0.7792133092880249), ('nurse/NN', 0.778285
```

In [18]:
```python
print("similarity of doctor + woman - man")
wv.most_similar(positive=[lem("doctor"),lem("woman")], negative=[lem("man")])
```

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

Out[18]:
```
[('midwife/NN', 0.6090542078018188),
 ('nurse/NN', 0.5804013609886169),
 ('physician/NN', 0.5530248880386353),
 ('gynaecologist/NN', 0.5421075820922852),
 ('obstetrician/NN', 0.5344318151473999),
 ('medical/JJ', 0.5299170017242432),
 ('midwive/VB', 0.5122523903846741),
 ('anaesthetist/NN', 0.502942681312561),
 ('nursing/NN', 0.5021981000900269),
 ('naakudu/RB', 0.5021182298660278)]
```

In [17]:
```python
# Get cosmul of logic
print("cosmul of doctor + woman - man")
wv.most_similar_cosmul(positive=[lem("doctor"),lem("woman")], negative=[lem("man")])
```

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

Out[17]:
```
[('midwife/NN', 0.9296931624412537),
 ('nurse/NN', 0.8866435289382935),
 ('gynaecologist/NN', 0.8841131329536438),
 ('midwive/VB', 0.8803321123123169),
 ('obstetrician/NN', 0.8797454237937927),
 ('physician/NN', 0.8750578165054321),
 ('medical/JJ', 0.8747599124908447),
 ('midwifery/NN', 0.874646008014679),
 ('nursing/NN', 0.867769181728363),
 ('naturopathic/JJ', 0.8633977770805359)]
```

In [19]:
```python
# Ways to retrive word vector
print("Get item dog")
vec_dog = wv.__getitem__("dog/NN")
vec_dog = wv.get_vector("dog/NN")
vec_dog = wv.word_vec("dog/NN")
print("vec_dog", vec_dog.shape, vec_dog[:10])
```

```
Get item dog
vec_dog (300,) [-0.13817333 -1.8090004  -0.45946687 -2.2184215   1.4197063   0.19401991
 -0.4230487  -2.7905297  -3.1192808   0.02542385]
```

```
In [20]: # 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/NN")
         print(wv.most_similar(positive=[vec_cat]))
```

```
Similar by vector to dog vector at top 10
[('dog/NN', 1.0000001192092896), ('cat/NN', 0.7325705289840698), ('puppy/NN', 0.70179593563079
Most similar to dog vector
[('dog/NN', 1.0000001192092896), ('cat/NN', 0.7325705289840698), ('puppy/NN', 0.70179593563079
Similar to cat vector
[('cat/NN', 1.0), ('dog/NN', 0.7325705885887146), ('meow/VB', 0.6924092769622803), ('kitten/NN
```

```
In [21]: # closer to __ than __
         print("closer to dog than cat")
         print(wv.words_closer_than("dog/NN", "cat/NN"))
         print("\ncloser to cat than dog")
         print(wv.words_closer_than("cat/NN", "dog/NN"))
```

```
closer to dog than cat
[]

closer to cat than dog
[]
```

```
In [22]: # Normalized Vector
         vec_king_norm = wv.word_vec("king/NN", 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/NN", use_norm=False)
         print("vec_king_unnorm:",vec_king_norm.shape, vec_king_unnorm[:10])
```

```
vec_king_norm: (300,) [ 0.02464886  0.09053605  0.00468578 -0.01604057  0.0808396   0.10550086
  0.01262516 -0.0464116  -0.06513052 -0.08347644]
vec_king_unnorm: (300,) [ 0.6677862   2.4528      0.12694712 -0.43457067  2.190104    2.858226
  0.34204054 -1.2573817  -1.764514   -2.2615411 ]
```

```
In [23]: wv.most_similar(positive=[vec_king_norm], negative=[vec_king_unnorm])
```

Out[23]: [('/NN', 0.3219989538192749),
 ('scheidermantel/NN', 0.3141171336174011),
 ('pafnutyevich/JJ', 0.3104780912399292),
 ('samodeyatelnost/NN', 0.3033001720905304),
 ('/JJ', 0.29464849829673767),
 ('zakhozha/NN', 0.2945646047592163),
 ('/NN', 0.29284998774528503),
 ('joenni/NN', 0.28914523124694824),
 ('lyubarina/NN', 0.2868795692920685),
 ('rsheuski/NN', 0.28535693883895874)]

In [24]:
```
# 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
[('parigine/VB', 0.28092506527900696), ('nmcue/NN', 0.2804727852344513), ('mozart_kv/NN', 0.278

