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-11 Сідак
Кирил з ФІОТ

Київ-2023

ЗМІСТ

1. ЗАВДАННЯ

1.1 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;

1.2 Додаткове завдання

Інтелектуальний аналіз текстів

- провести сантимент аналіз (визначення емоційної тональності – позитивний / негативний) для даних ukr_text.csv.
- провести категоризацію (визначення категорій тексту) даних методом LSA.

Обробка даних оповідань А.К. Дойля та Е.По

- 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?

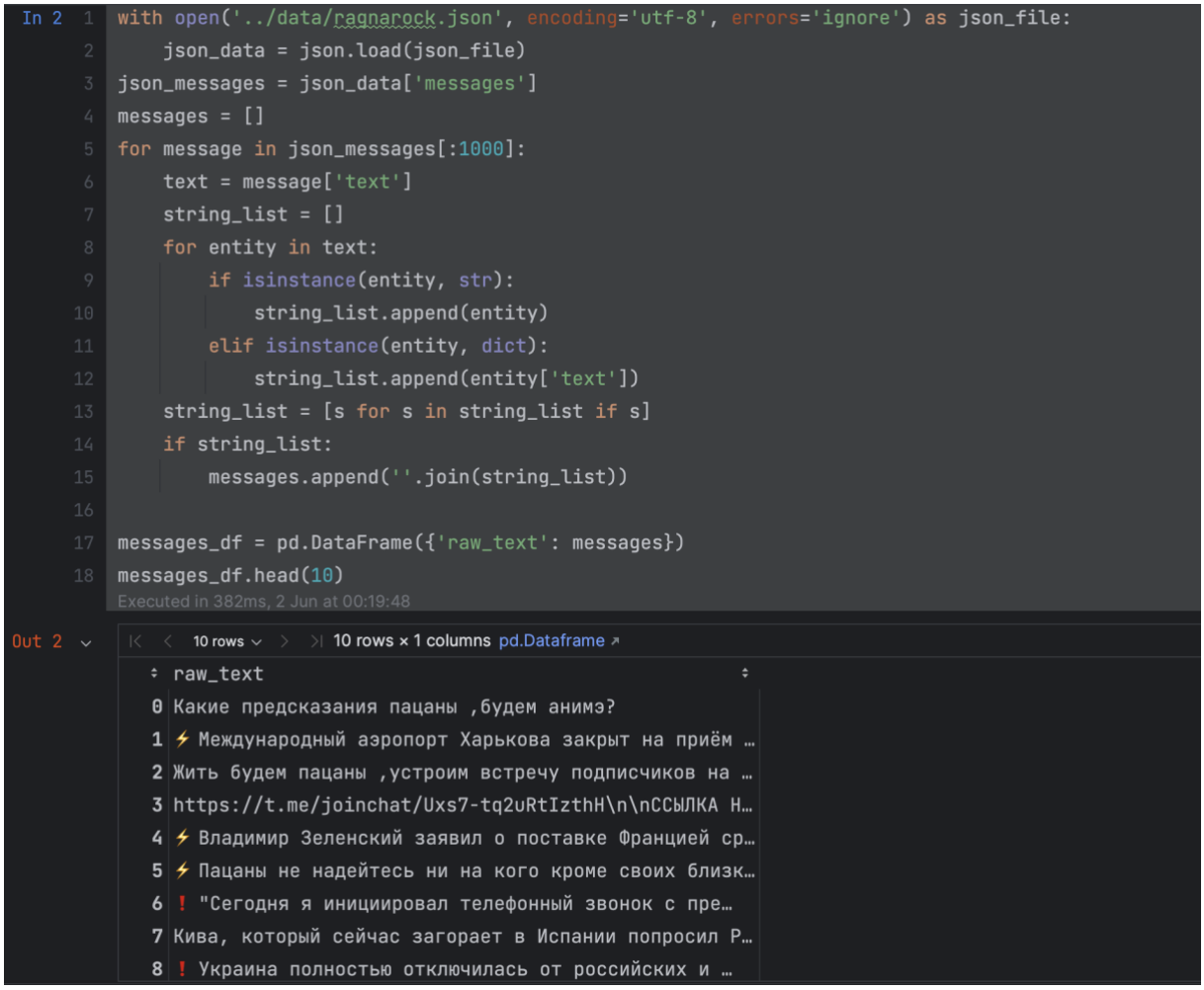
2. ОСНОВНЕ ЗАВДАННЯ

Спочатку імпортуємо усі необхідні пакети, модулі класи та функції для подальшої роботи.

```
In 1
import pandas as pd
import matplotlib.pyplot as plt
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import nltk
import re
import json
import string
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from langdetect import detect
import numpy as np
from pymorphy2 import MorphAnalyzer
from nltk.stem import WordNetLemmatizer
from functools import reduce
import pickle

nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
```

Рисунок 2.1 – Імпортування усіх необхідних пакетів

Наступним кроком завантажимо дані з json-файлу, що містить текстові повідомлення з телеграм-каналу, у датафрейм. Для нашої задачі будуть використані текстові дані з українського телеграм-каналу «RAGNAROCK PRIVET», який загалом містить різні новини.



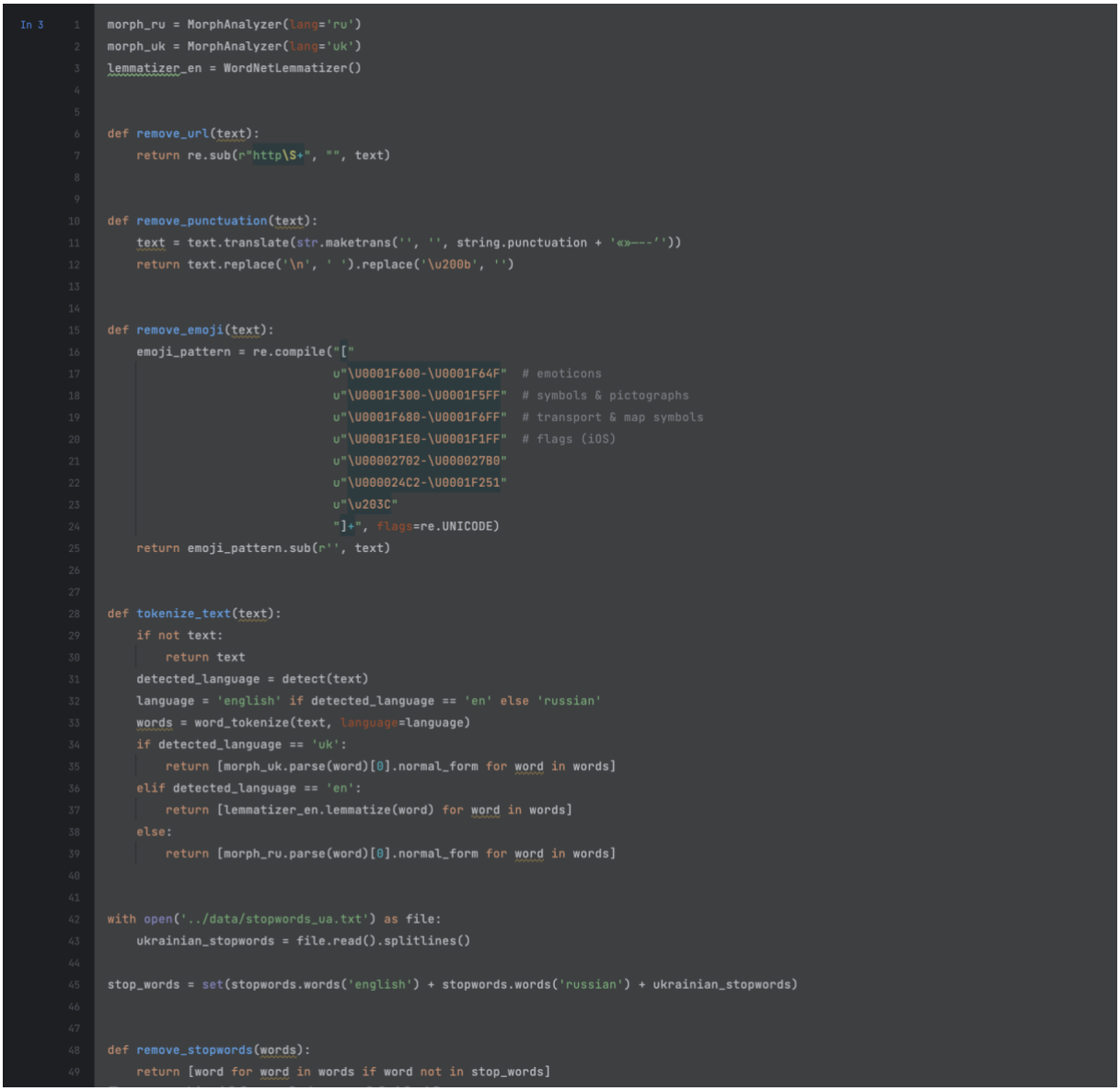
```
In 2
with open('../data/ragnarock.json', encoding='utf-8', errors='ignore') as json_file:
    json_data = json.load(json_file)
json_messages = json_data['messages']
messages = []
for message in json_messages[:1000]:
    text = message['text']
    string_list = []
    for entity in text:
        if isinstance(entity, str):
            string_list.append(entity)
        elif isinstance(entity, dict):
            string_list.append(entity['text'])
    string_list = [s for s in string_list if s]
    if string_list:
        messages.append(''.join(string_list))

messages_df = pd.DataFrame({'raw_text': messages})
messages_df.head(10)
Executed in 382ms, 2 Jun at 00:19:48
```

