pa - w2v sentence generator

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 19:10:49,507 : INFO : 'pattern' package found; tag filters are available for Englisl
2019-05-09 19:10:49,518 : INFO : loading Word2Vec object from models/wiki-en-190409-s300-w5-mc
2019-05-09 19:11:49,686 : INFO : loading wv recursively from models/wiki-en-190409-s300-w5-mc1-
2019-05-09 19:11:49,689 : INFO : loading vectors from models/wiki-en-190409-s300-w5-mc1-bw1000
2019-05-09 19:11:49,693 : INFO : setting ignored attribute vectors_norm to None
2019-05-09 19:11:49,697 : INFO : loading vocabulary recursively from models/wiki-en-190409-s30
2019-05-09 19:11:49,699 : INFO : loading trainables recursively from models/wiki-en-190409-s30
2019-05-09 19:11:49,700 : INFO : loading syn1neg from models/wiki-en-190409-s300-w5-mc1-bw1000
2019-05-09 19:11:49,703 : INFO : setting ignored attribute cum_table to None
2019-05-09 19:11:49,704 : 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 [16]: from gensim.utils import simple_preprocess
         import numpy as np
         def tokemmized(sentence, vocabulary):
             return np.array([word for word in simple_preprocess(sentence) if word in vocabula

         def compute_sentence_similarity(sentence_1, sentence_2):
             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 wv.n_similarity(tokens_1, tokens_2)
```

```
In [17]: s1 = 'This room is dirty'
         s2 = 'dirty and disgusting room'

         s1 = 'this is a sentence'
         s2 ='this is also a sentence'

         #from gensim import utils
         #print(utils.lemmatize("The quick brown fox jumps over the lazy dog."))
         #print(utils.lemmatize(s1))

         similarity = compute_sentence_similarity(s1, s2)
         print(similarity,"\n")
```

```
['this' 'is' 'sentence'] ['this' 'is' 'also' 'sentence']
0.9266693658411185
```

```
In [18]: # Translate a string
         vocabulary = set(model.wv.index2word)
         #del vocabulary
```

```
In [19]: def add_vectors(vector_1, vector_2):
             return vector_1 + vector_2 if(vector_1.shape == vector_2.shape) else None


         vec_random_1 = np.random.rand(2,)
         vec_random_2 = np.random.rand(1,)

         print(add_vectors(vec_random_1,vec_random_2))
```

```
None
```

```
In [95]: def translate_sentence(sentence, vector, operation, verbose=False):
             tokens = tokemmized(sentence, vocabulary)
             #print(operation(tokens,tokens))
             #print(tokens.shape)
```

```
        if verbose: print(tokens,"\n")
        #print(vector)
        #print(np.array([wv.word_vec(token) for token in tokens]))
        #print(np.array([wv.word_vec(token, use_norm=False) for token in tokens]))

        sentence_vector = np.array([wv.word_vec(token, use_norm=False) for token in tokens
        normed_sentence_vector = np.array(sentence_vector / np.linalg.norm(sentence_vecto
        normed_vector = np.array(vector / np.linalg.norm(vector))

        #print(sentence_vector)
        #print(normed_vector)
        #tokens_vector = np.array([my_vector+normed_vector for my_vector in sentence_vect
        #print(tokens_vector.shape)

        generated_sentence = []
        if verbose: print(wv.most_similar(positive=[normed_vector]),"\n")
        for token_id in range(len(normed_sentence_vector)):
            #print(token_id)
            output = normed_sentence_vector[token_id]+normed_vector[token_id]
            #print(output)
            if verbose: print(wv.most_similar(positive=[normed_sentence_vector[token_id]]
            if verbose: print(wv.most_similar(positive=[output]))
            if verbose: print("\n")
            #print(model.wv.most_similar(positive=[tokens_vector[0]]))
            #wv.most_similar([model.wv.word_vec['capital'] + model.wv.word_vec['science']
            generated_sentence.append(wv.most_similar(positive=[output], topn=1)[0][0])
            #print(generated_sentence)

        return generated_sentence
```

```
In [ ]: #def get_random_word(vocabulary):
        #    np.random.choice(vocabulary, 1)
```