 similar to nomalized random vector
[('parigine/VB', 0.28092506527900696), ('nmcue/NN', 0.2804727852344513), ('mozart_kv/NN', 0.278
```

In [25]:
```
# 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[25]: [('parigine/VB', 0.2886022925376892),
 ('kalfhani/NN', 0.27668145298957825),
 ('kriesinger/NN', 0.27662235498428345),
 ('/VB', 0.27649563550949097),
 ('nmcue/NN', 0.27467527985572815),
 ('regent/NN', 0.27348271012306213),
 ('mozart_kv/NN', 0.27183622121810913),
 ('shhtei/NN', 0.2708197832107544),
 ('tabuur/VB', 0.27058061957359314),
 ('hangedup/JJ', 0.2701559066772461)]

In [26]:
```
# 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[26]: [('king/NN', 0.9415208697319031),
          ('prince/NN', 0.6032038927078247),
          ('queen/NN', 0.5944242477416992),
          ('monarch/NN', 0.5747672319412231),
          ('throne/NN', 0.5345853567123413),
          ('crown/NN', 0.5192743539810181),
          ('ruler/NN', 0.5041853189468384),
          ('emperor/NN', 0.48576343059539795),
          ('coronation/NN', 0.475940078496933),
          ('lord/NN', 0.47265052795410156)]
```

```
In [27]: # 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[27]: array([0.10765238])
```

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

```
In [28]: # The the distance of two words
         print("distance between dog and cat")
         wv.distance("dog/NN","cat/NN")
```

```
distance between dog and cat
```

```
Out[28]: 0.2674294870033782
```

```
In [29]: # Get the distance of a word for the list of word
         print("distance from dog to king and cat")
         wv.distances("dog/NN",["king/NN","cat/NN"])
```

```
distance from dog to king and cat
```

```
Out[29]: array([0.81238294, 0.26742947], dtype=float32)
```

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

In [30]:
```
# 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 senten
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)
```

```
['be/VB', 'sentence/NN'] ['be/VB', 'also/RB', 'sentence/NN']
0.9267933550381176

['cat/NN', 'be/VB', 'mammal/NN'] ['bird/NN', 'be/VB', 'ave/NN']
0.6503839221443558

['cat/NN', 'be/VB', 'mammal/NN'] ['dog/NN', 'be/VB', 'mammal/NN']
0.9425444280677167
```

In [31]:
```
# Analogy with not normalized vectors
print("france is to paris as berlin is to ?")
wv.most_similar([wv['france/NN'] - wv['paris/NN'] + wv['berlin/NN']])
```

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

Out[31]:
```
[('germany/NN', 0.7672240138053894),
 ('berlin/NN', 0.6933715343475342),
 ('france/NN', 0.5758201479911804),
 ('uedem/NN', 0.5712798833847046),
 ('gdr/NN', 0.5634602308273315),
 ('osnabrueck/NN', 0.5577783584594727),
```

```
 ('cottbu/NN', 0.5571167469024658),
 ('najallah/NN', 0.5441399812698364),
 ('hüttenjazz/NN', 0.539354145526886),
 ('german/NN', 0.5388710498809814)]
```

In [32]:
```
# Analogy with normalized Vector
vec_france_norm = wv.word_vec('france/NN', use_norm=True)
vec_paris_norm = wv.word_vec('paris/NN', use_norm=True)
vec_berlin_norm = wv.word_vec('berlin/NN', use_norm=True)
vec_germany_norm = wv.word_vec('germany/NN', use_norm=True)
vec_country_norm = wv.word_vec('country/NN', 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[32]:
```
[('germany/NN', 0.7600144743919373),
 ('berlin/NN', 0.6725304126739502),
 ('uedem/NN', 0.5701783299446106),
 ('france/NN', 0.5680463910102844),
 ('gdr/NN', 0.5581510663032532),
 ('cottbu/NN', 0.5506802797317505),
 ('osnabrueck/NN', 0.5495263338088989),
 ('najallah/NN', 0.5433506965637207),
 ('hüttenjazz/NN', 0.537042498588562),
 ('german/NN', 0.5326942801475525)]
```

In [43]:
```
# Cosine Similarities
print("cosine_similarities of france and paris")
print(wv.cosine_similarities(vec_france_norm, [vec_paris_norm]),wv.distance("france/NI
print("cosine_similarities of france and berlin")
print(wv.cosine_similarities(vec_france_norm, [vec_berlin_norm]),wv.distance("france/I
print("cosine_similarities of france and germany")
print(wv.cosine_similarities(vec_france_norm, [vec_germany_norm]),wv.distance("france,
print("cosine_similarities of france and country")
print(wv.cosine_similarities(vec_france_norm, [vec_country_norm]),wv.distance("france,
```