Out 2 — 10 rows × 1 columns pd.Dataframe

	raw_text
0	Какие предсказания пацаны ,будем анимэ?
1	⚡ Международный аэропорт Харькова закрыт на приём …
2	Жить будем пацаны ,устроим встречу подписчиков на …
3	https://t.me/joinchat/Uxs7-tq2uRtIzthH\n\nССЫЛКА Н…
4	⚡ Владимир Зеленский заявил о поставке Францией ср…
5	⚡ Пацаны не надейтесь ни на кого кроме своих близк…
6	❗ "Сегодня я инициировал телефонный звонок с пре…
7	Кива, который сейчас загорает в Испании попросил Р…
8	❗ Украина полностью отключилась от российских и …

Рисунок 2.2 – Завантаження текстових даних телеграм-каналу у датафрейм

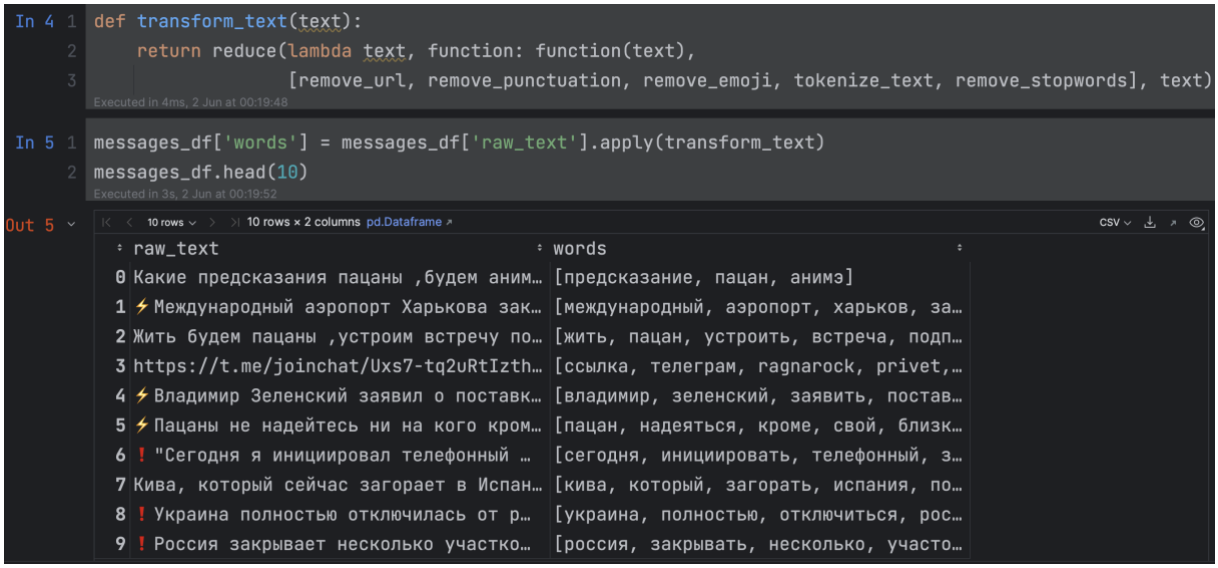
Після цього створимо функції для попередньої обробки даних, а саме: видалення url, видалення пунктуації, видалення емодзі, токенізація, лематизція в залежності від мови та безпосередньо видалення стоп-слів.



```
In 3
morph_ru = MorphAnalyzer(lang='ru')
morph_uk = MorphAnalyzer(lang='uk')
lemmatizer_en = WordNetLemmatizer()


def remove_url(text):
    return re.sub(r"http\S+", "", text)


def remove_punctuation(text):
    text = text.translate(str.maketrans('', '', string.punctuation + '«»—–’'))
    return text.replace('\n', ' ').replace('\u200b', '')


def remove_emoji(text):
    emoji_pattern = re.compile("["
                               u"\U0001F600-\U0001F64F"  # emoticons
                               u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                               u"\U0001F680-\U0001F6FF"  # transport & map symbols
                               u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                               u"\U00002702-\U000027B0"
                               u"\U000024C2-\U0001F251"
                               u"\u203C"
                               "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', text)


def tokenize_text(text):
    if not text:
        return text
    detected_language = detect(text)
    language = 'english' if detected_language == 'en' else 'russian'
    words = word_tokenize(text, language=language)
    if detected_language == 'uk':
        return [morph_uk.parse(word)[0].normal_form for word in words]
    elif detected_language == 'en':
        return [lemmatizer_en.lemmatize(word) for word in words]
    else:
        return [morph_ru.parse(word)[0].normal_form for word in words]


with open('../data/stopwords_ua.txt') as file:
    ukrainian_stopwords = file.read().splitlines()

stop_words = set(stopwords.words('english') + stopwords.words('russian') + ukrainian_stopwords)


def remove_stopwords(words):
    return [word for word in words if word not in stop_words]
```

Рисунок 2.3 – Визначення функцій для попередньої обробки текстових даних

Наступним кроком визначимо функцію, яка буде послідовно застосовувати вище визначені функції до тексту (рядку датафрейму), та застосуємо її до датафрейму, створивши новий стовпець, що містить для кожного повідомлення список нормалізованих форм слів.



```
In 4
def transform_text(text):
    return reduce(lambda text, function: function(text),
                  [remove_url, remove_punctuation, remove_emoji, tokenize_text, remove_stopwords], text)
Executed in 4ms, 2 Jun at 00:19:48

In 5
messages_df['words'] = messages_df['raw_text'].apply(transform_text)
messages_df.head(10)
Executed in 3s, 2 Jun at 00:19:52

Out 5   10 rows   10 rows × 2 columns  pd.Dataframe                                                    CSV
```

	raw_text	words
0	Какие предсказания пацаны ,будем аним…	[предсказание, пацан, анимэ]
1	⚡Международный аэропорт Харькова зак…	[международный, аэропорт, харьков, за…
2	Жить будем пацаны ,устроим встречу по…	[жить, пацан, устроить, встреча, подп…
3	https://t.me/joinchat/Uxs7-tq2uRtIzth…	[ссылка, телеграм, ragnarock, privet,…
4	⚡Владимир Зеленский заявил о поставк…	[владимир, зеленский, заявить, постав…
5	⚡Пацаны не надейтесь ни на кого кром…	[пацан, надеяться, кроме, свой, близк…
6	❗"Сегодня я инициировал телефонный …	[сегодня, инициировать, телефонный, з…
7	Кива, который сейчас загорает в Испан…	[кива, который, загорать, испания, по…
8	❗Украина полностью отключилась от р…	[украина, полностью, отключиться, рос…
9	❗Россия закрывает несколько участко…	[россия, закрывать, несколько, участо…

Рисунок 2.4 – Визначення функції для попередньої обробки текстових даних та її застосування до датафрейму

Тепер обчислимо частоту кожного слова серед усіх виділених слів.

```
In 6
words = pd.Series(np.concatenate(messages_df['words']))
words_count = words.value_counts()
words_count.head()
Executed in 8ms, 2 Jun at 00:19:52

Out 6   5 rows   Length: 5, dtype: int64  pd.Series                                                  CSV
```

	<unnamed>
украина	174
российский	112
канал	93
киев	86
это	80

Рисунок 2.5 – Обчислення частоти кожного слова

Після цього за допомогою горизонтальної стовпчастої діаграми візуалізуємо частоти 20-ти найбільш частих слів.

