Natural Language processing

TP 4 – Word2vec

LE Do Thanh Dat, YOU Borachhun

Exercice 1 - Entrainer son propre Word2vec

1. Importer les dépendances

In [5]:
```
# Importing all necessary modules
from nltk.tokenize import sent_tokenize, word_tokenize
import warnings
import gensim
from gensim.models import Word2Vec

warnings.filterwarnings(action='ignore')
```

2. Importer les données et déclarer quelques variables

In [7]:
```
# Read 'alice.txt' file
sample = open("./data/alice_wonderland.txt")
s = sample.read()
```

3. Prétraitement des données

In [8]:
```
# Replaces escape character with space
f = s.replace("\n", " ")
data = []

# Iterate through each sentence in the file
for i in sent_tokenize(f):
    temp = []

    # Tokenize the sentence into words
    for j in word_tokenize(i):
        temp.append(j.lower())

    data.append(temp)
```

4. Entrainement du CBOW

In [9]:
```
# Create CBOW model
model1 = gensim.models.Word2Vec(data, min_count=1, vector_size=100, window=5)

# Print results
print("Consine similarity between 'alice' " + "and 'wonderland' - CBOW :", model

print("Consine similarity between 'alice' " + "and 'machines' - CBOW :", model1.
```

```
Consine similarity between 'alice' and 'wonderland' - CBOW : 0.97724813
Consine similarity between 'alice' and 'machines' - CBOW : 0.7810761
```

5. Entrainement du Skip-gram

In [10]:
```
# Create Skip Gram model
model2 = gensim.models.Word2Vec(data, min_count=1, vector_size=100, window=5, sg

# Print results
print("Consine similarity between 'alice' " + "and 'wonderland' - Skip Gram :",

print("Consine similarity between 'alice' " + "and 'machines' - Skip Gram :", mc
```

```
Consine similarity between 'alice' and 'wonderland' - Skip Gram : 0.6048101
Consine similarity between 'alice' and 'machines' - Skip Gram : 0.8113064
```

- Jouez avec le parametre vector_size =[2,10,500] sur le Skipgram et CBOW, quel est l'effet sur les distances ?

In [11]:
```
vector_size_list = [2,10,500]

for vs in vector_size_list:
    print("########## vector_size =", vs, "##########")
    model_cbow = gensim.models.Word2Vec(data, min_count=1, vector_size=vs, windc
    print("Consine similarity between 'alice' " + "and 'wonderland' - CBOW :", m
    print("Consine similarity between 'alice' " + "and 'machines' - CBOW :", moc

    model_skip_gram = gensim.models.Word2Vec(data, min_count=1, vector_size=vs,
    print("Consine similarity between 'alice' " + "and 'wonderland' - Skip Gram
    print("Consine similarity between 'alice' " + "and 'machines' - Skip Gram :"
```

```
########## vector_size = 2 ##########
Consine similarity between 'alice' and 'wonderland' - CBOW : 0.5203754
Consine similarity between 'alice' and 'machines' - CBOW : 0.7028143
Consine similarity between 'alice' and 'wonderland' - Skip Gram : 0.9609333
Consine similarity between 'alice' and 'machines' - Skip Gram : 0.98721325
########## vector_size = 10 ##########
Consine similarity between 'alice' and 'wonderland' - CBOW : 0.9256681
Consine similarity between 'alice' and 'machines' - CBOW : 0.56509876
Consine similarity between 'alice' and 'wonderland' - Skip Gram : 0.5811062
Consine similarity between 'alice' and 'machines' - Skip Gram : 0.77608305
########## vector_size = 500 ##########
Consine similarity between 'alice' and 'wonderland' - CBOW : 0.99073654
Consine similarity between 'alice' and 'machines' - CBOW : 0.9146096
Consine similarity between 'alice' and 'wonderland' - Skip Gram : 0.7262976
Consine similarity between 'alice' and 'machines' - Skip Gram : 0.8663808
```

For CBOW, the similarities between the words increase as the vector_size increases. For Skip Gram, the similarities decrease significantly when vector_size goes from 2 to 10, but then increase back when vector_size equals 500.

