08IrTheory

November 4, 2018

1 英文 NLP 基礎教學

```
In [1]: import pandas as pd

        import nltk

        from nltk.stem.porter import PorterStemmer
        porter_stemmer = PorterStemmer()
        from nltk.stem.lancaster import LancasterStemmer
        lancaster_stemmer = LancasterStemmer()
        from nltk.stem import SnowballStemmer
        snowball_stemmer = SnowballStemmer('english')
        from nltk.stem import WordNetLemmatizer
        wordnet_lemmatizer = WordNetLemmatizer()


        from nltk.corpus import stopwords
        stops = stopwords.words('english')
        from string import punctuation
```

```
In [2]: testStr = "This value is also called cut-off in the literature. If float, the paramete
        # 請使用 nltk.word_tokenize 及 nltk.wordpunct_tokenize 進行分詞，並比較其中差異。
        #=============your works starts===============#
        word_tokenize_tokens = nltk.word_tokenize(testStr)
        wordpunct_tokenize_tokens = nltk.wordpunct_tokenize(testStr)
        #==============your works ends================#

        print("/".join(word_tokenize_tokens))
        print("/".join(wordpunct_tokenize_tokens))
        # This/value/is/also/called/cut-off/in/the/literature/./If/float/,/the/parameter/repre
        # This/value/is/also/called/cut/-/off/in/the/literature/./If/float/,/the/parameter/rep
```

```
This/value/is/also/called/cut-off/in/the/literature/./If/float/,/the/parameter/represents/a/pro
This/value/is/also/called/cut/-/off/in/the/literature/./If/float/,/the/parameter/represents/a/p
```

```
In [3]: tokens = wordpunct_tokenize_tokens
        df = pd.DataFrame(index = tokens)
```

```
# 請使用 porter_stemmer, lancaster_stemmer, snowball_stemmer, wordnet_lemmatizer，進行
#=============your works starts===============#
df['porter_stemmer'] = [porter_stemmer.stem(t) for t in tokens]
df['lancaster_stemmer'] = [lancaster_stemmer.stem(t) for t in tokens]
df['snowball_stemmer'] = [snowball_stemmer.stem(t) for t in tokens]
df['wordnet_lemmatizer'] = [wordnet_lemmatizer.lemmatize(t) for t in tokens]
#==============your works ends================#

print(df.iloc[0].tolist())
print(df.iloc[1].tolist())
# ['thi', 'thi', 'this', 'This']
# ['valu', 'valu', 'valu', 'value']

df
```

```
['thi', 'thi', 'this', 'This']
['valu', 'valu', 'valu', 'value']
```

```
Out[3]:            porter_stemmer lancaster_stemmer snowball_stemmer  \
        This                  thi               thi             this
        value                valu              valu             valu
        is                     is                is               is
        also                 also              also             also
        called               call               cal             call
        cut                   cut               cut              cut
        -                       -                 -                -
        off                   off               off              off
        in                     in                in               in
        the                   the               the              the
        literature      literatur               lit        literatur
        .                       .                 .                .
        If                     If                if               if
        float               float               flo            float
        ,                       ,                 ,                ,
        the                   the               the              the
        parameter         paramet           paramet          paramet
        represents         repres            repres           repres
        a                       a                 a                a
        proportion        proport           proport          proport
        of                     of                of               of
        documents        document              docu         document
        ,                       ,                 ,                ,
        integer             integ             integ            integ
        absolute          absolut            absolv          absolut
        counts              count             count            count
        .                       .                 .                .
```

```
           wordnet_lemmatizer
This                     This
value                   value
is                         is
also                     also
called                 called
cut                       cut
-                           -
off                       off
