МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

**ПРАКТИЧНЕ ЗАНЯТТЯ № 3 ЗА ТЕМОЮ:**

**Базові засади аналізу текстів систем рекомендацій контенту**

МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗВІТ З ПРАКТИЧНОЇ РОБОТИ № 3 ЗА ТЕМОЮ:

Базові засади аналізу текстів систем рекомендацій контенту

Група \_\_\_\_\_11\_\_\_\_\_\_

Курс \_\_\_\_\_1\_\_\_\_\_\_\_\_\_

Студент (ка) \_\_\_\_Супруненко М. І. \_\_\_\_

Дата оформлення \_\_\_\_\_\_28.02.2025\_\_\_\_\_\_\_

Перевірив \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

МЕТА: Метою практичної роботи є формування професійних вмінь та навичок щодо базових засад аналізу текстів, вміння застосовувати отримані знання на практиці в практичних задачах інтелектуального аналізу даних

ОСНАЩЕННЯ:

ТЕОРІЯ: Оброблені (tidy) «акуратні» дані мають певну структуру:

* Кожна змінна є стовпцем.
* Кожне спостереження є рядком.
* Кожен вид одиниці спостереження являє собою таблицю.

Таким чином, ми визначаємо оброблений текстовий формат як таблицю з одним маркером (токеном) на рядок (**a table with one-token-per-row**).

Токен — це значуща одиниця тексту, наприклад слово, яку ми хочемо використовувати для аналізу, а токенізація — це процес поділу тексту на токени. Ця структура «один маркер на рядок» залежить від того, як текст часто зберігається в поточному аналізі, можливо, у вигляді рядків або в матриці термінів документа.

Для акуратного видобутку (mining) тексту токен, який зберігається в кожному рядку, найчастіше є окремим словом, але також може бути n-грамою, реченням або абзацом. У пакеті «tidytext» ми реалізовуємо функціональність для токенізації часто використовуваних одиниць тексту, і перетворюємо у формат «один терм на рядок».

Оброблені (tidy) набори даних дозволяють маніпулювати стандартним набором «tidy» інструментів, включаючи такі популярні пакети, як dplyr (Wickham and Francois 2016), tidyr (Wickham 2016), ggplot2 (Wickham 2009) і broom (Robinson 2017).

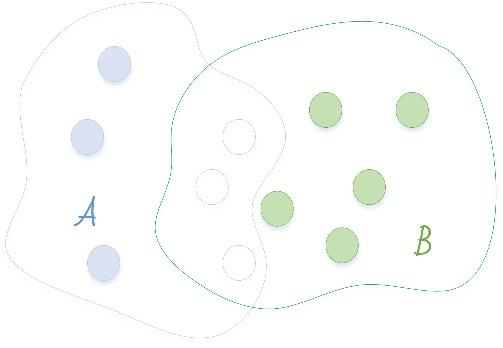
Структурування текстових даних таким чином означає, що вони відповідають принципам акуратності даних і ними можна маніпулювати за допомогою набору узгоджених інструментів.

* **Рядок (String)**: Текст можна, звичайно, зберігати як рядки, тобто вектори символів, у R, і часто текстові дані спочатку зчитуються в пам’ять у цій формі.
* **Корпус (Corpus)**: ці типи об’єктів зазвичай містять необроблені рядки, анотовані додатковими метаданими та деталями.
* **Матриця документ-термін** (**Document-term matrix**): це розріджена матриця, що описує набір (тобто корпус) документів з одним рядком для кожного документа та одним стовпцем для кожного терміна. Значення в матриці зазвичай є кількістю слів або tf-idf.

Подібність Жаккара множин A та B визначається таким рівнянням:

𝑆𝐼𝑀(𝐴,𝐵)=|𝐴∩𝐵|/|𝐴∪𝐵|, її можна позначити SIM(A,B).

Подібність Жаккара описує відношення розміру перетинів множин A та B до розміру їх об'єднання.

Подібність між двома документами можна виміряти, обчисливши косинус подібності між їхніми векторними представленнями. 

Документи можна представити як набір векторів у векторному просторі, де кожному терміну відповідає одна вісь. Для кожного слова в документі використовується хеш-функція, яка зіставляє його з цілочисельним індексом. Після цього документ можна перетворити на набір векторів, що представляють слова в документі. Векторне представлення документів не містить порядку інформації - воно відображає, скільки разів слово з'являється в документі. При цьому втрачається порядок слів. У більшості випадків векторного представлення достатньо, щоб передати зміст документа, незважаючи на те, що порядок інформації втрачено.

