Appendix

.1 Jupyter Notebooks

Appendix . Appendix

.1.1 pa-build-dictionary

pa - build dictionary

June 2, 2019

In [1]:
```
# Start logging process at root level
import logging
log_filename = "logs/pa-build-dictionary.log"
logging.basicConfig(filename=log_filename, format='%(asctime)s : %(levelname)s : %(mes
logging.root.setLevel(level=logging.INFO)
```

In [ ]:
```
# Build dictionary
from gensim import corpora
from gensim.corpora import WikiCorpus
import gensim.downloader as api

datafile_src = "datasets/enwiki-latest-pages-articles.xml.bz2"
#datafile_name = "datasets/chunks/enwiki-chunks-999/enwiki-chunk-999-1.xml.bz2"
#unfiltered_dictionary_name = "dictionaries/enwiki-chunk-999-1_lem.txt.bz2"
#filtered_dictionary_name = "dictionaries/enwiki-filtered-20-10-100000.txt.bz2"
unlemmatized_dictionary_name = "dictionaries/enwiki-20190409-dict-unlemmatized.txt.bz2
lemmatized_dictionary_name = "dictionaries/enwiki-20190409-dict-lemmatized.txt.bz2"

must_lemmatize = True

if must_lemmatize:
    dictionary_name = lemmatized_dictionary_name
else:
    dictionary_name = unlemmatized_dictionary_name

#datafile_name = api.load("text8")
#datafile_name = [wd for wd in datafile_name]
#unfiltered_dictionary_name = "dictionaries/text8.txt.bz2"
#filtered_dictionary_name = "dictionaries/text8-filtered-20-10-100000.txt.bz2"

#dct = corpora.Dictionary(datafile_name)
wiki = WikiCorpus(datafile_src, lemmatize=must_lemmatize) #False to no use lemmatizati
wiki.dictionary.save_as_text(dictionary_name)
#dct.save_as_text(unfiltered_dictionary_name)

# Remove words occuring less than 20 times, and words occuring in more than 10% of the
#wiki.dictionary.filter_extremes(no_below=20, no_above=0.1, keep_n=100000)
```

pa-build-dictionary

```
        #wiki.dictionary.save_as_text(filtered_dictionary_name)

        del wiki
        #del dct
```

In [ ]:

In [ ]:

In [ ]:
```
# Load dictionary from file
datafile_name_2 = "datasets/chunks/enwiki-chunks-9999/enwiki-chunk-9999-1.xml.bz2"
unfiltered_dictionary_name_2 = "dictionaries/enwiki-chunk-9999-1_2.txt.bz2"

from gensim.corpora import Dictionary
dictionary = Dictionary.load_from_text(unfiltered_dictionary_name)

wiki_2 = WikiCorpus(datafile_name_2)
dictionary.add_documents(wiki_2)

dictionary.save_as_text(unfiltered_dictionary_name_2)

del wiki_2
```

In [ ]:

.1.2 pa-build-word2vec-on-splits

pa - build word2vec on splits

June 2, 2019

```
In [1]: # Start logging process at root level
        import logging
        logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.
        logging.root.setLevel(level=logging.INFO)

In [2]: import multiprocessing
        import os, os.path

        chunk_size = 99999
        chunk_start_at = 5
        chunks_basepath = "datasets/chunks/enwiki-chunks-"+str(chunk_size)
        model_path = "models/enwiki-full-dict-"+str(chunk_size)+".model"
        dictionary_path = "dictionaries/enwiki-english-lemmatized.txt.bz2"
        chunks_count = len([name for name in os.listdir(chunks_basepath) if os.path.isfile(os.
        #chunks_count = 4

        lemmatization = True
        w2v_size = 300
        w2v_window = 5
        #w2v_cores = multiprocessing.cpu_count()-10
        w2v_cores = 20
        w2v_min_count = 1
        w2v_epochs = 5

In [3]: # Load dictionary from file
        from gensim.corpora import Dictionary
        dictionary = Dictionary.load_from_text(dictionary_path)

2019-04-03 00:21:45,633 : INFO : 'pattern' package found; tag filters are available for Englis

In [4]: # Initialize simple sentence iterator required for the Word2Vec model / memory savior
        class SentencesIterator:
            def __init__(self, wiki):
                self.wiki = wiki

            def __iter__(self):
                for sentence in self.wiki.get_texts():
                    yield list(map(lambda x: x.decode('utf-8'), sentence)) #set encode('utf-8'
```

pa-build-word2vec-on-splits

```
In [ ]: import os
        from gensim.models import Word2Vec
        from gensim.corpora import WikiCorpus

        for e in range(chunk_start_at,chunks_count+1):
            chunk_name = chunks_basepath+"/"+"enwiki-chunk-"+str(chunk_size)+"-"+str(e)+".xml.l
            model_backup_basepath = "models/enwiki-full-dict-parts-"+str(chunk_size)
            model_backup_path = model_backup_basepath+"/enwiki-full-dict-"+str(chunk_size)+"-pa

            if not os.path.exists(model_backup_basepath):
                print("Mkdir the model base path")
                os.mkdir(model_backup_basepath)

            # Build WikiCorpus based on the dictionary
            wiki = WikiCorpus(chunk_name, dictionary=dictionary,lemmatize=lemmatization)

            # Set generator
            sentences = SentencesIterator(wiki)

            print("Running with: ",str(w2v_cores)," cores")
            print("Selected model:",model_path)
            if not os.path.exists(model_path):
                print("Building model on:",chunk_name)
                # Build model on first part
                model = Word2Vec(sentences=sentences,
                                 size=w2v_size,
                                 min_count=w2v_min_count,
                                 window=w2v_window,
                                 workers=w2v_cores,
                                 iter=w2v_epochs
                                )
                model.save(model_path)
                model.save(model_backup_path)
                print("Model saved")
            else:
                # Train model on anthor part
                print("Training model on:",chunk_name)
                model = Word2Vec.load(model_path)
                model.train(sentences=sentences, total_examples=model.corpus_count, epochs=mod
                model.save(model_path)
                model.save(model_backup_path)
                print("Model updated")

            print("Backup model saved:",model_backup_path)

            del wiki
            del model
            del sentences
```

Appendix . Appendix

pa-build-word2vec-on-splits

```
        del dictionary
        print("Done.")
```

In [ ]:

.1.3 pa-play-with-w2v-enwiki-lemmatized

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.
        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.bz

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 Englis
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-le
2019-04-23 12:52:19,137 : INFO : loading trainables recursively from models/enwiki-20190409-le
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.
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)
```

Appendix . Appendix

pa-play-with-w2v-enwiki-lemmatized

```
        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
```

pa-play-with-w2v-enwiki-lemmatized

```
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]: 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]: # 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]: # 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])
```

Appendix . Appendix

pa-play-with-w2v-enwiki-lemmatized

```
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])
```

pa-play-with-w2v-enwiki-lemmatized

```
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])
```

Appendix . Appendix

pa-play-with-w2v-enwiki-lemmatized

```
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")
```

pa-play-with-w2v-enwiki-lemmatized

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),
```

pa-play-with-w2v-enwiki-lemmatized

```
          ('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/NN
         print("cosine_similarities of france and berlin")
         print(wv.cosine_similarities(vec_france_norm, [vec_berlin_norm]),wv.distance("france/N
         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 ?")
```

pa-play-with-w2v-enwiki-lemmatized

```
        #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),
```

Appendix . Appendix

pa-play-with-w2v-enwiki-lemmatized

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

In [49]: # 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]: # 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]: 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),
```

pa-play-with-w2v-enwiki-lemmatized

```
          ('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),
```

Appendix . Appendix

pa-play-with-w2v-enwiki-lemmatized

```
          ('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["e

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

        # education, natality, salubrity, economy

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

```
In [ ]:
```

.1.4 pa-play-with-w2v-enwiki-unlemmatized

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.
        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):
```

pa-play-with-w2v-enwiki-unlemmatized

```
            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"]))
```

pa-play-with-w2v-enwiki-unlemmatized

```
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),
```

Appendix . Appendix

pa-play-with-w2v-enwiki-unlemmatized

```
          ('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),
```

pa-play-with-w2v-enwiki-unlemmatized

```
          ('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
[]
```

Appendix . Appendix

pa-play-with-w2v-enwiki-unlemmatized

```
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', 0.26340371370315

 similar to nomalized random vector
[('vicarios', 0.27943286299705505), ('by', 0.2677918076515198), ('pyrophore', 0.26340371370315


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
```

pa-play-with-w2v-enwiki-unlemmatized

```
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
```

Appendix . Appendix

pa-play-with-w2v-enwiki-unlemmatized