in                         in
the                       the
literature         literature
.                           .
If                         If
float                   float
,                           ,
the                       the
parameter           parameter
represents         represents
a                           a
proportion         proportion
of                         of
documents            document
,                           ,
integer               integer
absolute             absolute
counts                  count
.                           .
```

```
In [4]: print("標點符號")
        print(punctuation)
        print("停用字")
        print(stops)
```

```
標點符號
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
停用字
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'
```

```
In [5]: df = pd.DataFrame(index = [t for t in tokens if t not in stops and t not in punctuatio
        # 請使用 porter_stemmer, lancaster_stemmer, snowball_stemmer, wordnet_lemmatizer，進行
        # 請去除標點符號與停用字
        #============your works starts===============#
        df['porter_stemmer'] = [porter_stemmer.stem(t.lower()) for t in tokens if t not in sto
        df['lancaster_stemmer'] = [lancaster_stemmer.stem(t.lower()) for t in tokens if t not
        df['snowball_stemmer'] = [snowball_stemmer.stem(t.lower()) for t in tokens if t not in
        df['wordnet_lemmatizer'] = [wordnet_lemmatizer.lemmatize(t.lower()) for t in tokens if
        #==============your works ends================#

        df
```

```
Out[5]:             porter_stemmer lancaster_stemmer snowball_stemmer  \
        This                   thi               thi             this
        value                 valu              valu             valu
        also                  also              also             also
        called                call               cal             call
        cut                    cut               cut              cut
        literature       literatur               lit        literatur
        If                      if                if               if
        float                float               flo            float
        parameter          paramet           paramet          paramet
        represents          repres            repres           repres
        proportion         proport           proport          proport
        documents         document              docu         document
        integer              integ             integ            integ
        absolute           absolut            absolv          absolut
        counts               count             count            count

                   wordnet_lemmatizer
        This                     this
        value                   value
        also                     also
        called                 called
        cut                       cut
        literature         literature
        If                         if
        float                   float
        parameter           parameter
        represents         represents
        proportion         proportion
        documents            document
        integer               integer
        absolute             absolute
        counts                  count
```

```
In [6]: df_tag = pd.DataFrame(index = tokens)
        # 請使用 nltk.pos_tag 進行詞性標記，並嘗試設定參數 tagset='universal'
        #============your works starts==============#
        df_tag['default'] = [tag for term, tag in nltk.pos_tag(tokens)]
        df_tag['universal'] = [tag for term, tag in nltk.pos_tag(tokens, tagset='universal')]
        #==============your works ends================#

        df_tag
```

```
Out[6]:             default universal
        This             DT       DET
```

```
value           NN      NOUN
is             VBZ      VERB
also            RB       ADV
called         VBN      VERB
cut            VBN      VERB
-                :         .
off             RB       ADV
in              IN       ADP
the             DT       DET
literature      NN      NOUN
.                .         .
If              IN       ADP
float           NN      NOUN
,                ,         .
the             DT       DET
parameter       NN      NOUN
represents     VBZ      VERB
a               DT       DET
proportion      NN      NOUN
of              IN       ADP
documents      NNS      NOUN
,                ,         .
integer         NN      NOUN
absolute        NN      NOUN
counts         NNS      NOUN
.                .         .
```

