**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**

**Звіт**

з лабораторної роботи №5 з дисципліни

«Обробка та аналіз текстових даних на Python»

„**Моделювання тем**”

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-11 Головня Олександр Ростиславович*

**Перевірив**

(прізвище, ім'я, по батькові)

Київ 2024

**Мета роботи:** Ознайомитись з вирішенням задач пошуку ключових слів та

моделювання тем.

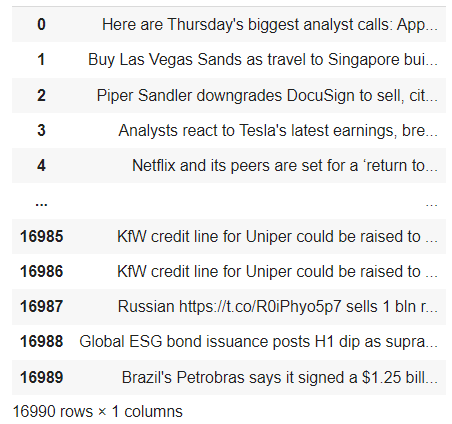
**Завдання до лабораторної роботи:**

Створити програму, яка виконує завдання відповідно до варіанту.

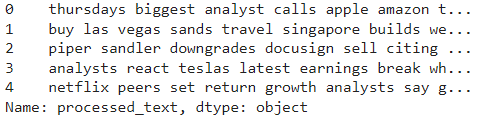
1.Застосувати приховане семантичне індексування бібліотеки scikit-learn для моделювання тем. Вивести документи, що описують кожну з тем. Вивести найбільш важливі теми для випадково обраних чотирьох документів.  
2. Використати текст carroll-alice.txt з корпусу gutenberg бібліотеки nltk та вивести ключові біграми.

**Результат виконання коду:**

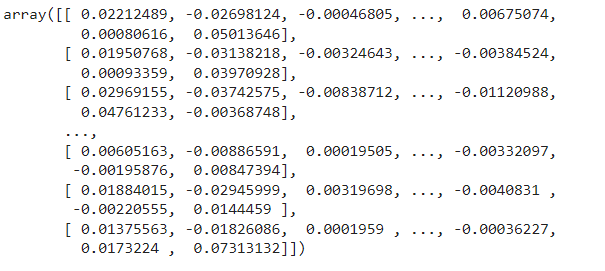
**Завантаження файлу у датафрейм**



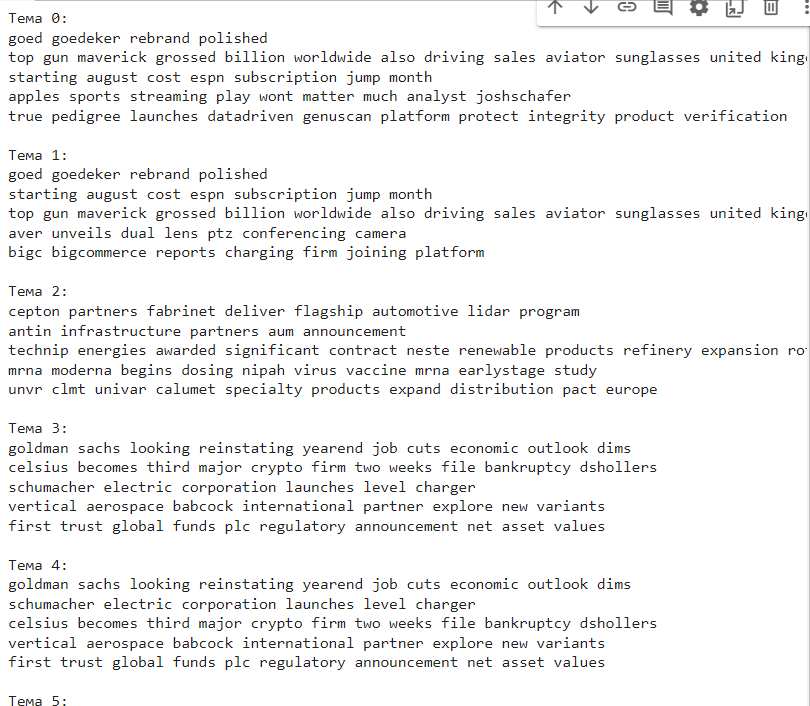
**Попередня обробка**



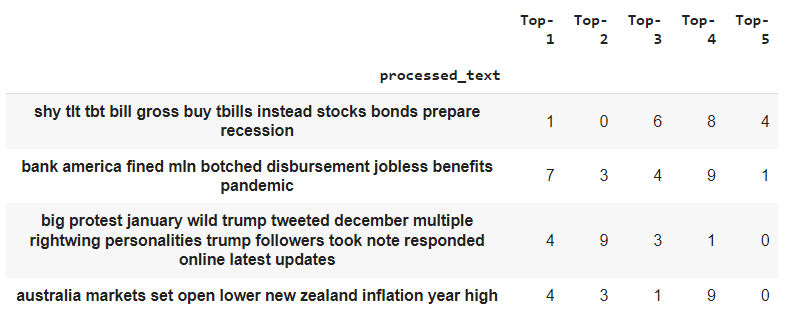
**Використання моделі TF-IDF та застосування прихованого семантичного індексування**



