МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Практикум №8

з курсу «Аналіз даних в інформаційнних системах» на тему:

«Аналіз текстів»

Викладач:
Олійник Ю.О.

Виконав:
студент 2 курсу групи
ІП-15 ФІОТ
Мєшков Андрій
Ігорович

Київ-2023

Практикум №8

Аналіз текстів

Мета роботи: ознайомитись з методами аналізу текстів.

Завдання:

Основне завдання

Дані для виконання: текстові дані у форматі csv-файлів або дані з відкритих джерел (телеграм-канали, RSS-канали тощо). Приклад даних за посиланням

1. Нормалізація та попередня обробка даних.
2. провести очищення текстових даних від стоп-слів/тегів/розмітки;
3. виконати токенізацію текстових елементів;
4. провести лематизацію текстових елементів (можна використати бібліотеку Spacy - приклад роботи за посиланням). Зберегти результат в окремий файл.
5. Створити Bag of Words для всіх нормалізованих слів. Зберегти результат в окремий файл.
6. Порахувати метрику TF-IDF для 10 слів, що найчастіше зустрічаються в корпусі;

Додаткове завдання

Обробка даних оповідань А.К. Дойля та Е.По (+1 бал):

- Завантажити потрібні дані.
- Завантажити оповідання А.К. Дойля та Е.По з папки Texts/Task.
- Виконати попередню обробку текстів.
- Побудувати дві хмари слів, що використовують А.К. Дойль та Е.По.
- Який з письменників написав більш похмурі оповідання?

Хід роботи:

Основне завдання:

Імпортуємо потрібні бібліотеки.

```
import nltk
nltk.download('stopwords')
nltk.download('punkt')
from bs4 import BeautifulSoup
import re
from nltk.tokenize import word_tokenize
import spacy
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from typing import List
```

Зчитуємо HTML-файл та витягуємо текст з тегів <p>, <h2> та <h1>

```
with open('kotsiubynskyy-mykhaylo-mykhaylovych-tini-zabutykh-predkiv1058.html', 'r',
encoding='utf-8') as f:
    soup = BeautifulSoup(f, 'html.parser')
    paragraphs = soup.find_all(['p', 'h1', 'h2'])
    text = '\n'.join([p.get_text() for p in paragraphs])
```

Створемо функції для запису файлу та створення списку сліз з їх частотами

```
def write(tokens: List[str], filename: str) -> None:
    with open(filename, 'w', encoding='utf-8') as f:
        f.write('\n'.join(tokens))
def word_list_to_freq_dict(wordlist):
    res = {}
    for word in wordlist:
        if word not in res:
            res[word] = 1
        else:
            res[word] += 1
    return res
```

Очищаємо текст від знаків пунктуації та цифр

```
text = re.sub(r"[^\w\s]", '', text)
text = re.sub(r"\d+", '', text)
```