2 建立詞向量

In [7]:
```
import numpy as np
import pandas as pd
import nltk
from nltk.stem.porter import PorterStemmer
porter_stemmer = PorterStemmer()

from nltk.corpus import stopwords
stops = stopwords.words('english')
from string import punctuation
```

In [8]:
```
corpus = ["what time is it?", "how long has it been since we started?", "that's a long
df = pd.DataFrame(corpus, columns=['sentence'])
df
```

Out[8]:
```
                                 sentence
0                       what time is it?
1  how long has it been since we started?
2                 that's a long time ago
```

```
In [9]: # 請使用 nltk.word_tokenize 將每一行的詞彙切開
        #============your works starts===============#
        df['tokenize'] = df['sentence'].apply(nltk.word_tokenize)
        #=============your works ends================#


        df['tokenize']
        # 0                               [what, time, is, it, ?]
        # 1     [how, long, has, it, been, since, we, started, ?]
        # 2                        [that, 's, a, long, time, ago]
        # Name: tokenize, dtype: object

Out[9]: 0                               [what, time, is, it, ?]
        1     [how, long, has, it, been, since, we, started, ?]
        2                        [that, 's, a, long, time, ago]
        Name: tokenize, dtype: object
```

```
In [10]: # 請找出不重複的所有出現過的字
         #============your works starts===============#
         word_index = set(np.hstack([tokens for tokens in df['tokenize']]))
         #=============your works ends================#

         print("len(word_index)", len(word_index))
         print(word_index)
         # len(word_index) 16
         # {'is', 'that', 'time', 'long', 'we', 'ago', 'started', 'has', 'been', 'a', "'s", 'h

len(word_index) 16
{'long', 'what', "'s", 'that', 'time', 'ago', 'since', '?', 'a', 'started', 'has', 'how', 'we'
```

```
In [11]: for column in word_index:
             # 請幫每一個字創造一個欄位並指派為 0
             #============your works starts===============#
             df[column] = 0
             #=============your works ends================#


         print(df.columns)
         df
         # Index(['sentence', 'tokenize', 'is', 'that', 'time', 'long', 'we', 'ago',
         #        'started', 'has', 'been', 'a', ''s', 'how', 'what', 'it', 'since', '?'],
         #       dtype='object')

Index(['sentence', 'tokenize', 'long', 'what', ''s', 'that', 'time', 'ago',
       'since', '?', 'a', 'started', 'has', 'how', 'we', 'it', 'been', 'is'],
      dtype='object')
```

```
Out[11]:                                      sentence  \
         0                             what time is it?
         1   how long has it been since we started?
         2                       that's a long time ago

                                              tokenize  long  what  's  that  \
         0                          [what, time, is, it, ?]     0     0   0     0
         1  [how, long, has, it, been, since, we, started, ?]     0     0   0     0
         2                     [that, 's, a, long, time, ago]     0     0   0     0

            time  ago  since  ?  a  started  has  how  we  it  been  is
         0     0    0      0  0  0        0    0    0   0   0     0   0
         1     0    0      0  0  0        0    0    0   0   0     0   0
         2     0    0      0  0  0        0    0    0   0   0     0   0
```

```
In [12]: def build_word_index(row):
             tokens = row['tokenize']
             for token in tokens:
                 # 請幫計算每個字，在這個句子中出現的次數
                 #=============your works starts===============#
                 row[token] += 1
                 #==============your works ends================#
             return row

         df_processed = df.apply(build_word_index, axis=1)

         print(df_processed.iloc[0].tolist())
         # ['what time is it?', ['what', 'time', 'is', 'it', '?'], 1, 0, 1, 0, 0, 0, 0, 0, 0,
         df_processed
```

```
['what time is it?', ['what', 'time', 'is', 'it', '?'], 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
```

```
Out[12]:                                      sentence  \
         0                             what time is it?
         1   how long has it been since we started?
         2                       that's a long time ago

                                              tokenize  long  what  's  that  \
         0                          [what, time, is, it, ?]     0     1   0     0
         1  [how, long, has, it, been, since, we, started, ?]     1     0   0     0
         2                     [that, 's, a, long, time, ago]     1     0   1     1

            time  ago  since  ?  a  started  has  how  we  it  been  is
         0     1    0      0  1  0        0    0    0   0   1     0   1
         1     0    0      1  1  0        1    1    1   1   1     1   0
         2     1    1      0  0  1        0    0    0   0   0     0   0
```

3 中文 NLP 教學

```
In [13]: import jieba
         jieba.set_dictionary('dict.txt.big')  # 如果是使用繁體文字，請記得去下載繁體字典來使用
         with open('stops.txt', 'r', encoding='utf8') as f:  # 中文的停用字，我也忘記從哪裡拿到的
             stops = f.read().split('\n')
```

```
In [14]: # 請使用 jieba.cut 進行斷詞，並嘗試使用全斷詞模式 (cut_all=True)
         #=============your works starts===============#
         result_cut = [t for t in jieba.cut('下雨天留客天留我不留')]
         result_cutall = [t for t in jieba.cut('下雨天留客天留我不留', cut_all=True)]
         #==============your works ends================#


         print(result_cut)
         print(result_cutall)
         # ['下雨天', '留客', '天留', '我', '不留']
         # ['下雨', '下雨天', '雨天', '留客', '天', '留', '我', '不留']
```

```
Building prefix dict from D:\Projects\IIIMaterial\08_IrTheory\dict.txt.big ...
Loading model from cache C:\Users\user\AppData\Local\Temp\jieba.u113aa1b94b8c59a47d44fb9a8d37f
Loading model cost 1.589 seconds.
Prefix dict has been built succesfully.


['下雨天', '留客', '天留', '我', '不留']
['下雨', '下雨天', '雨天', '留客', '天', '留', '我', '不留']
```

```
In [15]: print("停用字")
         print(stops[:100])
```