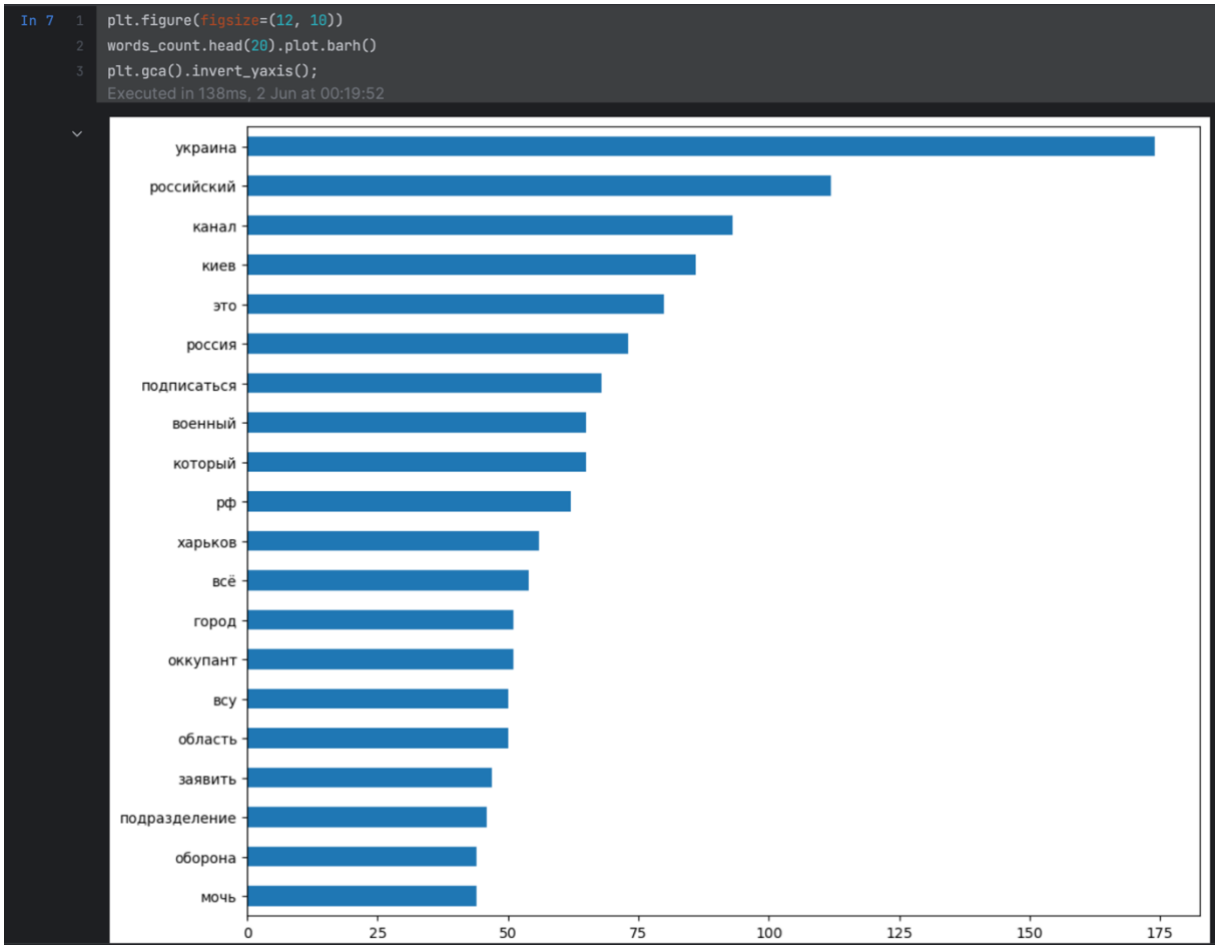


Рисунок 2.6 – Візуалізація частоти 20-ти найбільш частих слів

На основі даної візуалізації можна побачити, що слова «это», «подписаться», «всё» часто зустрічаються , проте загалом не несуть смислового навантаження, тому їх також можна віднести до стоп-слів та їх варто видалити.

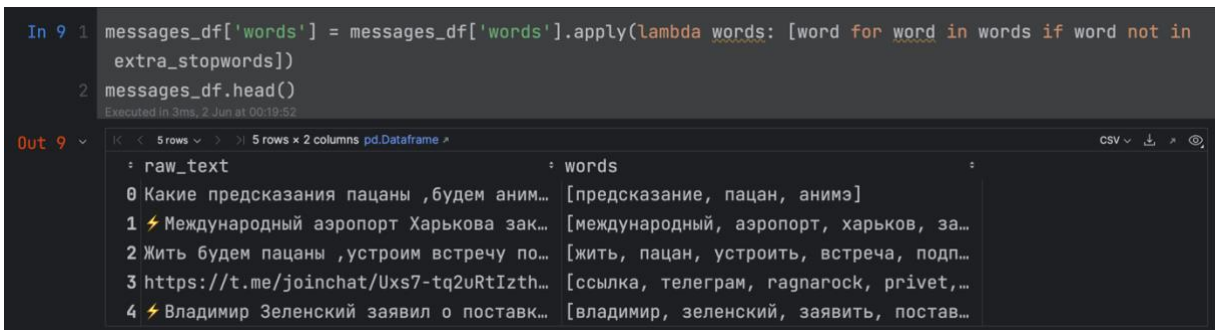


Рисунок 2.7 – Видалення стоп-слів «это», «подписаться», «всё»

Тепер створимо стовпець, що буде містити текст з нормалізованих форм слів, записаних через пробіл.

```
In 10  messages_df['normalized_text'] = messages_df['words'].str.join(' ')
       messages_df.head()
```

Out 10 — 5 rows × 3 columns pd.Dataframe

	raw_text	words	normalized_text
0	Какие предсказания пацаны ,будем аним…	[предсказание, пацан, анимэ]	предсказание пацан ани
1	⚡Международный аэропорт Харькова зак…	[международный, аэропорт, харьков, за…	международный аэропорт
2	Жить будем пацаны ,устроим встречу по…	[жить, пацан, устроить, встреча, подп…	жить пацан устроить вс
3	https://t.me/joinchat/Uxs7-tq2uRtIzth…	[ссылка, телеграм, ragnarock, privet,…	ссылка телеграм ragnar
4	⚡Владимир Зеленский заявил о поставк…	[владимир, зеленский, заявить, постав…	владимир зеленский зая

Рисунок 2.8 – Створення стовпця з нормалізованим текстом

На основі даного стовпця створимо та навчимо Bag of Words (торбу слів) та відобразимо отримані результати у вигляді датафрейма. Також збережемо матрицю та словник у відповідних файлах.

```
In 11  cv = CountVectorizer()
       bag_of_words = cv.fit_transform(messages_df['normalized_text'])
       bag_of_words = pd.DataFrame(bag_of_words.toarray(), columns=cv.get_feature_names_out())
       bag_of_words.head()
```

Out 11 — 5 rows × 4298 columns pd.Dataframe

	000	0139	015	0185	020	025	031	04	050	055	0551	0600	0612870164	0616536417
0	0	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	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0

```
In 12  with open('../data/bag_of_words_matrix.pkl', 'wb') as file:
           pickle.dump(bag_of_words, file)
       with open('../data/vocabulary.pkl', 'wb') as file:
           pickle.dump(cv.vocabulary_, file)
```

