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](https://drive.google.com/drive/folders/17Qnj87hJWpI20CK_7sYPzQ9jguQBZm25)

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](https://github.com/d9nchik/tokenizer_example)). 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](https://drive.google.com/drive/folders/1kYY4pxkeLXoszCX8icpFUrhFCnwgAyhz?usp=sharing).
* 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

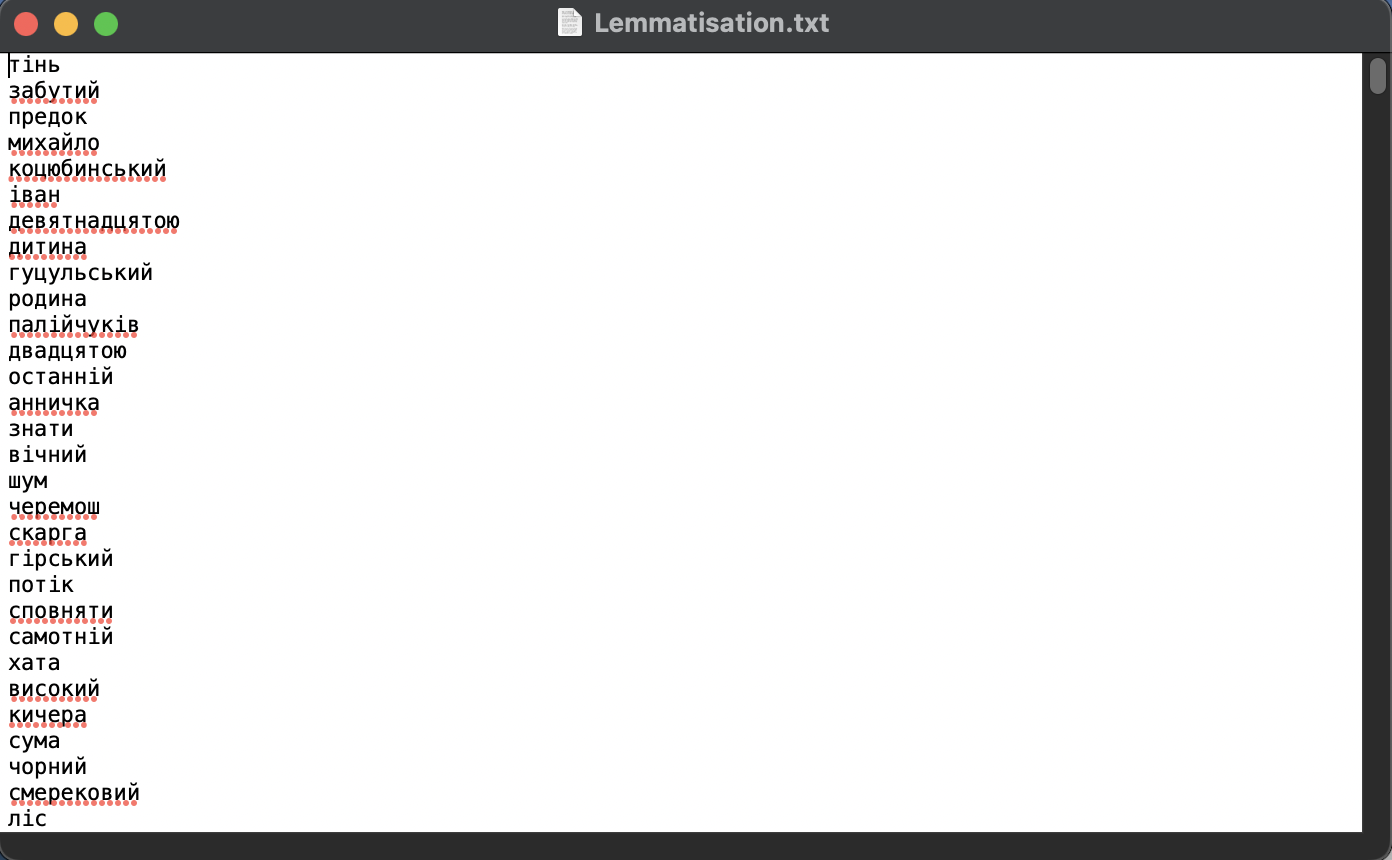
*Лематизація текстових елементів*

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()