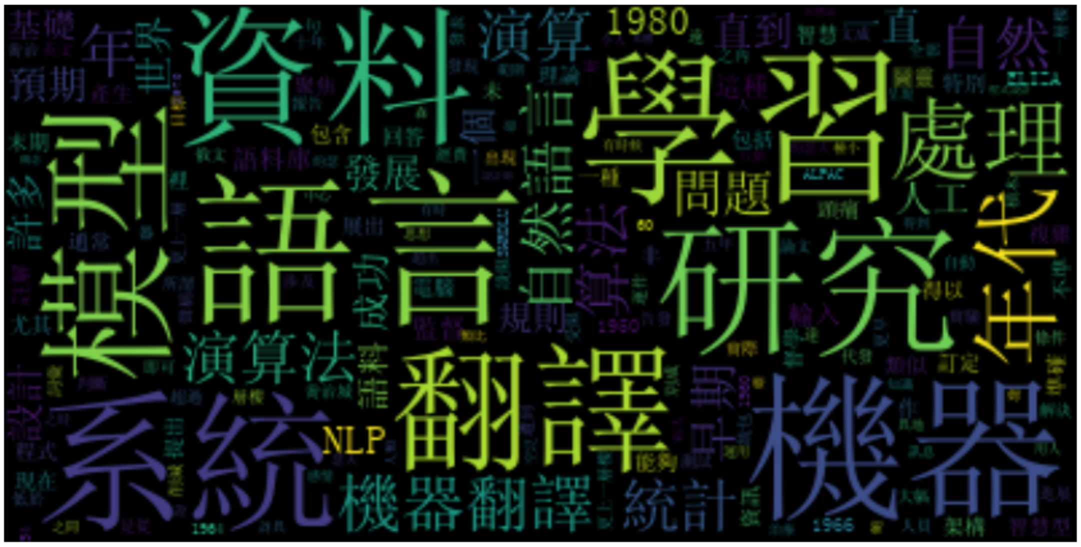
```
停用字
['\ufeff\ufeff,', '', '、', '。', ' “', '” ', '《', '》', '!', ',', ':', ';', '?', '人民',
```

```
In [16]: from collections import Counter
         from wordcloud import WordCloud
         from matplotlib import pyplot as plt

         testStr = """自然語言處理大體是從 1950 年代開始，雖然更早期也有作為。1950 年，圖靈發表論
         1954 年的喬治城實驗涉及全部自動翻譯超過 60 句俄文成為英文。研究人員聲稱三到五年之內即可解
         1960 年代發展特別成功的 NLP 系統包括 SHRDLU——一個詞彙設限、運作於受限如「積木世界」的一
         1970 年代，程式設計師開始設計「概念本體論」(conceptual ontologies)的程式，將現實世界的資
         一直到 1980 年代，多數自然語言處理系統是以一套複雜、人工訂定的規則為基礎。不過從 1980 年代
         許多早期的成功屬於機器翻譯領域，尤其歸功 IBM 的研究，漸次發展出更複雜的統計模型。這些系統
         近來的研究更加聚焦於非監督式學習和半監督學習的演算法。這種演算法，能夠從沒有人工註解理想答
         近年來, 深度學習技巧紛紛出爐 [2][3] 在自然語言處理方面獲得最尖端的成果，例如語言模型 [4]
```

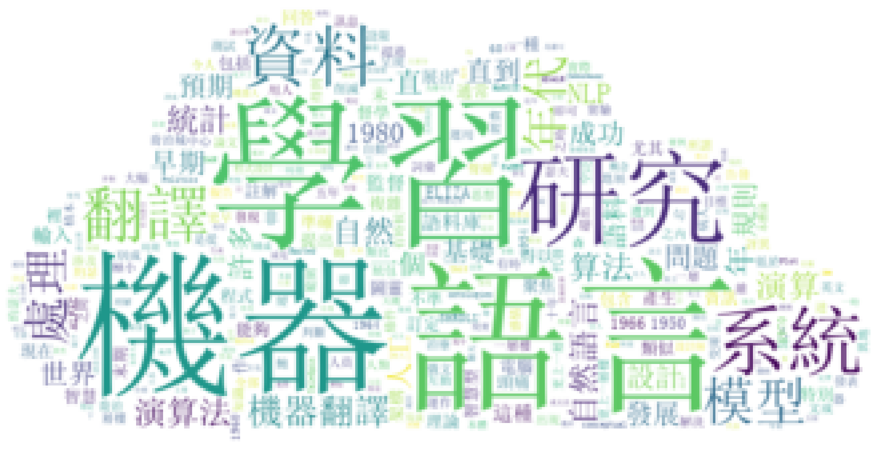
```
        # 請用全斷詞模式對上面文章段落進行分詞
        # 然後計算每一個詞彙出現過的次數，並將出現較多次的排到前面
        #=============your works starts===============#
        terms = [t for t in jieba.cut(testStr, cut_all=True) if t not in stops]
        terms_sorted =  sorted(Counter(terms).items(), key=lambda x:x[1], reverse=True)
        #==============your works ends================#

        print(terms_sorted)
```

```
[(' 語言', 10), (' 學習', 10), (' 機器', 9), (' 研究', 8), (' 系統', 8), (' 資料', 8), (' 翻譯',
```

```
In [17]: wordcloud = WordCloud(font_path="simsun.ttf") # 注意必須放數中文字體，否則會變亂碼
         wordcloud.generate_from_frequencies(frequencies=Counter(terms))
         plt.figure(figsize=(15,15))
         plt.imshow(wordcloud, interpolation="bilinear")
         plt.axis("off")
         plt.show()
```



```
In [18]: from PIL import Image

         alice_mask = np.array(Image.open("cloud_mask7.png"))
         wc = WordCloud(background_color="white", max_words=2000, mask=alice_mask, font_path="
         wc.generate_from_frequencies(Counter(terms))

         # show
         plt.imshow(wc, interpolation='bilinear')
         plt.axis("off")
         plt.figure()
         plt.imshow(alice_mask, cmap=plt.cm.gray, interpolation='bilinear')
         plt.axis("off")
         plt.show()
```