```
In [57]: #print(translate_sentence(s1, np.random.rand(300,)*np.random.rand(300,),add_vectors,v
         print(translate_sentence(s1, np.random.rand(300,)+wv["king"],add_vectors,verbose=False
         print(translate_sentence(s1, np.random.rand(300,)-wv["king"],add_vectors,verbose=False
         print(translate_sentence(s1, np.random.rand(300,)*wv["king"],add_vectors,verbose=False
         print(translate_sentence(s1, np.random.rand(300,)/wv["king"],add_vectors,verbose=False
         print(translate_sentence(s1, np.random.rand(300,)%wv["king"],add_vectors,verbose=False
```

```
[('king', 0.9372287392616272), ('queen', 0.6690690517425537), ('prince', 0.6481649875640869),

['this', 'convinced', 'sentence']
[('knocknaskeharoe', 0.2645658850669861), ('incom', 0.2635432481765747), ('neshwille', 0.26234

['this', 'is', 'sentence']
[('king', 0.8530193567276001), ('prince', 0.5861936807632446), ('queen', 0.5642796158790588),
['this', 'convinced', 'danceworks']
[('', 0.30970412492752075), ('vcissara', 0.29018083214759827), ('bobtails', 0.28184133768081665

['this', 'convinced', 'danceworks']
[('king', 0.8441299200057983), ('queen', 0.5206315517425537), ('prince', 0.5197598934173584),

['this', 'convinced', 'danceworks']
```

```
In [ ]: #print(translate_sentence(s1, np.random.rand(300,)*np.random.rand(300,),add_vectors,ve
        #print(translate_sentence(s1, np.random.rand(300,)+wv["king"],add_vectors,verbose=Fals
        #print(translate_sentence(s1, np.random.rand(300,)-wv["king"],add_vectors,verbose=Fals
        #print(translate_sentence(s1, np.random.rand(300,)*wv["king"],add_vectors,verbose=Fals
        #print(translate_sentence(s1, np.random.rand(300,)/wv["king"],add_vectors,verbose=Fals
        #print(translate_sentence(s1, np.random.rand(300,)%wv["king"],add_vectors,verbose=Fals
```

```
In [63]: print(translate_sentence(s1, wv["king"],add_vectors,verbose=False),"\n")
         print(translate_sentence(s1, wv["science"],add_vectors,verbose=False),"\n")
         print(translate_sentence(s1, wv["dog"],add_vectors,verbose=False),"\n")
```

```
[('king', 0.9999999403953552), ('queen', 0.6651210784912109), ('prince', 0.6620543599128723),

['this', 'convinced', 'danceworks']

[('science', 0.9999999403953552), ('sciences', 0.7399516105651855), ('physics', 0.637518525123

['this', 'convinced', 'sentence']

[('dog', 1.0), ('dogs', 0.7544571161270142), ('cat', 0.7354434728622437), ('puppy', 0.68804031

['attended', 'is', 'sentence']
```

```
In [69]: sentence = "Capital of science"
         print(translate_sentence(sentence, np.random.rand(300,)*np.random.rand(300,)+10*np.ra
```

```
[('feminique', 0.2670329213142395), ('gawding', 0.26683396100997925), ('mystic', 0.26484486460

['capital', 'of', 'science']
```

```
In [70]: sentence = "Capital of science"
         print(translate_sentence(sentence, wv["dog"],add_vectors,verbose=False),"\n")
```

```
[('dog', 1.0), ('dogs', 0.7544571161270142), ('cat', 0.7354434728622437), ('puppy', 0.68804031

['annie', 'of', 'science']
```

```
In [71]: #definition of capital
         sentence = "A town or city that is the official seat of government in a political enti
         print(translate_sentence(sentence, wv["science"],add_vectors,verbose=False),"\n")

[('science', 0.9999999403953552), ('sciences', 0.7399516105651855), ('physics', 0.637518525123

['town', 'tetlow', 'city', 'that', 'convinced', 'the', 'overwhelm', 'seat', 'of', 'lovestruck'
```