```
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),
```

pa-play-with-w2v-enwiki-unlemmatized

```
          ('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']])
```

Appendix . Appendix

pa-play-with-w2v-enwiki-unlemmatized

```
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)]
```

pa-play-with-w2v-enwiki-unlemmatized

```
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)]
```

Appendix . Appendix

pa-play-with-w2v-enwiki-unlemmatized

```
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 [ ]:
```

.1.5 pa-w2v-explore-models

pa - w2v explore models

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.
        logging.root.setLevel(level=logging.INFO)

In [3]: # Load model and dictionary
        dictrionary_root_path = "dictionaries/"
        dictionary_unlem_path = dictrionary_root_path+"enwiki-20190409-dict-unlemmatized.txt.b
        dictionary_lem_path = dictrionary_root_path+"enwiki-20190409-dict-lemmatized.txt.bz2"
        is_lemmatized = False

In [4]: import os
        model_root_path = "models/"
        models_list = [name for name in os.listdir(model_root_path) if os.path.isfile(os.path.
        print(len(models_list))

52


In [5]: from sklearn.manifold import TSNE
        tsne_model = TSNE(perplexity=40, n_components=2, init='pca', n_iter=2500, random_state
        #fitted_values = tsne_model.fit_transform(tokens)

In [6]: from gensim.models import Word2Vec

        word_to_plot = "woman"
        top_similar = 100

        for model_name in models_list:
            if "-lem" in model_name:
                dictionary_path = dictionary_lem_path
            else:
                dictionary_path = dictionary_unlem_path

            #print("loading model", model)
```

Appendix . Appendix

pa-w2v-explore-models

```
            model = Word2Vec.load(os.path.join(model_root_path, model_name), mmap='r')
            print("model loaded")

            word_vector = model.wv.most_similar(positive=[word_to_plot], topn=top_similar)
            print("word_vector loaded")

            word_vocabulary = [word_to_plot]
            for element in word_vector:
                element_name = element[0]
                if element_name not in word_vocabulary:
                    word_vocabulary.append(element_name)
            #print(word_vocabulary)
            print("word_vocabulary loaded")

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

            #print(tokens)
            #print(labels)

            fitted_values = tsne_model.fit_transform(tokens)

            break
```

```
2019-05-20 18:27:32,255 : INFO : 'pattern' package found; tag filters are available for Englis
2019-05-20 18:27:32,271 : INFO : loading Word2Vec object from models/wiki-en-190409-s300-w5-mc
2019-05-20 18:27:45,912 : INFO : loading wv recursively from models/wiki-en-190409-s300-w5-mc5
2019-05-20 18:27:45,914 : INFO : loading vectors from models/wiki-en-190409-s300-w5-mc5-bw1000
2019-05-20 18:27:45,919 : INFO : setting ignored attribute vectors_norm to None
2019-05-20 18:27:45,922 : INFO : loading vocabulary recursively from models/wiki-en-190409-s30
2019-05-20 18:27:45,924 : INFO : loading trainables recursively from models/wiki-en-190409-s30
2019-05-20 18:27:45,926 : INFO : loading syn1neg from models/wiki-en-190409-s300-w5-mc5-bw1000
2019-05-20 18:27:45,929 : INFO : setting ignored attribute cum_table to None
2019-05-20 18:27:45,930 : INFO : loaded models/wiki-en-190409-s300-w5-mc5-bw10000-cbow-i5-c1-u
2019-05-20 18:27:56,551 : INFO : precomputing L2-norms of word weight vectors

model loaded
word_vector loaded
word_vocabulary loaded

/home/rclaret/anaconda3/envs/py36/lib/python3.6/site-packages/ipykernel_launcher.py:30: Deprec
```

```
In [13]: from gensim.models import Word2Vec
```

pa-w2v-explore-models

```
word_to_plot = "man"
top_similar = 100

for model_name in models_list:
    if "-lem" in model_name:
        dictionary_path = dictionary_lem_path
    else:
        dictionary_path = dictionary_unlem_path

    #print("loading model", model)
    model = Word2Vec.load(os.path.join(model_root_path, model_name), mmap='r')
    print("model loaded")

    word_vector = model.wv.most_similar(positive=[word_to_plot], topn=top_similar)
    print("word_vector loaded")

    word_vocabulary = [word_to_plot]
    for element in word_vector:
        element_name = element[0]
        if element_name not in word_vocabulary:
            word_vocabulary.append(element_name)
    #print(word_vocabulary)
    print("word_vocabulary loaded")

    labels = []
    tokens = []
    banned_words = ["creature", "monster"]
    for word in word_vocabulary:
        if word not in banned_words:
            tokens.append(model[word])
            labels.append(word)

    #print(tokens)
    #print(labels)

    fitted_values = tsne_model.fit_transform(tokens)

    break
```

```
2019-05-20 19:58:23,469 : INFO : loading Word2Vec object from models/wiki-en-190409-s300-w5-mc
2019-05-20 19:58:35,566 : INFO : loading wv recursively from models/wiki-en-190409-s300-w5-mc5
2019-05-20 19:58:35,568 : INFO : loading vectors from models/wiki-en-190409-s300-w5-mc5-bw1000
2019-05-20 19:58:35,573 : INFO : setting ignored attribute vectors_norm to None
2019-05-20 19:58:35,575 : INFO : loading vocabulary recursively from models/wiki-en-190409-s30
2019-05-20 19:58:35,576 : INFO : loading trainables recursively from models/wiki-en-190409-s30
2019-05-20 19:58:35,578 : INFO : loading syn1neg from models/wiki-en-190409-s300-w5-mc5-bw1000
2019-05-20 19:58:35,582 : INFO : setting ignored attribute cum_table to None
2019-05-20 19:58:35,583 : INFO : loaded models/wiki-en-190409-s300-w5-mc5-bw10000-cbow-i5-c1-u
```

Appendix . Appendix

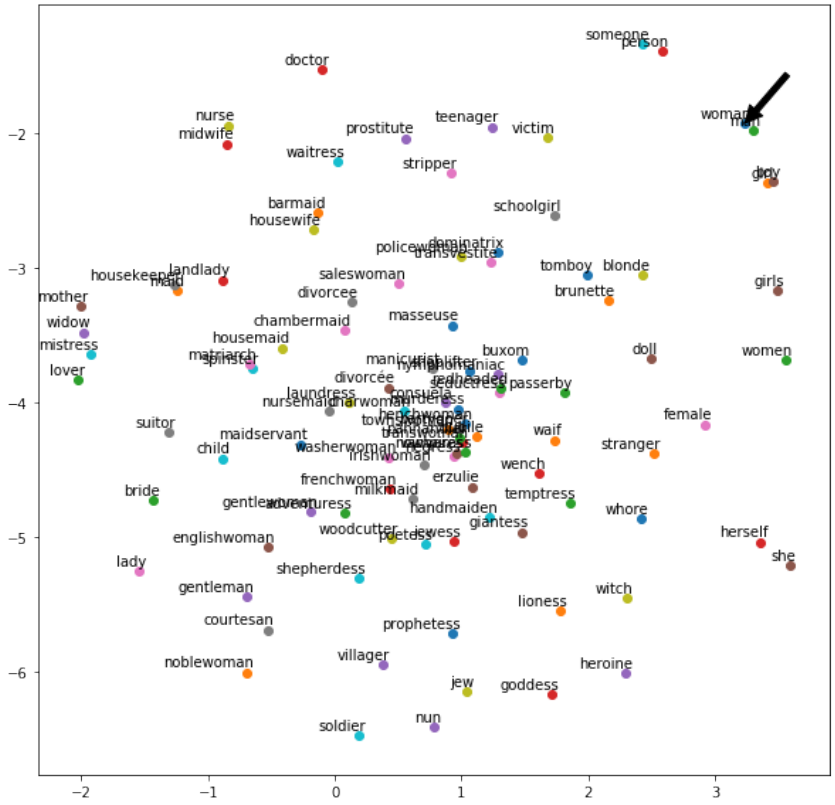
pa-w2v-explore-models

```
2019-05-20 19:58:50,790 : INFO : precomputing L2-norms of word weight vectors


model loaded
word_vector loaded
word_vocabulary loaded


/home/rclaret/anaconda3/envs/py36/lib/python3.6/site-packages/ipykernel_launcher.py:32: Deprec
```

```
In [11]: import matplotlib.pyplot as plt
         #%matplotlib widget
         #%matplotlib notebook
         %matplotlib inline
```

```
In [8]: plt.figure(figsize=(10, 10))
        def plot_word_vector(fitted_values):
            x = []
            y = []
            for value in fitted_values:
                x.append(value[0])
                y.append(value[1])


            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')
                if labels[i]==word_to_plot:
                    plt.annotate(word_to_plot, xy=(x[i], y[i]), xytext=(5, 0),
                    arrowprops=dict(facecolor='black', shrink=0.8),fontsize = 1,
                    )
            plt.show()
        plot_word_vector(fitted_values)
```