Токенізація тексту

```
text = word_tokenize(text)
```

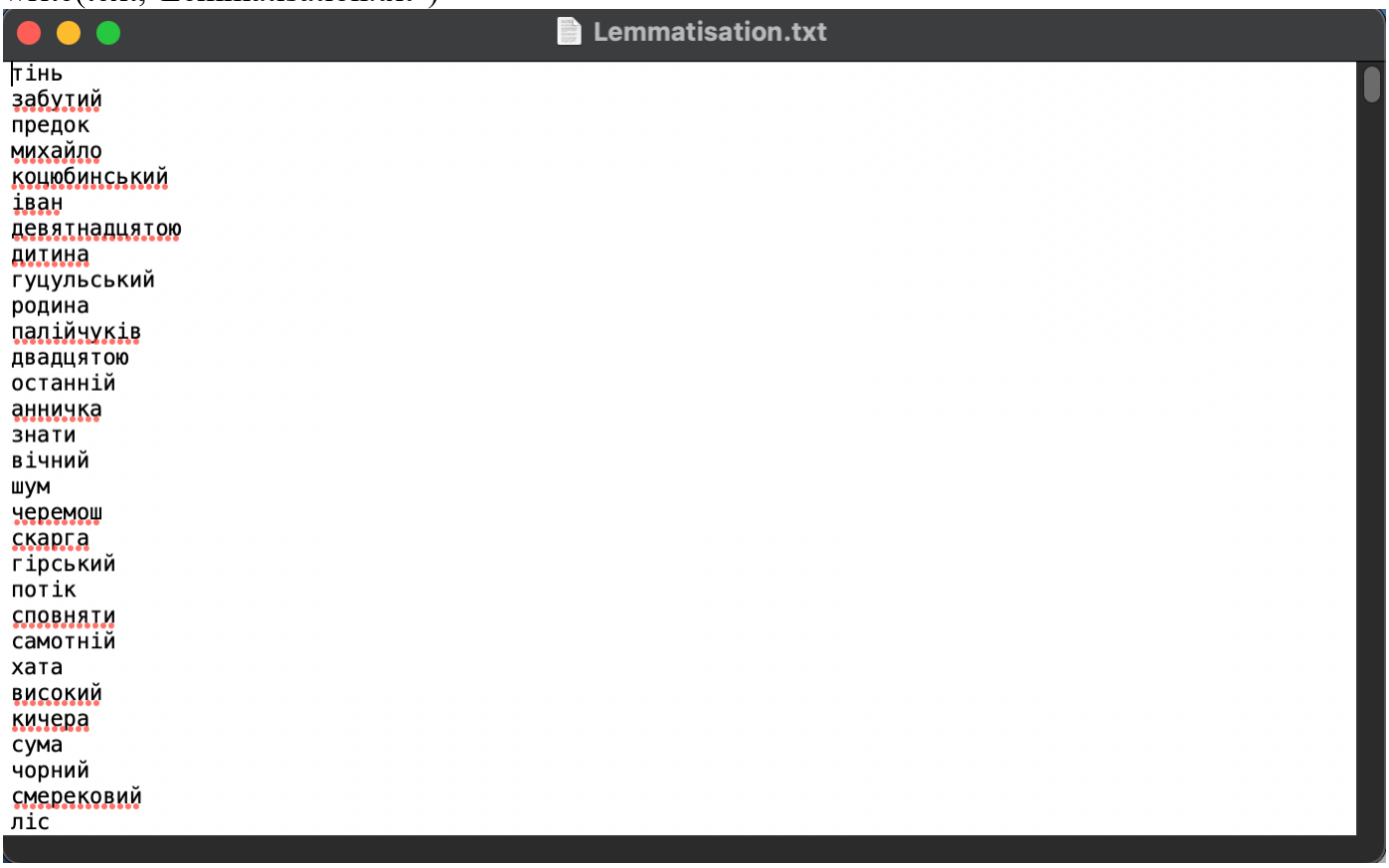
Завантажуємо список стоп-слів

```
nlp = spacy.load('uk_core_news_sm')
stop_words = nlp.Defaults.stop_words
```

Лематизація текстових елементів

```
lemmatized_words = [doc.lemma_ for doc in nlp(' '.join(text))]
```

