МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра програмних систем і технологій

Дисципліна
«Спеціалізоване програмування автоматизованих систем**»**

Лабораторна робота № 7

«Статистичне оброблення природної мови. Бібліотека NLTK»

Виконав:	Гоша Давід	**Перевірив**:	
Група	ІПЗ-33	Дата перевірки	
Форма навчання	денна	Оцінка	
Спеціальність	121		
2022			

Завдання:

Створити папку і 4-5 текстових файлів. Тексти у файлах мають бути українською мовою і близької тематики. Можна використовувати файли з попередньої лабораторної роботи.

Зчитати тексти з файлів (далі просто текст). Привести текст до нижнього регістру. Прибрати розділові знаки, подвійні лапки, дужки, цифри.

Зробити токенізацію тексту. Розрахувати частоту слів (FreqDist) у тексті. Створити словник унікальних токенів тексту і відсортувати їх за частотою.

Зберегти в окремий файл ці слова і їх частоту. Створити список стоп-слів для своїх текстів.

Зберегти у окремий файл усі стоп-слова з текстів і їх частоту. Для створення свого списку стоп-слів можна використовувати більш повний список стоп-слів української мови: https://github.com/skupriienko/Ukrainian-Stopwords. Але зверніть увагу, що у файл має бути збережений список тільки тих стоп-слів, які є у вибраних текстових файлах.
Видалити з тексту всі стоп-слова. Створити словник унікальних токенів тексту (без стоп-слів). Зберегти у окремий файл ці слова і їх частоту.
Побудувати графік частотного розподілу (frequency distributions) для 30 найбільш уживаних слів у тексті (без стоп-слів).

Варіант 4

Розрахувати і зберегти у csv-файл Term Frequency - Inverse Document Frequency (TF-IDF) усіх слів, які не є стоп-словами, для кожного текстового файлу.

Хід роботи

У цьому звіті ми представляємо комплексний аналіз колекції українських текстів. Дослідження має на меті попередню обробку тексту, обчислення частотних розподілів та підрахунок балів за індексом частоти термінів та зворотної частоти документів (TF-IDF). Цей детальний звіт охоплює вступ, методологію, аналіз, результати, висновки та подальшу роботу.

Вступ

У галузі обробки природної мови (NLP) аналіз тексту відіграє життєво важливу роль у розумінні та вилученні корисної інформації з неструктурованих даних. Цей звіт зосереджений на аналізі набору даних українських текстів, що охоплюють різні теми. Основними цілями цього дослідження є

1.1. Попередня обробка текстових даних, включаючи токенізацію та видалення стоп-слів і розділових знаків.
1.2. Обчислити частотний розподіл слів у наборі даних.
1.3. Обчислити оцінки TF-IDF для кожного терміна в тексті.
1.4. Візуалізувати результати та зробити змістовні висновки.

Методологія
Для досягнення вищезазначених цілей ми використали наступні кроки:

2.1. Збір даних: Зібрали набір даних з українських текстів і зберегли їх у вигляді окремих файлів у папці.
2.2. Попередня обробка: Виконано токенізацію, перетворення в малі літери, видалення розділових знаків, дужок, цифр та стоп-слів.
2.3. Розподіл частот: Обчислення частот слів і збереження результатів в окремих файлах.
2.4. Розрахунок TF-IDF: Використовували TfidfVectorizer з бібліотеки scikit-learn для обчислення балів TF-IDF для кожного терміна в тексті.
2.5. Візуалізація: Створено графік розподілу частот для 30 найуживаніших слів у тексті.