pa-w2v-explore-models

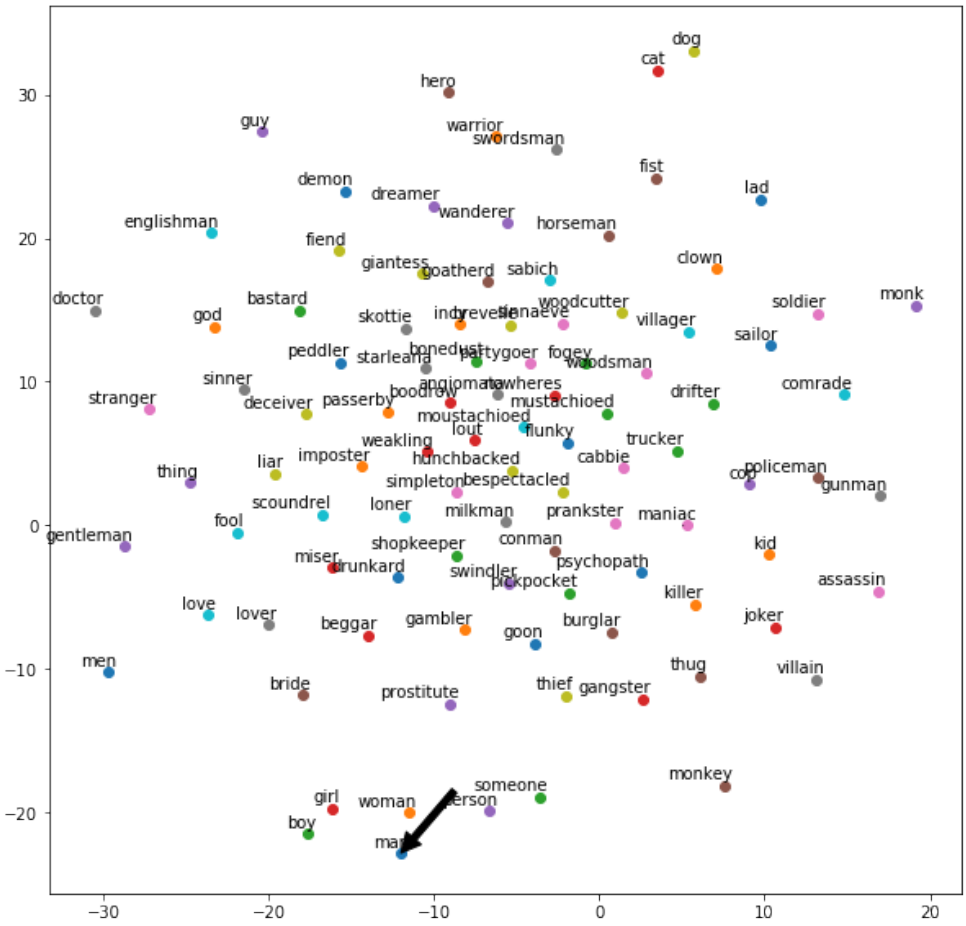



```
In [14]: plt.figure(figsize=(10, 10))
         def plot_word_vector(fitted_values):
             x = []
             y = []
             for value in fitted_values:
                 x.append(value[0])
                 y.append(value[1])


             for i in range(len(x)):
                 plt.scatter(x[i],y[i])
                 plt.annotate(labels[i],
                              xy=(x[i], y[i]),
                              xytext=(5, 2),
```

Appendix . Appendix

pa-w2v-explore-models

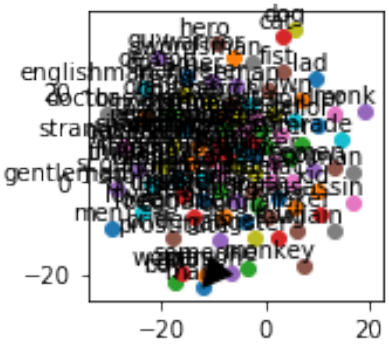
```
                        textcoords='offset points',
                        ha='right',
                        va='bottom')
            if labels[i]==word_to_plot:
                plt.annotate(word_to_plot, xy=(x[i], y[i]), xytext=(5, 0),
                arrowprops=dict(facecolor='black', shrink=0.8),fontsize = 1,
                )
        plt.show()
plot_word_vector(fitted_values)
```

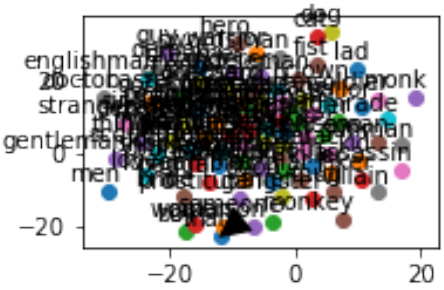



```
In [20]: from sklearn.manifold import TSNE

         plt.figure(figsize=(5, 5))
```

pa-w2v-explore-models

```
tsne_model = TSNE(perplexity=40, n_components=2, init='pca', n_iter=2500, random_stat
plt.subplot(2,2,1)
plot_word_vector(tsne_model.fit_transform(tokens))

tsne_model = TSNE(perplexity=40, n_components=2, init='pca', n_iter=2500, random_stat
plt.subplot(2,2,2)
plot_word_vector(tsne_model.fit_transform(tokens))
```






```
In [14]: from ipywidgets import *
         import numpy as np
         import matplotlib.pyplot as plt

In [15]: def update(w):
             line.set_ydata(np.sin(w * x))
             fig.canvas.draw()

         x = np.linspace(0, 2 * np.pi)
         fig = plt.figure()
         ax = fig.add_subplot(1, 1, 1)
```

Appendix . Appendix

pa-w2v-explore-models

```
        line, = ax.plot(x, np.sin(x))
        interact(update, w=widgets.IntSlider(min=-10,max=30,step=1,value=10,continuous_update=
```

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

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

```
interactive(children=(IntSlider(value=10, continuous_update=False, description='w', max=30, mi
```

In [ ]:

.1.6 pa-w2v-explore-vector-influence

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.
        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 Englis
2019-05-09 21:43:53,254 : INFO : loading Word2Vec object from models/wiki-en-190409-s300-w5-mc
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
```

Appendix . Appendix

pa-w2v-explore-vector-influence

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]:
```
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,
```

pa-w2v-explore-vector-influence

```
 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,
```

Appendix . Appendix

pa-w2v-explore-vector-influence

```
                -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()
```

pa-w2v-explore-vector-influence

```
        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]: 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-vect
                      break
```

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

1 : [('federer', 0.9674688577651978), ('nadal', 0.7478001117706299), ('djokovic', 0.7464911341
2 : [('federer', 0.981936514377594), ('nadal', 0.7617033123970032), ('djokovic', 0.75779414176
3 : [('federer', 0.9720087051391602), ('nadal', 0.7415935397148132), ('djokovic', 0.7396956682
4 : [('federer', 0.9875750541687012), ('djokovic', 0.7640445828437805), ('nadal', 0.7609711885
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
```