```
In [72]: sentence = "the financial capital of the world is wall street"
         print(translate_sentence(sentence, wv["science"],add_vectors,verbose=False),"\n")

[('science', 0.9999999403953552), ('sciences', 0.7399516105651855), ('physics', 0.637518525123

['the', 'fallenbrunnen', 'capital', 'of', 'attended', 'world', 'convinced', 'wall', 'street']
```

```
In [96]: sentence = "the financial capital of the world is wall street"
         print(translate_sentence(sentence, wv["science"],add_vectors,verbose=False),"\n")

['gawding', 'financial', 'capital', 'gawding', 'the', 'world', 'is', 'wall', 'street']
```

```
In [91]: random_vector = np.random.choice(np.array(list(vocabulary)))
         print(wv.most_similar(positive=[random_vector]),"\n")

[('', 0.9640800952911377), ('', 0.9301100969314575), ('', 0.9295684099197388), ('yakimaki', 0.
```

```
In [92]: sentence = "the financial capital of the world is wall street"
         print(translate_sentence(sentence, wv[np.random.choice(np.array(list(vocabulary)))],a

['the', 'fallenbrunnen', 'annie', 'recalled', 'the', 'world', 'is', 'wall', 'street']
```

```
In [94]: sentence = "the financial capital of the world is wall street"
         my_random_vector = np.random.choice(np.array(list(vocabulary)))
         print(wv.most_similar(positive=[my_random_vector]),"\n")
         print(translate_sentence(sentence, wv[my_random_vector],add_vectors,verbose=False),"\

[('talgo', 0.4247443675994873), ('renfe', 0.4004928171634674), ('feve', 0.3839436173439026), (

['attended', 'financial', 'capital', 'recalled', 'attended', 'world', 'convinced', 'urbanathlo
```

```
In [59]: #https://www.phrasemix.com/collections/the-50-most-important-english-proverbs
         print(translate_sentence("Two wrongs don't make a right.", wv[random_vector],add_vecto

[('dog', 1.0), ('dogs', 0.7544571161270142), ('cat', 0.7354434728622437), ('puppy', 0.68804031

['snobbishness', 'wrongs', 'don', 'make', 'meritoriois']
```

```
In [60]: print(translate_sentence("The pen is mightier than the sword.", wv[random_vector],add_

[('dog', 1.0), ('dogs', 0.7544571161270142), ('cat', 0.7354434728622437), ('puppy', 0.68804031

['attended', 'pen', 'is', 'mightier', 'anaharlick', 'attended', 'sword']
```

```
In [61]: print(translate_sentence("When in Rome, do as the Romans.", wv[random_vector],add_vec

[('dog', 1.0), ('dogs', 0.7544571161270142), ('cat', 0.7354434728622437), ('puppy', 0.68804031

['serves', 'in', 'rome', 'do', 'appealed', 'attended', 'romans']
```

```
In [62]: print(translate_sentence("The squeaky wheel gets the grease.", wv[random_vector],add_

[('dog', 1.0), ('dogs', 0.7544571161270142), ('cat', 0.7354434728622437), ('puppy', 0.68804031

['attended', 'squeaky', 'wheel', 'gets', 'attended', 'nanzan']
```

```
In [64]: sentence = "When the going gets tough, the tough get going."
         print(translate_sentence(sentence, wv[random_vector],add_vectors,verbose=False),"\n")

[('dog', 1.0), ('dogs', 0.7544571161270142), ('cat', 0.7354434728622437), ('puppy', 0.68804031

['serves', 'the', 'going', 'gets', 'treferig', 'attended', 'tough', 'founded', '']
```

```
In [65]: sentence = "No man is an island."
         print(translate_sentence(sentence, wv[random_vector],add_vectors,verbose=False),"\n")

[('dog', 1.0), ('dogs', 0.7544571161270142), ('cat', 0.7354434728622437), ('puppy', 0.68804031

['umuokiri', 'man', 'is', 'an', 'xylochaerus']
```

```
In [66]: sentence = "Fortune favors the bold."
         print(translate_sentence(sentence, wv[random_vector],add_vectors,verbose=False),"\n")

