Word2Vec-以 gensim 訓練中文詞向量

參考及引用資料來源

- [1] zake7749-使用 gensim 訓練中文詞向量 (http://zake7749.github.io/2016/08/28/word2vec-with-gensim/)
- [2] gensim/corpora/wikicorpus (https://radimrehurek.com/gensim/corpora/wikicorpus.html)
- Word2Vec的簡易教學與參數調整指南 (https://www.kaggle.com/jerrykuo7727/word2vec)
- zhconv (https://pypi.org/project/zhconv/)
- jieba (https://pypi.org/project/jieba/)

In [2]:

```
%load_ext memory_profiler
!pip install -q zhconv
```

確認相關 Packages

In [3]:

```
import os

# Packages
import gensim
import jieba
import zhconv
from gensim.corpora import WikiCorpus
from datetime import datetime as dt
from typing import List


if not os.path.isfile('dict.txt.big'):
    !wget https://github.com/fxsjy/jieba/raw/master/extra_dict/dict.txt.big
jieba.set_dictionary('dict.txt.big')

print("gensim", gensim.__version__)
print("jieba", jieba.__version__)
```

```
gensim 4.1.2
jieba 0.42.1
```

準備中文訓練文本

訓練文本來源: 維基百科資料庫

(https://zh.wikipedia.org/wiki/Wikipedia:%E6%95%B0%E6%8D%A

> 要訓練詞向量，第一步當然是取得資料集。由於 word2vec 是基於非監督式學習，**訓練集一定一定要越大越好，語料涵蓋的越全面，訓練出來的結果也會越漂亮**。[1] (http://zake7749.github.io/2016/08/28/word2vec-with-gensim/)

- zhwiki-20210101-pages-articles.xml.bz2 (https://dumps.wikimedia.org/zhwiki/20210101/zhwiki-20210101-pages-articles.xml.bz2) (1.9 GB)

目前已經使用另一份 Notebook (維基百科中文語料庫 zhWiki_20210101 (https://www.kaggle.com/bbqlp33/zhwiki-20210101)) 下載好中文維基百科語料，並可以直接引用

In [17]:

```
#ZhWiki = "/kaggle/input/zhwiki-20210101/zhwiki-20210101-pages-articles.xml.bz2"
ZhWiki = r"C:\Users\user\Desktop\exercise\NLP\zhwiki-20230501-pages-articles.xml.bz2"
#"C:\Users\user\Desktop\exercise\NLP\zhwiki-20230501-pages-articles.xml.bz2"
!du -sh $ZhWiki
!md5sum $ZhWiki
!file $ZhWiki
```

```
'du' 不是內部或外部命令、可執行的程式或批次檔。
'md5sum' 不是內部或外部命令、可執行的程式或批次檔。
'file' 不是內部或外部命令、可執行的程式或批次檔。
```

中文文本前處理

在正式訓練 `Word2Vec` 之前，其實涉及了文本的前處理，本篇的處理包括如下三點 (而實務上對應的不同使用情境，可能會有不同的前處理流程):

- 簡轉繁: zhconv (https://pypi.org/project/zhconv/)
- 中文斷詞: jieba (https://pypi.org/project/jieba/)
- 停用詞

簡繁轉換

wiki 文本其實摻雜了簡體與繁體中文，比如「数学」與「數學」，這會被 word2vec 當成兩個不同的詞。[1] (http://zake7749.github.io/2016/08/28/word2vec-with-gensim/)
所以我們在斷詞前，需要加上簡繁轉換的手續

以下範例使用了較輕量的 Package zhconv (https://pypi.org/project/zhconv/)，
若需要更高的精準度，則可以參考 OpenCC (https://github.com/BYVoid/OpenCC)

In [6]:

```
zhconv.convert("这原本是一段简体中文", "zh-tw")
```

Out[6]:

```
'這原本是一段簡體中文'
```

中文斷詞

使用 jieba (https://pypi.org/project/jieba/) `jieba.cut` 來進行中文斷詞，
並簡單介紹 jieba 的兩種分詞模式:

- `cut_all=False` **精確模式**，試圖將句子最精確地切開，適合文本分析；
- `cut_all=True` **全模式**，把句子中所有的可以成詞的詞語都掃描出來, 速度非常快，但是不能解決歧義；

而本篇文本訓練採用**精確模式** `cut_all=False`

In [7]:

```
seg_list = jieba.cut("我来到北京清华大学", cut_all=True)
print("Full Mode: " + "/ ".join(seg_list))  # 全模式

seg_list = jieba.cut("我来到北京清华大学", cut_all=False)
print("Default Mode: " + "/ ".join(seg_list))  # 精確模式
```

```
Building prefix dict from C:\Users\user\Desktop\exercise\NLP\dict.txt.big
...
Dumping model to file cache C:\Users\user\AppData\Local\Temp\jieba.u1f62a
42fe1738d51f5bb243f550ba545.cache
Loading model cost 1.276 seconds.
Prefix dict has been built successfully.

Full Mode: 我/ 来到/ 北京/ 清华/ 清华大学/ 华大/ 大学
Default Mode: 我/ 来到/ 北京/ 清华大学
```

In [8]:

```
print(list(jieba.cut("中英夾雜的example，Word2Vec應該很interesting吧?")))
```

```
['中', '英', '夾雜', '的', 'example', '，', 'Word2Vec', '應該', '很', 'inte
resting', '吧', '?']
```

引入停用詞表

停用詞就是像英文中的 **the,a,this**，中文的**你我他**，與其他詞相比顯得不怎麼重要，對文章主題也無關緊要的，
是否要使用停用詞表，其實還是要看你的應用，也有可能保留這些停用詞更能達到你的目標。[1] (http://zake7749.github.io/2016/08/28/word2vec-with-gensim/)

- Is it compulsory to remove stop words with word2vec? (https://www.quora.com/Is-it-compulsory-to-remove-stop-words-with-word2vec)
- The Effect of Stopword Filtering prior to Word Embedding Training (https://stats.stackexchange.com/questions/201372/the-effect-of-stopword-filtering-prior-to-word-

embedding-training)

以下範例還是示範引入停用詞表，而停用詞表網路上有各種各樣的資源
剛好 `kaggle` ，環境預設有裝 spacy (https://pypi.org/project/spacy/)，
就順道引用 `spacy` 提供的停用詞表吧 (實務上stopwords 應為另外準備好且檢視過的靜態文檔)

In [10]:

```
import spacy

# 下載語言模組
spacy.cli.download("zh_core_web_sm")  # 下載 spacy 中文模組
spacy.cli.download("en_core_web_sm")  # 下載 spacy 英文模組

nlp_zh = spacy.load("zh_core_web_sm") # 載入 spacy 中文模組
nlp_en = spacy.load("en_core_web_sm") # 載入 spacy 英文模組

# 印出前20個停用詞
print('--\n')
print(f"中文停用詞 Total={len(nlp_zh.Defaults.stop_words)}: {list(nlp_zh.Defaults.stop_wo
print("--")
print(f"英文停用詞 Total={len(nlp_en.Defaults.stop_words)}: {list(nlp_en.Defaults.stop_wo
```

```
✔ Download and installation successful
You can now load the package via spacy.load('zh_core_web_sm')
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
--

中文停用詞 Total=1891: ['诚然', '同', '怎样', '■', '变成', '呜呼', '一般',
'犹且', '是以', '论说', '什么', '如次', '倘若', '︿', '咳', '至今', '多么',
'最后', '啊哟', '且不说'] ...
--
英文停用詞 Total=326: ['‘re', 'empty', 'forty', 'ten', 'few', 'thru',
'‘s', 'meanwhile', 'no', 'perhaps', 'upon', 'although', 'call', 'it',
"'m", 'off', 'same', 'sometimes', 'eight', 'using'] ...
```