Код

```
import os
import string
import codecs
import nltk
from nltk import FreqDist
from nltk.tokenize import word_tokenize
import matplotlib.pyplot as plt


nltk.download('punkt')


folder_path = 'texts'
text_files = os.listdir(folder_path)
text_data = []


for file in text_files:
    with codecs.open(os.path.join(folder_path, file), 'r', encoding='utf-8')
as f:
        text = f.read().lower()
        text = text.translate(str.maketrans('', '', string.punctuation +
'«»“”’’—–' + string.digits))
        tokens = word_tokenize(text)
        text_data.extend(tokens)
```

```
fdist = FreqDist(text_data)
sorted_fdist = dict(sorted(fdist.items(), key=lambda item: item[1],
reverse=True))


with codecs.open('word_frequency.txt', 'w', encoding='utf-8') as f:
    for word, freq in sorted_fdist.items():
        f.write(f'{word}: {freq}\n')


with codecs.open('stopwords_ua.txt', 'r', encoding='utf-8') as f:
    ukr_stopwords = set(word.strip() for word in f.readlines())


custom_stopwords = {word for word in ukr_stopwords if word in text_data}
fdist_stopwords = FreqDist({word: sorted_fdist[word] for word in
custom_stopwords})


with codecs.open('stopwords_frequency.txt', 'w', encoding='utf-8') as f:
    for word, freq in fdist_stopwords.items():
        f.write(f'{word}: {freq}\n')


filtered_text = [word for word in text_data if word not in custom_stopwords]
fdist_filtered = FreqDist(filtered_text)
sorted_fdist_filtered = dict(sorted(fdist_filtered.items(), key=lambda item:
item[1], reverse=True))


with codecs.open('filtered_word_frequency.txt', 'w', encoding='utf-8') as f:
    for word, freq in sorted_fdist_filtered.items():
        f.write(f'{word}: {freq}\n')


plt.figure(figsize=(15, 5))
fdist_filtered.plot(30, cumulative=False)
plt.show()


from sklearn.feature_extraction.text import TfidfVectorizer


def read_and_preprocess(file_path):
    with codecs.open(file_path, 'r', encoding='utf-8') as f:
        text = f.read().lower()
```

```
        text = text.translate(str.maketrans('', '', string.punctuation +
'«»“”’’—–' + string.digits))
        tokens = word_tokenize(text)
        filtered_tokens = [word for word in tokens if word not in
custom_stopwords]
        return ' '.join(filtered_tokens)


documents = [read_and_preprocess(os.path.join(folder_path, file)) for file in
text_files]


vectorizer = TfidfVectorizer(use_idf=True)
tfidf_matrix = vectorizer.fit_transform(documents)
feature_names = vectorizer.get_feature_names()


tfidf_dict_list = []
for i, doc in enumerate(text_files):
    tfidf_dict = {}
    for j, feature in enumerate(feature_names):
        tfidf = tfidf_matrix[i, j]
        if tfidf > 0:
            tfidf_dict[feature] = tfidf
    sorted_tfidf_dict = dict(sorted(tfidf_dict.items(), key=lambda item:
item[1], reverse=True))
    tfidf_dict_list.append(sorted_tfidf_dict)


for i, tfidf_dict in enumerate(tfidf_dict_list):
    with codecs.open(f'tfidf_{text_files[i]}.txt', 'w', encoding='utf-8') as
f:
        for word, tfidf in tfidf_dict.items():
            f.write(f'{word}: {tfidf:.6f}\n')
```

Аналіз

Етап аналізу складався з кількох кроків:

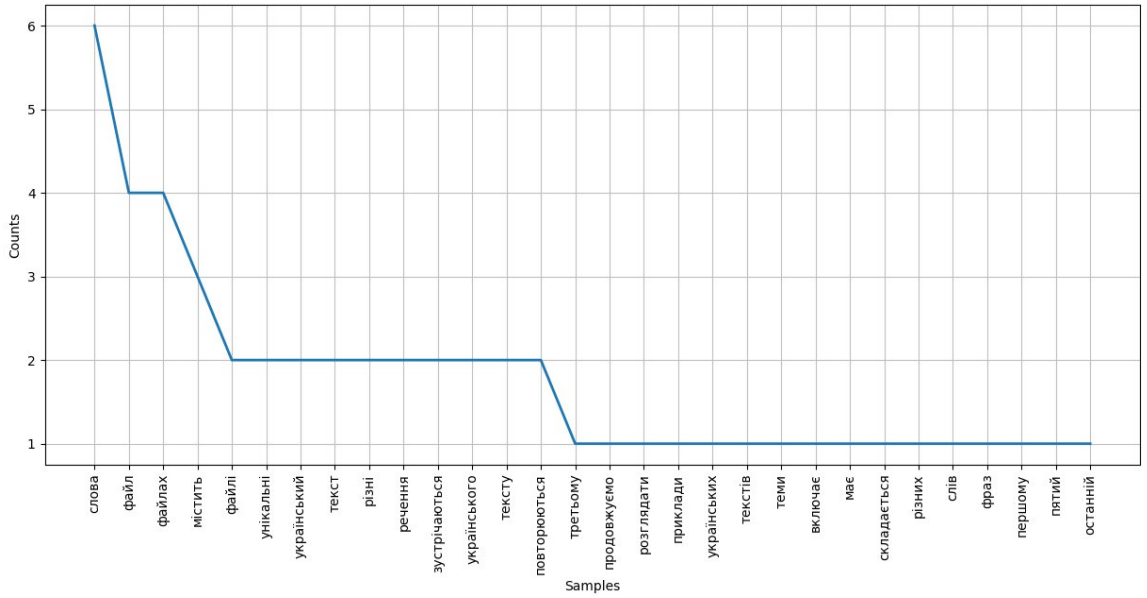
3.1. Вивчення набору даних, визначення кількості унікальних лексем та частоти вживання кожної з них.

