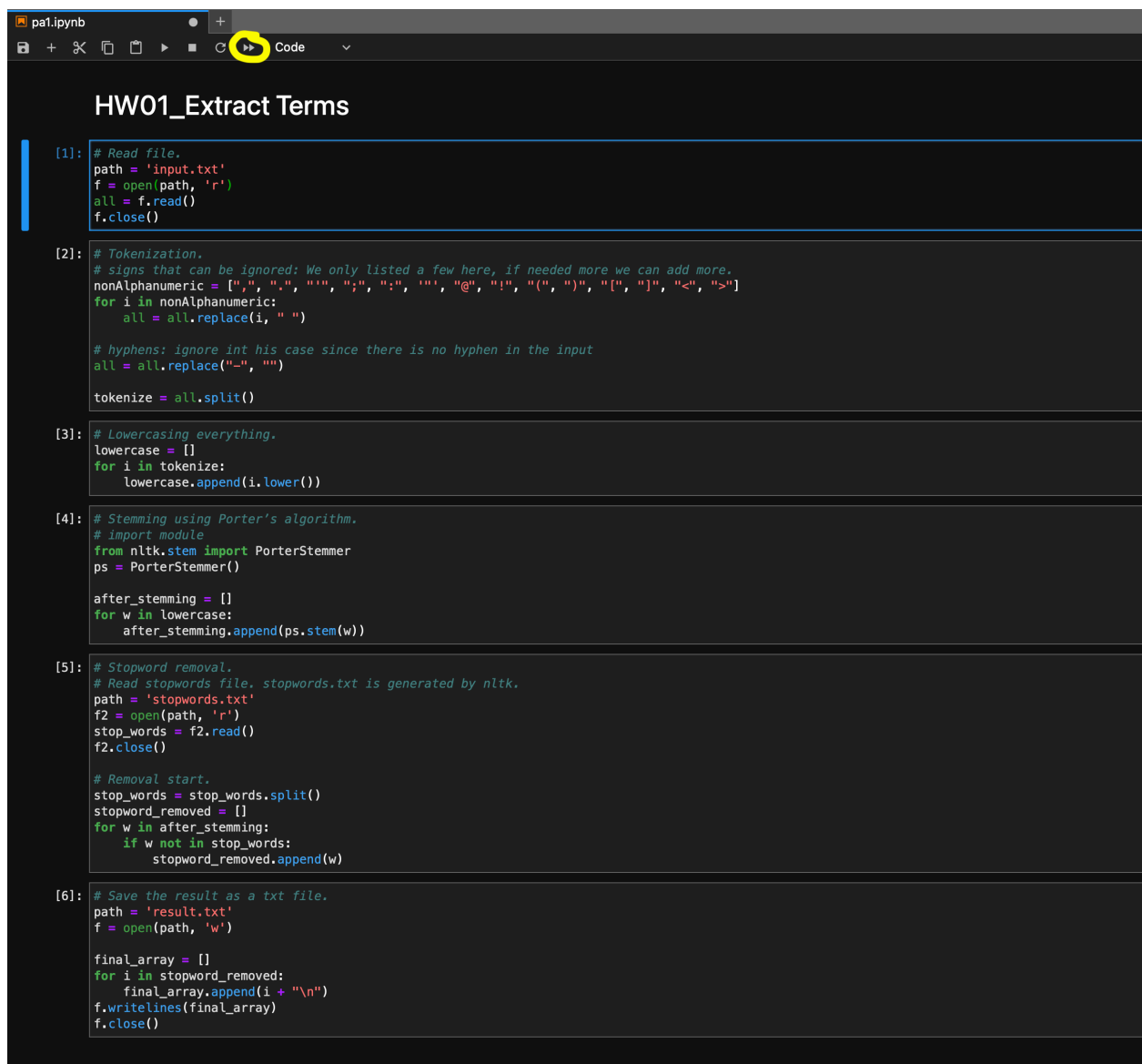


1. 執行環境：Jupyter Lab
2. 程式語言：Python 3.10.5
3. 執行方式：

使用 pa4.ipynb 檔

- (1) 使用 Jupyter Lab 或 notebook 開啟 pa4.ipynb 檔
- (2) pip3 install nltk 用於 Porter's Algorithm
- (3) 將整個檔案全部執行即可



```
[1]: # Read file.
path = 'input.txt'
f = open(path, 'r')
all = f.read()
f.close()

[2]: # Tokenization.
# signs that can be ignored: We only listed a few here, if needed more we can add more.
nonAlphanumeric = [" ", ".", "!", ":", ";", ",", "'", "@", "(", ")", "[", "]", "<", ">"]
for i in nonAlphanumeric:
    all = all.replace(i, " ")

# hyphens: ignore int his case since there is no hyphen in the input
all = all.replace("-", " ")

tokenize = all.split()

[3]: # Lowercasing everything.
lowercase = []
for i in tokenize:
    lowercase.append(i.lower())

[4]: # Stemming using Porter's algorithm.
# import module
from nltk.stem import PorterStemmer
ps = PorterStemmer()

after_stemming = []
for w in lowercase:
    after_stemming.append(ps.stem(w))

[5]: # Stopword removal.
# Read stopwords file. stopwords.txt is generated by nltk.
path = 'stopwords.txt'
f2 = open(path, 'r')
stop_words = f2.read()
f2.close()

# Removal start.
stop_words = stop_words.split()
stopword_removed = []
for w in after_stemming:
    if w not in stop_words:
        stopword_removed.append(w)

[6]: # Save the result as a txt file.
path = 'result.txt'
f = open(path, 'w')

final_array = []
for i in stopword_removed:
    final_array.append(i + "\n")
f.writelines(final_array)
f.close()
```

4. 作業處理邏輯說明：

A. Read all files in data.

B. Tokenize all doc:

(1) Read file: Use `open()` read only to read file and `close()` to finish.

(2) Tokenization: Craft a list with signs we want to split as `nonAlphanumeric`. Then, replace the signs with blank and eliminate the hyphens and dots among words. Finally, use `".split()"` to split string in to a list of words whenever the next element is a `"\n"` or a blank.

(3) Remove Digits: Remove all digits in the document.

(4) Lowercasing everything: Use `".lower()"` to lower case everything.