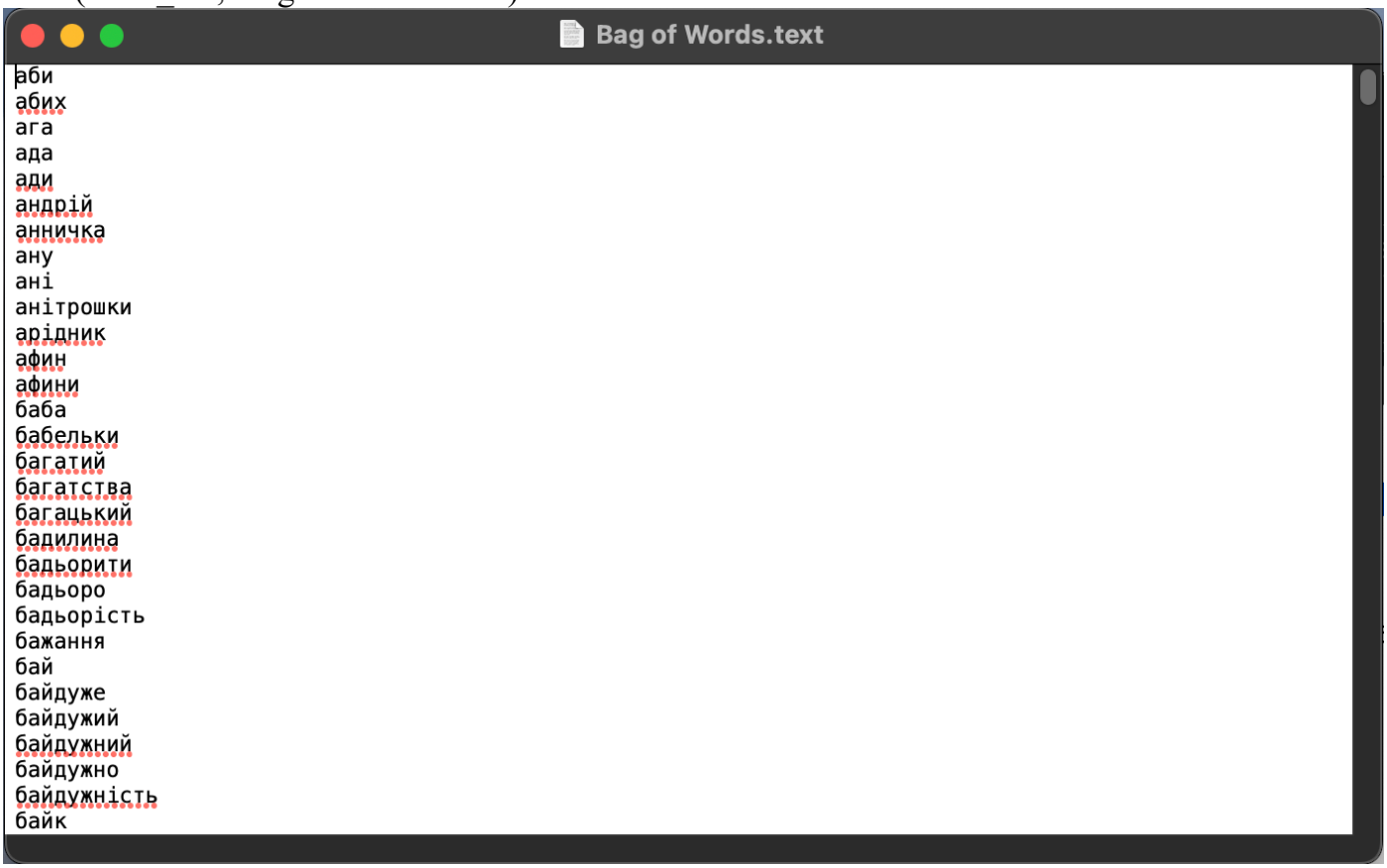
Виконуємо очищення тексту від стоп-слів
text = [word *for* word *in* lemmatized_words *if* word not in stop_words and re.match(r'\w+', word)]
write(text,"Lemmatisation.txt")



Ініціалізуємо об'єкт CountVectorizer
vectorizer = CountVectorizer()

Виконуємо векторизацію тексту
bag_of_words = vectorizer.fit_transform(text)

Отримуємо список всіх слів у "Bag of Words"
word_list = vectorizer.get_feature_names_out()
write(word_list,"Bag of Words.text")



Ініціалізуємо об'єкт TfidfVectorizer
top_words = word_list_to_freq_dict(text)
top_words_list = sorted(top_words.items(), key=lambda x: x[1], reverse=True)[:10]
top_words_only = [word *for* word, _ *in* top_words_list]
text = " ".join(text)

vectorizer = TfidfVectorizer(vocabulary=top_words_only)
tfidf_matrix = vectorizer.fit_transform([text])
tfidf_values = tfidf_matrix.toarray()[0]

word_tfidf = dict(zip(top_words_only, tfidf_values))