4 中文檢索系統

1. TFIDF

$$TFIDF_{td} = TF_{td} \times log(\frac{N}{DF_t})$$

- 所謂 TFIDF 應分成兩個部分來理解：TF(Term Frequency) 以及 IDF(Inverted Document Frequency)。
- TF(Term Frequency): TF_{td} 指得是在特定的文章 d 中特定的字 t 出現了幾次。這個部分同時，也表示了一個文字在一篇文章的重要性，依但出現越多次，這個字也就越能代表這篇文章。
- IDF(Inverted Document Frequency): N 指得是總共有機篇文章，DF_t 中的 DF 是 Document Frequency 的意思，DFt 則是詞彙 t 在幾篇文章中出現過。$\frac{DF_t}{N}$ 也就是所有文章當中，詞彙 t 在幾篇文章出現過，而其倒數則是 Inverted Documnet Index，表著這個詞彙如果在很多文章裏面都出現過，則其重要性會受到懲罰，而取 log 則只是讓他在分數的影響上比較平滑而已。

2. Cosine Similarity

$$\cos\theta = \frac{A \cdot B}{\|A\|_2 \|B\|_2}$$

- if $A = [1, 2, 0, 4]$ and $B = [3, 2, 1, 0]$
- $\cos\theta = \frac{1 \cdot 3 + 2 \cdot 2 + 0 \cdot 1 + 4 \cdot 0}{\sqrt{1^2+2^2+0^2+4^2} \cdot \sqrt{3^2+2^2+1^2+0^2}}$

In [19]:
```
import jieba
jieba.set_dictionary('dict.txt.big')   # 如果是使用繁體文字，請記得去下載繁體字典來使用
import numpy as np
import pandas as pd
```

In [20]:
```
# 把檔案讀出來 (原始資料: https://society.hccg.gov.tw/ch/home.jsp?id=43&parentpath=0,5
df_QA = pd.read_json('ProcessedData.json', encoding='utf8')
# 我們這次只會使用到 question 跟 ans 這兩個欄位
df_question = df_QA[['question', 'ans']].copy()   ## 不要更動到原始的 DataFrame
df_question.drop_duplicates(inplace=True)   ## 丟掉重複的資料
df_question.head(5)   ## show 出來
```