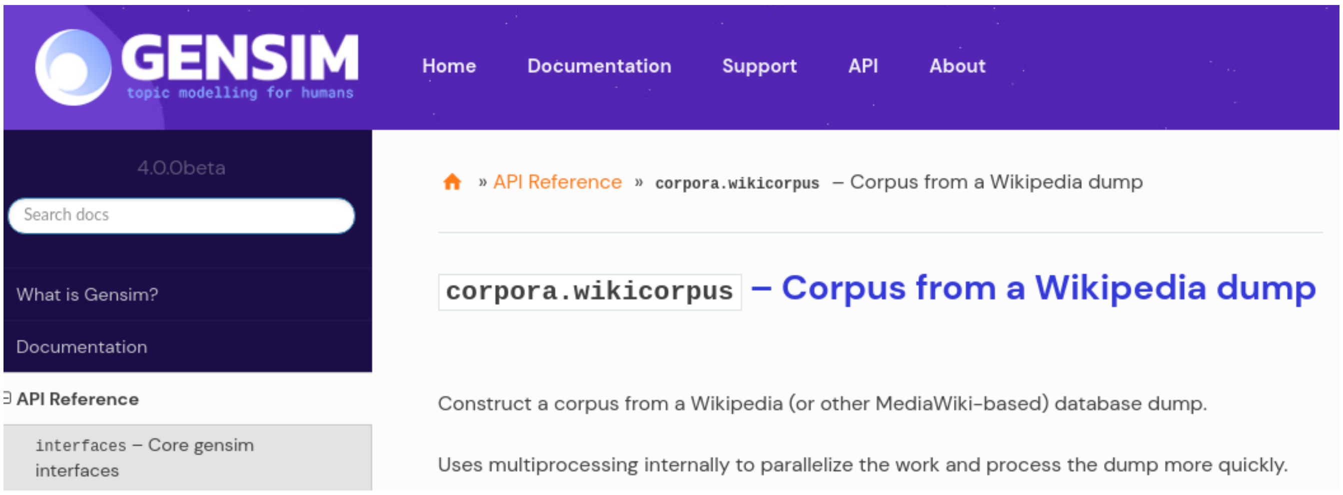
In [11]:

```
STOPWORDS =  nlp_zh.Defaults.stop_words | \
             nlp_en.Defaults.stop_words | \
             set(["\n", "\r\n", "\t", " ", ""])
print(len(STOPWORDS))

# 將簡體停用詞轉成繁體，擴充停用詞表
for word in STOPWORDS.copy():
    STOPWORDS.add(zhconv.convert(word, "zh-tw"))

print(len(STOPWORDS))
```

```
2222
3005
```

讀取 wiki 語料庫，並且進行前處理和斷詞

維基百科（`wiki.xml.bz2`）下載好後，先別急著解壓縮，因為這是一份 xml 文件，裏頭佈滿了各式各樣的標籤，我們得先想辦法送走這群不速之客，不過也別太擔心，`gensim` 早已看穿了一切，藉由調用 [wikiCorpus (https://radimrehurek.com/gensim/corpora/wikicorpus.html)](https://radimrehurek.com/gensim/corpora/wikicorpus.html)，我們能很輕鬆的只取出文章的標題和內容。[[1] (http://zake7749.github.io/2016/08/28/word2vec-with-gensim/)](http://zake7749.github.io/2016/08/28/word2vec-with-gensim/)


GENSIM
topic modelling for humans
Home
Documentation
Support
API
About
4.0.0beta
Search docs
What is Gensim?
Documentation
API Reference
interfaces – Core gensim interfaces
» API Reference » corpora.wikicorpus – Corpus from a Wikipedia dump

corpora.wikicorpus – Corpus from a Wikipedia dump

Construct a corpus from a Wikipedia (or other MediaWiki-based) database dump.

Uses multiprocessing internally to parallelize the work and process the dump more quickly.

[[2] (https://radimrehurek.com/gensim/corpora/wikicorpus.html)](https://radimrehurek.com/gensim/corpora/wikicorpus.html)

Supported dump formats:

- `<LANG>wiki-<YYYYMMDD>-pages-articles.xml.bz2`
- `<LANG>wiki-latest-pages-articles.xml.bz2`

In [12]:

```
def preprocess_and_tokenize(
    text: str, token_min_len: int=1, token_max_len: int=15, lower: bool=True) -> List[st
    if lower:
        text  = text.lower()
    text = zhconv.convert(text, "zh-tw")
    return [
        token for token in jieba.cut(text, cut_all=False)
        if token_min_len <= len(token) <= token_max_len and \
            token not in STOPWORDS
    ]
```

In [13]:

```
print(preprocess_and_tokenize("歐幾里得，西元前三世紀的古希臘數學家，現在被認為是幾何之父，此
print(preprocess_and_tokenize("我来到北京清华大学"))
print(preprocess_and_tokenize("中英夾雜的example，Word2Vec應該很interesting吧?"))
```

```
['歐幾', '裡得', '西元前', '世紀', '古希臘', '數學家', '幾何', '父', '此書',
'拉斐爾']
['來到', '北京', '清華大學']
['中', '英', '夾雜', 'example', 'word2vec', 'interesting']
```

In [ ]:

```
%%time
%%memit

print(f"Parsing {ZhWiki}...")
wiki_corpus = WikiCorpus(ZhWiki, tokenizer_func=preprocess_and_tokenize, token_min_len=1
```

In [ ]:

```

```

In [*]:

```
g = wiki_corpus.get_texts()
print(next(g)[:10])
print(next(g)[:10])
print(next(g)[:10])


# print(jieba.lcut("".join(next(g))[:50]))
# print(jieba.lcut("".join(next(g))[:50]))
```

初始化 `WikiCorpus` 後，能藉由 `get_texts()` 可迭代每一篇文章，它所回傳的是一個 `tokens list` ，我以空白符將這些 `tokens` 串接起來，統一輸出到同一份文字檔裡。這邊要注意一件事， `get_texts()` 受 `article_min_tokens` 參數的限制，只會回傳內容長度大於 **50** (default) 的文章。

- **article_min_tokens** *(int, optional)* – Minimum tokens in article. Article will be ignored if number of tokens is less.

秀出前 3 偏文章的前10 個 token

將處理完的語料集存下來，供後續使用

In [*]:

```
WIKI_SEG_TXT = "wiki_seg.txt"

generator = wiki_corpus.get_texts()

with open(WIKI_SEG_TXT, "w", encoding='utf-8') as output:
    for texts_num, tokens in enumerate(generator):
        output.write(" ".join(tokens) + "\n")

        if (texts_num + 1) % 100000 == 0:
            print(f"[{str(dt.now()):.19}] 已寫入 {texts_num} 篇斷詞文章")
```

訓練 Word2Vec

In [14]:

```
%%time

from gensim.models import word2vec
import multiprocessing

max_cpu_counts = multiprocessing.cpu_count()
word_dim_size = 300  #  設定 word vector 維度
print(f"Use {max_cpu_counts} workers to train Word2Vec (dim={word_dim_size})")


# 讀取訓練語句
sentences = word2vec.LineSentence(WIKI_SEG_TXT)

# 訓練模型
model = word2vec.Word2Vec(sentences, size=word_dim_size, workers=max_cpu_counts)

# 儲存模型
output_model = f"word2vec.zh.{word_dim_size}.model"
model.save(output_model)
```

```
Use 4 workers to train Word2Vec (dim=300)
CPU times: user 3h 8min 9s, sys: 1min 25s, total: 3h 9min 34s
Wall time: 1h 4min 34s
```

儲存的模型總共會產生三份檔案

In [15]:

```
! ls word2vec.zh*
```

```
word2vec.zh.300.model
word2vec.zh.300.model.trainables.syn1neg.npy
word2vec.zh.300.model.wv.vectors.npy
```

In [16]:

```
!du -sh word2vec.zh*
```

```
71M     word2vec.zh.300.model
1.3G    word2vec.zh.300.model.trainables.syn1neg.npy
1.3G    word2vec.zh.300.model.wv.vectors.npy
```

查看模型以及詞向量實驗

模型其實就是巨大的 Embedding Matrix

In [17]:

```
print(model.wv.vectors.shape)
model.wv.vectors
```

```
(1138562, 300)
```

Out[17]:

```
array([[-1.9005029e+00, -2.8552267e-01, -1.6861261e+00, ...,
        -2.3989561e+00,  7.9282230e-01, -1.7689761e+00],
       [-8.9374363e-01,  2.7187495e+00, -2.8586307e+00, ...,
         7.6278639e-01, -1.2364342e+00,  4.6035990e-01],
       [-1.3627478e+00,  9.0678096e-01, -2.4168897e+00, ...,
         1.9464742e-01,  2.3467582e-01, -4.3651393e-01],
       ...,
       [ 1.1225304e-01, -2.3689542e-02,  5.3460799e-02, ...,
         1.8911289e-02,  6.1728880e-02, -2.9928887e-02],
       [ 1.0599125e-01,  1.4494479e-03, -7.2507304e-03, ...,
         3.7081163e-02,  1.5778864e-02, -1.6071750e-02],
       [-5.2874412e-02, -6.3391119e-02, -1.4798551e-02, ...,
        -7.8865103e-02,  3.2674141e-02, -2.9911553e-02]], dtype=float32)
```

收錄的詞彙

In [18]:

```
print(f"總共收錄了 {len(model.wv.vocab)} 個詞彙")

print("印出 20 個收錄詞彙:")
print(list(model.wv.vocab.keys())[:10])
```

```
總共收錄了 1138562 個詞彙
印出 20 個收錄詞彙:
['歐幾里得', '西元前', '世紀', '古希臘', '數學家', '認為', '幾何', '父', '此
畫', '為']
```

詞彙的向量

In [19]:

```
vec = model.wv['數學家']
print(vec.shape)
vec
```

```
(300,)
```

Out[19]:

沒見過的詞彙

In [20]:

```
word = "這肯定沒見過 "

# 若強行取值會報錯
try:
    vec = model.wv[word]
except KeyError as e:
    print(e)
```

```
"word '這肯定沒見過 ' not in vocabulary"
```

查看前 10 名相似詞

`model.wv.most_similar` 的 `topn` 預設為 10

In [21]:

```
model.wv.most_similar("飲料", topn=10)
```

Out[21]:

```
[('飲品', 0.7986035346984863),
 ('果汁', 0.6965157389640808),
 ('含酒精', 0.6784194707870483),
 ('提神', 0.6489840149879456),
 ('酒精類', 0.6488814353942871),
 ('罐裝', 0.6478314995765686),
 ('軟飲料', 0.6374701261520386),
 ('啤酒', 0.6282455921173096),
 ('酒類', 0.6256542801856995),
 ('優格', 0.6210787892341614)]
```

In [22]:

```
model.wv.most_similar("car")
```

Out[22]:

```
[('truck', 0.6967549324035645),
 ('motor', 0.6519397497177124),
 ('motorcycle', 0.6470927596092224),
 ('seat', 0.6259925866127014),
 ('vehicle', 0.6339040994644165),
 ('chevrolet', 0.6203585267066956),
 ('wagon', 0.6203234105709229),
 ('volkswagen', 0.6184985637664795),
 ('cab', 0.6164403560791016),
 ('carriage', 0.6120896339416504)]
```

In [23]:

```
model.wv.most_similar("facebook")
```

Out[23]:

```
[('臉書', 0.8205897212028503),
 ('專頁', 0.7610922455787659),
 ('面書', 0.7100078773498535),
 ('instagram', 0.7310690879821777),
 ('貼文', 0.7073481372966766),
 ('推特', 0.6892963647842407),
 ('粉絲團', 0.6821428537368774),
 ('twitter', 0.6560337543487549),
 ('網誌', 0.6300784349441528),
 ('youtube', 0.626787543296814)]
```

In [24]:

```
model.wv.most_similar("詐欺")
```

Out[24]:

```
[('欺詐', [illegible]),
 ('詐騙', [illegible]),
 ('竊盜', [illegible]),
 ('慣犯', [illegible]),
 ('詐欺罪', [illegible]),
 ('信用調查', [illegible]),
 ('詐術', [illegible]),
 ('逃稅', [illegible]),
 ('逃漏稅', [illegible]),
 ('賭博', [illegible])]
```

```
array([-0.15254694, -1.7895197 ,  2.8389933 ,  0.30419526,  0.94147015,
       -0.924163  ,  0.05298484, -0.13174261, -1.4274364 , -0.85435706,
        1.8479271 , -1.0273514 ,  0.05875092,  1.6739838 ,  2.3618689 ,
        2.0175905 , -1.5817889 , -0.8472086 ,  1.1092311 , -0.24629189,
        0.49799377,  0.36100635, -0.281036  , -0.5806378 , -1.8780046 ,
        0.2837012 , -0.11682022, -0.22787744,  0.11489099, -0.14591609,
       -0.20482585, -1.1259157 , -0.01348399, -1.0098934 ,  3.2184417 ,
       -0.7147375 , -0.7956163 ,  1.9969889 ,  0.14035068,  3.6188982 ,
        1.7953275 , -1.4445941 ,  0.37768763, -0.77543026, -0.69721234,
        0.8814372 , -0.14605104,  1.7810897 ,  1.9077462 ,  0.33889568,
        2.4869808 ,  0.05023988, -1.7290001 ,  0.52984065,  1.971243  ,
        1.042739  , -2.2993985 , -0.15407254,  1.0061789 ,  0.36432067,
       -0.12965468,  0.22315088, -0.6530562 ,  0.5096923 ,  1.6476531 ,
        2.8283326 , -0.41265598,  3.710713  , -2.182122  , -1.2232319 ,
       -0.13470766,  1.6847824 ,  0.3732413 ,  0.38064095,  3.7629342 ,
        0.5054948 ,  1.7682022 ,  0.3355076 , -0.82843256, -0.01899206,
        1.3599714 , -0.3263021 ,  0.5632072 , -1.6275458 ,  0.18896504,
       -2.8486328 ,  0.4079449 ,  0.76394415,  0.39884079, -1.1291069 ,
        1.616785  , -2.2050192 , -0.68297696, -0.7727057 ,  1.8106886 ,
        0.3693791 ,  1.4199053 ,  0.9511317 ,  0.3185586 , -2.787385  ,
       -2.046976  ,  1.1417929 ,  0.65914726, -0.09776347,  1.291978  ,
        0.9779469 , -2.1436791 ,  1.177742  ,  2.2546558 , -1.7125031 ,
       -0.95545936, -0.11377065, -1.223585  , -1.3839793 , -1.6620296 ,
        0.9213591 ,  0.16553962,  2.6440482 , -0.19194634,  0.38509196,
        1.714671  , -0.81765043,  0.23664996,  1.3536583 ,  2.3723216 ,
       -3.0491447 , -0.9825735 , -1.3566401 , -1.283516  ,  0.4361322 ,
        2.008308  , -0.06008134,  2.1489894 , -0.470945  ,  0.28165355,
        1.0654688 ,  0.7665213 , -0.47257465, -0.8030565 , -1.4145786 ,
       -1.7558275 ,  2.0288863 , -3.5372381 ,  1.4158623 ,  0.3096819 ,
        0.704036  , -1.3761473 ,  1.6208903 , -1.2401619 , -0.13551526,
       -1.9005125 ,  0.8854844 ,  0.4481448 , -0.4172336 , -1.206034  ,
        0.2372846 ,  0.4517836 , -1.966441  , -0.44706354,  0.34252086,
        1.6452466 ,  0.10298152, -0.6640831 , -0.80406773, -1.0047605 ,
        0.25708237, -1.2658246 ,  0.72502005, -0.16557129, -2.1745164 ,