Якщо отримано косинусоїдальну подібність +1, два досліджувані документи можна вважати однаковими. Однак, зверніть увагу, що ці два документи не є фактично однаковими, але схожими на основі обраних термінів. Косинусоїдальна подібність -1 вказує на два вектори, які є діаметрально протилежними, тобто документи за абсолютно різною тематикою.

ПЛАН ПРОВЕДЕННЯ ПРАКТИЧНОГО ЗАНЯТТЯ:

1. Відповіді на вхідні питання.
2. Проведення досліджень невеликих текстів українських класиків. Виявлення схожості текстів, підрахунок частоти використання слів.
3. Підготовка звіту за результатами дослідження.
4. Підготовка домашнього завдання.
5. Захист домашнього завдання.

ПИТАННЯ ДЛЯ ВХІДНОГО КОНТРОЛЮ СТУДЕНТІВ:

1. Важливість слова. Показники Inverse Document Frequency, IDF та term frequency.
2. Хеш-функції. Хеш-функції.
3. Індекси.
4. Що таке схожі сутності і як їх ідентифікувати?
5. Подібність документів.
6. Представлення документів у вигляді наборів символів.
7. Побудова shingle наборів на основі стоп слів.
8. Матричне представлення множин.
9. MinHash та сігнатурна матриці.
10. Матриця рейтингів (вподобань).
11. Системи Спільної фільтрації.
12. Системи рекомендацій на основі змісту.
13. Ефективність рекомендаційних систем.

ХІД ПРОВЕДЕННЯ ПРАКТИЧНОГО ЗАНЯТТЯ:

1. Завантажте в середовище аналізу два невеликих твори українських письменників, наприклад, Л.Українка та Т.Шевченко. Можна виконувати завантаження з ресурсів, а також можливо створити окрему змінну, в яку завантажити рядки наприклад віршів, на більше 100 строк.

# To be or not to be?.. Л.Українка

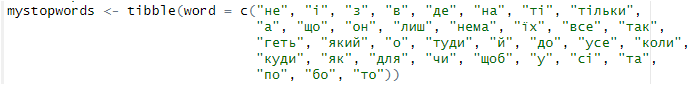
Text\_1 <- c("Стій, серце, стій! не бийся так шалено.",

"Вгамуйся, думко, не літай так буйно!",

"Не бий крильми в порожньому просторі.",

" …………….")

1. Завантажте необхідні бібліотеки.
2. Для перетворення вихідного тексту в акуратний (tidy) набір даних (data set), його необхідно помістити у фрейм даних (data frame) за допомогою функції tibble().
3. Здійсніть токенізацію завантажених текстів за допомогою функції unnest\_tokens().
4. Проаналізуйте завантажені тексти та створіть власний список стоп слів, наприклад,



1. Очистіть завантажені та оброблені (токенізовані) тексти від стоп слів з використанням функції anti\_join ()
2. Порахуйте частоту вживання кожного слова та побудуйте діаграми.
3. Порахуйте показник **term frequency** кожного слова (токену):

- кількість разів зустрічі слова в документі ;

- максимальна кількість разів зустрічі будь-якого слова в документі .

1. Побудуйте функцію для розрахунку подібності Жаккара двох текстів:
2. Порівняйте схожість завантажених вами текстів.

ДОМАШНЄ ЗАВДАННЯ: Оформлення звіту та підготовка до захисту.

**Практична робота 1**

**1. Завантаження необхідних бібліотек**

|  |
| --- |
| # Встановлення бібліотек, якщо вони ще не встановлені (розкоментуйте при необхідності)  #!pip install pandas re matplotlib seaborn scikit-learn  import pandas as pd  import re  import matplotlib.pyplot as plt  import seaborn as sns  from nltk.corpus import stopwords  from nltk.tokenize import word\_tokenize  from sklearn.feature\_extraction.text import CountVectorizer, TfidfVectorizer  from collections import Counter  # Власний набір стоп-слів для української мови  ukrainian\_stopwords = set([      "я", "ти", "він", "вона", "ми", "ви", "вони", "це", "і", "на", "з", "до", "від", "по",      "для", "що", "як", "коли", "де", "не", "вже", "так", "був", "була", "було", "буде", "будуть"  ]) |

1. **Завантаження текстів**

|  |
| --- |
| # Завантажуємо невеликі уривки з творів Лесі Українки та Тараса Шевченка  text\_1 = """Стій, серце, стій! не бийся так шалено.  Вгамуйся, думко, не літай так буйно!  Не бий крильми в порожньому просторі."""  text\_2 = """Як умру, то поховайте мене на могилі.  Серед степу широкого, на Вкраїні милій.  Щоб лани широкополі, і Дніпро, і кручі,  Було видно, було чути, як реве ревучий.""" |