```
cosine_similarities of france and paris
[0.62629485] 0.37370521250384203
cosine_similarities of france and berlin
[0.26217574] 0.7378242844644337
cosine_similarities of france and germany
[0.56096226] 0.4390377899399447
cosine_similarities of france and country
[0.35918537] 0.6408146341093731
```

In [45]:
```
# Analogy
print("king is to man what woman is to ?")
```

```
#wv.most_similar([wv['man/NN'] - wv['woman/NN'] + wv['king/NN']])
wv.most_similar([wv['king/NN'] - wv['man/NN'] + wv['woman/NN']])
```

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

```
Out[45]: [('king/NN', 0.7021986246109009),
          ('queen/NN', 0.5920202732086182),
          ('monarch/NN', 0.5383858680725098),
          ('woman/NN', 0.4814680218696594),
          ('crown/NN', 0.4538986086845398),
          ('ingwenyama/NN', 0.436950147151947),
          ('princess/NN', 0.43597322702407837),
          ('empress/NN', 0.42599907517433167),
          ('regnant/NN', 0.4184303283691406),
          ('ranavalona/NN', 0.416481077671051)]
```

```
In [46]: # Analogy
         print("paris is to france as germany is to ?")
         wv.most_similar([wv['paris/NN'] - wv['france/NN'] + wv['germany/NN']])
```

```
paris is to france as germany is to ?
```

```
Out[46]: [('berlin/NN', 0.7753599882125854),
          ('germany/NN', 0.7352864742279053),
          ('munich/JJ', 0.7241991758346558),
          ('berlin/VB', 0.7004410028457642),
          ('cologne/NN', 0.6728582382202148),
          ('düsseldorf/NN', 0.6541168093681335),
          ('bonn/NN', 0.6338502168655396),
          ('dresden/NN', 0.6333985328674316),
          ('hamburg/NN', 0.6157830953598022),
          ('leipzig/NN', 0.6134828329086304)]
```

```
In [48]: # Analogy
         print("cat is to mammal as sparrow is to ?")
         wv.most_similar([wv['cat/NN'] - wv['mammal/NN'] + wv['bird/NN']])
```

```
cat is to mammal as sparrow is to ?
```

```
Out[48]: [('cat/NN', 0.7435729503631592),
          ('dog/NN', 0.5758434534072876),
          ('kitten/NN', 0.551855742931366),
          ('bird/NN', 0.5491441488265991),
          ('kitty/NN', 0.5458417534828186),
          ('meow/VB', 0.5401268601417542),
          ('meow/NN', 0.5142310261726379),
```

```
 ('poodle/NN', 0.5134133100509644),
 ('goldfish/NN', 0.5111803412437439),
 ('slinky/JJ', 0.49928945302963257)]
```

In [49]:
```python
# Analogy
print("grass is to green as sky is to ?")
wv.most_similar([wv['sky/NN'] - wv['blue/NN'] + wv['green/NN']])
```

```
grass is to green as sky is to ?
```

Out[49]:
```
[('green/NN', 0.576596736907959),
 ('sky/NN', 0.5435831546783447),
 ('green/JJ', 0.3984556496143341),
 ('green/VB', 0.3885582387447357),
 ('jordannick/NN', 0.3508602976799011),
 ('horizon/NN', 0.3487999737262726),
 ('ukip/NN', 0.3445552587509155),
 ('percomi/NN', 0.34198832511901855),
 ('seneley/NN', 0.3393542170524597),
 ('sunlit/NN', 0.32632747292518616)]
```

In [51]:
```python
# Analogy
print("athens is to greece as baghdad is to ?")
wv.most_similar([wv['athens/NN'] - wv['greece/NN'] + wv['afghanistan/NN']])
```

```
athens is to greece as baghdad is to ?
```

Out[51]:
```
[('afghanistan/NN', 0.7056152820587158),
 ('afghan/NN', 0.6274094581604004),
 ('nangarhar/NN', 0.6010676622390747),
 ('taliban/JJ', 0.5929509401321411),
 ('kandahar/NN', 0.5881943702697754),
 ('taliban/VB', 0.5868856906890869),
 ('roghni/NN', 0.5827623009681702),
 ('khost/NN', 0.5813905000686646),
 ('kabul/NN', 0.5794689655303955),
 ('helmand/NN', 0.5781930685043335)]
```

In [56]:
```python
wv.most_similar([wv["country/NN"]])
```

Out[56]:
```
[('country/NN', 0.9999998807907104),
 ('nation/NN', 0.6845932602882385),
 ('region/NN', 0.5479593873023987),
 ('continent/NN', 0.54496169090271),
 ('europe/NN', 0.5181665420532227),
 ('have/VB', 0.4689757823944092),
 ('globally/RB', 0.43926775455474854),
```