[('dog', 1.0), ('dogs', 0.7544571161270142), ('cat', 0.7354434728622437), ('puppy', 0.68804031

['rccs', 'favors', 'the', 'bold']
```

```
In [67]: sentence = "People who live in glass houses should not throw stones."
         print(translate_sentence(sentence, wv[random_vector],add_vectors,verbose=False),"\n")

[('dog', 1.0), ('dogs', 0.7544571161270142), ('cat', 0.7354434728622437), ('puppy', 0.68804031

['lorrey', 'who', 'live', 'in', 'yrrl', 'ragone', 'should', 'attended', 'postindustrial', 'sto
```

```
In [68]: sentence = "Hope for the best, but prepare for the worst."
         print(translate_sentence(sentence, wv[random_vector],add_vectors,verbose=False),"\n")

[('dog', 1.0), ('dogs', 0.7544571161270142), ('cat', 0.7354434728622437), ('puppy', 0.68804031

['crystengcomm', 'for', 'the', 'best', 'attended', 'murmelschwein', 'for', 'attended', 'maikar
```

```
In [73]: sentence = "Better late than never."
         print(translate_sentence(sentence, wv[random_vector],add_vectors,verbose=False),"\n")

[('dog', 1.0), ('dogs', 0.7544571161270142), ('cat', 0.7354434728622437), ('puppy', 0.68804031

['kitchissipi', 'late', 'than', 'never']
```

```
In [74]: sentence = "Birds of a feather flock together."
         print(translate_sentence(sentence, wv[random_vector],add_vectors,verbose=False),"\n")

[('dog', 1.0), ('dogs', 0.7544571161270142), ('cat', 0.7354434728622437), ('puppy', 0.68804031

['osostolus', 'of', 'feather', 'flock', 'extortionary']
```

```
In [75]: sentence = "Keep your friends close and your enemies closer."
         print(translate_sentence(sentence, wv[random_vector],add_vectors,verbose=False),"\n")
```

```
[('dog', 1.0), ('dogs', 0.7544571161270142), ('cat', 0.7354434728622437), ('puppy', 0.68804031

['scholar', 'your', 'friends', 'close', 'recalled', 'pulaman', 'enemies', 'frankalmoinage']
```

```
In [76]: sentence = "A picture is worth a thousand words."
         print(translate_sentence(sentence, wv[random_vector],add_vectors,verbose=False),"\n")
```

```
[('dog', 1.0), ('dogs', 0.7544571161270142), ('cat', 0.7354434728622437), ('puppy', 0.68804031

['mobilized', 'is', 'worth', 'thousand', '']
```

```
In [77]: sentence = "There's no such thing as a free lunch."
         print(translate_sentence(sentence, wv[random_vector],add_vectors,verbose=False),"\n")
```

```
[('dog', 1.0), ('dogs', 0.7544571161270142), ('cat', 0.7354434728622437), ('puppy', 0.68804031

['mmcaa', 'no', 'such', 'thing', 'appealed', 'wurznbacher', 'lunch']
```

```
In [126]: sentence = "There's no place like home."
          print(translate_sentence(sentence, wv["yellow"],add_vectors,verbose=False),"\n")
```

```
['there', 'no', 'place', 'like', 'home']
```

```
In [ ]:

In [ ]:

In [101]: print("Man is to Woman what King is to ?")
          wv.most_similar([wv['wallstreet'] - wv['finance'] + wv['switzerland']])
```

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

```
Out[101]: [('switzerland', 0.6374906301498413),
           ('gälltofta', 0.4903566241264343),
           ('dubendorf', 0.48968037962913513),
           ('pratteln', 0.45561110973358154),
           ('notwil', 0.44998636841773987),
           ('dzhirkvelov', 0.44828879833221436),
           ('futuresgeneva', 0.44405433535575867),
           ('nottwil', 0.43685224652290344),
           ('gruyeres', 0.4326044023036957),
           ('macolin', 0.4323049783706665)]
```

```
In [ ]:
```