Appendix . Appendix

pa-w2v-explore-vector-influence

```
96 : [('federer', 0.9877071976661682), ('nadal', 0.76472008228302), ('djokovic', 0.76281607151
97 : [('federer', 0.9792643785476685), ('djokovic', 0.7504821419715881), ('nadal', 0.747939825
99 : [('federer', 0.9359966516494751), ('nadal', 0.7525387406349182), ('djokovic', 0.737351238
109 : [('federer', 0.9804996848106384), ('nadal', 0.7527401447296143), ('djokovic', 0.75010502
110 : [('federer', 0.983569860458374), ('nadal', 0.7643581032752991), ('djokovic', 0.764225661
111 : [('federer', 0.9852062463760376), ('nadal', 0.751778244972229), ('djokovic', 0.749609708
112 : [('federer', 0.9834575653076172), ('djokovic', 0.7717949151992798), ('nadal', 0.75975418
114 : [('federer', 0.971716582775116), ('nadal', 0.7597029209136963), ('djokovic', 0.752267718
118 : [('federer', 0.9890128374099731), ('nadal', 0.7692031860351562), ('djokovic', 0.76162457
121 : [('federer', 0.9828811883926392), ('nadal', 0.7608731985092163), ('djokovic', 0.75685125
122 : [('federer', 0.9623067378997803), ('nadal', 0.7562921047210693), ('djokovic', 0.75496476
127 : [('federer', 0.9811326265335083), ('djokovic', 0.7630206346511841), ('nadal', 0.75700682
140 : [('federer', 0.9737919569015503), ('nadal', 0.7469925284385681), ('djokovic', 0.73864781
141 : [('federer', 0.9820785522460938), ('djokovic', 0.7541807889938354), ('nadal', 0.74478393
143 : [('federer', 0.9725000858306885), ('djokovic', 0.7532610297203064), ('nadal', 0.74560236
149 : [('federer', 0.9821915626525879), ('nadal', 0.7587418556213379), ('djokovic', 0.75582319
153 : [('federer', 0.9912641048431396), ('nadal', 0.7679131031036377), ('djokovic', 0.76557159
154 : [('federer', 0.9611778855323792), ('nadal', 0.7640374302864075), ('djokovic', 0.74699139
159 : [('federer', 0.9867867231369019), ('nadal', 0.761846125125885), ('djokovic', 0.758842587
161 : [('federer', 0.972281813621521), ('djokovic', 0.7496495842933655), ('nadal', 0.733103394
168 : [('federer', 0.9834516048431396), ('nadal', 0.7455811500549316), ('djokovic', 0.74503374
169 : [('federer', 0.9918667078018188), ('nadal', 0.7636914253234863), ('djokovic', 0.76304543
170 : [('federer', 0.9738223552703857), ('nadal', 0.7491594552993774), ('djokovic', 0.74786573
177 : [('federer', 0.9743062257766724), ('nadal', 0.7564150094985962), ('djokovic', 0.75096976
183 : [('federer', 0.9886819124221802), ('nadal', 0.7658869028091431), ('djokovic', 0.75255298
185 : [('federer', 0.9869067668914795), ('nadal', 0.7604039907455444), ('djokovic', 0.75559377
188 : [('federer', 0.988337516784668), ('nadal', 0.7713549137115479), ('djokovic', 0.769763052
190 : [('federer', 0.9796915054321289), ('nadal', 0.75789475440979), ('djokovic', 0.7564153671
191 : [('federer', 0.945972740650177), ('djokovic', 0.7349231243133545), ('nadal', 0.723672151
192 : [('federer', 0.9749240279197693), ('djokovic', 0.7552589178085327), ('nadal', 0.74376481
194 : [('federer', 0.9790613651275635), ('nadal', 0.7561460137367249), ('djokovic', 0.75368356
205 : [('federer', 0.9703332185745239), ('nadal', 0.7397903800010681), ('djokovic', 0.73921775
209 : [('federer', 0.9666629433631897), ('djokovic', 0.7642092108726501), ('nadal', 0.75245058
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', 0.74858951
234 : [('federer', 0.9915372133255005), ('nadal', 0.7670965790748596), ('djokovic', 0.76422190
249 : [('federer', 0.979834258556366), ('nadal', 0.7571359872817993), ('djokovic', 0.757085561
250 : [('federer', 0.9855191707611084), ('nadal', 0.7594801783561707), ('djokovic', 0.75543314
254 : [('federer', 0.9868241548538208), ('nadal', 0.7586868405342102), ('djokovic', 0.75754880
256 : [('federer', 0.980610728263855), ('nadal', 0.7567037343978882), ('djokovic', 0.753770232
283 : [('federer', 0.9677611589431763), ('djokovic', 0.765521049495117), ('nadal', 0.75493657
289 : [('federer', 0.9651498794555664), ('nadal', 0.7487481832504272), ('djokovic', 0.74829834
291 : [('federer', 0.9793593883514404), ('nadal', 0.7567390203475952), ('djokovic', 0.75478601
297 : [('federer', 0.9643127918243408), ('nadal', 0.7438105344772339), ('djokovic', 0.74333423
```

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

pa-w2v-explore-vector-influence

```
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 [ ]:
```

Appendix . Appendix

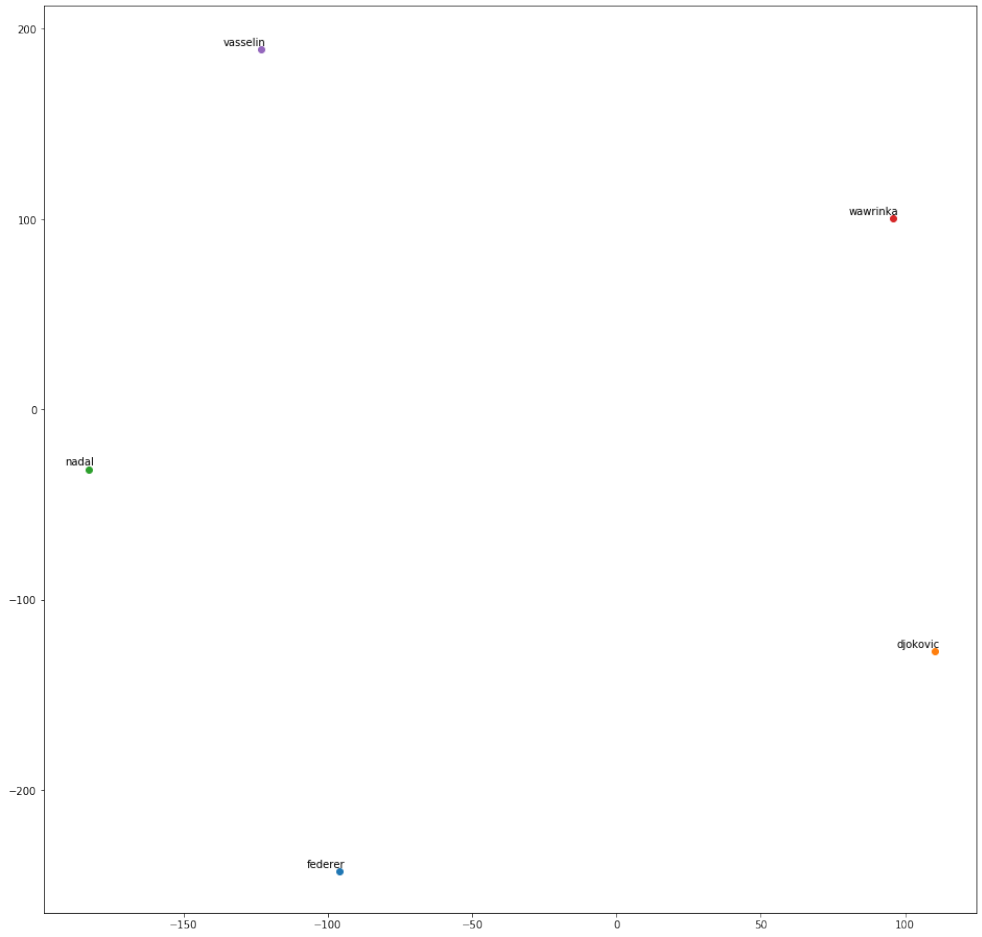
pa-w2v-explore-vector-influence

```
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()
```

pa-w2v-explore-vector-influence



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', 'verda
```

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

pa-w2v-explore-vector-influence

```
            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', 'verda
```

```
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
```

pa-w2v-explore-vector-influence

```
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
```

pa-w2v-explore-vector-influence

```
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)
```

pa-w2v-explore-vector-influence

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

```
/home/rclaret/anaconda3/envs/py36/lib/python3.6/site-packages/ipykernel_launcher.py:13: Deprec
  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])
```

Appendix . Appendix

pa-w2v-explore-vector-influence

```
        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
```

pa-w2v-explore-vector-influence

```
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
```

pa-w2v-explore-vector-influence

```
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: Deprec
  del sys.path[0]
```

pa-w2v-explore-vector-influence

```
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
```

Appendix . Appendix

pa-w2v-explore-vector-influence

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

In [ ]:

.1.7 pa-w2v-mono-training

pa - w2v mono training 1

June 2, 2019

1 Gensim Training Experiments

- **Machines:**
  - HEIA-FR GPU-2 (32 cpu dual threaded)
  - CPU Monster at HEIA-FR (48 cpu single threaded)
- **Dataset:**
  - wikipedia english dump from 2019-03-19 (16GB)
  - wikipedia english dump from 2019-04-09 (16GB)
- **Dictionary:**
  - lemmatized dictionary(16MB)
  - unlemmatized dictionary(16MB)

1.1 What's going on

- Training a Word2Vec on the full wikipedia english dataset using its pre-extracted lemmatized and unlemmatized dictionary.

In [1]:
```
# Word2Vec settings
import multiprocessing