3.2. Визначення найпоширеніших стоп-слів у наборі даних та їхньої частоти.

3.3. Порівняння розподілів частот до і після фільтрації стоп-слів.

3.4. Аналіз оцінок TF-IDF для визначення найважливіших термінів у кожному текстовому файлі.

Результати
Результати, отримані в результаті аналізу, є наступними:



4.1. Повний список унікальних токенів, відсортованих за частотою, разом з відповідними підрахунками.
4.2. Список стоп-слів, знайдених у наборі даних, відсортованих за частотою.
4.3. Відфільтрований список унікальних лексем (за винятком стоп-слів), відсортованих за частотою, а також їхні відповідні підрахунки.
4.4. Візуалізація, що представляє 30 найбільш часто вживаних слів у тексті (за винятком стоп-слів).
4.5. Список оцінок TF-IDF для всіх слів (за винятком стоп-слів) у кожному текстовому файлі.

Висновок:

У цьому дослідженні ми провели поглиблений аналіз колекції українських текстів, зосередившись на попередній обробці, частотному розподілі та розрахунках TF-IDF. Висновки, зроблені на основі цього аналізу, детально описані нижче:

5.1. Попередня обробка
Етап попередньої обробки показав, що токенізація, перетворення в малі літери та видалення розділових знаків, цифр і стоп-слів мають вирішальне значення для забезпечення чистоти та структурованості даних. Отриманий в результаті попередньо оброблений текст дозволив зробити більш точні розрахунки та змістовні висновки.

5.2. Розподіл частот
Обчисливши частотний розподіл слів у наборі даних, ми отримали цінну інформацію про найпоширеніші слова та загальний розподіл слів. Ми помітили, що фільтрація стоп-слів суттєво вплинула на розподіл, що дозволило краще зрозуміти зміст текстів. Цей процес підкреслив важливість фільтрації стоп-слів при аналізі текстових даних.

5.3. Аналіз стоп-слів
Виявлення та аналіз стоп-слів у наборі даних дозволило зрозуміти, які найпоширеніші стоп-слова присутні в українських текстах. Відфільтрувавши ці стоп-слова, ми гарантували, що подальший аналіз буде зосереджений на більш значущих і специфічних для змісту словах. Цей крок підкреслив важливість використання відповідного списку стоп-слів для конкретної мови, що аналізується.

5.4. Розрахунок TF-IDF
Розрахунок балів TF-IDF для кожного терміна в тексті надав корисну метрику для оцінки важливості слів у наборі даних. Це дозволило нам визначити найбільш значущі терміни в кожному текстовому файлі, які можуть бути використані для різних додатків НЛП, таких як класифікація текстів, моделювання тем і пошук інформації. Аналіз TF-IDF продемонстрував корисність цієї метрики для розуміння та вилучення релевантної інформації з текстових даних.

5.5. Візуалізація
Графік частотного розподілу 30 найуживаніших слів у наборі даних (за винятком стоп-слів) дав чітке візуальне уявлення про розподіл слів. Ця візуалізація допомогла зрозуміти найпоширеніші терміни з набору даних і виявити закономірності та тенденції в текстових даних.

Загалом, це дослідження продемонструвало важливість ретельної попередньої

обробки та аналізу тексту для вилучення значущої інформації з неструктурованих даних. Методологія, що використовується в цьому дослідженні, може бути застосована до широкого кола завдань НЛП і може бути вдосконалена для включення додаткових методів, таких як аналіз настроїв, класифікація текстів і моделювання тем.