**3. Перетворення текстів у формат DataFrame**

|  |
| --- |
| df = pd.DataFrame({"author": ["Леся Українка"]\*3 + ["Тарас Шевченко"]\*4,                     "text": text\_1.split("\n") + text\_2.split("\n")})  df |
|  |

**4. Токенізація тексту**

|  |
| --- |
| # Простий метод токенізації з використанням регулярних виразів  def tokenize(text):      # Перетворюємо текст в нижній регістр      text = text.lower()      # Використовуємо регулярний вираз для розбиття тексту на слова      tokens = re.findall(r'\b\w+\b', text)  # Витягнення слів (не враховує розділові знаки)      # Видаляємо стоп-слова      tokens = [word for word in tokens if word not in ukrainian\_stopwords]      return tokens  # Токенізація тексту  df["tokens"] = df["text"].apply(lambda x: tokenize(x))  # Якщо є порожні значення, заміняємо їх на порожні списки  df["tokens"] = df["tokens"].apply(lambda x: [] if x is None else x)  # Використовуємо explode для розгортання списку токенів  df\_exploded = df.explode("tokens").reset\_index(drop=True)  # Переглянути результат  df\_exploded |
| |  | **author** | **text** | **tokens** | | --- | --- | --- | --- | | 0 | Леся Українка | Стій, серце, стій! не бийся так шалено. | стій | | 1 | Леся Українка | Стій, серце, стій! не бийся так шалено. | серце | | 2 | Леся Українка | Стій, серце, стій! не бийся так шалено. | стій | | 3 | Леся Українка | Стій, серце, стій! не бийся так шалено. | бийся | | 4 | Леся Українка | Стій, серце, стій! не бийся так шалено. | шалено | | 5 | Леся Українка | Вгамуйся, думко, не літай так буйно! | вгамуйся | | 6 | Леся Українка | Вгамуйся, думко, не літай так буйно! | думко | | 7 | Леся Українка | Вгамуйся, думко, не літай так буйно! | літай | | 8 | Леся Українка | Вгамуйся, думко, не літай так буйно! | буйно | | 9 | Леся Українка | Не бий крильми в порожньому просторі. | бий | | 10 | Леся Українка | Не бий крильми в порожньому просторі. | крильми | | 11 | Леся Українка | Не бий крильми в порожньому просторі. | в | | 12 | Леся Українка | Не бий крильми в порожньому просторі. | порожньому | | 13 | Леся Українка | Не бий крильми в порожньому просторі. | просторі | | 14 | Тарас Шевченко | Як умру, то поховайте мене на могилі. | умру | | 15 | Тарас Шевченко | Як умру, то поховайте мене на могилі. | то | | 16 | Тарас Шевченко | Як умру, то поховайте мене на могилі. | поховайте | | 17 | Тарас Шевченко | Як умру, то поховайте мене на могилі. | мене | | 18 | Тарас Шевченко | Як умру, то поховайте мене на могилі. | могилі | | 19 | Тарас Шевченко | Серед степу широкого, на Вкраїні милій. | серед | | 20 | Тарас Шевченко | Серед степу широкого, на Вкраїні милій. | степу | | 21 | Тарас Шевченко | Серед степу широкого, на Вкраїні милій. | широкого | | 22 | Тарас Шевченко | Серед степу широкого, на Вкраїні милій. | вкраїні | | 23 | Тарас Шевченко | Серед степу широкого, на Вкраїні милій. | милій | | 24 | Тарас Шевченко | Щоб лани широкополі, і Дніпро, і кручі, | щоб | | 25 | Тарас Шевченко | Щоб лани широкополі, і Дніпро, і кручі, | лани | | 26 | Тарас Шевченко | Щоб лани широкополі, і Дніпро, і кручі, | широкополі | | 27 | Тарас Шевченко | Щоб лани широкополі, і Дніпро, і кручі, | дніпро | | 28 | Тарас Шевченко | Щоб лани широкополі, і Дніпро, і кручі, | кручі | | 29 | Тарас Шевченко | Було видно, було чути, як реве ревучий. | видно | | 30 | Тарас Шевченко | Було видно, було чути, як реве ревучий. | чути | | 31 | Тарас Шевченко | Було видно, було чути, як реве ревучий. | реве | | 32 | Тарас Шевченко | Було видно, було чути, як реве ревучий. | ревучий | |
| # Перевірка на те, що після tokenization кожен елемент є списком  df["tokens"] = df["text"].apply(lambda x: tokenize(x) if isinstance(tokenize(x), list) else [])  # Якщо є порожні значення, заміняємо їх на порожні списки  df["tokens"] = df["tokens"].apply(lambda x: [] if x is None else x)  # Використовуємо explode  df\_exploded = df.explode("tokens").reset\_index(drop=True) |