#w2v_w2v_sentences=None
#w2v_corpus_file=None
w2v_size=300 # (default: 100)
#w2v_alpha=0.025
w2v_window=10 # (default: 5)
w2v_min_count=1 # (default: 5)
#w2v_max_vocab_size=None
#w2v_sample=0.001
#w2v_seed=1
w2v_workers=4 # (default: 3) # multiprocessing.cpu_count()
#w2v_min_alpha=0.0001
w2v_sg=0 # if sg=0 CBOW is used (default); if sg=1 skip-gram is used
#w2v_hs=0
#w2v_negative=5
```

Appendix . Appendix

pa-w2v-mono-training

```
#w2v_ns_exponent=0.75
#w2v_cbow_mean=1
#w2v_hashfxn=<built-in function hash>
w2v_iter=5 # (default: 5)
#w2v_null_word=0
#w2v_trim_rule=None
#w2v_sorted_vocab=1
w2v_batch_words=10000 # (default: 10000)
#w2v_compute_loss=False
#w2v_callbacks=()
#w2v_max_final_vocab=None
```

In [2]:
```
# General settings
lemmatization = False
run_corpus = "wiki"
run_lang = "en"
run_date = "190409"
run_log_prefix = "train"

run_model_dir = "models/"
run_dict_dir = "dictionaries/"
run_datasets_dir = "datasets/"
run_log_dir = "logs/"
```

In [3]:
```
run_w2v_algo = "cbow" if w2v_sg==0 else "sg"

run_options = "s"+str(w2v_size)+"-w"+str(w2v_window)+"-mc"+str(w2v_min_count)+"-bw"+st
print(run_options)

run_base_name = run_corpus+"-"+run_lang+"-"+run_date # wiki-en-190409
run_model_name = run_model_dir+run_base_name+"-"+run_options

run_dict_name = run_dict_dir+run_base_name+"-dict"
run_dataset_name = run_datasets_dir+run_base_name+"-latest-pages-articles.xml.bz2"
run_log_name = run_log_dir+run_log_prefix+"-"+run_base_name+"-"+run_options

run_lem = "-lem" if lemmatization else "-unlem"

run_model_name += run_lem+".model"
run_dict_name += run_lem+".txt.bz2"
run_log_name += run_lem+".log"

print(run_model_name)
print(run_dict_name)
print(run_dataset_name)
print(run_log_name)
```

```
s300-w10-mc1-bw10000-cbow-i5-c4
models/wiki-en-190409-s300-w10-mc1-bw10000-cbow-i5-c4-unlem.model
```

pa-w2v-mono-training

```
dictionaries/wiki-en-190409-dict-unlem.txt.bz2
datasets/wiki-en-190409-latest-pages-articles.xml.bz2
logs/train-wiki-en-190409-s300-w10-mc1-bw10000-cbow-i5-c4-unlem.log
```

In [4]:
```
# Start logging process at root level
import logging
logging.basicConfig(filename=run_log_name, format='%(asctime)s : %(levelname)s : %(mes
#logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging
logging.root.setLevel(level=logging.INFO)
```

In [ ]:

In [5]:
```
# Load dictionary from file
from gensim.corpora import Dictionary
dictionary = Dictionary.load_from_text(run_dict_name)
```

In [ ]:
```
# Build WikiCorpus based on the dictionary
from gensim.corpora import WikiCorpus

wc_fname=run_dataset_name
#wc_processes=None
wc_lemmatize=lemmatization
wc_dictionary=dictionary
#wc_filter_namespaces=('0', )
#wc_tokenizer_func=<function tokenize>
#wc_article_min_tokens=50
#wc_token_min_len=2
#wc_token_max_len=15
#wc_lower=True
#wc_filter_articles=None

wiki = WikiCorpus(fname=wc_fname, dictionary=wc_dictionary,lemmatize=wc_lemmatize)
```

In [ ]:
```
# Initialize simple sentence iterator required for the Word2Vec model
# Trying to bypass memory errors

if lemmatization:
    class SentencesIterator:
        def __init__(self, wiki):
            self.wiki = wiki

        def __iter__(self):
            for sentence in self.wiki.get_texts():
                yield list(map(lambda x: x.decode('utf-8'), sentence))
                #yield gensim.utils.simple_preprocess(line)

else:
    class SentencesIterator:
```

Appendix . Appendix

pa-w2v-mono-training

```
                def __init__(self, wiki):
                    self.wiki = wiki

                def __iter__(self):
                    for sentence in self.wiki.get_texts():
                        yield list(map(lambda x: x.encode('utf-8').decode('utf-8'), sentence))

        sentences = SentencesIterator(wiki)
```

In [ ]:
```
        # Train model
        from gensim.models import Word2Vec

        print("Running with: " + str(w2v_workers) + " cores")

        model = Word2Vec(sentences=sentences,
                         size=w2v_size,
                         window=w2v_window,
                         min_count=w2v_min_count,
                         workers=w2v_workers,
                         sg=w2v_sg,
                         iter=w2v_iter
                        )
        model.save(run_model_name)

        del model
        del wiki
        del sentences
        del dictionary
```

```
Running with: 4 cores
```

In [ ]:

.1.8 pa-w2v-sentence-generator

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.
        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 Englis
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
```

Appendix . Appendix

pa-w2v-sentence-generator

```
        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)
```

pa-w2v-sentence-generator

```
            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 token
            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=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
```

```
[('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),
```

Appendix . Appendix

pa-w2v-sentence-generator

```
['this', 'convinced', 'danceworks']
[('', 0.30970412492752075), ('vcissara', 0.29018083214759827), ('bobtails', 0.2818413376808166
```

```
['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', 0.26703292131423950), ('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']
```

pa-w2v-sentence-generator

```
In [71]: #definition of capital
         sentence = "A town or city that is the official seat of government in a political ent
         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', 0.9

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
```

Appendix . Appendix

pa-w2v-sentence-generator

```
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_vect
```

```
[('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']
```

pa-w2v-sentence-generator

```
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")
```

Appendix . Appendix

pa-w2v-sentence-generator

```
[('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 [ ]:
```

.1.9 pa-wikidump-splitter

pa - wikidump splitter

June 2, 2019

```
In [1]: # Start logging process at root level
        import logging
        logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.
        logging.root.setLevel(level=logging.INFO)
```

```
In [2]: total_lines = 1092633438
        chunk_pages = 99999 # 999999: ~2.3GB, 99999: ~300MB, 9999: ~89MB, 999: ~8MB
        chunks_folder = "datasets/chunks/"
        folder_name = chunks_folder+"enwiki-chunks-"+str(chunk_pages)+"/"
        chunk_basename = "enwiki-chunk-"+str(chunk_pages)+"-"
```

```
In [3]: import os
        # Check and create chunk diretory
        if not os.path.exists(chunks_folder):
            print("Chunks folder was not present.")
            os.mkdir(chunks_folder)
        if not os.path.exists(folder_name):
            print("Data chunk folder was not present.")
            os.mkdir(folder_name)
```

```
Data folder was not present.
```

```
In [4]: # Based on:
        # https://stackoverflow.com/questions/6184912/how-to-split-large-wikipedia-dump-xml-bz

        import os
        import bz2
        from timeit import default_timer as timer

        #print("expecting: " +  + " parts")

        def split_xml(filename):
            ''' The function gets the filename of wiktionary.xml.bz2 file as  input and create
            smallers chunks of it in a the diretory chunks
            '''
            # Counters
            pagecount = 0
```

pa-wikidump-splitter

```
            filecount = 1
            total_pages = 0
            # open chunkfile in write mode
            chunkname = lambda filecount: os.path.join(folder_name,chunk_basename+str(filecoun
            chunkfile = bz2.BZ2File(chunkname(filecount), 'w')
            # Read line by line
            bzfile = bz2.BZ2File(filename)
            #print(sum(1 for _ in bzfile))
            print("Chunking...")
            start = timer()
            for line in bzfile:
                chunkfile.write(line)
                # the </page> determines new wiki page
                if b'</page>' in line:
                    pagecount += 1
                    total_pages += 1
                if pagecount > chunk_pages:
                    chunkfile.write(b'</mediawiki>') # add end tag
                    end = timer()
                    print(datetime.datetime.now(),":",filecount,"->", round(end - start,2), "s
                    chunkfile.close()
                    pagecount = 0
                    filecount += 1
                    chunkfile = bz2.BZ2File(chunkname(filecount), 'w')
                    start = timer()
                    # add start tag
                    chunkfile.write(b'<mediawiki xmlns="http://www.mediawiki.org/xml/export-0.
            try:
                chunkfile.close()
            except:
                print('Files already close')

            print("Done.")
```

```
In [ ]: split_xml('datasets/enwiki-latest-pages-articles.xml.bz2')
```

```
In [ ]:
```