Виводимо слова та значения метрики TF-IDF

```
for word, tfidf in word_tfidf.items():
    print(f"Слово: {word}, TF-IDF: {tfidf}")
```

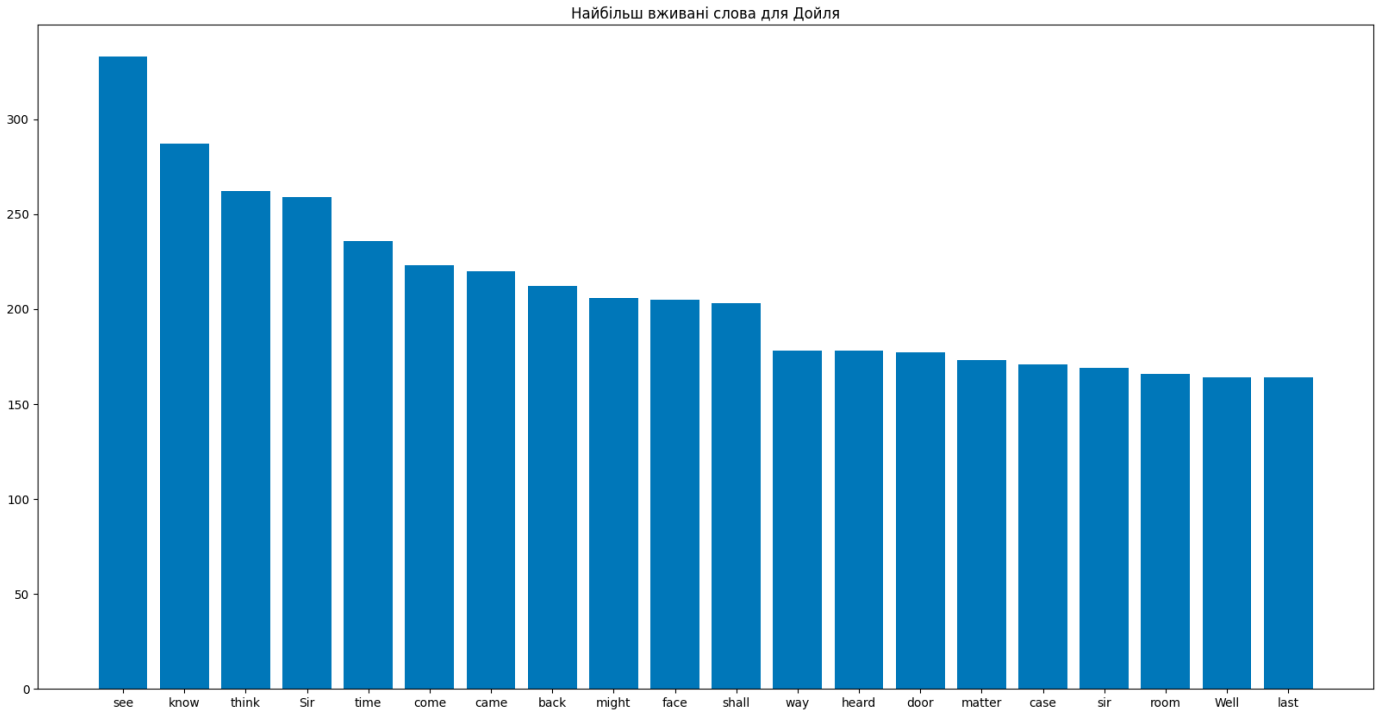
```
Слово: іван, TF-IDF: 0.6143246159081622
Слово: гора, TF-IDF: 0.3751915439439112
Слово: свій, TF-IDF: 0.2721169439593202
Слово: ліс, TF-IDF: 0.251502023962402
Слово: марічка, TF-IDF: 0.251502023962402
Слово: око, TF-IDF: 0.24737903996301835
Слово: нога, TF-IDF: 0.24737903996301835
Слово: од, TF-IDF: 0.251502023962402
Слово: рука, TF-IDF: 0.22676411996610016
Слово: вівця, TF-IDF: 0.2102721839685656
```