**5. Підрахунок частоти вживання слів**

|  |
| --- |
| # Об'єднуємо всі токени в один список  all\_tokens = [token for sublist in df["tokens"] for token in sublist]  # Рахуємо частоту появи кожного слова  word\_freq = Counter(all\_tokens)  # Створюємо DataFrame для візуалізації  word\_freq\_df = pd.DataFrame(word\_freq.items(), columns=["word", "count"]).sort\_values(by="count", ascending=False)  # Побудова діаграми  plt.figure(figsize=(10, 5))  sns.barplot(x=word\_freq\_df["word"][:10], y=word\_freq\_df["count"][:10])  plt.xticks(rotation=45)  plt.title("Найчастіше вживані слова")  plt.show() |
|  |

6. Обчислення Term Frequency (TF)

|  |
| --- |
| def term\_frequency(text):      words = tokenize(text)      word\_count = Counter(words)      max\_freq = max(word\_count.values()) if word\_count else 1  # Захист від ділення на нуль      return {word: freq / max\_freq for word, freq in word\_count.items()}  df["tf"] = df["text"].apply(term\_frequency)  df |
| |  | **author** | **text** | **tokens** | **tf** | | --- | --- | --- | --- | --- | | 0 | Леся Українка | Стій, серце, стій! не бийся так шалено. | [стій, серце, стій, бийся, шалено] | {'стій': 1.0, 'серце': 0.5, 'бийся': 0.5, 'шал... | | 1 | Леся Українка | Вгамуйся, думко, не літай так буйно! | [вгамуйся, думко, літай, буйно] | {'вгамуйся': 1.0, 'думко': 1.0, 'літай': 1.0, ... | | 2 | Леся Українка | Не бий крильми в порожньому просторі. | [бий, крильми, в, порожньому, просторі] | {'бий': 1.0, 'крильми': 1.0, 'в': 1.0, 'порожн... | | 3 | Тарас Шевченко | Як умру, то поховайте мене на могилі. | [умру, то, поховайте, мене, могилі] | {'умру': 1.0, 'то': 1.0, 'поховайте': 1.0, 'ме... | | 4 | Тарас Шевченко | Серед степу широкого, на Вкраїні милій. | [серед, степу, широкого, вкраїні, милій] | {'серед': 1.0, 'степу': 1.0, 'широкого': 1.0, ... | | 5 | Тарас Шевченко | Щоб лани широкополі, і Дніпро, і кручі, | [щоб, лани, широкополі, дніпро, кручі] | {'щоб': 1.0, 'лани': 1.0, 'широкополі': 1.0, '... | | 6 | Тарас Шевченко | Було видно, було чути, як реве ревучий. | [видно, чути, реве, ревучий] | {'видно': 1.0, 'чути': 1.0, 'реве': 1.0, 'реву... | |

7. Функція для розрахунку подібності Жаккара

|  |
| --- |
| def jaccard\_similarity(text1, text2):      set1, set2 = set(tokenize(text1)), set(tokenize(text2))      intersection = len(set1.intersection(set2))      union = len(set1.union(set2))      return intersection / union if union != 0 else 0  similarity = jaccard\_similarity(text\_1, text\_2)  print(f"Подібність Жаккара між текстами: {similarity:.2f}") |
| Подібність Жаккара між текстами: 0.00 |

8. Косинусна подібність

|  |
| --- |
| vectorizer = CountVectorizer().fit([text\_1, text\_2])  vectors = vectorizer.transform([text\_1, text\_2])  from sklearn.metrics.pairwise import cosine\_similarity  cos\_sim = cosine\_similarity(vectors)[0, 1]  print(f"Косинусна подібність між текстами: {cos\_sim:.2f}") |
| Косинусна подібність між текстами: 0.00 |

**Висновок:**  
В результаті виконання практичної роботи було проведено токенізацію текстів українською мовою за допомогою різних підходів. Спочатку була спроба використати бібліотеку NLTK для обробки тексту, однак виникли проблеми із завантаженням необхідних ресурсів для токенізації, зокрема стоп-слів українською мовою. Як альтернативу, було застосовано простіший метод токенізації на основі регулярних виразів, що дозволило успішно розбити тексти на окремі слова та видалити стоп-слова.

У підсумку, текст було успішно токенізовано, отримано список слів для кожного уривка, а також продемонстровано методи обробки тексту з використанням регулярних виразів для токенізації. Цей підхід виявився надійним і зручним, що дозволяє уникнути проблем із зовнішніми бібліотеками, особливо для української мови.