6. Mots les plus similaires

In [12]:
```
# Top 10 words contributing positively or negatively
print(model1.wv.most_similar(positive="wonderland"))
print(model1.wv.most_similar(negative="wonderland"))
```

```
[('end', 0.9956433773040771), ('adventures', 0.9926746487617493), ('[', 0.98996
34718894958), ('provide', 0.989370584487915), (']', 0.9886643886566162), ('term
s', 0.9859185218811035), ('ebook', 0.9858258366584778), ('associated', 0.985429
1081428528), ('chapter', 0.9852020144462585), ('gutenberg-tm', 0.98494404554367
07)]
[('accounts', 0.8507004380226135), ('occur', 0.7214661836624146), ('sternly',
0.6745487451553345), ('locked', 0.6474383473396301), ('books', 0.63704901933670
04), ('telescope.', 0.5843905210494995), ('treacle-well.', 0.5788701176643372),
('execution.', 0.576991081237793), ('swallowing', 0.5746933221817017), ('poky',
0.5299243330955505)]
```

Exercice 2 - Utiliser un modèle pré entrainé

1. Télécharger le modèle pré entrainé (5 à 10 minutes)

In [13]:
```
import gensim.downloader as api
wv = api.load('word2vec-google-news-300')
```

```
[==================================================-] 100.0% 1662.4/1662.8MB dow
nloaded
```

2. Trouver le vocabulaire du modele

In [14]:
```
# Retrieve the vocabulary of a model
for index, word in enumerate(wv.index_to_key):
    if index == 10:
        break
    print(f"word #{index}/{len(wv.index_to_key)} is {word}")
```

```
word #0/3000000 is </s>
word #1/3000000 is in
word #2/3000000 is for
word #3/3000000 is that
word #4/3000000 is is
word #5/3000000 is on
word #6/3000000 is ##
word #7/3000000 is The
word #8/3000000 is with
word #9/3000000 is said
```