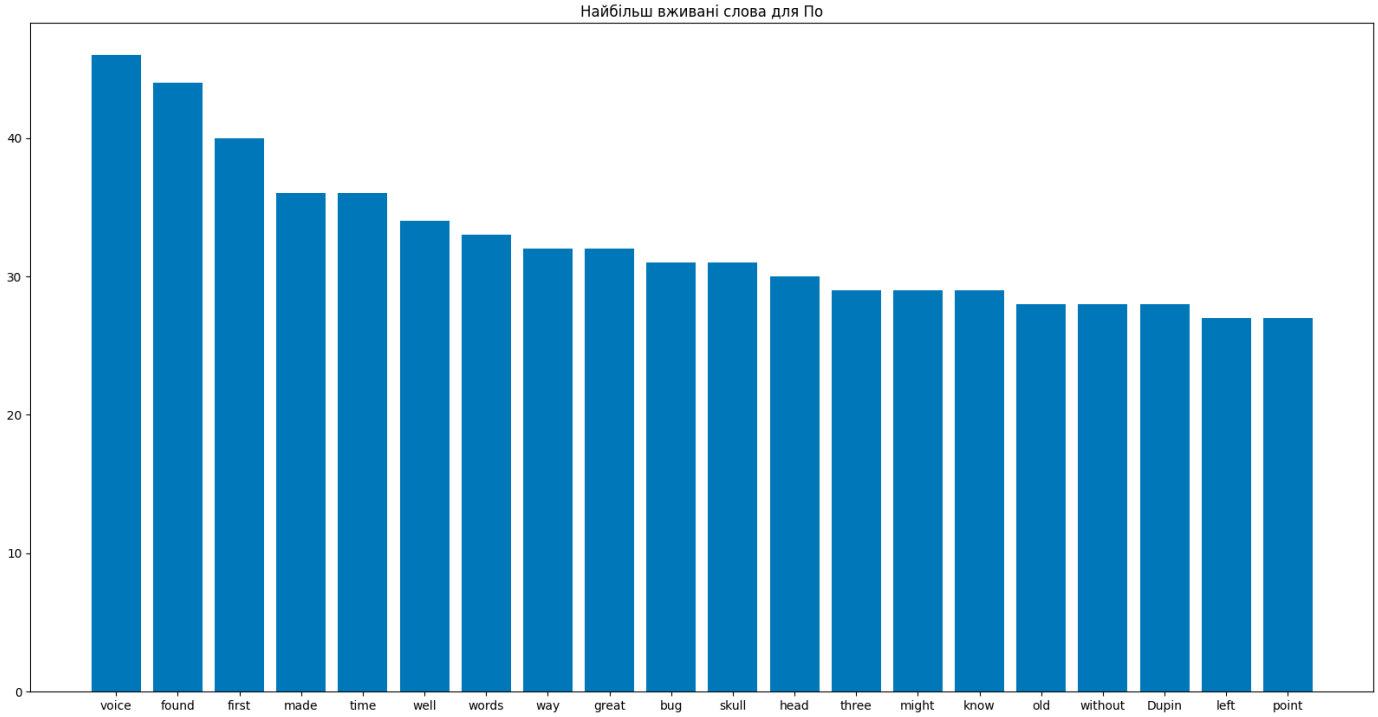
Додаткове завдання:

Імпортуємо потрібні бібліотеки.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
import nltk
nltk.download('vader_lexicon')
from nltk.sentiment import SentimentIntensityAnalyzer
from wordcloud import WordCloud
import string
from functools import reduce

Скорочення слів до їх основи або кореневої форми
porter = PorterStemmer()

Зчитування текстового файлу.
def read_text(path):
    with open(path, 'r') as file:
        return file.read()

Читаємо оповідання Дойля й По
doyle = read_text('doyle.txt')
doyle += read_text('doyle-2.txt')
poe = read_text('poe.txt')
poe += read_text('poe-2.txt')

Повертає функцію для видалення заданих стоп-слів.
def get_stopwords_remover(stop_words):
    return lambda words: [word for word in words if not word.lower() in stop_words]

Список стоп-слів
stop_words = set(stopwords.words('english'))
remove_stopwords = get_stopwords_remover(stop_words)

Видалення знаків пунктуації.
def clear_punctuation(text):
    return text.translate(str.maketrans('', '', string.punctuation)

Стемізація методом Портера.
def porter_stem(words):
    return [porter.stem(word) for word in words]

Функція для об'єднання етапів обробки в єдиний конвеєр.
def preprocessing_pipeline(steps):
```

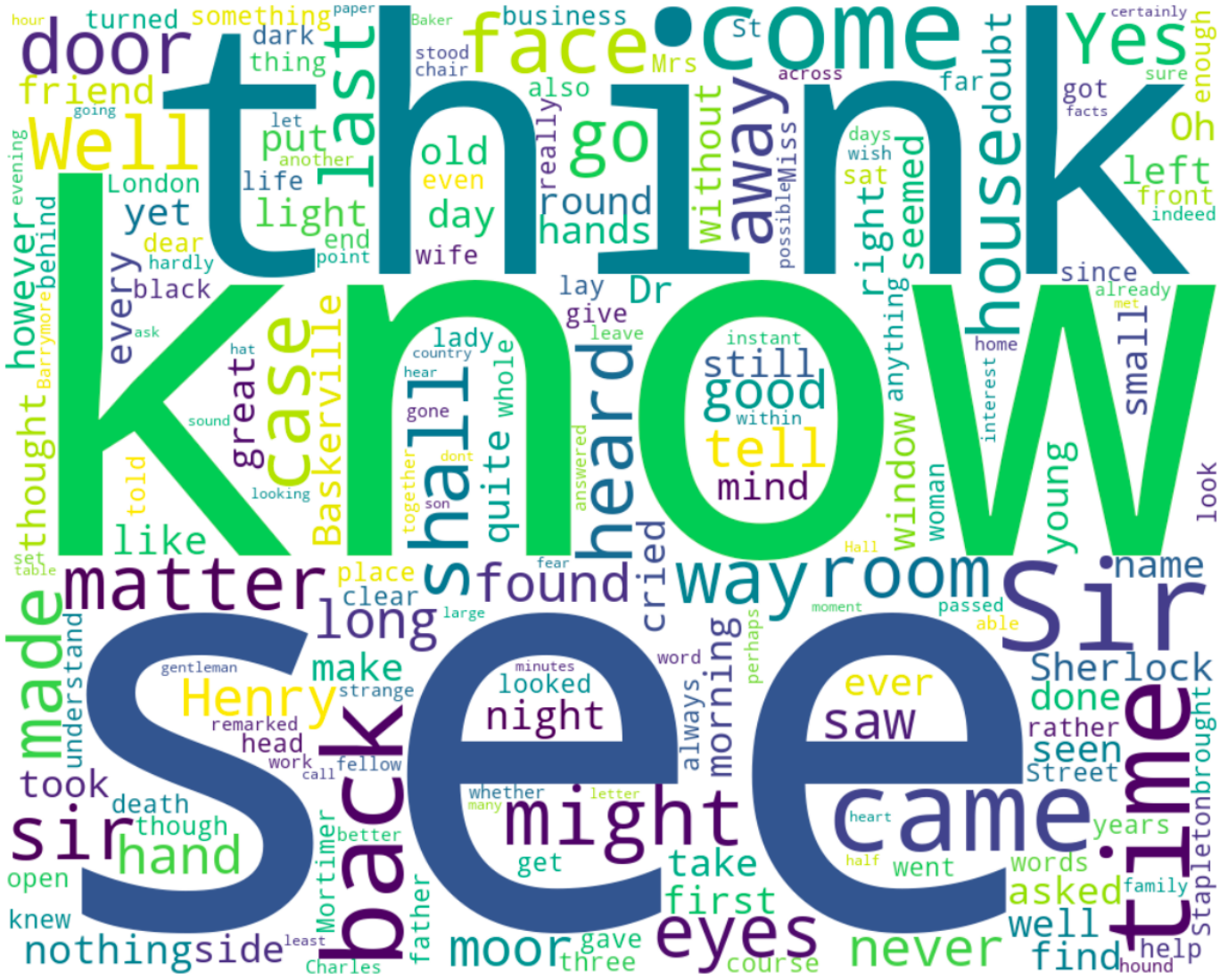
```
    return lambda raw_text: reduce(lambda data, func: func(data), steps, raw_text)

Видалення пунктуації з тексту, токенізація та видалення стоп-слів
pipe = preprocessing_pipeline([
    clear_punctuation,
    word_tokenize,
    remove_stopwords
])
doyle_words = pipe(doyle)
poe_words = pipe(poe)

Додамо кастомні стоп-слова та видалимо їх
custom_stopwords = {'upon', 'one', 'said', 'could', 'would', 'us', 'man', 'mr', 'de', 'may', 'must', 'thus',
'say', 'much', 'little', 'two', 'holmes', 'legrand', 'jupiter', 'watson'}
remove_custom_stopwords = get_stopwords_remover(custom_stopwords)
doyle_words = remove_custom_stopwords(doyle_words)
poe_words = remove_custom_stopwords(poe_words)

Список слів з їх частотами
def word_list_to_freq_dict(wordlist):
    res = {}
    for word in wordlist:
        if word not in res:
            res[word] = 1
        else:
            res[word] += 1
    return res
doyle_words_freq = word_list_to_freq_dict(doyle_words)
poe_words_freq = word_list_to_freq_dict(poe_words)

Графік найбільш вживаних слів, для більш точного аналізу хмар
doyle_words_freq_arr = np.array(sorted(doyle_words_freq.items(), key=lambda item: -item[1])).T
poe_words_freq_arr = np.array(sorted(poe_words_freq.items(), key=lambda item: -item[1])).T
n = 20
plt.figure(figsize=(20, 10))
plt.title('Найбільш вживані слова для Дойля')
plt.bar(doyle_words_freq_arr[0][:n], doyle_words_freq_arr[1][:n].astype(int))

plt.figure(figsize=(20, 10))
plt.title('Найбільш вживані слова для По')
plt.bar(poe_words_freq_arr[0][:n], poe_words_freq_arr[1][:n].astype(int))

plt.show()
```


Найбільш вживані слова для Дойля
0
50
100
150
200
250
300
see
know
think
Sir
time
come
came
back
might
face
shall
way
heard
door
matter
case
sir
room
Well
last


Найбільш вживані слова для По
0
10
20
30
40
voice
found
first
made
time
well
words
way
great
bug
skull
head
three
might
know
old
without
Dupin
left
point

Будуємо хмари слів

```
wordcloud = WordCloud(
    background_color='white',
    width=1000, height=800
)
doyle_cloud = wordcloud.generate_from_frequencies(word_list_to_freq_dict(doyle_words))

plt.figure(figsize=(15, 15))
plt.imshow(doyle_cloud, interpolation='bilinear')
plt.axis('off')
plt.show()

doyle_cloud = wordcloud.generate_from_frequencies(word_list_to_freq_dict(poe_words))

plt.figure(figsize=(15, 15))
plt.imshow(doyle_cloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



Шукаємо сумніші оповідання
Аналіз тексту Дойля

```
sia_doyle = SentimentIntensityAnalyzer()
sentimental_doyle = sia_doyle.polarity_scores(" ".join(doyle_words))
print('Compound sentiment score Doyle: ', sentimental_doyle['compound'])
```

Аналіз тексту "По"

```
sia_poe = SentimentIntensityAnalyzer()
sentimenta_poe = sia_poe.polarity_scores(" ".join(poe_words))
print('Compound sentiment score Poe: ',sentimenta_poe['compound'])
```

```
Compound sentiment score Doyle:  1.0
Compound sentiment score Poe:  0.9999
```

Висновок

За отриманими даними можна зробити висновок, що

- В основному завданні найуживанішим словом є “іван” з метрикою TF-IDF= 0.6143246159081622;

- В додатковому завданні А.К.Дойль написав більш похмурі оповідання.