.2 Speadsheets

Appendix . Appendix

.2.1 pa-models-created-and-time-benchmarks

Genbot - Deepening Project - Summary of Word2Vec Models Created + Time Benchmark

TIME	SIZE	WINDOW	MIN_COUNT	BAG_OF_WORDS	SG	ITERATION	CORES	LEMMATIZATION	TIME_START	TIME_END
20h 19m	300	5	1	10000	0	5	10	0	2019-04-16 14:09	2019-04-17 10:28
20h 28m	300	5	1	10000	0	5	20	0	2019-04-23 20:04	2019-04-24 16:32
20h 29m	300	5	1	10000	0	5	30	0	2019-04-24 23:10	2019-04-25 19:39
20h 29m	300	5	10	10000	0	5	10	0	2019-05-02 17:33	2019-05-03 14:02
20h 31m	300	5	1	10000	0	5	40	0	2019-04-25 22:18	2019-04-26 18:49
20h 51m	300	1	1	10000	0	5	10	0	2019-05-06 22:32	2019-05-07 19:23
21h 4m	300	5	5	10000	0	5	40	0	2019-04-28 23:13	2019-04-29 20:17
21h 7m	300	5	5	10000	0	5	4	0	2019-05-06 21:57	2019-05-07 19:04
21h 9m	300	5	5	100000	0	5	40	0	2019-05-01 15:31	2019-05-02 12:40
21h 10m	300	5	5	10000	0	5	5	0	2019-05-06 22:03	2019-05-07 19:13
21h 26m	300	3	3	10000	0	5	10	0	2019-05-06 22:45	2019-05-07 20:11
21h 28m	300	10	5	10000	0	5	10	0	2019-05-05 23:03	2019-05-06 20:31
21h 34m	300	15	5	10000	0	5	10	0	2019-05-05 23:04	2019-05-06 20:38
21h 38m	300	2	5	10000	0	5	10	0	2019-05-05 23:05	2019-05-06 20:43
21h 42m	300	20	5	10000	0	5	10	0	2019-05-05 23:04	2019-05-06 20:46
21h 46m	300	2	2	10000	0	5	10	0	2019-05-06 22:40	2019-05-07 20:26
21h 46m	300	3	1	10000	0	5	5	0	2019-05-14 16:28	2019-05-15 14:14
21h 48m	300	8	1	10000	0	5	5	0	2019-05-14 16:30	2019-05-15 14:18
21h 52m	300	8	5	10000	0	5	5	0	2019-05-14 16:32	2019-05-15 14:24
22h	300	8	8	10000	0	5	5	0	2019-05-14 16:32	2019-05-15 14:32
22h 6m	300	3	8	10000	0	5	5	0	2019-05-14 16:29	2019-05-15 14:35
22h 9m	300	8	3	10000	0	5	5	0	2019-05-14 16:31	2019-05-15 14:40
22h 11m	300	5	5	10000	0	5	3	0	2019-05-05 23:05	2019-05-06 21:16
22h 16m	300	5	5	10000	0	5	6	0	2019-05-09 17:02	2019-05-10 15:18
22h 19m	300	5	5	10000	0	5	9	0	2019-05-09 17:07	2019-05-10 15:26
22h 20m	300	5	5	10000	0	5	11	0	2019-05-09 17:13	2019-05-10 15:33
22h 24m	300	5	5	10000	0	5	8	0	2019-05-09 17:03	2019-05-10 15:27
22h 43m	300	5	5	10000	0	5	7	0	2019-05-09 17:02	2019-05-10 15:45
22h 49m	300	6	5	10000	0	5	5	0	2019-05-13 16:24	2019-05-14 15:13
22h 52m	300	10	8	10000	0	5	5	0	2019-05-16 15:41	2019-05-17 14:33
22h 53m	300	4	5	10000	0	5	5	0	2019-05-13 16:18	2019-05-14 15:11
22h 54m	300	9	5	10000	0	5	5	0	2019-05-13 16:30	2019-05-14 15:24
22h 55m	300	10	1	10000	0	5	4	0	2019-05-16 15:40	2019-05-17 14:35
22h 55m	300	8	15	10000	0	5	5	0	2019-05-16 15:46	2019-05-17 14:41
22h 58m	300	8	5	10000	0	5	5	0	2019-05-13 16:28	2019-05-14 15:26
22h 58m	300	5	15	10000	0	5	5	0	2019-05-16 15:44	2019-05-17 14:42
23h 3m	300	10	15	10000	0	5	5	0	2019-05-16 15:47	2019-05-17 14:50
23h 4m	300	3	5	10000	0	5	5	0	2019-05-13 16:15	2019-05-14 15:19
23h 4m	300	7	5	10000	0	5	5	0	2019-05-13 16:27	2019-05-14 15:31
23h 5m	300	5	5	10000	0	5	5	0	2019-05-13 16:21	2019-05-14 15:26
23h 5m	300	3	15	10000	0	5	5	0	2019-05-16 15:45	2019-05-17 14:50
23h 17m	300	10	3	10000	0	5	5	0	2019-05-16 15:40	2019-05-17 14:57
23h 26m	300	5	1	10000	1	5	40	0	2019-04-27 17:15	2019-04-28 16:41
1d 2h 42m	300	5	5	10000	0	5	2	0	2019-05-09 17:01	2019-05-10 19:43
1d 4h 16m	300	5	8	10000	0	5	4	0	2019-05-11 23:53	2019-5-13 4:09
1d 4h 25m	300	5	9	10000	0	5	4	0	2019-05-11 23:53	2019-5-13 4:18
1d 4h 26m	300	5	5	10000	0	5	4	0	2019-05-11 23:46	2019-5-13 4:12
1d 4h 26m	300	5	6	10000	0	5	4	0	2019-05-11 23:51	2019-5-13 4:17
1d 4h 27m	300	5	7	10000	0	5	4	0	2019-05-11 23:52	2019-5-13 4:19
1d 4h 39m	300	5	5	10000	0	5	3	0	2019-05-11 23:46	2019-05-13 4:25
1d 4h 41m	300	5	4	10000	0	5	4	0	2019-05-11 23:50	2019-5-13 4:31
1d 4h 52m	300	5	3	10000	0	5	4	0	2019-05-11 23:50	2019-5-13 4:42
1d 5h 1m	300	5	5	10000	0	5	7	0	2019-05-11 23:47	2019-5-13 4:48
1d 5h 9m	300	5	2	10000	0	5	4	0	2019-05-11 23:48	2019-5-13 4:57
1d 18h 50m	300	10	10	10000	0	5	1	0	2019-05-09 17:01	2019-05-11 11:51
3d 21h 40m	300	5	1	10000	0	5	10	1	2019-04-17 15:11	2019-04-21 12:51

.3 Meeting Notes

.3.1 02_18_19_meeting

Meeting PA, with Jean by Skype

But du meeting

-> déterminer Semaine 1
↳ Cahier des charges
↳ Jusqu'à 2 semaines
↳ Lire des articles
↳ Fixer un plan

Idée Romain

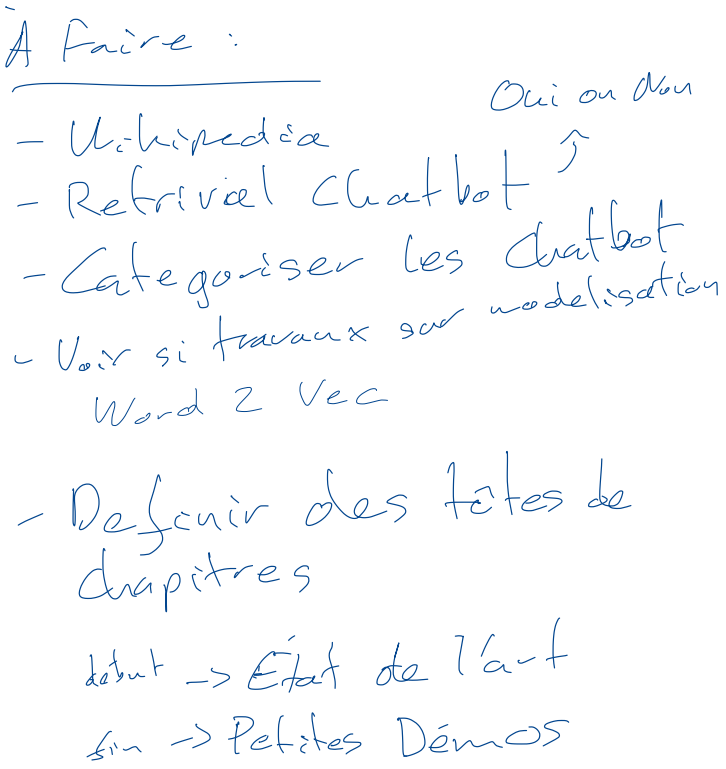
-> n ODE
↳ à approfondir
↳ applications chatbot

Idées Jean

- Travailler sur Wikipedia
- Word 2 Vec (Initialement supervisé)
  - -> utilise word embedding
  - -> Initiative du Chatbot
    - -> "Lausanne ou Genève"
      - ↳ Promenade dans l'espace Vect
  - -> Initialisé supervisé
  - -> les chemins initiaux supervisé
    - ↳ l'utilisateur ajoute des chemins
- Double RNN (se souvenir plus longtemps)
  - ① -> Generer text et dialogue
  - ① -> Contexte long terme
  - -> Etat de l'art ?