       -0.37656462, -1.1152672 ,  1.0244526 ,  1.1565366 , -1.3018371 ,
       -0.61735666, -0.47080624,  1.0154963 ,  0.27704978, -0.43282175,
        2.8569677 ,  0.9180066 , -1.0534698 ,  0.8483542 , -2.0334537 ,
       -1.710315  , -1.3449622 ,  2.2180285 , -0.22329706,  0.3695973 ,
       -1.5329542 , -0.46323976, -1.4827193 ,  0.73403674, -0.34677625,
        1.0670941 ,  0.88510114, -1.906598  , -0.5413286 , -3.2423143 ,
       -0.9742989 , [illegible] , -2.0051272 ,  0.8977296 , -0.39034557,
        0.4364136 , [illegible] ,  0.86995924, -0.42781776, -0.79106665,
       -1.8506816 , [illegible] ,  1.2932605 , -1.6092048 , -1.5371683 ,
       -0.9567902 , [illegible] , -1.6739159 ,  0.9763718 , -0.8025273 ,
        0.2590659 , [illegible] ,  0.2761859 ,  2.3008378 , -0.7928667 ,
        0.5515494 , [illegible] ,  0.14855805,  3.2365885 , -3.3151088 ,
       -1.3501531 , [illegible] , -1.2938559 , -2.1719165 ,  0.03638348,
       -0.6302786 , [illegible] ,  1.6141753 , -0.09968105, -2.897564  ,
        2.0326308 , [illegible] ,  2.1948044 ,  1.0531849 ,  1.6850697 ,
       -2.8527457 , [illegible] , -0.9736342 ,  0.47620815, -0.53191006,
        2.4831314 , -1.8135078 , -0.09655156, -0.26880732, -1.5746521 ,
        3.447743  ,  0.7701042 , -2.079731  , -1.1631368 ,  0.29982182,
       -0.07716776, -3.0327997 ,  3.0748208 , -1.4359124 ,  1.0085474 ,
        0.0964113 ,  0.32251447,  3.6297605 , -1.2651255 , -0.76287466,
        0.31288242,  0.9797535 ,  0.22678709,  0.11903057,  0.6259719 ,
       -2.007877  , -0.33391798, -1.288079  ,  1.391751  ,  0.90182006,
        1.5843096 , -0.94183683, -0.56292737, -1.1241164 , -1.5128164 ,
       -1.3753456 , -2.1157842 ,  1.3935559 , -1.5951293 ,  1.7871724 ,
       -1.3674058 , -0.77202106,  0.08307242,  0.7245565 ,  2.0208385 ,
       -0.72172374, -1.9296253 ,  0.6796764 ,  2.4959662 , -0.91734695],
      dtype=float32)
```

In [25]:

```
model.wv.most_similar("合約")
```

Out[25]:

```
[('合同', 0.7740179896354675),
 ('簽約', 0.6996151208877563),
 ('續約', 0.6664248704910278),
 ('租約', 0.616113007068634),
 ('簽下', 0.615209698677063),
 ('續簽', 0.6073216199874878),
 ('買斷', 0.5997638702392578),
 ('新東家', 0.5763907432556152),
 ('選擇權', 0.5660616159439087),
 ('勞動合同', 0.5596709251403809)]
```

計算 Cosine 相似度

In [26]:

```
model.wv.similarity("連結", "鍵接")
```

Out[26]:

```
0.709352
```

In [27]:

```
model.wv.similarity("連結", "陰天")
```

Out[27]:

```
-0.023800302
```

讀取模型

In [28]:

```
print(f"Loading {output_model}...")
new_model = word2vec.Word2Vec.load(output_model)
```

```
Loading word2vec.zh.300.model...
```

In [29]:

```
model.wv.similarity("連結", "陰天") == new_model.wv.similarity("連結", "陰天")
```

Out[29]:

```
True
```