3. Retrouver un vecteur pour un mot

In [15]:
```
# Obtain vectors for terms the model is familiar with:
vec_king = wv['king']
vec_king
```

```
Out[15]: array([ 1.25976562e-01,  2.97851562e-02,  8.60595703e-03,  1.39648438e-01,
        -2.56347656e-02, -3.61328125e-02,  1.11816406e-01, -1.98242188e-01,
         5.12695312e-02,  3.63281250e-01, -2.42187500e-01, -3.02734375e-01,
        -1.77734375e-01, -2.49023438e-02, -1.67968750e-01, -1.69921875e-01,
         3.46679688e-02,  5.21850586e-03,  4.63867188e-02,  1.28906250e-01,
         1.36718750e-01,  1.12792969e-01,  5.95703125e-02,  1.36718750e-01,
         1.01074219e-01, -1.76757812e-01, -2.51953125e-01,  5.98144531e-02,
         3.41796875e-01, -3.11279297e-02,  1.04492188e-01,  6.17675781e-02,
         1.24511719e-01,  4.00390625e-01, -3.22265625e-01,  8.39843750e-02,
         3.90625000e-02,  5.85937500e-03,  7.03125000e-02,  1.72851562e-01,
         1.38671875e-01, -2.31445312e-01,  2.83203125e-01,  1.42578125e-01,
         3.41796875e-01, -2.39257812e-02, -1.09863281e-01,  3.32031250e-02,
        -5.46875000e-02,  1.53198242e-02, -1.62109375e-01,  1.58203125e-01,
        -2.59765625e-01,  2.01416016e-02, -1.63085938e-01,  1.35803223e-03,
        -1.44531250e-01, -5.68847656e-02,  4.29687500e-02, -2.46582031e-02,
         1.85546875e-01,  4.47265625e-01,  9.58251953e-03,  1.31835938e-01,
         9.86328125e-02, -1.85546875e-01, -1.00097656e-01, -1.33789062e-01,
        -1.25000000e-01,  2.83203125e-01,  1.23046875e-01,  5.32226562e-02,
        -1.77734375e-01,  8.59375000e-02, -2.18505859e-02,  2.05078125e-02,
        -1.39648438e-01,  2.51464844e-02,  1.38671875e-01, -1.05468750e-01,
         1.38671875e-01,  8.88671875e-02, -7.51953125e-02, -2.13623047e-02,
         1.72851562e-01,  4.63867188e-02, -2.65625000e-01,  8.91113281e-03,
         1.49414062e-01,  3.78417969e-02,  2.38281250e-01, -1.24511719e-01,
        -2.17773438e-01, -1.81640625e-01,  2.97851562e-02,  5.71289062e-02,
        -2.89306641e-02,  1.24511719e-02,  9.66796875e-02, -2.31445312e-01,
         5.81054688e-02,  6.68945312e-02,  7.08007812e-02, -3.08593750e-01,
        -2.14843750e-01,  1.45507812e-01, -4.27734375e-01, -9.39941406e-03,
         1.54296875e-01, -7.66601562e-02,  2.89062500e-01,  2.77343750e-01,
        -4.86373901e-04, -1.36718750e-01,  3.24218750e-01, -2.46093750e-01,
        -3.03649902e-03, -2.11914062e-01,  1.25000000e-01,  2.69531250e-01,
         2.04101562e-01,  8.25195312e-02, -2.01171875e-01, -1.60156250e-01,
        -3.78417969e-02, -1.20117188e-01,  1.15234375e-01, -4.10156250e-02,
        -3.95507812e-02, -8.98437500e-02,  6.34765625e-03,  2.03125000e-01,
         1.86523438e-01,  2.73437500e-01,  6.29882812e-02,  1.41601562e-01,
        -9.81445312e-02,  1.38671875e-01,  1.82617188e-01,  1.73828125e-01,
         1.73828125e-01, -2.37304688e-01,  1.78710938e-01,  6.34765625e-02,
         2.36328125e-01, -2.08984375e-01,  8.74023438e-02, -1.66015625e-01,
        -7.91015625e-02,  2.43164062e-01, -8.88671875e-02,  1.26953125e-01,
        -2.16796875e-01, -1.73828125e-01, -3.59375000e-01, -8.25195312e-02,
        -6.49414062e-02,  5.07812500e-02,  1.35742188e-01, -7.47070312e-02,
        -1.64062500e-01,  1.15356445e-02,  4.45312500e-01, -2.15820312e-01,
        -1.11328125e-01, -1.92382812e-01,  1.70898438e-01, -1.25000000e-01,
         2.65502930e-03,  1.92382812e-01, -1.74804688e-01,  1.39648438e-01,
         2.92968750e-01,  1.13281250e-01,  5.95703125e-02, -6.39648438e-02,
         9.96093750e-02, -2.72216797e-02,  1.96533203e-02,  4.27246094e-02,
        -2.46093750e-01,  6.39648438e-02, -2.25585938e-01, -1.68945312e-01,
         2.89916992e-03,  8.20312500e-02,  3.41796875e-01,  4.32128906e-02,
         1.32812500e-01,  1.42578125e-01,  7.61718750e-02,  5.98144531e-02,
        -1.19140625e-01,  2.74658203e-03, -6.29882812e-02, -2.72216797e-02,
        -4.82177734e-03, -8.20312500e-02, -2.49023438e-02, -4.00390625e-01,
        -1.06933594e-01,  4.24804688e-02,  7.76367188e-02, -1.16699219e-01,
         7.37304688e-02, -9.22851562e-02,  1.07910156e-01,  1.58203125e-01,
         4.24804688e-02,  1.26953125e-01,  3.61328125e-02,  2.67578125e-01,
        -1.01074219e-01, -3.02734375e-01, -5.76171875e-02,  5.05371094e-02,
         5.26428223e-04, -2.07031250e-01, -1.38671875e-01, -8.97216797e-03,
        -2.78320312e-02, -1.41601562e-01,  2.07031250e-01, -1.58203125e-01,
         1.27929688e-01,  1.49414062e-01, -2.24609375e-02, -8.44726562e-02,
         1.22558594e-01,  2.15820312e-01, -2.13867188e-01, -3.12500000e-01,
        -3.73046875e-01,  4.08935547e-03,  1.07421875e-01,  1.06933594e-01,
         7.32421875e-02,  8.97216797e-03, -3.88183594e-02, -1.29882812e-01,
```

```
 1.49414062e-01, -2.14843750e-01, -1.83868408e-03,  9.91210938e-02,
 1.57226562e-01, -1.14257812e-01, -2.05078125e-01,  9.91210938e-02,
 3.69140625e-01, -1.97265625e-01,  3.54003906e-02,  1.09375000e-01,
 1.31835938e-01,  1.66992188e-01,  2.35351562e-01,  1.04980469e-01,
-4.96093750e-01, -1.64062500e-01, -1.56250000e-01, -5.22460938e-02,
 1.03027344e-01,  2.43164062e-01, -1.88476562e-01,  5.07812500e-02,
-9.37500000e-02, -6.68945312e-02,  2.27050781e-02,  7.61718750e-02,
 2.89062500e-01,  3.10546875e-01, -5.37109375e-02,  2.28515625e-01,
 2.51464844e-02,  6.78710938e-02, -1.21093750e-01, -2.15820312e-01,
-2.73437500e-01, -3.07617188e-02, -3.37890625e-01,  1.53320312e-01,
 2.33398438e-01, -2.08007812e-01,  3.73046875e-01,  8.20312500e-02,
 2.51953125e-01, -7.61718750e-02, -4.66308594e-02, -2.23388672e-02,
 2.99072266e-02, -5.93261719e-02, -4.66918945e-03, -2.44140625e-01,
-2.09960938e-01, -2.87109375e-01, -4.54101562e-02, -1.77734375e-01,
-2.79296875e-01, -8.59375000e-02,  9.13085938e-02,  2.51953125e-01],
dtype=float32)
```