Out[20]:
```
                          question  \
0             小孩出生後應於何時申請育兒津貼?
1             小孩出生後應於何時申請育兒津貼?
2                育兒津貼申請應備文件為何?
3  若民眾夫妻雙方均失業，是否可申請家庭育兒津貼費用補助
4                  育兒津貼補助對象為何？


                                                 ans
0  1. 幼兒家長在戶政事務所完成新生兒出生登記後，即可向所轄區公所社政課提出育兒津貼申請。2
1  隨時提出; 津貼經審查通過後，追溯自受理申請之當月起發給。兒童出生後六十日內向戶政事務所
2  申請資料應備齊:(一) 兒童之戶口名簿影本。(二) 申請人之郵局存摺封面影本。(三) 父母雙方身
3  一、育兒津貼補助對象：1. 育有二足歲以下兒童。2. 兒童之父母至少一方因育兒需要，致未能就
4  育兒津貼補助對象，應符合下列規定：(一) 育有二足歲以下兒童。(二) 兒童之父母 (或監護人)
```

```
In [21]: # 前處理
         all_terms = []
         def preprocess(item):  ## 定義前處理的 function
             # 請把將每一行用 jieba.cut 進行分詞 (記得將 cut_all 設定為 True)
             # 同時建立所有詞彙的 list(all_terms)
             #=============your works starts===============#
             terms = [t for t in jieba.cut(item, cut_all=True)]  ## 把全切分模式打開，可以比對的
             all_terms.extend(terms)  ## 收集所有出現過的字
             #==============your works ends================#
             return terms

         df_question['processed'] = df_question['question'].apply(preprocess)
         print(df_question.iloc[0])
         # question                                   小孩出生後應於何時申請育兒津貼?
         # ans              1. 幼兒家長在戶政事務所完成新生兒出生登記後，即可向所轄區公所社政課提出育兒
         # processed                       [小孩, 出生, 後, 應於, 何時, 申請, 育兒, 津貼, , ]
         # Name: 0, dtype: object

         df_question.head()
```

```
Building prefix dict from D:\Projects\IIIMaterial\08_IrTheory\dict.txt.big ...
Loading model from cache C:\Users\user\AppData\Local\Temp\jieba.u113aa1b94b8c59a47d44fb9a8d37f
Loading model cost 1.931 seconds.
Prefix dict has been built succesfully.


question                                    小孩出生後應於何時申請育兒津貼?
ans             1. 幼兒家長在戶政事務所完成新生兒出生登記後，即可向所轄區公所社政課提出育兒津貼申請。
processed                      [小孩, 出生, 後, 應於, 何時, 申請, 育兒, 津貼, , ]
Name: 0, dtype: object
```

```
Out[21]:                         question  \
         0            小孩出生後應於何時申請育兒津貼?
         1            小孩出生後應於何時申請育兒津貼?
         2               育兒津貼申請應備文件為何?
         3   若民眾夫妻雙方均失業，是否可申請家庭育兒津貼費用補助
         4                 育兒津貼補助對象為何？

                                                      ans  \
         0  1. 幼兒家長在戶政事務所完成新生兒出生登記後，即可向所轄區公所社政課提出育兒津貼申請。2
         1  隨時提出；津貼經審查通過後，追溯自受理申請之當月起發給。兒童出生後六十日內向戶政事務所
         2  申請資料應備齊:(一) 兒童之戶口名簿影本。(二) 申請人之郵局存摺封面影本。(三) 父母雙方身
         3  一、育兒津貼補助對象：1. 育有二足歲以下兒童。2. 兒童之父母至少一方因育兒需要，致未能就
         4  育兒津貼補助對象，應符合下列規定：(一) 育有二足歲以下兒童。(二) 兒童之父母 (或監護人)

                                            processed
         0           [小孩, 出生, 後, 應於, 何時, 申請, 育兒, 津貼, , ]
```

```
1                  [小孩, 出生, 後, 應於, 何時, 申請, 育兒, 津貼, , ]
2                     [育兒, 津貼, 申請, 應, 備, 文件, 為, 何, , ]
3  [若, 民, 眾, 夫妻, 雙方, 均, 失業, , , 是否, 可, 申請, 家庭, 育兒...
4                       [育兒, 津貼, 貼補, 補助, 對象, 為, 何, , ]
```

```
In [22]: # 建立 termindex: 將 all_terms 取出不重複的詞彙,並轉換型別為 list(避免順序亂掉)
         #=============your works starts===============#
         termindex = list(set(all_terms))
         #==============your works ends================#

         print("len(termindex)", len(termindex))
         print(termindex[:10])
         # len(termindex) 1012
         # ['', ' 耗材', ' 被', ' 其他', ' 發', ' 發現', ' 申請人', ' 遭遇', ' 環境', ' 您好']
```

```
len(termindex) 1012
['', ' 證', ' 災害', ' 重要', ' 之人', ' 偶或', ' 到', ' 民', ' 耗材', ' 幫忙']
```

```
In [23]: # 建立 IDF vector
         Doc_Length = len(df_question)  ## 計算出共有幾篇文章
         Idf_vector = []  ## 初始化 IDF 向量
         for term in termindex:  ## 對 index 中的詞彙跑回圈
             num_of_doc_contains_term = 0  ## 計算有機篇文章出現過這個詞彙
             for terms in df_question['processed']:
                 if term in terms:
                     num_of_doc_contains_term += 1
             idf = np.log(Doc_Length/num_of_doc_contains_term)  ## 計算該詞彙的 IDF 值
             Idf_vector.append(idf)
         print(len(Idf_vector))
         print(Idf_vector[:10])
```

```
1012
[0.04710446918747347, 5.093750200806762, 5.786897381366708, 5.786897381366708, 5.78689738136670