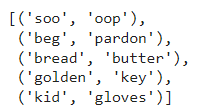
**Вивід документів, що описують кожну з тем**



**Найбільш важливі теми для випадково обраних чотирьох документів**



**Використати текст carroll-alice.txt з корпусу gutenberg бібліотеки nltk та вивести ключові біграми.**



**Код програми:**

import pandas as pd

news\_corpus = pd.read\_csv('news.csv', sep='\t')

news\_corpus

from nltk.tokenize import WordPunctTokenizer

from nltk.corpus import stopwords

import re

import nltk

nltk.download('stopwords')

stop\_words = set(stopwords.words('english'))

tokenizer = WordPunctTokenizer()

def preprocess\_document(doc):

    doc = re.sub(r'https?:\/\/\S+', '', doc)  # Видаляємо посилання

    doc = re.sub(r'[^a-zA-Z\s]', '', doc, re.I | re.A)

    doc = doc.lower()

    doc = doc.strip()

    tokens = tokenizer.tokenize(doc)

    filtered\_tokens = [token for token in tokens if token not in stop\_words and len(token) > 2]

    doc = ' '.join(filtered\_tokens)

    return doc

# Застосовуємо функцію до всіх документів у корпусі

news\_corpus['processed\_text'] = news\_corpus['text,label'].apply(preprocess\_document)

news\_corpus['processed\_text'].head()

from sklearn.feature\_extraction.text import TfidfVectorizer

import numpy as np

cv = TfidfVectorizer(min\_df=10, max\_df=0.8, ngram\_range=(1, 2))

cv\_features = cv.fit\_transform(news\_corpus['processed\_text'])

vocabulary = np.array(cv.get\_feature\_names\_out())

from sklearn.decomposition import TruncatedSVD

n\_topics = 10

lsi\_model = TruncatedSVD(n\_components=n\_topics, random\_state=1234)

document\_topics = lsi\_model.fit\_transform(cv\_features)

document\_topics

topic\_docs = {}

for i, topic in enumerate(lsi\_model.components\_):

    top\_docs\_idx = np.argsort(topic)[::-1][:5]  # Вибираємо топ-5 документів для кожної теми

    topic\_docs[i] = top\_docs\_idx

for topic, docs\_idx in topic\_docs.items():

    print(f"Тема {topic}:")

    for doc\_idx in docs\_idx:

        print(news\_corpus.iloc[doc\_idx]['processed\_text'])

    print()

top\_topics = 5

documents\_indices = np.random.choice(document\_topics.shape[0], size=4, replace=False)

random\_documents = document\_topics[documents\_indices]

most\_important\_topics = np.argsort(np.abs(random\_documents), axis=1)[:, ::-1][:, :top\_topics]

most\_important\_topics = pd.DataFrame(most\_important\_topics, index=[news\_corpus['processed\_text'][documents\_indices]],

                                     columns=[f'Top-{i + 1}' for i in range(top\_topics)])

most\_important\_topics

from nltk.corpus import gutenberg

nltk.download('gutenberg')

nltk.download('punkt')

alice\_corpus = gutenberg.sents('carroll-alice.txt')

def preprocess\_sentence(sentence):

    doc = ' '.join(sentence)

    doc = re.sub(r'[^a-zA-Z\s]', '', doc, re.I | re.A)

    doc = doc.lower()

    doc = doc.strip()

    tokens = tokenizer.tokenize(doc)

    filtered\_tokens = [token for token in tokens if token not in stop\_words and len(token) > 2]

    return filtered\_tokens

alice\_corpus = [preprocess\_sentence(sentence) for sentence in alice\_corpus]

print(alice\_corpus[:5])

from nltk.collocations import BigramCollocationFinder, BigramAssocMeasures

bigram\_measures = BigramAssocMeasures()

finder = BigramCollocationFinder.from\_documents(alice\_corpus)

finder.apply\_freq\_filter(5)

finder.nbest(bigram\_measures.pmi, 5)

**Висновок:** Під час розробки лабораторної роботи, я ознайомився на практиці з вирішенням задач пошуку ключових слів та моделювання тем.