Рисунок 2.9 – Створення торби слів та збереження матриці й словника у файлі

Наступним кроком створимо векторизатор TF-IDF та навчимо його на нормалізованому тексті (реченнях).

```
In 13  tfidf_vectorizer = TfidfVectorizer()

In 14  tfidf_matrix = tfidf_vectorizer.fit_transform(messages_df['normalized_text'])
       feature_names = tfidf_vectorizer.get_feature_names_out()

In 15  words = pd.Series(np.concatenate(messages_df['words']))
       words_count = words.value_counts()
```

Рисунок 2.10 – Створення та навчання векторизатора TF-IDF

Тепер, використовуючи отриману вище матрицю метрики TF-IDF, створимо датафрейм з показниками цієї метрики для 10 найбільш вживаних слів для кожного документа (речення).

```
In 15
words = pd.Series(np.concatenate(messages_df['words']))
words_count = words.value_counts()
Executed in 74ms, 2 Jun at 01:20:32

In 16
top_words_df = messages_df[['raw_text']].copy()
for word in words_count.head(10).index:
    top_words_df[word] = 0
    word_index = np.where(feature_names == word)[0][0]
    for document_index, _ in top_words_df.iterrows():
        top_words_df.loc[document_index, word] = tfidf_matrix[document_index, word_index]
top_words_df
Executed in 244ms, 2 Jun at 01:20:33
```

Out 16 — 1-10 — 821 rows × 11 columns pd.Dataframe — CSV

	raw_text	украина	российский	канал	киев	россия	военный	к
0	Какие предсказания пацаны ,будем аним…	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0
1	⚡Международный аэропорт Харькова зак…	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0
2	Жить будем пацаны ,устроим встречу по…	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0
3	https://t.me/joinchat/Uxs7-tq2uRtIzth…	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0
4	⚡Владимир Зеленский заявил о поставк…	0.101376	0.000000	0.0	0.000000	0.000000	0.000000	0
5	⚡Пацаны не надейтесь ни на кого кром…	0.071535	0.000000	0.0	0.000000	0.000000	0.000000	0
6	❗"Сегодня я инициировал телефонный …	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0
7	Кива, который сейчас загорает в Испан…	0.129291	0.000000	0.0	0.000000	0.170061	0.000000	0
8	❗Украина полностью отключилась от р…	0.109078	0.123481	0.0	0.000000	0.000000	0.000000	0
9	❗Россия закрывает несколько участко…	0.000000	0.000000	0.0	0.000000	0.164962	0.000000	0

Рисунок 2.11 – Обчислення TF-IDF метрики для 10 найбільш вживаних у корпусі слів

3. ПЕРШЕ ДОДАТКОВЕ ЗАВДАННЯ

Спочатку аналогічно імпортуємо усі необхідні пакети для подальшої роботи.

```
In 1
import numpy as np
import pandas as pd
from nltk.tokenize import word_tokenize
import nltk
from functools import reduce
import string
from pymorphy2 import MorphAnalyzer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD

nltk.download('stopwords')
nltk.download('punkt')
```

Рисунок 3.1 – Імпортування усіх необхідних пакетів

Потім визначимо аналогічні функції для попередньої обробки наших текстових даних.

```
In 2
morph_uk = MorphAnalyzer(lang='uk')


def remove_punctuation(text):
    text = text.translate(str.maketrans('', '', string.punctuation + '«»—–-’'))
    return text.replace('\n', ' ').replace('NBSP', ' ')


def tokenize_text(text):
    words = word_tokenize(text, language='russian')
    return [morph_uk.parse(word)[0].normal_form for word in words]


with open('../data/stopwords_ua.txt') as file:
    ukrainian_stopwords = set(file.read().splitlines())


def remove_stopwords(words):
    return [word for word in words if word not in ukrainian_stopwords]
```

Рисунок 3.2 – Визначення функцій для попередньої обробки текстових даних

Наступним кроком визначимо функцію, яка послідовно викликатиме функції для обробки текстових даних.

```
In 3  1  def transform_text(text):
      2      return reduce(lambda text, function: function(text),
      3                    [remove_punctuation, tokenize_text, remove_stopwords], text)
```

Рисунок 3.3 – Визначення функції для попередньої обробки тексту

Тепер зчитаємо дані з csv-файлу в датафрейм, приведемо назви стовпців до нижнього регістру та застосуємо функцію обробки тексту для стовпця з самим текстом (body).

```
In 4  1  ukr_text_df = pd.read_csv('../data/ukr_text.csv')
      2  ukr_text_df.columns = ukr_text_df.columns.str.lower()
      3  ukr_text_df.head()
```

Out 4 — 5 rows × 3 columns pd.Dataframe

	id	title	body
0	http://k.img.com.ua/rss/ua/4013798	Кличко покликав німецьких інвесторів …	Київ - перспективний і від
1	http://k.img.com.ua/rss/ua/4001679	З'явилося відео, як байкер почав стрі…	З'явилося відео конфлікту
2	http://k.img.com.ua/rss/ua/4001390	У центрі Києва посеред вулиці помер ч…	У Києві на Бессарабській п
3	http://k.img.com.ua/rss/ua/4001239	Нічний ураган перетворив Хрещатик на …	Київ вночі 16 серпня перех
4	http://k.img.com.ua/rss/ua/4001227	Потоп у Києві: столицю накрив ураган …	Уночі Київ вкотре накрила

```
In 5  1  ukr_text_df['words'] = ukr_text_df['body'].apply(transform_text)
      2  ukr_text_df.head()
```

Out 5 — 5 rows × 4 columns pd.Dataframe

	body	words
зв німецьких інвесторів …	Київ - перспективний і відкритий рино…	[перспективний, відкритий, ринок, біз…
ео, як байкер почав стрі…	З'явилося відео конфлікту між мотоцик…	[зявитися, відео, конфлікт, мотоциклі…
а посеред вулиці помер ч…	У Києві на Бессарабській площі вранці…	[бессарабський, площа, вранці, четвер…
перетворив Хрещатик на …	Київ вночі 16 серпня пережив найсильн…	[вночі, 16, серпень, пережити, найсил…
: столицю накрив ураган …	Уночі Київ вкотре накрила негода. Най…	[уночі, вкотре, накрити, негода, найс…

Рисунок 3.4 – Зчитування даних та обробка тексту

Після цього завантажимо словник тональності та визначимо функцію для її обчислення шляхом додавання всіх значень тональності, сума яких буде поділена на загальну кількість слів у тексті (новині) з метою врівноваження оцінку для текстів (новин), що містять різну кількість слів.

```
In 6
tone_df = pd.read_csv('https://raw.githubusercontent.com/lang-uk/tone-dict-uk/master/tone-dict-uk.tsv',
 delimiter='\t', names=['word', 'tone'], index_col=0)
tone_df.head()
```

Executed in 377ms, 2 Jun at 01:41:48

Out 6 — 5 rows × 1 columns pd.Dataframe

word	tone
Всевишній	1
Господь	1
Христовий	1
аборт	-1
абсурд	-1

Рисунок 3.5 – Завантаження словника тональності

```
In 7
def calculate_tone(words):
    tone_words = [word for word in words if word in tone_df.index]
    if len(set(tone_words)) < 5:
        return 0
    return sum([tone_df.loc[word].tone if word in tone_df.index else 0 for word in words]) / len(words)

ukr_text_df['tone'] = ukr_text_df['words'].apply(calculate_tone)
ukr_text_df.head()
```

Executed in 339ms, 2 Jun at 01:58:36

Out 7 — 5 rows × 5 columns pd.Dataframe

	body	words	tone
інвесторів …	Київ - перспективний і відкритий рино…	[перспективний, відкритий, ринок, біз…	0.026163
р почав стрі…	З'явилося відео конфлікту між мотоцик…	[зявитися, відео, конфлікт, мотоциклі…	-0.090909
лиці помер ч…	У Києві на Бессарабській площі вранці…	[бессарабський, площа, вранці, четвер…	-0.137255
Хрещатик на …	Київ вночі 16 серпня пережив найсильн…	[вночі, 16, серпень, пережити, найсил…	0.000000
крив ураган …	Уночі Київ вкотре накрила негода. Най…	[уночі, вкотре, накрити, негода, найс…	0.000000

Рисунок 3.6 – Обчислення показника тональності для кожної новини

Для наочності отриманих результатів виведемо найбільш негативну новину та найбільш позитивну.

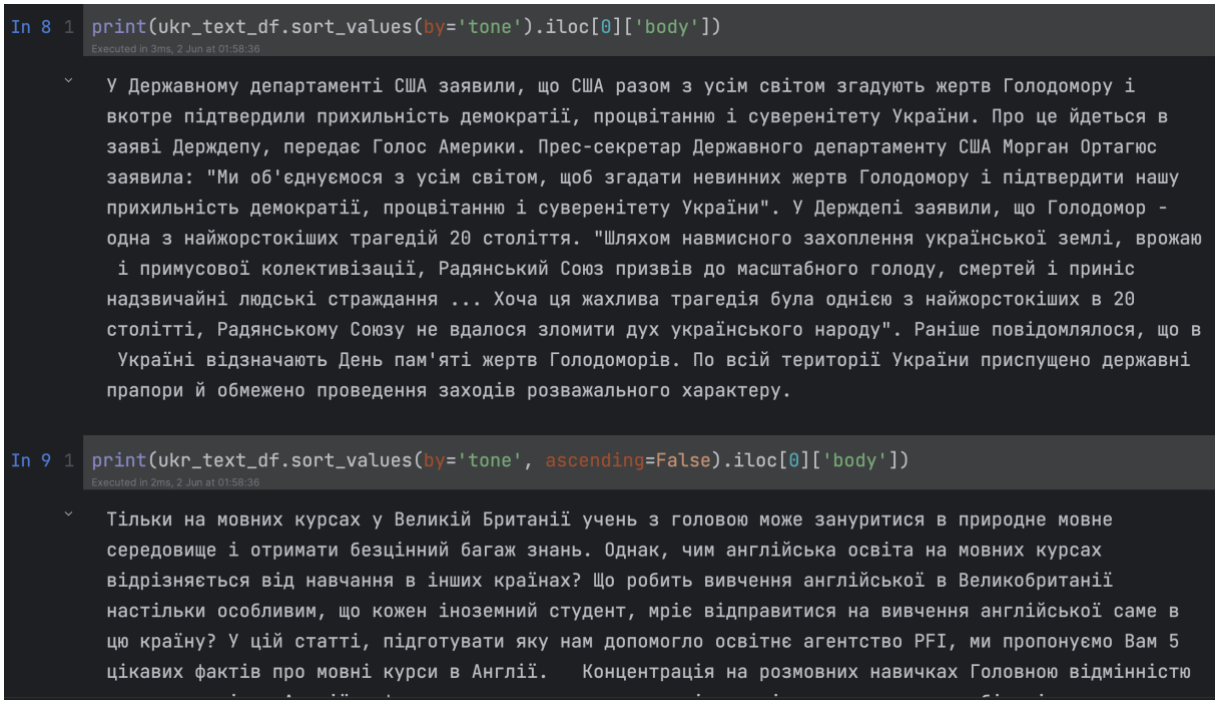


Рисунок 3.7 – Найбільш негативна та позитивна новина відповідно

Аналогічно до основного завдання створимо стовпець з нормалізованим текстом, тобто, з нормалізованими формами слів, записаних через пробіл.

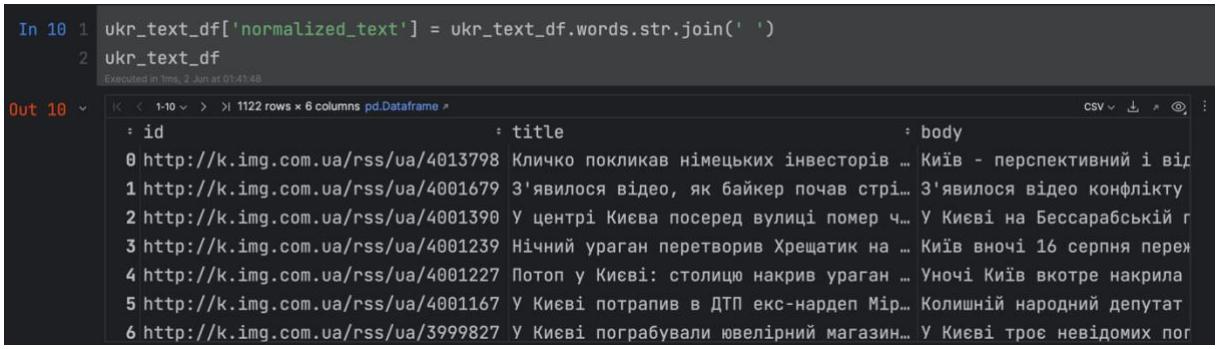


Рисунок 3.8 – Створення стовпця з нормалізованим текстом

Далі створимо та навчимо векторизатор TF-IDF на основі даного стовпця.

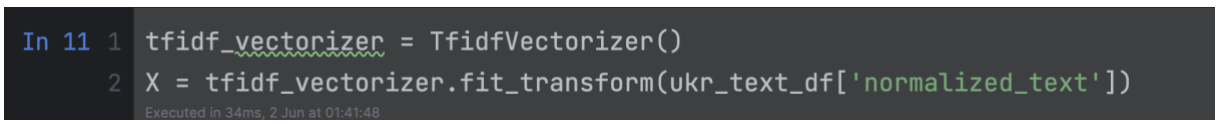


Рисунок 3.9 – Створення та навчання векторизатора TF-IDF

Наступним кроком створимо категоризатор та оберемо кількість тем рівну 100 для методу LSA. Навчимо цей перетворювач на основі матриці метрики TF-IDF та виведемо нову отриману матрицю.

```
In 12  1  svd_vectorizer = TruncatedSVD(n_components=100, random_state=42)
       2  X_lsa = svd_vectorizer.fit_transform(X)
          Executed in 763ms, 2 Jun at 01:58:37

In 13  1  X_lsa
          Executed in 2ms, 2 Jun at 01:58:37

Out 13    1-10 > >| 1,122 rows × 100 columns np.ndarray ↗        CSV
```

	0	1	2	3	4	5	6	7	8	9
3	0.043385	-0.012315	-0.014534	-0.012375	-0.003493	0.033754	-0.028741	-0.009413	0.014932	-0.005994
4	0.059402	-0.021692	-0.029251	-0.016541	-0.018693	0.019230	-0.054412	-0.001225	0.029294	-0.029618
5	0.113500	-0.038551	-0.024676	-0.001565	0.001591	0.069859	-0.034043	0.008222	0.019380	-0.025668
6	0.080400	-0.028267	-0.021085	-0.017458	-0.018908	0.027732	-0.049721	-0.017733	0.005627	-0.008596
7	0.071619	-0.015004	-0.004601	-0.001071	-0.001383	0.055652	-0.008666	-0.007712	0.019281	0.002495
8	0.110948	-0.037762	-0.033153	-0.011700	-0.003115	0.055305	-0.058385	-0.003451	0.029156	-0.016927
9	0.129260	-0.056900	-0.088444	-0.013063	-0.000221	0.047951	-0.022126	0.131658	-0.014638	-0.017116

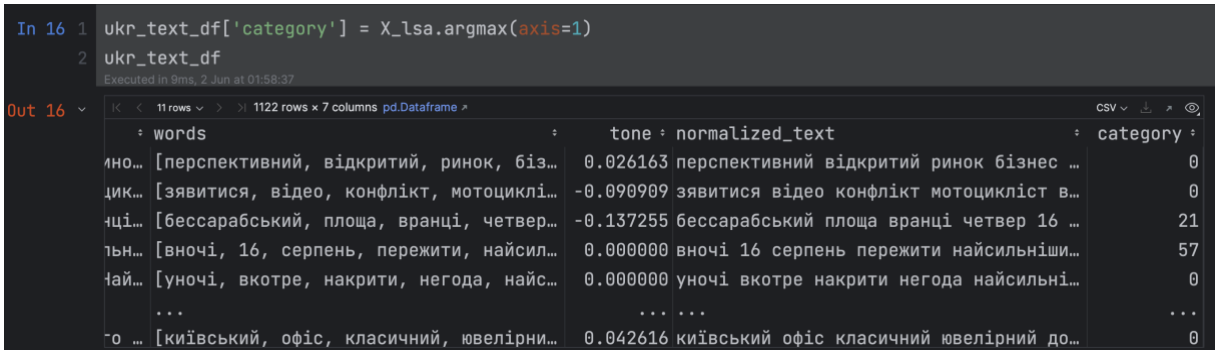
Рисунок 3.10 – Створення та навчання категоризатора (LSA)

Тепер визначимо функцію для виведення найважливіших слів по кожній категорії та застосуємо її.

```
In 14  1  def get_category_words(vectorizer, svd, n_top_words):
       2      words = vectorizer.get_feature_names_out()
       3      topics = []
       4      for component in svd.components_:
       5          top_words_idx = np.argsort(component)[::-1][:n_top_words]
       6          top_words = [words[i] for i in top_words_idx]
       7          topics.append(top_words)
       8      return topics
          Executed in 1ms, 2 Jun at 01:58:37

In 15  1  categories = get_category_words(tfidf_vectorizer, svd_vectorizer, 10)
       2  for i, category in enumerate(categories):
       3      print(f'Words in category {i}: {", ".join(category)}')
          Executed in 174ms, 2 Jun at 01:58:37

   Words in category 0: україна, рок, долар, компанія, росія, новий, йога, країна, область, грудень
   Words in category 1: нафта, біржа, долар, індекс, барель, пункт, марка, торг, ньюйоркський, brent
   Words in category 2: область, регіон, інвестиція, газа, млрд, грн, населення, душити, транзит,
   україна
   Words in category 3: долар, газа, транзит, росія, україна, нафтогаз, газпром, нафта, російський,
   барель
   Words in category 4: індекс, пункт, jones, dow, nasdaq, фондовий, 500, виборчий, ринок, цвк
   Words in category 5: виборчий, округа, цвк, комісія, депутат, народний, серпень, обраний,
   зареєструвати, голос
   Words in category 6: гонка, область, кубок, україна, регіон, збірний, підручний, естафета,
```

Рисунок 3.11 – Найважливіші слова для кожної категорії

Потім створимо стовпець категорії, що містить категорію, до якої належить кожна з новин.



```
In 16  1  ukr_text_df['category'] = X_lsa.argmax(axis=1)
       2  ukr_text_df
Executed in 9ms, 2 Jun at 01:58:37
```

Out 16 — 11 rows — 1122 rows × 7 columns pd.Dataframe — CSV

	words	tone	normalized_text	category
ино…	[перспективний, відкритий, ринок, біз…	0.026163	перспективний відкритий ринок бізнес …	0
дик…	[зявитися, відео, конфлікт, мотоцикліс…	-0.090909	зявитися відео конфлікт мотоцикліст в…	0
яці…	[бессарабський, площа, вранці, четвер…	-0.137255	бессарабський площа вранці четвер 16 …	21
ьын…	[вночі, 16, серпень, пережити, найсил…	0.000000	вночі 16 серпень пережити найсильніши…	57
дай…	[уночі, вкотре, накрити, негода, найс…	0.000000	уночі вкотре накрити негода найсильні…	0
	...	...	...	...
го …	[київський, офіс, класичний, ювелірни…	0.042616	київський офіс класичний ювелірний до…	0

Рисунок 3.12 – Створення стовпця з категорією для кожної новини

4. ДРУГЕ ДОДАТКОВЕ ЗАВДАННЯ

Аналогічно до попередніх завдань імпортуємо усі необхідні пакети для подальшої роботи.

```
In 1  1  import pandas as pd
      2  import matplotlib.pyplot as plt
      3  from nltk.tokenize import word_tokenize
      4  from nltk.corpus import stopwords
      5  from nltk.corpus import sentiwordnet as swn
      6  import nltk
      7  from wordcloud import WordCloud
      8  import string
      9  from functools import reduce
     10  from nltk.stem import WordNetLemmatizer
     11
     12  nltk.download('stopwords')
     13  nltk.download('punkt')
     14  nltk.download('averaged_perceptron_tagger')
     15  nltk.download('sentiwordnet')
```

Рисунок 4.1 – Імпортування усіх необхідних пакетів

Далі визначимо аналогічні функції для попередньої обробки наших текстових даних.

```
In 2
lemmatizer = WordNetLemmatizer()

def remove_punctuation(text):
    return text.translate(str.maketrans('', '', string.punctuation))


def get_stopwords_remover(stop_words):
    return lambda words: words[~words.isin(stop_words)]


def lemmatize(words):
    return words.apply(lemmatizer.lemmatize)


def preprocessing_pipeline(steps):
    return lambda raw_text: reduce(lambda data, func: func(data), steps,
     raw_text)


def lowercase(words):
    return words.str.lower()
Executed in 23ms, 2 Jun at 02:11:42
```

Рисунок 4.2 – Функції для попередньої обробки тексту

Наступним кроком створимо пайплайн для послідовного застосування цих функцій до тексту.

```
In 3
stop_words = set(stopwords.words('english'))
remove_stopwords = get_stopwords_remover(stop_words)
Executed in 23ms, 2 Jun at 02:11:42

In 4
pipe = preprocessing_pipeline([
    remove_punctuation,
    word_tokenize,
    pd.Series,
    lowercase,
    remove_stopwords,
    lemmatize
])
```

Рисунок 4.3 – Створення пайплайну для попередньої обробки тексту

Після цього завантажимо оповідання Дойла та По з текстових файлів у рядки та застосуємо пайплан для попередньої обробки цих даних.

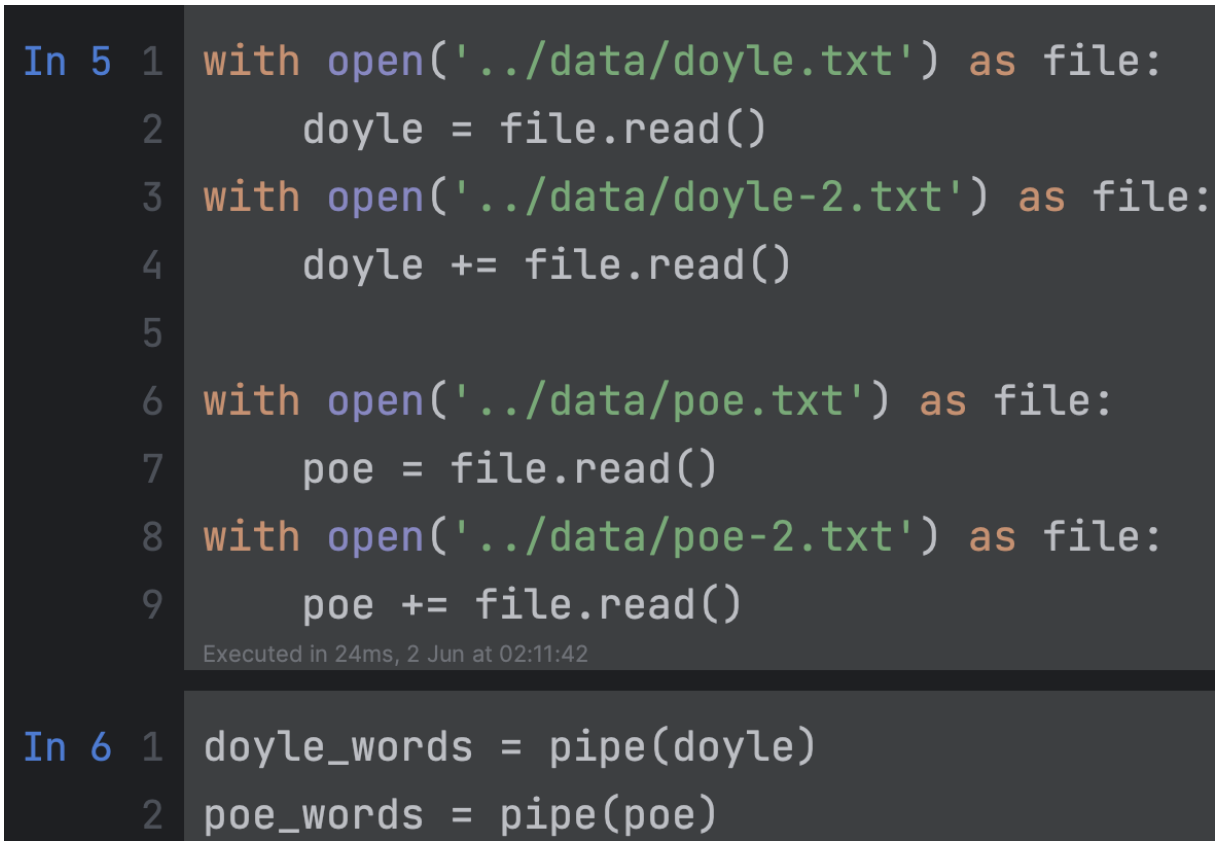


Рисунок 4.4 – Завантаження даних та їх попередня обробка

Тепер виведемо 30 найбільш вживаних слів для кожного автора у вигляді горизонтальної стовпчастої діаграми.

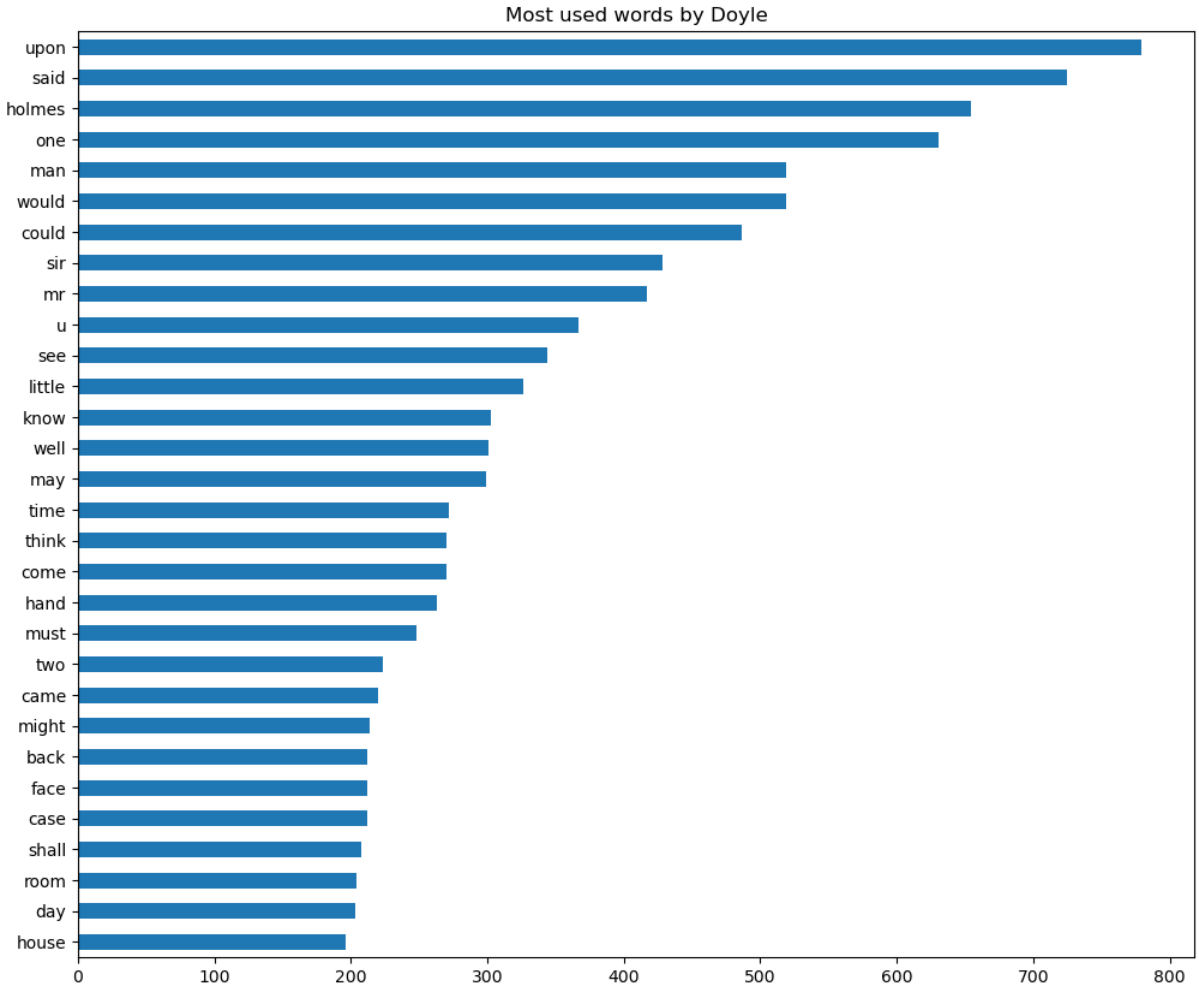


Рисунок 4.5 – 30 найбільш вживаних слів Дойла

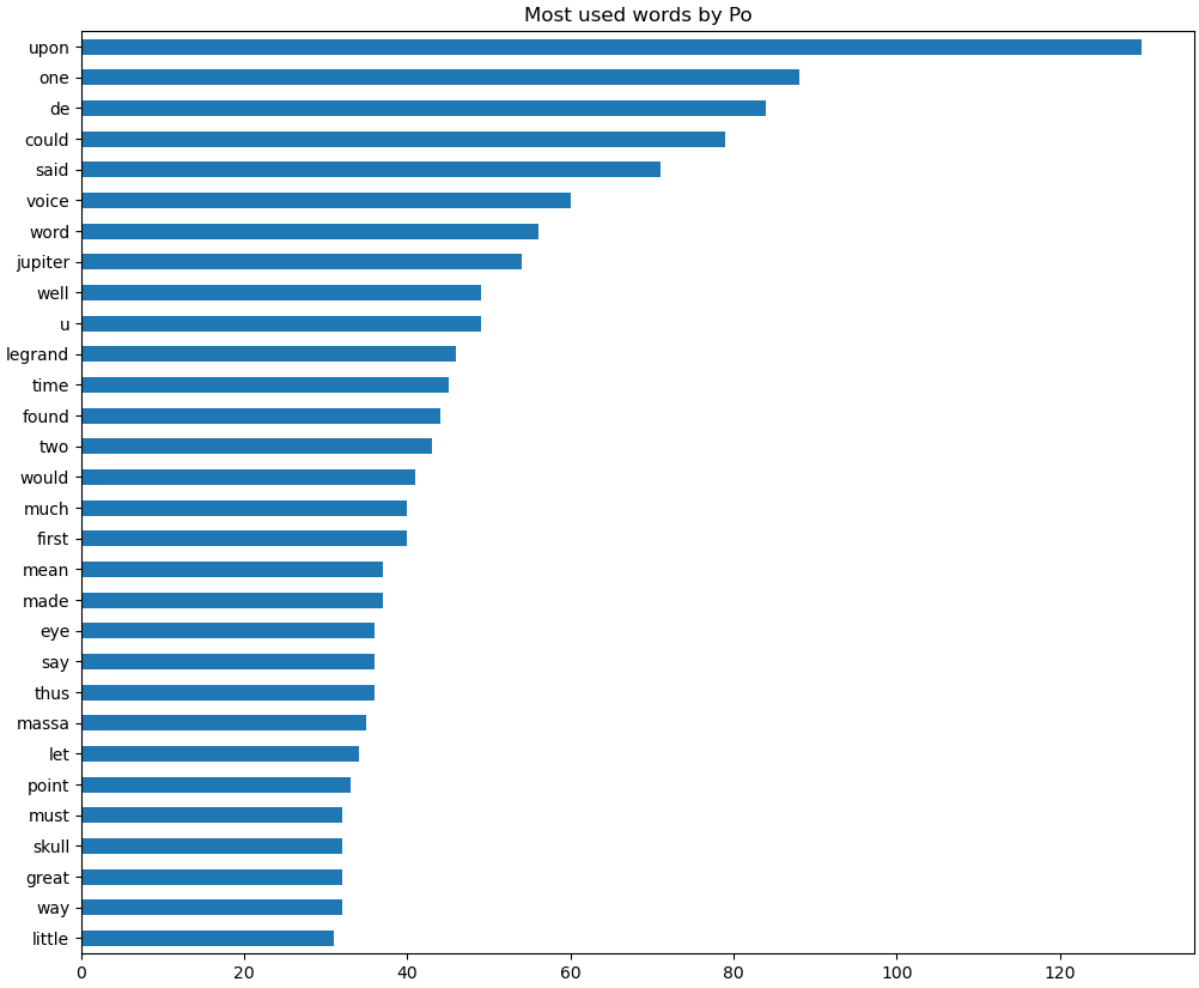


Рисунок 4.6 – 30 найбільш вживаних слів По

Можемо побачити, що тут присутні слова, які не несуть особливого смислового навантаження, тому їх також варто віднести до стоп-слів та видалити.

```
In 9  1 extra_stopwords = {'upon', 'could', 'would', 'us', 'u', 'man', 'mr', 'de', 'may', 'must', 'thus', 'say',
        'much'}
      2 remove_extra_stopwords = get_stopwords_remover(extra_stopwords)
        Executed in 2ms, 2 Jun at 02:11:44

In 10 1 doyle_words = remove_extra_stopwords(doyle_words)
      2 poe_words = remove_extra_stopwords(poe_words)
```

Рисунок 4.7 – Видалення нових стоп-слів для обох авторів

Тепер побудуємо хмари слів для обох авторів із заданням параметрів ширини, висоти кольором фону.

```
In 11  1  wordcloud = WordCloud(
       2      background_color='white',
       3      width=1000, height=1000
       4  )
```

Рисунок 4.8 – Визначення хмари слів з параметрами розміру та кольором

Рисунок 4.9 – Хмара слів для Дойла

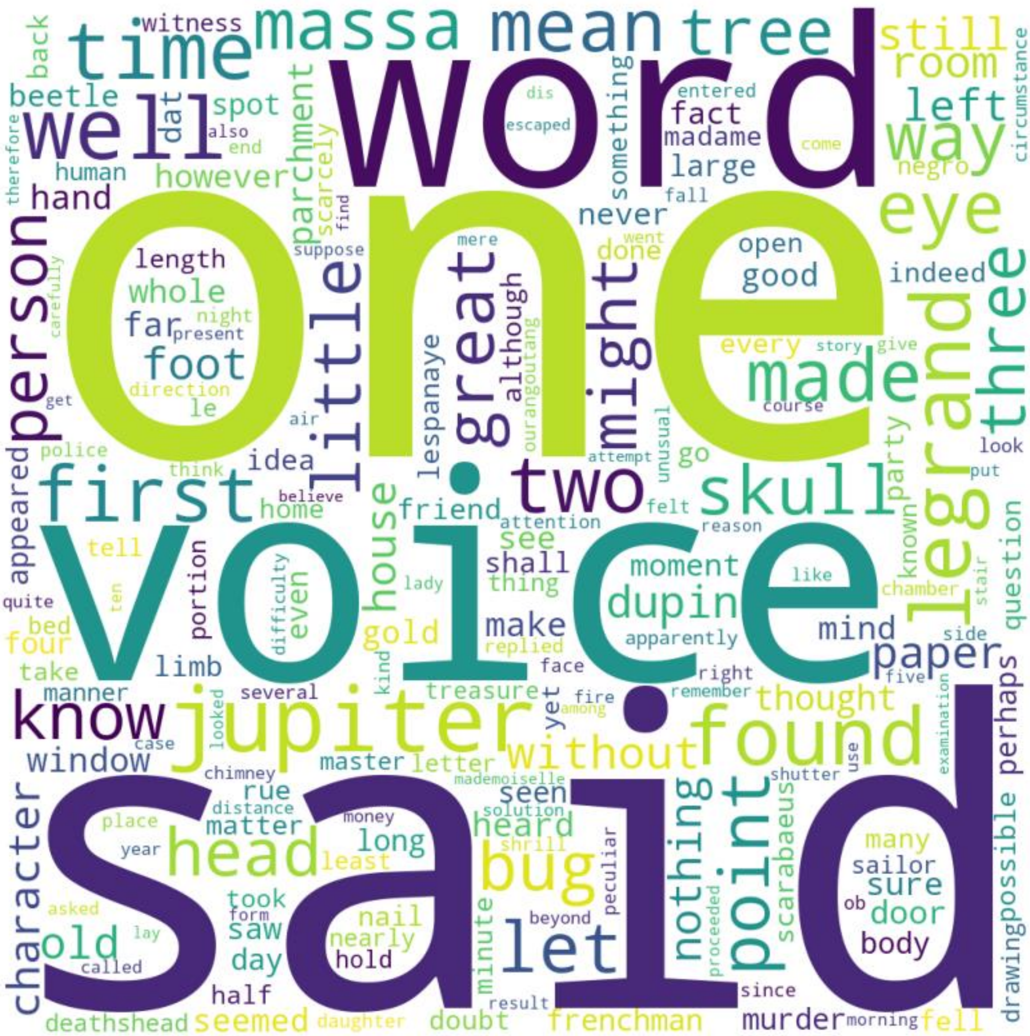


Рисунок 4.10 – Хмара слів для По

Наступним кроком обчислимо показник похмурості для обох авторів ітеративним шляхом. Для кожного слова кожного автора визначимо його частину мови та за допомогою wordnet визначимо тональність цього. Суму значень тональностей поділимо на кількість слів з метою збалансування отриманої оцінки.

```
def get_wordnet_pos(w):
    treebank_tag = nltk.pos_tag([w])[0][1]
    if treebank_tag.startswith('J'):
        return 'a'
    elif treebank_tag.startswith('V'):
        return 'v'
    elif treebank_tag.startswith('N'):
        return 'n'
    elif treebank_tag.startswith('R'):
        return 'r'
    else:
        return 'n'
```

Рисунок 4.11 – Функція для визначення частини мови слова

```
In 15
doyle_score = 0
poe_score = 0
doyle_count = 0
poe_count = 0

for word, count in list(doyle_words.value_counts().items()):
    synsets = list(swn.senti_synsets(word, get_wordnet_pos(word)))
    if synsets:
        score = synsets[0].pos_score() - synsets[0].neg_score()
        doyle_score += score * count
        doyle_count += count

for word, count in list(poe_words.value_counts().items()):
    synsets = list(swn.senti_synsets(word, get_wordnet_pos(word)))
    if synsets:
        score = synsets[0].pos_score() - synsets[0].neg_score()
        poe_score += score * count
        poe_count += count

doyle_score /= len(doyle_words)
poe_score /= len(poe_words)

print(f'Doyle score: {doyle_score}, Poe score: {poe_score}')
Executed in 2s, 2 Jun at 02:11:48

Doyle score: 0.0028875187754810096, Poe score: 0.0028881887219259976
```

Рисунок 4.12 – Обчислення показника похмурості для кожного автора

```
In 16
author = 'Poe' if poe_score < doyle_score else 'Doyle'
print(f'{author} wrote more gloomy novels')
Executed in 1ms, 2 Jun at 02:11:48

Doyle wrote more gloomy novels
```

Рисунок 4.13 – Визначення автора з більш похмурими оповіданнями (Дойл)

5. ВИСНОВОК

Отже, при виконанні даної лабораторної було обробленом текстові дані з різних джерел на різних мовах, створено торбу слів для текстових даних з телеграм каналу, обчислено показники метрики TF-IDF для різних даних, обчислено тональності для різних новин, категоризовано текстові дані з використанням методу LSA, а також побудовано хмари слів для авторів Артура Конан Дойла й Едгара Алана По. Крім того, я дослідив оповідання цих авторів на похмурість та в результаті отримав, що Дойл писав більш похмурі оповідання, хоча варто також зазначити, що різниця між показниками для цих авторів досить невелика.