(5) Stop word removal: Read file `"stopwords.txt"`, which is generated by `"nltk"` (approved by the professor). Whenever the word before stemming is not a stop word, we keep it to the next section.

(6) Stemming using Porter's algorithm: Use the api `"from nltk.stem import PorterStemmer"` allowed by the professor for stemming words.

(7) Save the results in `all_doc` array.

C. Count tf, df:

(1) Count the term frequency of every word in every document and save it as python dictionary, then put them into `"tf_dict_arr"`.

(2) Count df of every term and save it as a large dictionary `"df_dict"`.

D. Count tf-idf unit vector:

(1) Calculate al the idf number in `"df_dict"` through the formula given and sort in alphabetical order.

(2) Calculate each document's tf-idf unit vector. First, we multiply tf from `"tf_dict_arr"` and idf from `"df_dict"` and save the sum of every value's square so that we can get the norm of the document after traveling all the elements. Then, we travel the elements again and divide them by the norm of the document so that every document is a unit vector now. Finally, we save the results in `"tf_idf_norm"` which is also an array of python dictionary.

E. Count cosine similarity matrix: Calculate the cosine similarity matrix of all documents by the `"tf_idf_norm"` in the previous step.

F. HAC Algorithm:

(1) Find the largest similarity among the similarity matrix and combine the two clusters by adding the pair to A.

(2) Update the similarity of the new cluster by changing the similarity to the minimum similarity of number in the pair (This is complete-link clustering method).

(3) Then we can obtain A and the new similarity matrix by the steps above.

G. Change A into target clusters:

(1) Initialize a cluster set with 1095 clusters.

(2) Merge the two clusters by the pair indicated in A in the same order as the first element to the last element.

(3) If the cluster equals target cluster number K, stop merging.

H. Output the results of clusters to file by sorting it in ascending order.

Results: 8.txt, 13.txt, 20.txt