4. Trouver la similitude entre 2 mots

In [16]:
```
# Word2Vec supports several word similarity tasks out of the box.
# You can see how the similarity intuitively decreases as the words get less and

pairs = [
    ('car', 'minivan'),     # a minivan is a kind of car
    ('car', 'bicycle'),     # still a wheeled vehicle
    ('car', 'airplane'),    # ok, no wheels, but still a vehicle
    ('car', 'cereal'),      # ... and so on
    ('car', 'communism'),
]
for w1, w2 in pairs:
    print('%r\t%r\t%.2f' % (w1, w2, wv.similarity(w1, w2)))
```

```
'car'	'minivan'	0.69
'car'	'bicycle'	0.54
'car'	'airplane'	0.42
'car'	'cereal'	0.14
'car'	'communism'	0.06
```

5. Jouez un peu avec la similitude des mots

In [17]:
```
# Print the 5 most similar words to "car" or "minivan"
print(wv.most_similar(positive=['car', 'minivan'], topn=5))
```

```
[('SUV', 0.8532192707061768), ('vehicle', 0.8175783753395081), ('pickup_truck', 0.7763688564300537), ('Jeep', 0.7567334175109863), ('Ford_Explorer', 0.7565720081329346)]
```

In [18]:
```
# Which of the below does not belong in the sequence?
print(wv.doesnt_match(['fire', 'water', 'land', 'sea', 'air', 'car']))
```

```
car
```