- Prochain meeting
  - -> 25.02.19
  - -> 10h
  - -> à déterminer l'endroit
- Inter meeting
  - -> 22.02.19
  - -> 17h
  - -> Bullet point
    - ↳ Rapport / Objectifs
- Framework Potentiel
  - -> RASA (support interne)
- Projet
  - -> 10h par semaine
  - -> sur 14 semaines
  - -> Standalone
    - ↳ Potential API
  - -> Cahier des charges flexible
    - ↳ "se faire plaisir"
  - Evaluation
    - ↳ Gestion du projet
      - ↳ Rapport
      - ↳ Comportement
      - ↳ Agile ?
    - ↳ Analyse
      - ↳ Etat de l'art
    - ↳ Dev
      - ↳ test
      - ↳ proposition de mise en place
    - ↳ Documentation / Rapport

02_18_19_meeting



.3.2 02_22_19_meeting

With Jean by Skype, 22.02.2019, 17h

- Model Transformer, to see
- Moteur de recherche évaluation
  - Relevance on pas
  - S'inspirer pour évaluer

- Questions too large:
  - Language -> Word2Vec Multi dimensionnels
  - Espace represente géométriquement
  - Operation logique
  - Inferer une trajectoire en inferant un corpus (dans un poème)
  -

- Challange:

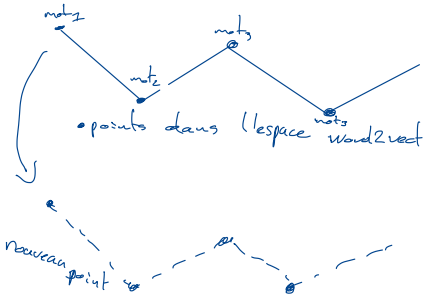
- Abstract language is Good

Lundi
-> 10h à Fribourg
-> On enchaine avec christophe

.3.3 02_25_19_meeting

with Jean, at iCoSys, 25.02.19 at 10:00
avec Christophe.
- Wikipedia FR Word2Vec
-> Aide Wikipedia FR et Word2Vec (~2015)

Todo:
- Trouver equilibre entre explicative et implementation
- Trouver une méthode principale et tourner autour,
- Encrer un objectif technologique qu'on peut atteindre.
  -> creer un garde fou
- Evaluer côté technique
- Idée de Jean:
  - Word2Vec, pas perte de temps.
    -> exploiter espace sous-jacent
      -> context, words,
    -> Utiliser des citations:
    -> Extraire avec NLP, NLTK



- Est-ce que dans tout les cas
- GAN?
- Inserer notion du part of speech random
  -> VERBE (reste verbe)
- "modelise le monde, plus que juste la langue"
- Common langue <- (inter langual)
- "la langue precède la pensé"?"

Plannifier agendant
x Qu'est ce qui s'est passé
x Contenue
x avancé général

x Context
x Objectifs
  x Principaux Principaux technologies
  x Secondaire: explo
x Plan d'attaque
  x MVP,
    -> date "demo with api" "chatbot"

-> Resultat: Rendre chatbot pro actif.
-> Visualisation graphique?
-> Base for other algorithms

2 objectifs
- ~~x~~ Word2Vec
- ~~x~~ Demonstrator, with the chatbot that as initive

.3.4 03_01_19_meeting

Skype meeting, with Jean,

Context : + Driver
-> AI-News, avec la liberté
-> C'institut a Fribourg est par ailleurs en collab avec un l'innovation suisse contexte de proto démontable

Objectif principaux :
⊖ 1 ou plusieurs prototype
⊕ Evaluation des chatbots généraux

Plan initial ← Formel

Structure Potentiel du Rapport à un moment dans le PA

Profs qui suivent — Philipe Joy Pasereau

-> Envoyer à jean invitation recurrente lundi 10h

-> Envoyer plan + spec avant 10h lundi
-> A annuler formellement lundi tout en accord

.3.5 03_11_19_meeting

ajouter les PV en annexes, dans rapport

Problème avec l'évaluation du chatbot,
-> on ne peut probablement pas scripter

carte collaborateur ?

ajouter à ? Github Jean use
+ Christophe
+ hennebert

How to evaluate Word 2 Vect
-> What are the extensions
-> operation geometrique? -> est-ce qu'on recupère du sense
-> Phrase vrai (verité "le roi est le mari de la reine")
possible et limite ("la chat est le mari de la chatte")
limité sur le corpus d'entrainement ? essayer avec autre corpus
1er etape reentrainer word 2 vec
2eme etape valider la word 2 vec avec la litterature
3eme jouer avec les trasfations

emoticon pour sentiment

IDea: Pro active
-> Dictionaire de phrase de verité
-> Evaluer le trigger de la conversation (initialisation)
-> Give to user the ability evaluate the chatbot behavior
->

Partir sur de la technologie standard avant l'avondadiste
->

-> Snippet du code dans Rapport

Court terme
Template latex
-> Word 2 Vec avec Gensin sur wikipedia
-> English only
-> Word 2 Vec avec des bouquins
-> possible de faire du fine tuning avec Gensim
-> structure du rapport, avec une premiere structure
long terme

-> S'appuyer sur des Romans.
->

.3.6 03_18_19_meeting

x Qu'est ce qui s'est passé

- la semaine passé
  - approuvé le cahier des charges
  - discussion sur l'initialisation du projet et la suite du projet
- Objectifs Globaux
  - Chatbot avec initiative
- Objectifs :
  - avoir un word2vec avec gensim et wikipedia en anglais
    - -> toujours en cours
      - -> surpris par les datasets
      - -> ça tourne sur le jupyter de icolab_gpu_2
      - ->
- la semaine:
  - lecture
    - Word2Char
  - fini le template latex
  - ajout au repertoire des profs

x Courtenne

- faire marcher ce word2vec et documenter
- tester avec fastText

x avancé générale:

- prise en main du word2vec
-

* Requêtes / Questions
  - Machine dedié ?
  - Limitation du jupyter

- script python sur terminal / background

Semaine prochaine Objectifs

* Operation Geometric sur Word2Vec
* Cister les questions scientifiques
  - Bullet point -> plus restreint que originaux
    -> est qu'il est possible d'avoir des phrases cohérente.

03_18_19_meeting

* Proverbe / Metaphore très imagé
  -> Essayer avec les proverbes
  -> Anglais ou Français
*

.3.7 03_25_19_meeting

what has been done?

- Gensim
  - memory error on large wiki dataset (full)
  - small dataset working
  - Geometric operations working, but not always relavante due to training set
  - transfered the specifications to later + Questions
  -

short terme

- dropping the idea of training on full wikipedia
  - let run on 1 cpu core training for test
- train on larger pretrained datasets from Gensim
- Combine with ANN
- Document in report Gensim life
- Send mail to christophe for Github
- Compare with : fastText, word2Vec, Glove

overall progress

- Gensim is not working as expected on large dataset
  fall back on smaller datasets and officially pretained
-

- Wikipedia FR
- Conversational agent framework

Question :
Deliverable next week

.3.8 04_01_19_meeting

what has been done?