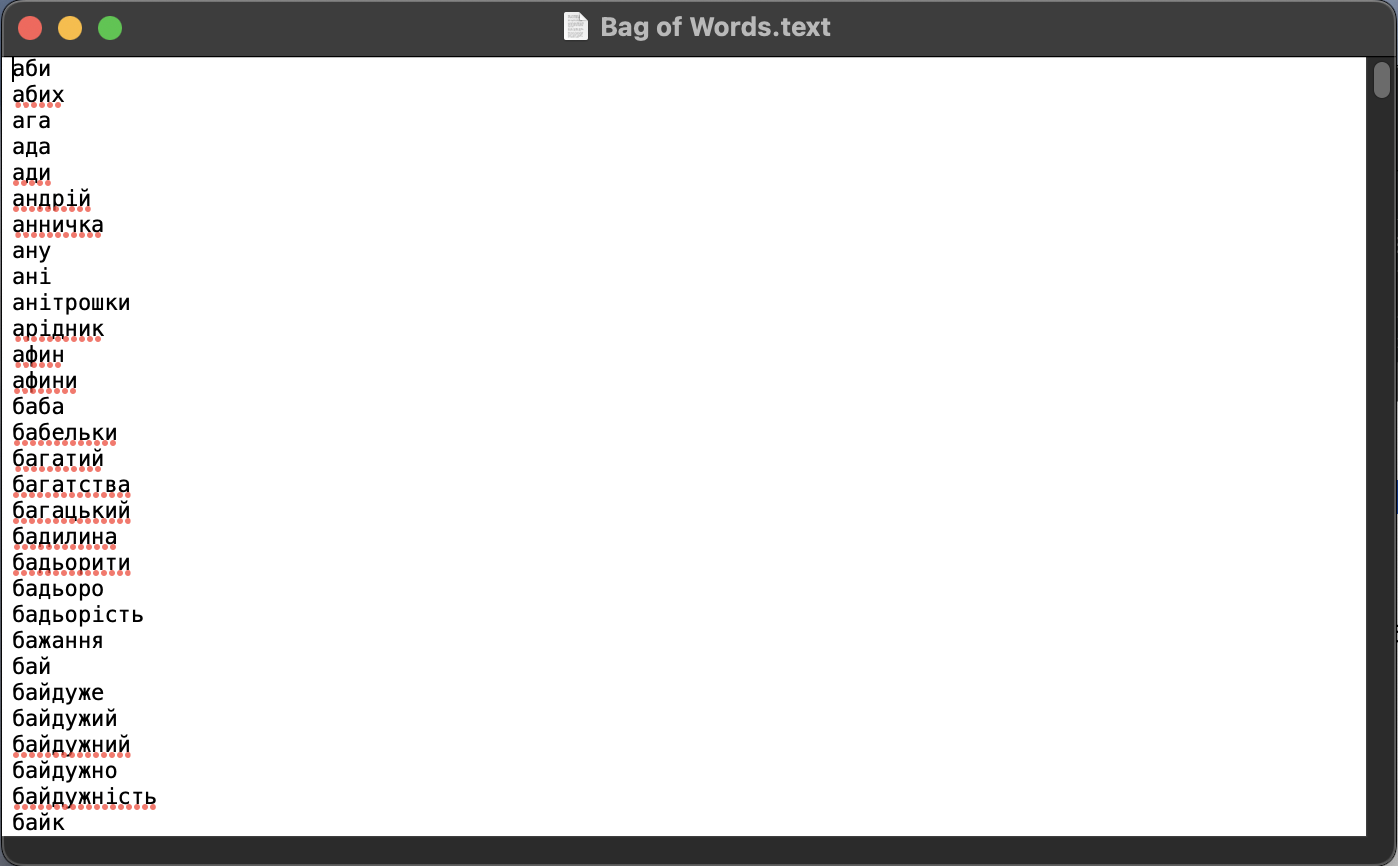
*Виконуємо векторизацію тексту*

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}")



**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')

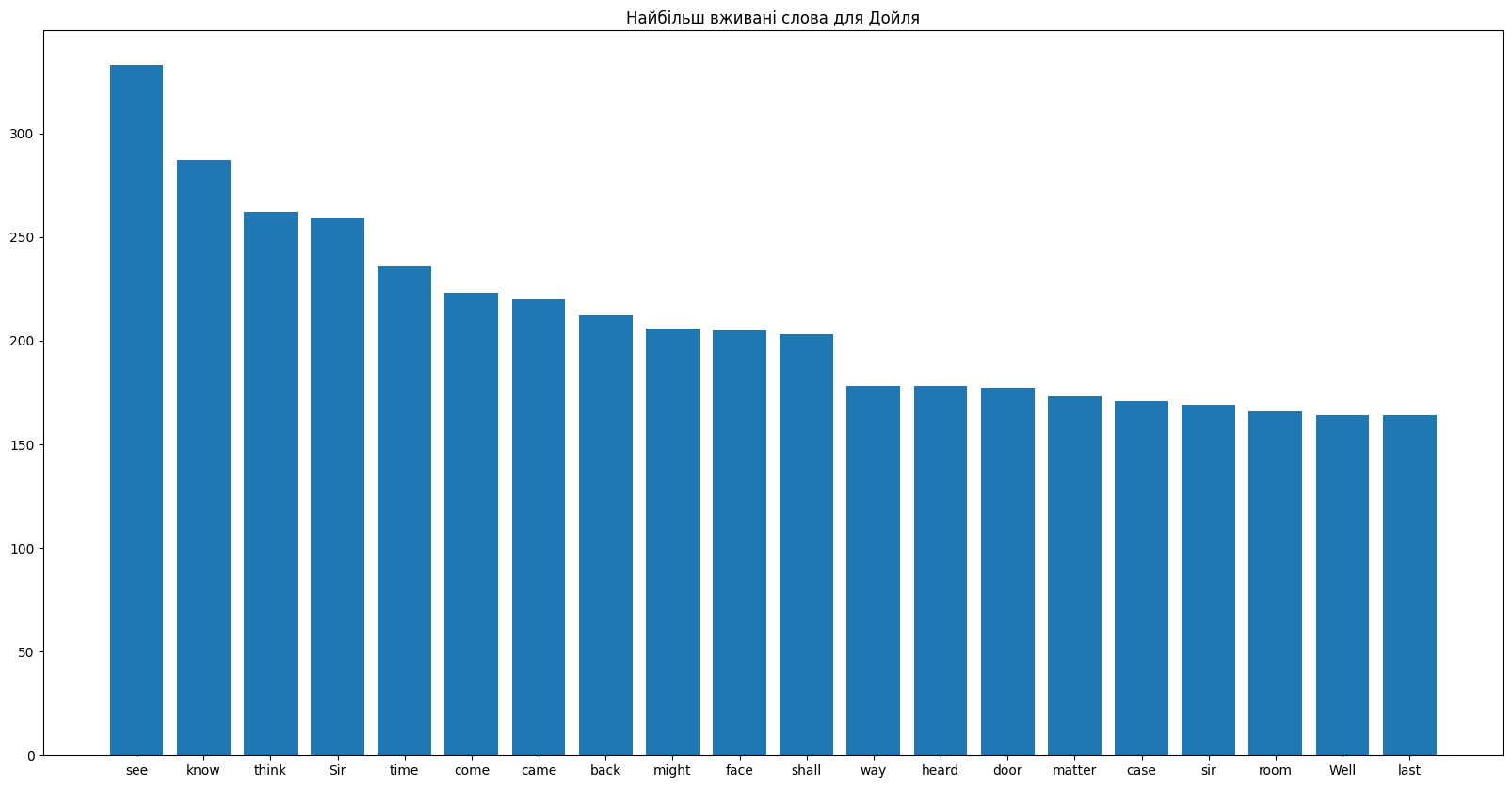
plt.bar(doyle\_words\_freq\_arr[0][:n], doyle\_words\_freq\_arr[1][:n].astype(int))

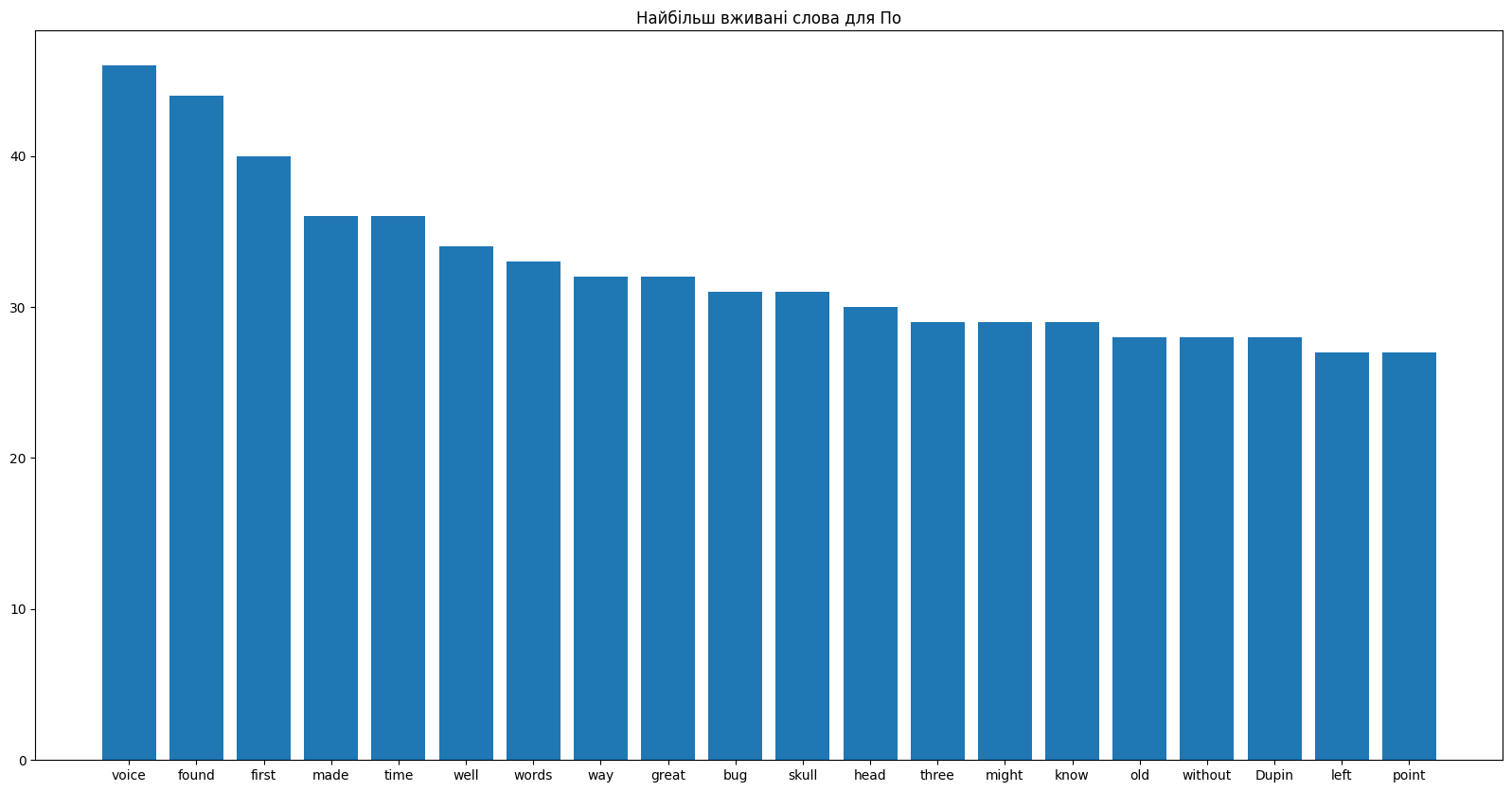
plt.figure(figsize=(20, 10))

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()





*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))

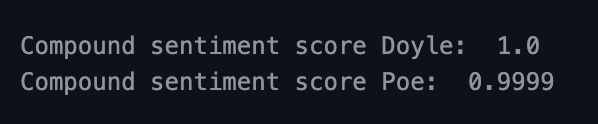
print('Compound sentiment score Doyle: ', sentimental\_doyle['compound'])

*Аналіз тексту "По"*

sia\_poe = SentimentIntensityAnalyzer()

sentimenta\_poe = sia\_poe.polarity\_scores(" ".join(poe\_words))

print('Compound sentiment score Poe: ',sentimenta\_poe['compound'])



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.