```

```
In [24]: # 建立 document vector
         def terms_to_vector(terms):  ## 定義把 terms 轉換成向量的 function
             ## 建立一條與 termsindex 等長、但值全部為零的向量 (hint:dtype=np.float32)
             #=============your works starts===============#
             vector = np.zeros_like(termindex, dtype=np.float32)
             #==============your works ends================#

             for term, count in Counter(terms).items():
                 # 計算 vector 上每一個字的 tf 值
                 #=============your works starts===============#
                 vector[termindex.index(term)] = count
                 #==============your works ends================#
```

```
    # 計算 tfidf，element-wise 的將 vector 與 Idf_vector 相乘
    ## hint: 如果兩個 vector 的型別都是 np.array，把兩條 vector 相乘，就會自動把向量中的
    #=============your works starts===============#
    vector = vector * Idf_vector
    #==============your works ends================#
    return vector



df_question['vector'] = df_question['processed'].apply(terms_to_vector)  ## 將上面定義
df_question['vector'][:10]
# 0    [0.09420893837494694, 0.0, 0.0, 0.0, 0.0, 0.0,...
# 1    [0.09420893837494694, 0.0, 0.0, 0.0, 0.0, 0.0,...
# 2    [0.09420893837494694, 0.0, 0.0, 0.0, 0.0, 0.0,...
# 3    [0.09420893837494694, 0.0, 0.0, 0.0, 0.0, 0.0,...
# 4    [0.09420893837494694, 0.0, 0.0, 0.0, 0.0, 0.0,...
# 5    [0.09420893837494694, 0.0, 0.0, 0.0, 0.0, 0.0,...
# 6    [0.2826268151248408, 0.0, 0.0, 0.0, 0.0, 0.0,...
# 7    [0.2826268151248408, 0.0, 0.0, 0.0, 0.0, 0.0,...
# 8    [0.2826268151248408, 0.0, 0.0, 0.0, 0.0, 0.0,...
# 9    [0.2826268151248408, 0.0, 0.0, 0.0, 0.0, 0.0,...
# Name: vector, dtype: object
```

```
Out[24]: 0    [0.09420893837494694, 0.0, 0.0, 0.0, 0.0, 0.0,...
         1    [0.09420893837494694, 0.0, 0.0, 0.0, 0.0, 0.0,...
         2    [0.09420893837494694, 0.0, 0.0, 0.0, 0.0, 0.0,...
         3    [0.09420893837494694, 0.0, 0.0, 0.0, 0.0, 0.0,...
         4    [0.09420893837494694, 0.0, 0.0, 0.0, 0.0, 0.0,...
         5    [0.09420893837494694, 0.0, 0.0, 0.0, 0.0, 0.0,...
         6    [0.2826268151248408, 0.0, 0.0, 0.0, 0.0, 0.0,...
         7    [0.2826268151248408, 0.0, 0.0, 0.0, 0.0, 0.0,...
         8    [0.2826268151248408, 0.0, 0.0, 0.0, 0.0, 0.0,...
         9    [0.2826268151248408, 0.0, 0.0, 0.0, 0.0, 0.0,...
         Name: vector, dtype: object
```

```
In [25]: from numpy.linalg import norm

         def cosine_similarity(vector1, vector2):  ## 定義 cosine 相似度的計算公式
             # 使用 np.dot 與 norm 計算 cosine score
             #=============your works starts===============#
             score = np.dot(vector1, vector2)  / (norm(vector1) * norm(vector2))
             #==============your works ends================#
             return score

         sentence1 = df_question.loc[0]  ## 取出第零個的問題
         sentence2 = df_question.loc[2]  ## 取出第二個的問題
         print(sentence1['question'])
         print(sentence2['question'])
```

```
        print(cosine_similarity(sentence1['vector'], sentence2['vector']))  ## 計算兩者的相似度
        # 0.2032278479377731
```

```
小孩出生後應於何時申請育兒津貼?
育兒津貼申請應備文件為何?
0.20322784793773094
```

```
In [26]: def retrieve(testing_sentence, return_num=3):  ## 定義出檢索引擎
             # 請使用前面定義的 terms_to_vector 與 preprocess 兩個 function，計算出 testing_sent
             # 計算其與資料庫每一的問句的相似度
             # 依分數進行排序，找到分數最高的三個句子
             #=============your works starts===============#
             testing_vector = terms_to_vector(preprocess(testing_sentence))  ## 把剛剛的前處理、
             idx_score_mapping = [(idx, cosine_similarity(testing_vector, vec)) for idx, vec in
             top3_idxs = np.array(sorted(idx_score_mapping, key=lambda x:x[1], reverse=True))[
             #==============your works ends================#

             return df_question.loc[top3_idxs, ['question', 'ans']]


         print(retrieve("老人年金").index)
         # Float64Index([100.0, 111.0, 321.0], dtype='float64')
```

```
Float64Index([100.0, 111.0, 321.0], dtype='float64')
```

