```python
from gensim.models.coherencemodel import CoherenceModel
import pandas as pd
import re
from gensim import corpora, models
from gensim.parsing.preprocessing import preprocess_string
import warnings
import matplotlib.pyplot as plt
import matplotlib.style as style
style.use('fivethirtyeight')
warnings.filterwarnings('ignore')

pd.set_option('display.max_columns', None)
# pd.set_option('display.width', None)
# pd.set_option('display.max_colwidth', None)
pd.set_option('display.unicode.ambiguous_as_wide', True)
pd.set_option('display.unicode.east_asian_width', True)
# 引入上次已經斷詞的小王子資料且依照章節進行區分document
# 更新部分：去除長度為一的字母和某些表示數量的詞彙（中文）以及某些可能較為沒意義的詞彙
# list = ('一', '二', '三', '四', '五', '六', '七', '八', '九','十', '兩', '幾','小王子'
#  '是', '只', '那', '在', '再', '說', '這','第')開頭的字(除了章節名稱之後會用到)
# 由於程式碼過於冗長就不附上

# first secion

# data = open('output.txt', 'r', encoding='utf8')
# document = ['']*27
# tempword = []
# countchapters = ["第一章 \n", "第二章 \n",
#                  "第三章 \n", "第四章 \n", "第五章 \n", "第六章 \n",
#                  "第七章 \n", "第八章 \n", "第九章 \n", "第十章 \n",
#                  "第十一章 \n", "第十二章 \n", "第十三章 \n", "第十四章 \n",
#                  "第十五章 \n", "第十六章 \n", "第十七章 \n", "第十八章 \n",
#                  "第十九章 \n", "第二十章 \n", "第二十一章 \n", "第二十二章 \n", "第二十三章 \n", "第二十四章 \n",
#                  "第二十五章 \n", "第二十六章 \n", "第二十七章 \n", "作者 \n"]

# for line in data.readlines():
#     tempword.append(line)  # 每行加入tempwords
# print(len(tempword))

# chapter = 0
# for i in range(len(tempword)):
#     if chapter < 27:
#         if tempword[i] == countchapters[chapter]:
#             temp = i+2
#             while True:
#                 document[chapter] += tempword[temp]
#                 temp += 1
#                 if tempword[temp] == countchapters[chapter+1]:
#                     f = open("allchapter/chapter"+str(chapter+1)+".txt", "w+")
#                     document[chapter] = re.sub("\n", " ", document[chapter])
#                     str_list = document[chapter].split()
#                     new_str = ' '.join(str_list)
#                     f.write(new_str)
#                     i = temp
#                     break
```

```
#             chapter += 1

# second section

# df = pd.DataFrame(columns=['document', 'text'])
# for i in range(27):
#     temprow = pd.read_table("chapter"+str(i+1)+".txt", header=None)
#     df = df.append(pd.DataFrame(
#         {'document': i, 'text': [temprow.iloc[0][0]]}),
ignore_index=True)

# df.to_csv("outputcsv.csv", index=False)
# print(df.text)

# third section

df = pd.read_csv('outputcsv.csv')

train_data = []
for i in range(27):
    train_data.append((df.iloc[i, 1]).split())


def create_lsa_model(documents, dictionary, number_of_topics):
    print(f'Creating LDA Model with {number_of_topics} topics')
    document_terms = [dictionary.doc2bow(doc) for doc in documents]
    return models.LsiModel(document_terms,
                           num_topics=number_of_topics,
                           id2word=dictionary,)


def run_lsa_process(documents, number_of_topics=10):
    dictionary = corpora.Dictionary(documents)
    lsa_model = create_lsa_model(documents, dictionary,
                                 number_of_topics)
    return documents, dictionary, lsa_model


def calculate_coherence_score(documents, dictionary, model):
    coherence_model = CoherenceModel(model=model,
                                     texts=documents,
                                     dictionary=dictionary,
                                     coherence='c_v')
    return coherence_model.get_coherence()


def get_coherence_values(start, stop):
    for num_topics in range(start, stop):
        print(f'\nCalculating coherence for {num_topics} topics')
        documents, dictionary, model = run_lsa_process(train_data,

number_of_topics=num_topics)
        coherence = calculate_coherence_score(documents,
                                              dictionary,
                                              model)
        yield coherence


if __name__ == '__main__':

```

```
114     min_topics, max_topics = 1, 28
115     coherence_scores = list(get_coherence_values(min_topics, max_topics))
116     print("The highests coherence score: "+str(max(coherence_scores))+" " +
117           "The best number of topics:
"+str(coherence_scores.index(max(coherence_scores))+1))
118     x = [int(i) for i in range(min_topics, max_topics)]
119 
120     documents, dictionary, model = run_lsa_process(
121         train_data, coherence_scores.index(max(coherence_scores))+1)
122 
123     topics = pd.DataFrame(columns=['index', '主題內容'])
124 
125     finaltopics = model.print_topics(
126         num_topics=coherence_scores.index(max(coherence_scores))+1,
num_words=10)
127     for topic in finaltopics:
128         topics = topics.append(pd.DataFrame(
129             {'index': [topic[0]+1], '主題內容': [topic[1]]}))
130     topics.to_csv("topics.csv", index=False)
131     plt.figure(figsize=(10, 8))
132     plt.ylim([0, 1])
133     plt.xticks(x, x)
134     plt.plot(x, coherence_scores, marker='o')
135     plt.xlabel('Number of topics')
136     plt.ylabel('Coherence Value')
137     plt.title('Coherence Scores by number of Topics')
138 
 plt.annotate("Best"+"\n"+str(max(coherence_scores))+"\n"+str(coherence_sco
res.index(max(coherence_scores))+1),
139                  xy=(coherence_scores.index(max(coherence_scores))+1,
max(coherence_scores)), xycoords='data')
140     plt.savefig("outputplt.jpg")
141 
```