```
         ('abroad/RB', 0.4374126195907593),
         ('market/NN', 0.4372479021549225),
         ('number/NN', 0.4358119070529938)]
```

```
In [54]: wv.most_similar([wv["capital/NN"]])
```

```
Out[54]: [('capital/NN', 1.0),
         ('investment/NN', 0.5486246943473816),
         ('asset/NN', 0.4636381268501282),
         ('territory/NN', 0.45464810729026794),
         ('investor/NN', 0.4461580812931061),
         ('bank/NN', 0.4335326552391052),
         ('economy/NN', 0.4294159412384033),
         ('province/NN', 0.4275493323802948),
         ('region/NN', 0.4241411089897156),
         ('xingwang/NN', 0.42348968982696533)]
```

```
In [59]: wv.most_similar([wv["paris/NN"]-wv["capital/NN"]])
```

```
Out[59]: [('paris/NN', 0.5917073488235474),
         ('orsay/JJ', 0.4863584637641907),
         ('colette/VB', 0.4663430452346802),
         ('montparnasse/VB', 0.4581751525402069),
         ('montparnasse/JJ', 0.45304393768310547),
         ('gustave/JJ', 0.45213115215301514),
         ('léonce/VB', 0.4514530599117279),
         ('delpire/NN', 0.4500682055950165),
         ('parisian/JJ', 0.4495515823364258),
         ('romantique/JJ', 0.44724446535110474)]
```

```
In [60]: wv.most_similar([wv["bern/NN"]-wv["capital/NN"]])
```

```
Out[60]: [('bern/NN', 0.5901567935943604),
         ('bern/JJ', 0.5297247171401978),
         ('luzern/NN', 0.45419546961784363),
         ('jürg/NN', 0.45301809906959534),
         ('zurich/JJ', 0.4391050338745117),
         ('bern/VB', 0.41643407940864563),
         ('lüscher/NN', 0.4157090187072754),
         ('zürich/NN', 0.41449370980262756),
         ('basel/JJ', 0.4126679301261902),
         ('herisau/NN', 0.4125199019908905)]
```

```
In [62]: wv.most_similar([wv["switzerland/NN"]-wv["bern/NN"]])
```

```
Out[62]: [('switzerland/NN', 0.5533241033554077),
         ('italy/NN', 0.4750353991985321),
         ('belgium/NN', 0.4626234173774719),
         ('europe/NN', 0.4177972078323364),
```

```
          ('germany/NN', 0.41732004284858704),
          ('argentina/NN', 0.4152482748031616),
          ('finland/NN', 0.4150034487247467),
          ('econometriclink/VB', 0.3965272307395935),
          ('bioavena/NN', 0.3937831521034241),
          ('scandinavia/RB', 0.39317047595977783)]
```

```
In [77]: wv.distance("dog/NN", "dogs/NN")
```

```
Out[77]: 0.30662431795333833
```

```
In [76]: wv.cosine_similarities(wv["dog/NN"],[wv["dogs/NN"]])
```

```
Out[76]: array([0.6933757], dtype=float32)
```

```
In [63]: wv.distance("switzerland/NN", "bern/NN")
```

```
Out[63]: 0.4661005163408918
```

```
In [67]: wv.cosine_similarities(wv["switzerland/NN"],[wv["bern/NN"]])
```

```
Out[67]: array([0.5338995], dtype=float32)
```

```
In [71]: wv.distance("paris/NN", "bern/NN")
```

```
Out[71]: 0.7524347683335195
```

```
In [68]: wv.cosine_similarities(wv["paris/NN"],[wv["bern/NN"]])
```

```
Out[68]: array([0.24756521], dtype=float32)
```

```
In [70]: wv.cosine_similarities(wv["paris/NN"],[wv["dog/NN"]])
```

```
Out[70]: array([0.03346416], dtype=float32)
```

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

```
In [ ]:
```

```
In [ ]: wv.cosine_similarities(wv["education/NN"], [wv["natality/NN"],wv["salubrity/NN"],wv["ec

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

        # education, natality, salubrity, economy

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

```
In [ ]:
```