5 Use Scikit learn

```
In [27]: from sklearn.feature_extraction.text import TfidfVectorizer

In [28]: tfidf = TfidfVectorizer()
         # 使用 tfidf.fit_transform 將轉換 df_question['processed'] 為 vector
         #=============your works starts===============#
         df_question['sklearn_vector'] = list(tfidf.fit_transform(df_question['processed'].appl
         #==============your works ends================#

         print(df_question.loc[:10, 'sklearn_vector'].apply(sum).values)
         # [2.54619627 2.54619627 1.95695906 3.12409736 2.19106254 2.74144953
         #  3.82923767 2.54569516 3.4163518  2.98088982 2.35528293]
```

```
[2.54619627 2.54619627 1.95695906 3.12409736 2.19106254 2.74144953
 3.82923767 2.54569516 3.4163518  2.98088982 2.35528293]
```

```
In [29]: def sklearn_retrieve(testing_sentence, return_num=3):  ## 定義出檢索引擎
             # 請使用前面定義的 tfidf.transform 與 preprocess 兩個 function，計算出 testing_sent
             # 注意 tfidf.transform 必須是兩個維度的 array
```

```
        # 且 out 為 sparse metric，必需.toarray() 轉換為一般 np.array()
        # 計算其與資料庫每一的問句的相似度
        # 依分數進行排序，找到分數最高的三個句子
        #=============your works starts===============#
        testing_vector = tfidf.transform([" ".join(preprocess(testing_sentence))]).toarray
        idx_score_mapping = [(idx, cosine_similarity(testing_vector, vec)) for idx, vec in
        top3_idxs = np.array(sorted(idx_score_mapping, key=lambda x:x[1], reverse=True))[
        #==============your works ends================#
        return df_question.loc[top3_idxs, ['question', 'ans']]

    print(retrieve("老人年金")['question'])
    print(sklearn_retrieve("老人年金")['question'])
    # 100.0     我已經年滿 65 歲領有國民年金老人年金及基本保證年金 3628 元，因家境清寒還可以再
    # 111.0                             新竹市老人一般可領老人津貼 6628 元，該如何申請？
    # 321.0              國民年金保險被保險人如果是家庭收入較低者，國民年金保險費是否可以減免？補
    # Name: question, dtype: object
    # 100.0     我已經年滿 65 歲領有國民年金老人年金及基本保證年金 3628 元，因家境清寒還可以再
    # 111.0                             新竹市老人一般可領老人津貼 6628 元，該如何申請？
    # 321.0              國民年金保險被保險人如果是家庭收入較低者，國民年金保險費是否可以減免？補
    # Name: question, dtype: object
```

```
100.0     我已經年滿 65 歲領有國民年金老人年金及基本保證年金 3628 元，因家境清寒還可以再申請中低收入
111.0                             新竹市老人一般可領老人津貼 6628 元，該如何申請？
321.0              國民年金保險被保險人如果是家庭收入較低者，國民年金保險費是否可以減免？補助標準為何
Name: question, dtype: object
100.0     我已經年滿 65 歲領有國民年金老人年金及基本保證年金 3628 元，因家境清寒還可以再申請中低收入
111.0                             新竹市老人一般可領老人津貼 6628 元，該如何申請？
321.0              國民年金保險被保險人如果是家庭收入較低者，國民年金保險費是否可以減免？補助標準為何
Name: question, dtype: object
```

```
In [30]: print(retrieve("托育")['question'])
         print(sklearn_retrieve("托育")['question'])
```

```
15.0                    托育費用補助對象為何？
14.0                    托育費用補助標準為何？
19.0     要申請托育補助一定要找加入居家托育服務中心的托育人員嗎？
Name: question, dtype: object
0.0     小孩出生後應於何時申請育兒津貼?
1.0     小孩出生後應於何時申請育兒津貼?
2.0         育兒津貼申請應備文件為何?
Name: question, dtype: object
```

```
In [31]: print(retrieve("補助")['question'])
         print(sklearn_retrieve("補助")['question'])
```

```
214.0            中低收入醫療補助補助項目及標準為何？
108.0                 申請假牙補助的資格及補助內容
82.0      特殊境遇家庭法律訴訟補助如何申請？補助額度如何？
Name: question, dtype: object
108.0         申請假牙補助的資格及補助內容
15.0              托育費用補助對象為何？
214.0     中低收入醫療補助補助項目及標準為何？
Name: question, dtype: object
```

```
In [32]: print(retrieve("救助")['question'])
         print(sklearn_retrieve("救助")['question'])
```

```
204.0           社會福利-急難救助核發救助對象？
74.0      遭遇特殊境遇家庭如何申請救助 (申請方式)？
203.0         社會福利-我要到那裡申請急難救助？
Name: question, dtype: object
204.0           社會福利-急難救助核發救助對象？
203.0         社會福利-我要到那裡申請急難救助？
74.0      遭遇特殊境遇家庭如何申請救助 (申請方式)？
Name: question, dtype: object
```

```
In [ ]:
```