- Run full w2v on AWS, on strongest machine available → expensive
- Testing Azure Notebook → storage limitation
- Testing on my machine (PC) → ram limitation
- Made an iteration/generator methode to do line by line on dataset
- Made a wikipedia dataset splitter with balise preservation
- Pre-trained models are not retrainable, not full model → only keyed Vectors
- Dictionary unexpected: - Lemmatization missing → Dog/NN, Do/VB, ... "library not installed first time run, "pattern"
- 1 CPU Core not working
- Memory allocation error on splitter also

short terme

- starting new sprint on - ANN chatbot
  - parallel algorithm
  - protocole to evaluate proactive

Get
- full w2v wikipedia
- meanwhile use smaller dataset
- Circature and test of existing chatbot solution
- Fix memory error on splitter with a banner for the generator

Last time short terme

- Combine with ANN
- Document in report Gensim life
- Compare with: fastText, word2Vec, Glove

overall progress

- fixing the memory problem with gensim by splitting datasets by pages and use a generator by line
- Trying to use cloud based machines, but expensive and no results yet.
- Word2Vec operations working on partial model by dictionary is not as expected

Appendix . Appendix

04_01_19_meeting

Questions:

- Problematic de reentrainement
- Tester les translations -> avec un set de phrases
  -> 20 aine
- Case comprehision Word2Vec ->

Translation

-> perturbation mini

-> le chat est une mamifaire

-> Plus proche voisin

-> vecteur appliquer au reste de la phrase

-> vecteur de translation

-> translation dirigé -> shift

-> translation random

-> Word2Vec

-> A la base pretraitement des mots

-> Playing with word2vec

-> Dimensions -> concepts

-> Jump on 1 dimension

-> Multi dimensions

-> Synonime

-> point chat

-> sinonime de chat

-> quelles dimensions les plus impactés

-> diriger dans la direction

- But comprendre l'espace Word2Vec

-> Word2Sequence

04_01_19_meeting

- Chapitre sur le biais
-> mettre en evidance
Doctor - man + woman = Nurse
-> Facebook, Google, -> biais dans recommendation publicité
-> poussé vers des études

.3.9 04_08_19_meeting

what has been done?

- full wikipedia english training : 3 J 20 h on heia-fr
- lemmatization vs non-lemmatization
  -> without -> performance -> less shades
- playing with operations by vectors
  -> words
  -> still working on sentences
- tested analogies operations : Athens is to Greece as Baghdad is to Iraq
  -> translation random
  -> Plus proche voisin -> le chat est un mamifaire
  -> vecteur appliquer au reste de la phrase
  -> vecteur de translation

short terme

- starting new sprint on - ANN chatbot
  - parallel algorithm
  - protocol to evaluate proactive
- Literature and test of existing chatbot solution
- Combine with ANN
- Document in report Gensim life
- Compare with : fastText, word2Vec, Glove

-> translation dirigé -> shift

overall progress

- fixed the memory problem with gensim by splitting datasets by pages and use a generator by line
- Tryied to use cloud based machines. Expensive and not worth it for CPU computation, same perf as on heia-fr.
- Full wikipedia EN trained model : 3 J et 20 h on heia-fr
  -> with lemmitization
- Word2Vec operations working

04_08_19_meeting

Prendre note des examples
->

est-ce un
sur base d'exemple

Exemple 10ème 20ème de Capital

-> Average of dimension
-> Vecteur de différence
-> Capital [ ]
-> Capital of science
-> Est le marié de..

-> Est-ce que les relations W2V sont les mêmes que les relations des entités

-> la force des similarités
-> Cosine (direction)
-> Normalizé
-> Non normalisé

-> Articles avec des trucs bizarre qu'on a vu
-> Why people are not using full lemized vocab
-> Thinking a metric to compare via similar
->mine VS google

04_08_19_meeting

method (wv_a, wv_b, list_de_mot, top-n)
Ls ratio du nombre de mots commun
↑ jolie contribution

Idée :
-> Evolution de la femme dans le word2vec
-> avec les backups de wikipedia

.3.10 04_16_19_meeting

what has been done?

- Setup of new CPU machine -> ready and running
- Started Research on ANN and DNN methods to use Word Embadings
- started looking at alternative corpora data sets
- Currently building Word2Vec without lemmatization
- Playing with similarities
- Building Dictionnary is longer on CM? exacty 1h?

short terme

- playing with - ANN chatbot
  - parallel algorithm
  - protocol to evaluate proactive
- Literature and test of existing chatbot solution
- Combine with ANN
- Document in report Gensim life
- Compare with: fastText, word2Vec, Glove
- Build alternative Word2Vec with different corpora
- Play with Word2Vec
  - geometies
  - analogies (capital of science)
- Compare unlemmatized vs lemmatized
- Compare Premade W2VC google) vs mine

overall progress

- fixed the memory problem with gensim by splitting data sets by pages and use a generator by line
- Tryied to use cloud based machines. Expensive and not worth it for CPU computation, same perf as on heia-fr.
- Full wikipedia EN trained model: 3 5 et 20h on heia-fr -> with lemmitization
- Word2Vec operations working
- Testing on a CPU dedicated machine to train

Questions:

- Meeting next week? A partir mercred apres-midi

Appendix . Appendix

04_16_19_meeting

Faire :

-> - stopword

-> Generate similar sentences

-> Use Word2Vec as it is

-> "drunk guy in a pub"

Avancer sur la doc !

Écrire tout ce qui a été fait jusqu'à présent

-> dUP

1) Versions
Dans quel mesures peut-on exploiter le W2V, sans intelligence

2) Seq2Seq, article + Implementation, avec quelque chose tout fait
-> Comparer les modules

Explorer la generation de phrases
-> pas greffer

2 semaines de redaction

-> mettre à jour le plan

.3.11 05_03_19_meeting

what has been done?

- Ran multiple models ~10 per training
- Made sentence generator
- started report
- Played with analogies

Problems:

- Cannot do similarities with un semantically related words
  -> "Capital of Science" doesn't make sense with raw wikien
- Analogies refer to astracted words. Doctor -> "Doctor who charlas"
  -> often not in the right semantic space
- Pure random doesn't impact the geometrical operations
  -> Words from vocabulary are needed
- CPU Entrainements

short terme

- Plot the specific spaces
- Circature and test of existing chatbot solution
- Compare Premade W2VC google) vs mine
- Compare unlemmatized vs lemmatized
- playing with - ANN chatbot
  - parallel algorithm
  - protocol to evaluate proactive

overall progress

- fixed the memory problem with gensim by splitting data sets by pages and use a generator by line
- Tryied to use cloud based machines. Expensive and not worth it for CPU computation, same perf as on heia-fr.
- Full wikipedia EN trained model : 3 J et 20 h on heia-fr
  -> with lemmitization
- Word2Vec operations working
- Testing on a CPU dedicated machine to train
- Generating new sentences

Appendix . Appendix

05_03_19_meeting

Todo :

Custom :

Skemming and already in vector

- Prendre un vecteur
  -> que vaut dire les axes

Federer
-> dimension valeur la plus grand
-> varier uniquement cette variable
Ls si on se deplace sur l'axe est-ce
qu'on peut obtenir des choses
genre warrior ka

LS

Rapport

Démo
-> Chatbot qui prend la main

05_10_19_meeting

Word2Vec – meeting – 05 10 19

What have been done:
- More models with different parameters and per nb_core benchmark
- Graphical representation of the word vectors
- Influence the each element of the word vector
  - Basic influence with the min and max of each variable
  - Graphical representation
-

Problems:

Short term:
- Stemming vocabulary, remove Chinese characters etc.
- Add paddings to vocabulary
- Play with spacy
- Make a model benchmark comparator
- Try to influence vector elements with important values
- Make a Seq2Seq
- Chatbot based on Edgar Poe books LSTM (double?)
  - Wikipedia World Embedding
  - http://www.gutenberg.org/ebooks/search/?query=Edgar+poe
  - Narrow wikipedia vocabulary to Edgar poe literature
- Facebook chatbot?

Todo:

Notes:

MRU – Philippe Joy

Appendix . Appendix

05_17_19_meeting

Word2Vec – meeting – 05 17 19

What have been done:
– Started a model representation for all models
– Started to build a chatbot lstm + keras
– GPU compromised, trying on cpu machine

Problems:
– GPU OOM Bugs on icolab
– Timeout on google colab
– Bug with weird characters output

Short term:
– Stemming vocabulary, remove Chinese characters etc.
– Add paddings to vocabulary
– Play with spacy
– Make a model benchmark comparator
– Try to influence vector elements with important values
– Make a Seq2Seq
– Chatbot based on Edgar Poe books LSTM (double?)
    – Wikipedia World Embedding
    – http://www.gutenberg.org/ebooks/search/?query=Edgar+poe
    – Narrow wikipedia vocabulary to Edgar poe literature
– Facebook chatbot?

Todo:
– Talk about the T–SNE
    – Intuition of what is going on
– 3d navigation
– metrics
    – Model state of the art
        – Google
        – Gensim par default
        –
    – 100 mots
        – 10 plus proches voisins
            – Metric a quel point les mots sont different
    – Analyser la pertinence
–
Notes: