Лабораторний практикум

З дисципліни “Обробка та аналіз текстових даних на Python”

для студентів

заочної форми навчання

КПІ ФІОТ

Укладач: Мягкий М.Ю.

КИЇВ 2025

ТЕОРЕТИЧНІ ВІДОМОСТІ

Правила оформлення та подання звітів з лабораторних робіт

При виконанні лабораторних робіт студент повинен дотримуватись певних правил щодо вибору варіанта, структури звіту та формату подання. Нижче наведено основні вимоги.

1. Вибір варіанта лабораторних робіт

Кожен студент **обирає лише один варіант** і працює **за ним протягом усіх 6-ти лабораторних робіт**. Зміна варіанта під час курсу **не допускається**. Це дозволяє студенту послідовно опанувати відповідну тематику та закріпити знання у відповідній сфері аналізу текстових даних.

2. Формат подання звітів

Звіти повинні бути подані **у форматі PDF**. Назва файлу має відповідати наступному формату:

ПРІЗВИЩЕ_ГРУПА_НОМЕР_РОБОТИ.pdf

Наприклад, якщо студент Іваненко з групи КН-21 виконує **четверту лабораторну роботу**, його файл повинен мати назву:

ІВАНЕНКО_КН-21_ЛР_4.pdf

Будь-які інші формати або неправильно названі файли **не приймаються**.

3. Структура звіту

Звіт повинен містити наступні розділи:

3.1. Титульна сторінка

На титульній сторінці повинні бути вказані:

- Назва навчального закладу
- Назва дисципліни
- Номер лабораторної роботи

- Тема лабораторної роботи
- Прізвище, ім’я, по батькові студента
- Група
- Дата виконання
- Прізвище викладача

3.2. Теоретичні відомості (за бажанням)

У цьому розділі студент **може коротко** описати теоретичні аспекти, які були використані під час виконання завдання. Наприклад, пояснити використані алгоритми, бібліотеки або підходи.

3.3. Хід роботи

Цей розділ є основною частиною звіту. У ньому студент повинен:

1. **Описати етапи виконання завдання**, пояснюючи ключові моменти.
2. Надати **фрагменти коду**, які використовувалися для виконання лабораторної роботи. Код повинен бути відформатований і прокоментований, щоб було зрозуміло, яка його частина за що відповідає.
3. Додати **скріншоти результатів роботи** (наприклад, графіки, візуалізації, таблиці, отримані значення тощо).

3.4. Висновки

У цьому розділі студент повинен коротко підсумувати результати виконаної роботи, вказавши:

- Чого вдалося досягти
- Які труднощі виникли та як вони були подолані
- Які знання та навички отримані в процесі виконання лабораторної роботи

3.5. Відповіді на контрольні запитання

Студент **обов'язково** повинен надати відповіді на всі контрольні запитання, що були запропоновані у варіанті його лабораторної роботи. Відповіді мають бути структурованими та повними, з можливими прикладами.

4. Порядок захисту лабораторних робіт

Захист лабораторних робіт відбувається **лише після того, як студент надіслав всі 6 звітів**. Частковий захист окремих робіт **не передбачений**.

При захисті студент повинен:

- Продемонструвати розуміння виконаних завдань
- Пояснити свій код та результати роботи
- Відповісти на додаткові запитання викладача

Якщо під час захисту з'ясується, що робота виконана не самостійно (ідентичні звіти, неусвідомлене пояснення коду), вона може бути **відхилена**, і студент буде зобов'язаний виконати її заново.

5. Вимоги до оформлення тексту

- Основний текст – **шрифт Times New Roman, розмір 12, міжрядковий інтервал 1.5**
- Скріншоти повинні бути **читабельними та якісними**
- Усі сторінки повинні мати **нумерацію**

Зміст

Лабораторна робота №1

Тема: Обробка текстових даних у Python: основи

Варіант 1

Завдання:

1. Завантажити текстову новину з сайту pravda.com.ua.
2. Видалити всі зайві пробіли, символи пунктуації та цифри.
3. Перетворити текст у нижній регістр.
4. Підрахувати кількість слів у тексті.

Хід роботи:

Для отримання тексту новини з веб сайту необхідно скористатися бібліотекою **requests**, яка дозволяє завантажувати вміст веб-сторінок. Після цього слід виконати очищення тексту: видалити зайву пунктуацію за допомогою модуля **string**, а для позбавлення тексту від цифр використовувати **регулярні вирази**. Далі потрібно привести весь текст до нижнього регістру, застосувавши метод **.lower()**, що забезпечує однакове представлення всіх слів незалежно від їх початкового написання. Наступним кроком є розбиття тексту на окремі слова, що можна здійснити за допомогою методу **.split()**, який розділяє рядок на складові за пробілами. Завершальним етапом є підрахунок загальної кількості слів у тексті та виведення отриманого результату.

Контрольні питання:

1. Як завантажити текст із вебсайту в Python?
2. Що таке бібліотека `requests`?
3. Як видалити пунктуацію з тексту?
4. Як видалити цифри з тексту?
5. Що робить метод `.lower()`?
6. Як розбити текст на слова?
7. Які є способи підрахунку кількості слів?
8. Чому важливо очищати текст перед аналізом?
9. Як працює модуль `string`?
10. Яке кодування тексту краще використовувати для української мови?

Варіант 2

Завдання:

1. Завантажити текст новини з сайту bbc.com/ukrainian.
2. Видалити зайві пробіли та пунктуацію.
3. Розбити текст на речення.
4. Порахувати кількість речень у тексті.

Хід роботи:

Для отримання тексту новини слід скористатися бібліотекою **requests**, яка дозволяє завантажити веб-сторінку. Далі необхідно видалити зайві пробіли та пунктуацію, використовуючи **регулярні вирази** або модуль **string**. Після очищення тексту потрібно розбити його на окремі речення, застосувавши метод **.split('.')**, який розділить текст за крапками. На завершальному етапі слід визначити загальну кількість речень у тексті та вивести отриманий результат.

Контрольні питання:

1. Як завантажити текст із сайту?
2. Як видалити зайві пробіли в Python?
3. Що таке пунктуація і як її позбутися?
4. Як розбити текст на речення?
5. Що робить метод `.split('.')`?
6. Які проблеми можуть виникнути при розбитті тексту на речення?
7. Як Python обробляє кодування тексту?
8. Як перевірити результат розбиття?
9. Що таке регулярні вирази (`re`) у Python?
10. Який модуль найчастіше використовують для роботи з текстом?

Варіант 3

Завдання:

1. Завантажити текст новини з сайту nv.ua.
2. Очистити текст від цифр та пунктуації.
3. Порахувати кількість унікальних слів.

Хід роботи:

Щоб опрацювати текст новини, спочатку потрібно отримати його через бібліотеку **requests**. Після завантаження необхідно очистити текст від цифр і пунктуаційних символів, використовуючи **string.punctuation** та **регулярні вирази**. Далі очищений текст

слід розбити на окремі слова та помістити їх у **множину (set)**, що дозволить отримати унікальний набір слів. Останнім кроком є підрахунок кількості унікальних слів та виведення результату.

Контрольні питання:

1. Що таке множина (`set`) у Python?
2. Як створити множину слів з тексту?
3. Як позбутися цифр у тексті?
4. Як уникнути повторень слів?
5. Що робить функція `set()`?
6. Яка відмінність множини від списку?
7. Що таке унікальні слова?
8. Як видалити пунктуацію?
9. Як підрахувати елементи множини?
10. Чому унікальні слова важливі у текстовому аналізі?

Варіант 4

Завдання:

1. Завантажити текст новини з сайту tsn.ua.
2. Очистити текст від пунктуації.
3. Порахувати кількість слів, що починаються з голосних літер.

Хід роботи:

Щоб обробити текст новини, спочатку потрібно отримати його за допомогою **requests**. Після завантаження необхідно видалити всі **символи пунктуації**, використовуючи **string.punctuation**, а потім розбити текст на окремі слова за допомогою методу **.split()**. Далі слід створити **цикл for**, у якому кожне слово перевірятиметься на предмет того, чи починається воно з голосної літери (а, е, є, и, і, ї, о, у, ю, я). Нарешті, необхідно підрахувати загальну кількість таких слів та вивести результат.

Контрольні питання:

1. Як видалити пунктуацію з тексту?
2. Які голосні літери є в українській мові?
3. Що робить метод `.split()`?
4. Як перевірити першу літеру слова в Python?
5. Який цикл використовувати для перебору слів?
6. Чому важливо очистити текст від пунктуації?

7. Як враховувати регістр літер у Python?
8. Як підраховувати кількість елементів за умовою?
9. Що таке лямбда-функції та чи можна їх використати в цьому завданні?
10. Як Python працює з кодуванням тексту?

Варіант 5

Завдання:

1. Завантажити текст новини з сайту unian.ua.
2. Очистити текст від цифр.
3. Визначити найдовше слово у тексті.

Хід роботи:

Для обробки тексту новини спочатку потрібно отримати його через **requests**. Далі слід видалити всі **цифри**, застосувавши **регулярні вирази**, а потім розбити текст на окремі слова за допомогою **.split()**. Наступним кроком є визначення найдовшого слова у тексті, використовуючи функцію **max()** з параметром **key=len**, що дозволить знайти слово з найбільшою довжиною. У підсумку слід вивести знайдене найдовше слово.

Контрольні питання:

1. Як знайти найдовше слово у тексті?
2. Як видалити цифри з тексту?
3. Який метод розбиває текст на слова?
4. Як порівнювати довжину слів?
5. Що таке довжина слова?
6. Як знайти максимальний елемент списку?
7. Як уникнути помилок при пошуку найдовшого слова?
8. Як можна оптимізувати пошук найдовшого слова?
9. Які функції Python можна використовувати для цього завдання?
10. Що станеться, якщо у тексті є кілька найдовших слів?

Варіант 6

Завдання:

1. Завантажити текст новини з сайту radiosvoboda.org.
2. Очистити текст від пунктуації.
3. Порахувати частоту появи конкретного слова (наприклад, "Україна").

Хід роботи:

Для виконання аналізу тексту необхідно завантажити його за допомогою **requests**. Після цього слід очистити текст від пунктуаційних символів, використовуючи **string.punctuation**, а потім розбити його на слова за допомогою **.split()**. Далі потрібно підрахувати кількість входжень заданого слова в текст, використовуючи метод **count()** або клас **collections.Counter**, який дозволяє швидко визначити частоту появи кожного слова. На завершення слід вивести отриманий результат.

Контрольні питання:

1. Як видалити пунктуацію?
2. Що таке частота появи слова?
3. Як розбити текст на слова?
4. Як порахувати входження слова у тексті?
5. Які модулі Python можна використати для підрахунку частоти слів?
6. Що таке `collections.Counter`?
7. Чому важливо працювати з текстами без пунктуації?
8. Як можна представити частоту слів у вигляді діаграми?
9. Чому частотний аналіз важливий у лінгвістичних дослідженнях?
10. Як можна використовувати частотний аналіз для автоматичного аналізу текстів?

Лабораторна робота №2

Тема: Використання регулярних виразів для попередньої обробки тексту

Варіант 1

Завдання:

1. Завантажити текстову новину з сайту pravda.com.ua.
2. Використати регулярні вирази для видалення всіх числових значень з тексту.
3. Знайти та вивести всі email-адреси, які згадуються у тексті.

Хід роботи:

Для завантаження тексту новини слід скористатися бібліотекою **requests**, яка дозволяє отримати вміст веб-сторінки. Після отримання тексту необхідно видалити всі **числові значення** за допомогою **регулярного виразу** `\d+`. Далі слід знайти всі **email-адреси** у тексті, використовуючи **re.findall()** разом із шаблоном `[\w\.-]+@[\w\.-]+\.\w+`, який розпізнає стандартні формати електронної пошти. Після цього потрібно вивести очищений текст та список знайдених email-адрес.

Контрольні питання:

1. Що таке регулярний вираз?
2. Як використовувати функцію `re.findall()`?
3. Як виглядає регулярний вираз для пошуку email-адрес?
4. Яка функція використовується для заміни тексту в Python?
5. Як видалити цифри з тексту за допомогою регулярних виразів?
6. Що означає символ `\d` у регулярних виразах?
7. Як перевірити результат роботи регулярних виразів?
8. Які існують методи модуля `re`?
9. Що таке групування у регулярних виразах?
10. Які переваги використання регулярних виразів для обробки тексту?

Варіант 2

Завдання:

1. Завантажити текст новини з сайту bbc.com/ukrainian.
2. Використовуючи регулярні вирази, видалити всі знаки пунктуації.
3. Знайти та підрахувати кількість телефонних номерів у тексті.

Хід роботи:

Щоб обробити текст новини, його необхідно отримати за допомогою **requests**, а потім очистити, видаливши **пунктуацію** за допомогою **re.sub()** та регулярного виразу `\W+`. Наступним кроком є пошук телефонних номерів у тексті. Для цього слід скористатися регулярним виразом `\(?\d{3}\)?[-.\s]?\d{3}[-.\s]?\d{4}`, який дозволяє розпізнати номери у форматах **(XXX) XXX-XXXX, XXX-XXX-XXXX або XXX.XXX.XXXX**. У підсумку необхідно вивести очищений текст та загальну кількість знайдених телефонних номерів.

Контрольні питання:

1. Як виглядає регулярний вираз для пошуку телефонних номерів?
2. Як видалити пунктуацію за допомогою регулярних виразів?
3. Що означає символ `\W` у регулярних виразах?
4. Які функції модуля `re` можна використати для очищення тексту?
5. Які проблеми можуть виникнути при пошуку телефонних номерів?
6. Чому регулярні вирази ефективні для текстового аналізу?
7. Як перевірити коректність знайдених номерів?
8. Яка різниця між методами `re.search()` і `re.findall()`?
9. Як групуються символи у регулярних виразах?
10. Чому важливо правильно обирати регулярні вирази для аналізу тексту?

Варіант 3

Завдання:

1. Завантажити текст новини з сайту nv.ua.
2. Використати регулярні вирази для видалення всіх слів довжиною менше 4 символів.
3. Вивести список усіх знайдених дат у форматі **дд.мм.рррр**.

Хід роботи:

Для аналізу тексту спочатку слід **завантажити новину** та очистити її, видаливши всі слова довжиною **менше 4 символів** за допомогою **регулярного виразу** `\b\w{1,3}\b`. Далі необхідно знайти **дати у форматі "дд.мм.рррр"**, використовуючи **re.findall()** із виразом `\b\d{2}\.\d{2}\.\d{4}\b`. Після очищення тексту слід вивести список знайдених дат разом із оновленим текстом.

Контрольні питання:

1. Як сформувати регулярний вираз для пошуку дат у форматі **дд.мм.рррр**?
2. Як працює функція `re.sub()`?
3. Що означає вираз `\b` у регулярних виразах?
4. Які ще формати дат можуть зустрічатися в текстах?
5. Як видалити короткі слова з тексту?
6. Що таке квантифікатор у регулярних виразах?
7. Чому важливо видаляти короткі слова при аналізі тексту?
8. Як перевірити роботу регулярного виразу на тексті?
9. Що таке метасимвол у регулярних виразах?
10. Які переваги використання функції `re.sub()` для заміни тексту?

Варіант 4

Завдання:

1. Завантажити текст новини з сайту tsn.ua.
2. За допомогою регулярних виразів знайти всі URL-адреси, згадані у тексті.
3. Видалити всі спеціальні символи (крім пробілів та літер).

Хід роботи:

Щоб опрацювати текст новини, потрібно його отримати та виконати **пошук URL-адрес**, використовуючи **re.findall()** разом із регулярним виразом

`https?://[^\s]+`, який розпізнає веб-посилання. Після цього слід очистити текст, залишивши **лише літери та пробіли**. Це можна зробити за допомогою **re.sub()** із виразом `r'[^a-zA-Za-яA-ЯїЇєЄіІґҐ ]'`, що видаляє всі інші символи. У підсумку необхідно вивести очищений текст та список знайдених URL-адрес.

Контрольні питання:

1. Як знайти URL-адреси за допомогою регулярних виразів?
2. Які символи потрібно видалити з тексту при очищенні?
3. Як видалити спеціальні символи в тексті?
4. Що означає спеціальний символ `\w` у регулярних виразах?
5. Який регулярний вираз використати для пошуку URL-адрес?
6. Які ще спеціальні символи можна видалити регулярними виразами?
7. Які особливості роботи з українським текстом у Python?
8. Які функції модуля `re` можна застосовувати для очищення тексту?
9. Чому потрібно очищати текст перед аналізом?
10. Як перевірити результати роботи регулярних виразів?

Варіант 5

Завдання:

1. Завантажити текст новини з сайту unian.ua.
2. Використовуючи регулярні вирази, видалити всі слова, коротші за 4 символи.
3. Знайти та підрахувати кількість усіх згаданих у тексті власних імен (імена з великої літери).

Хід роботи:

Для обробки тексту новини потрібно завантажити його та **видалити всі слова, що містять менше 4 символів**, застосовуючи `\b\w{1,3}\b`. Далі необхідно виконати **пошук власних імен** у тексті, використовуючи **re.findall()** разом із виразом `r'\b[A-ZA-ЯЇЄІҐ][a-za-яїєіґ]+\b'`, який розпізнає слова, що починаються з великої літери. Наприкінці слід вивести очищений текст та підрахувати кількість знайдених власних назв.

Контрольні питання:

1. Як знайти слова за допомогою регулярних виразів?
2. Як видалити короткі слова з тексту?
3. Що означає символ `\b` у регулярних виразах?

4. Які символи позначають початок слова у регулярних виразах?
5. Чому важливо очищати текст від коротких слів?
6. Як працює модуль `re`?
7. Який регулярний вираз можна використати для пошуку власних назв?
8. Які функції використовуються для підрахунку кількості слів?
9. Чим відрізняється регулярний вираз від звичайного пошуку?
10. Що робити, якщо знайдені результати некоректні?

Варіант 6

Завдання:

1. Завантажити текст новини з сайту ukrinform.ua.
2. Використати регулярні вирази для пошуку та видалення повторюваних слів у тексті.
3. Порахувати та вивести кількість слів, що повторюються у тексті.

Хід роботи:

Щоб знайти повторювані слова у тексті, спочатку слід **завантажити його**, а потім **видалити дублікати** слів за допомогою **re.sub()** із виразом `r'\b(\w+)\b\s+\b\1\b'`, який заміняє повторення одного і того ж слова. Далі потрібно **підрахувати частоту входження слів**, використовуючи **collections.Counter()** разом із регулярним виразом `r'\b\w+\b'`, який витягує всі слова з тексту. У підсумку слід вивести очищений текст та кількість повторюваних слів.

Контрольні питання:

1. Як шукати повторювані слова регулярними виразами?
2. Що таке backreference у регулярних виразах?
3. Як видалити дублікати слів за допомогою регулярних виразів?
4. Яка функція Python дозволяє замінити знайдені вирази?
5. Як підрахувати кількість повторюваних слів у тексті?
6. Які труднощі можуть виникнути при пошуку повторень у тексті?
7. Що таке жадібні та ліниві квантифікатори у регулярних виразах?
8. Які існують альтернативи регулярним виразам для такого завдання?
9. Чому регулярні вирази корисні для попередньої обробки тексту?
10. Як перевірити правильність роботи регулярного виразу на прикладах?

Лабораторна робота №3

Тема: Аналіз тексту за допомогою NLTK

Варіант 1

Завдання:

1. Завантажити текстову новину з сайту pravda.com.ua.
2. Виконати токенізацію тексту.
3. Провести частотний аналіз слів.
4. Вивести список 10 найчастіших слів.

Хід роботи:

Щоб виконати аналіз тексту, необхідно завантажити його за допомогою бібліотеки **requests**. Після отримання тексту слід здійснити **токенізацію**, використовуючи **nltk.word_tokenize()**, яка розбиває текст на окремі слова. Далі, для підрахунку **частотного розподілу слів**, потрібно застосувати **nltk.FreqDist()**, що дозволяє визначити, які слова зустрічаються найчастіше. Завершальним кроком є виведення **10 найуживаніших слів** разом з їхньою частотою.

Контрольні питання:

1. Що таке токенізація?
2. Як проводиться частотний аналіз слів?
3. Яка роль класу `FreqDist` у NLTK?
4. Що таке стоп-слова?
5. Як вилучити стоп-слова з тексту?
6. Для чого проводять частотний аналіз слів?
7. Як побудувати графік частот слів у NLTK?
8. Чому важлива попередня обробка тексту перед аналізом?
9. Як працює метод `word_tokenize()`?
10. Що таке текстовий корпус і як він використовується в NLP?

Варіант 2

Завдання:

1. Завантажити статтю з сайту unian.ua.
2. Виконати токенізацію тексту.
3. Визначити частоту появи біграм.
4. Вивести 10 найчастіших біграм.

Хід роботи:

Щоб виконати аналіз біграм у тексті, спочатку необхідно завантажити статтю та провести її **токенізацію** за допомогою **nltk.word_tokenize()**. Після цього потрібно сформувати **біграми**, використовуючи **nltk.bigrams()**, що дозволяє створити пари послідовних слів. Далі, застосовуючи **nltk.FreqDist()**, можна визначити частоту появи кожної біграми в тексті. У підсумку слід вивести **10 найпоширеніших біграм**.

Контрольні питання:

1. Що таке біграма?
2. Як обчислювати частоту біграм?
3. Для чого використовуються біграми в аналізі текстів?
4. Які методи є в бібліотеці NLTK для роботи з біграмами?
5. Як перевірити якість токенізації?
6. Що означає контекст у текстовому аналізі?
7. Які переваги біграм перед уніграмами?
8. Які можливі помилки при аналізі біграм?
9. Як вивести результат роботи у зручному форматі?
10. Чим відрізняються токенізація на слова та на речення?

Варіант 3

Завдання:

1. Завантажити текст з сайту radiosvoboda.org.
2. Виконати токенізацію тексту на речення.
3. Порахувати кількість речень у тексті.
4. Вивести перші 5 речень тексту.

Хід роботи:

Для визначення кількості речень у тексті потрібно спочатку **завантажити** новину, а потім застосувати **nltk.sent_tokenize()**, яка розбиває текст на речення. Далі, використовуючи функцію **len()**, можна порахувати загальну кількість речень у тексті. Останнім кроком є виведення **перших 5 речень**, щоб оцінити правильність розбиття.

Контрольні питання:

1. Що таке `sent_tokenize()`?
2. Яке призначення розбиття тексту на речення?
3. Чому важливе розбиття тексту на речення в NLP?
4. Як визначити межі речень у тексті?

5. Як працює `sent_tokenize()` у NLTK?
6. Чим відрізняється `sent_tokenize()` від `word_tokenize()`?
7. Що може впливати на якість токенізації?
8. Як перевірити результати токенізації речень?
9. Що таке токен у контексті NLP?
10. Як завантажити текст із вебсайту для аналізу?

Варіант 4

Завдання:

1. Завантажити текст з сайту liga.net.
2. Провести токенізацію тексту на слова.
3. Вивести частоту появи конкретного слова **"Україна"**.

Хід роботи:

Щоб підрахувати частоту появи слова **"Україна"** у тексті, спочатку слід **отримати текст** новини та виконати **токенізацію**, застосовуючи **nltk.word_tokenize()**. Далі необхідно використати **nltk.FreqDist()** для підрахунку частоти входження кожного слова. У підсумку слід вивести кількість разів, коли слово **"Україна"** зустрічається у тексті.

Контрольні питання:

1. Як знайти частоту появи конкретного слова у тексті?
2. Як працює `word_tokenize()`?
3. Які є методи для аналізу частотності в NLTK?
4. Як виявити різницю між токенізованим текстом і вихідним текстом?
5. Що означає нормалізація тексту і чи потрібна вона у даному випадку?
6. Чому варто враховувати відмінки в аналізі частоти слова?
7. Як можна візуалізувати результати частотного аналізу?
8. Чи можна використовувати `Counter` з `collections` для частотного аналізу?
9. Як позбутися знаків пунктуації перед частотним аналізом?
10. Які ще методи NLP можна застосувати до аналізу окремих слів?

Варіант 5

Завдання:

1. Завантажити текст із сайту hromadske.ua.
2. Виконати токенізацію тексту.
3. Виявити і підрахувати частоту слів довших за 6 літер.

Хід роботи:

Для аналізу довгих слів у тексті спочатку потрібно **завантажити його** та виконати **токенізацію** за допомогою **nltk.word_tokenize()**. Далі необхідно **відфільтрувати** всі слова, довжина яких перевищує **6 літер**, щоб залишити лише довші слова. Потім, застосовуючи **nltk.FreqDist()**, можна визначити частотний розподіл таких слів. У підсумку слід вивести результати аналізу.

Контрольні питання:

1. Як знайти слова довші за 6 літер у тексті?
2. Як `word_tokenize()` обробляє складні слова?
3. Чому важливо враховувати довжину слів у текстовому аналізі?
4. Як використати `FreqDist()` для підрахунку слів певної довжини?
5. Чому варто видаляти стоп-слова перед таким аналізом?
6. Як можна нормалізувати токени для аналізу частоти довгих слів?
7. Чи можна використовувати `Counter` замість `FreqDist()`?
8. Як вивести частотний аналіз у зручному форматі?
9. Чи зміниться частота слів при лематизації?
10. Як впливає морфологія української мови на аналіз довгих слів?

Варіант 6

Завдання:

1. Завантажити текст з сайту ukrinform.ua.
2. Провести токенізацію тексту.
3. Виявити та вивести стоп-слова, що зустрічаються в тексті.
4. Підрахувати кількість стоп-слів.

Хід роботи:

Щоб знайти та підрахувати **стоп-слова** у тексті, спочатку слід **завантажити статтю** та виконати **токенізацію** за допомогою **nltk.word_tokenize()**. Далі необхідно завантажити список **стоп-слів** української мови за допомогою **nltk.corpus.stopwords.words('ukrainian')**. Потім слід **відфільтрувати всі стоп-слова** з тексту, після чого підрахувати їхню загальну кількість та вивести результат.

Контрольні питання:

1. Що таке стоп-слова у NLP?
2. Як отримати список стоп-слів у NLTK?

3. Чому видалення стоп-слів покращує аналіз тексту?
4. Які основні стоп-слова для української мови?
5. Чи можна створити власний список стоп-слів?
6. Як порахувати кількість стоп-слів у тексті?
7. Чи впливають стоп-слова на частотний аналіз?
8. Як можна використовувати `set()` для швидшої перевірки стоп-слів?
9. Як можна покращити якість аналізу тексту без видалення стоп-слів?
10. Які ще методи аналізу тексту можна поєднувати з обробкою стоп-слів?

Лабораторна робота №4

Тема: Векторизація тексту та застосування TF-IDF

Варіант 1

Завдання:

1. Завантажити текстову новину з сайту pravda.com.ua.
2. Використовуючи метод **Bag-of-Words**, векторизувати текст.
3. Вивести розмір отриманого вектора (кількість унікальних слів).

Хід роботи:

Щоб виконати векторизацію тексту, спочатку необхідно **завантажити новину** за допомогою **requests**. Далі слід скористатися **CountVectorizer** із бібліотеки **sklearn.feature_extraction.text**, що дозволяє перетворити текст у векторну форму, представивши його у вигляді **кількості входжень слів**. Після цього потрібно визначити **розмір вектора**, тобто підрахувати кількість **унікальних слів** у тексті. Завершальним кроком є виведення отриманих результатів.

Контрольні питання:

1. Що таке метод **Bag-of-Words**?
2. Яку бібліотеку використовують для векторизації тексту у Python?
3. Які параметри `CountVectorizer` ви знаєте?
4. Як підрахувати кількість унікальних слів у тексті?
5. Що таке вектор ознак у текстовій аналітиці?
6. Чим корисна векторизація тексту?
7. Чому метод **Bag-of-Words** ігнорує порядок слів?
8. Як завантажити текст із веб-сторінки у Python?
9. Яка роль попередньої обробки тексту перед векторизацією?

10. Як вивести отриманий вектор у вигляді масиву?

Варіант 2

Завдання:

1. Завантажити текст новини з сайту nv.ua.
2. Використати **TF-IDF** для векторизації тексту.
3. Вивести 10 слів з найвищими значеннями TF-IDF.

Хід роботи:

Щоб виконати TF-IDF аналіз тексту, спочатку потрібно **отримати статтю** за допомогою **requests**. Далі застосовується **TfidfVectorizer** із **sklearn.feature_extraction.text**, який обчислює **TF-IDF ваги** для кожного слова у тексті. Після векторизації необхідно визначити **10 слів із найвищими значеннями TF-IDF**, що дозволяє знайти найбільш значущі терміни. У підсумку слід вивести список знайдених слів разом із їхніми вагами.

Контрольні питання:

1. Що таке **TF-IDF**?
2. Яка різниця між **Bag-of-Words** і **TF-IDF**?
3. Як працює `TfidfVectorizer`?
4. Як визначається вага слова у TF-IDF?
5. Чому TF-IDF краще підходить для оцінки важливості слів?
6. Які недоліки має метод **TF-IDF**?
7. Як можна нормалізувати текст перед TF-IDF векторизацією?
8. Як знайти слова з найвищими TF-IDF значеннями?
9. Чому важливо видаляти стоп-слова перед обчисленням TF-IDF?
10. Як впливає розмір корпусу документів на TF-IDF?

Варіант 3

Завдання:

1. Завантажити статтю з сайту bbc.com/ukrainian.
2. Векторизувати текст за допомогою **CountVectorizer**.
3. Вивести частоту появи слова **"економіка"**.

Хід роботи:

Для підрахунку частоти конкретного слова в тексті необхідно **завантажити новину** та використати **CountVectorizer** для векторизації тексту. Після цього слід знайти, скільки

разів у векторизованому представленні зустрічається слово **"економіка"**. Завершальним кроком є виведення отриманого результату.

Контрольні питання:

1. Як працює `CountVectorizer`?
2. Що таке частота слова у тексті?
3. Як знайти частоту конкретного слова у векторизованому тексті?
4. Як працює матриця термін-документ (Term-Document Matrix)?
5. Чим корисна векторизація тексту для аналізу новин?
6. Чому потрібно нормалізувати текст перед векторизацією?
7. Як впливає розмір словника на продуктивність моделі?
8. Як обрати параметри `CountVectorizer`, щоб уникнути надмірного шуму?
9. Які обмеження має **Bag-of-Words** для аналізу тексту?
10. Як можна використати частоту слова для тематичного аналізу?

Варіант 4

Завдання:

1. Завантажити текст з сайту tsn.ua.
2. Використати **TF-IDF векторизацію**.
3. Визначити та вивести кількість слів, значення TF-IDF яких **більше 0.1**.

Хід роботи:

Щоб визначити слова з високими TF-IDF значеннями, спочатку необхідно **отримати текст новини** та обчислити **TF-IDF ваги** за допомогою **TfidfVectorizer**. Далі слід **відфільтрувати всі слова**, у яких **TF-IDF значення перевищує 0.1**, і вивести **кількість таких слів**, а також список відповідних термінів.

Контрольні питання:

1. Як визначити, які слова мають TF-IDF більше 0.1?
2. Що означає значення **TF-IDF = 0**?
3. Чому деякі слова отримують високі TF-IDF значення?
4. Як видалення стоп-слів впливає на TF-IDF?
5. Чому важливо масштабувати результати TF-IDF?
6. Як змінюється TF-IDF при додаванні нових документів?
7. Яка роль `max_df` та `min_df` у `TfidfVectorizer`?
8. Як можна візуалізувати результати TF-IDF?
9. Чим TF-IDF відрізняється від нормалізованої частоти термінів?

10. Як змінюється TF-IDF для слова, якщо воно зустрічається у кожному документі корпусу?

Варіант 5

Завдання:

1. Завантажити текст з сайту hromadske.ua.
2. Використати метод **Bag-of-Words**.
3. Визначити кількість слів, які зустрічаються лише **один раз** у тексті.

Хід роботи:

Щоб знайти слова, які зустрічаються лише **один раз** у тексті, спочатку потрібно **завантажити статтю** та використати **CountVectorizer** для векторизації. Далі необхідно визначити слова, частота появи яких дорівнює **1**, що дозволить виявити **рідковживані терміни**. У підсумку слід підрахувати їхню кількість та вивести список таких слів.

Контрольні питання:

1. Як знайти слова, які зустрічаються лише **один раз** у тексті?
2. Що таке **hapax legomena** у текстовому аналізі?
3. Як працює `Counter` для підрахунку слів?
4. Чому рідковживані слова можуть бути важливими у текстовому аналізі?
5. Як використовувати `CountVectorizer` для підрахунку частоти слів?
6. Чому важливо аналізувати рідковживані слова у корпусі текстів?
7. Як обробляти тексти перед побудовою `Bag-of-Words` моделі?
8. Як можна візуалізувати рідковживані слова у тексті?
9. Яка роль лематизації у побудові **Bag-of-Words**?
10. Як використовувати частоту рідковживаних слів для аналізу стилю автора?

Варіант 6

Завдання:

1. Завантажити текст з сайту ukrinform.ua.
2. Використати **TF-IDF** для векторизації тексту.
3. Вивести слова, значення TF-IDF яких знаходяться у **діапазоні 0.05–0.15**.

Хід роботи:

Для аналізу слів у певному **TF-IDF діапазоні** спочатку потрібно **завантажити текст новини** та використати **TfidfVectorizer** для створення **TF-IDF матриці**. Після цього слід **відфільтрувати слова**, значення TF-IDF яких **знаходяться у діапазоні 0.05–0.15**. Завершальним кроком є виведення списку знайдених слів.

Контрольні питання:

1. Як знайти слова з TF-IDF значенням у діапазоні **0.05–0.15**?
2. Чому слова з середніми TF-IDF значеннями можуть бути найбільш інформативними?
3. Як фільтрувати слова за TF-IDF у `TfidfVectorizer`?
4. Як працює `TfidfTransformer`?
5. Чому потрібно приводити текст до нижнього регістру перед обчисленням TF-IDF?
6. Які параметри у `TfidfVectorizer` впливають на результат?
7. Як використовувати `idf_` для аналізу впливу рідковживаних слів?
8. Як змінюється TF-IDF, якщо додати дублікати документа?
9. Як використовувати TF-IDF для побудови кластеризаційних моделей?
10. Чому важливо вибирати відповідний поріг для значень TF-IDF у текстовому аналізі?

Лабораторна робота №5

Тема: Тематичне моделювання з використанням LDA

Варіант 1

Завдання:

1. Завантажити **5 текстових новин** з сайту pravda.com.ua на тему політики.
2. Використати алгоритм **LDA** для побудови тематичної моделі.
3. Вивести **3 найголовніші теми** з ключовими словами.

Хід роботи:

Щоб побудувати тематичну модель для аналізу новин, спочатку необхідно **завантажити тексти** та виконати їх **очищення**, видаляючи зайві символи та пунктуацію. Далі потрібно створити **словник**, використовуючи **gensim.corpora.Dictionary**, що дозволяє перетворити текст у структуровану форму. Наступним етапом є побудова моделі **LDA** за допомогою **gensim.models.LdaModel**, яка визначає приховані теми у текстах. У підсумку слід вивести **три найголовніші теми** разом із відповідними **ключовими словами**.

Контрольні питання:

1. Що таке тематичне моделювання?
2. Як працює алгоритм **LDA**?
3. Які параметри потрібно задавати при створенні моделі **LDA**?
4. Що таке корпус текстів у тематичному моделюванні?
5. Як підготувати тексти до аналізу в **LDA**?
6. Як визначити **оптимальну кількість тем** у моделі?
7. Яка бібліотека Python використовується для LDA?
8. Які переваги тематичного моделювання?
9. Що таке **ключові слова тем** у LDA?
10. Як оцінити **якість моделі** у тематичному аналізі?

Варіант 2

Завдання:

1. Завантажити **5 текстів новин** з сайту bbc.com/ukrainian про економіку.
2. Використати модель **LDA** з **4 темами**.
3. Вивести отримані теми та їх ключові слова.

Хід роботи:

Для аналізу економічних новин необхідно **отримати тексти** та очистити їх від небажаних символів. Далі слід сформувати **корпус документів**, використовуючи **gensim.corpora.Dictionary**, який дозволяє підготувати дані до тематичного аналізу. На основі отриманого корпусу потрібно створити **LDA-модель**, що міститиме **чотири теми**, використовуючи **LdaModel**. Після побудови моделі необхідно вивести визначені теми разом із ключовими словами, що їх характеризують.

Контрольні питання:

1. Як працює алгоритм **LDA**?
2. Що означають слова в отриманих темах?
3. Як вибрати **оптимальну кількість тем** у моделі?
4. Що таке **токенізація** та як вона використовується в LDA?
5. Як оцінити якість моделі?
6. Які метрики використовують для **оцінки LDA**?
7. Що таке **когерентність** у LDA?
8. Як очистити текст перед тематичним аналізом?
9. Які бібліотеки використовуються для тематичного аналізу?
10. Як **інтерпретувати результати LDA**?

Варіант 3

Завдання:

1. Завантажити **5 текстів статей** з сайту unian.ua на медичну тематику.
2. Побудувати модель **LDA** на **4 теми**.
3. Вивести отримані теми з найбільш характерними словами.

Хід роботи:

Щоб провести тематичний аналіз текстів, спочатку слід **завантажити тексти** та виконати **попередню обробку**, яка включає **видалення пунктуації, стоп-слів та лематизацію**. Далі потрібно сформувати **корпус документів** за допомогою **gensim.corpora.Dictionary**, який використовується для структурування текстових даних. Після цього необхідно застосувати **LDA-модель** для виявлення тематичних груп у текстах. Завершальним етапом є **виведення списку тем** з **найбільш характерними словами** для кожної з них.

Контрольні питання:

1. Що таке тематичне моделювання?
2. Як обрати **кількість тем** у моделі LDA?
3. Яка структура результатів LDA-моделі?
4. Що таке **словник у Gensim**?
5. Що означає поняття **"прихована тема"**?
6. Які параметри можна **налаштовувати** в LDA?
7. Що впливає на **якість** моделі LDA?
8. Як оцінити **когерентність** отриманих тем?
9. Яка роль попередньої обробки тексту в LDA?
10. Що означають **вагові коефіцієнти слів** у темах?

Варіант 4

Завдання:

1. Завантажити **5 новин про спорт** з сайту sport.ua.
2. Створити модель **LDA** з **4 темами**.
3. Вивести основні теми і найважливіші слова.

Хід роботи:

Для тематичного аналізу спортивних новин потрібно спочатку **завантажити та очистити тексти**, видаляючи **спеціальні символи** та виконуючи **нормалізацію тексту**.

Далі необхідно створити **словник текстів** за допомогою **gensim.corpora.Dictionary**, що дозволяє систематизувати отримані дані. Після цього використовується **LdaModel** для побудови **тем з чотирма категоріями**, що відповідають основним темам у текстах. У підсумку слід **вивести отримані теми** разом із їхніми **ключовими словами**.

Контрольні питання:

1. Як підготувати корпус текстів для моделювання?
2. Як працює **Latent Dirichlet Allocation**?
3. Як впливає **кількість тем** на результати моделі?
4. Що таке **лематизація** і як вона допомагає в NLP?
5. Чому важливо **очищати тексти** перед моделюванням?
6. Як визначити **найважливіші слова** у кожній темі?
7. Що таке **розрідженість** текстових даних і як її враховує LDA?
8. Як змінюється розподіл тем при додаванні нових документів?
9. Як можна візуалізувати результати LDA?
10. Чим **LDA** відрізняється від інших методів кластеризації текстів?

Варіант 5

Завдання:

1. Завантажити **5 текстів про технології** з сайту itc.ua.
2. Використати **LDA** з **3 темами**.
3. Визначити **найбільш значущі слова** у кожній темі.

Хід роботи:

Щоб виконати тематичне моделювання для новин про технології, потрібно **отримати та очистити тексти**, після чого виконати **токенізацію**, тобто розбиття тексту на окремі слова. Далі слід створити **корпус текстів** за допомогою **gensim.corpora.Dictionary**, який використовується для структурування даних. Наступним кроком є побудова **моделі LDA**, що визначить **три тематичні групи** у текстах. У підсумку слід **визначити найбільш значущі слова у кожній темі** та вивести їх.

Контрольні питання:

1. Як використовувати **Gensim** для тематичного моделювання?
2. Що означає **гіперпараметр альфа** в LDA?
3. Як налаштовувати **кількість слів у темах**?
4. Чому **розмір словника** важливий у тематичному аналізі?
5. Як можна **автоматично визначити** оптимальну кількість тем?

6. Що таке **probability distribution** у контексті LDA?
7. Як порівняти результати тематичного аналізу між кількома моделями?
8. Які **обмеження** має LDA?
9. Як покращити **якість** отриманих тем?
10. Чим **LDA** відрізняється від **LDA2Vec** та інших методів?

Варіант 6

Завдання:

1. Завантажити **5 текстових новин** з сайту nv.ua на тему освіти.
2. Використати **LDA** для створення **тематичної моделі з 5 темами**.
3. Вивести **5 найважливіших слів** для кожної теми.

Хід роботи:

Щоб визначити основні теми у текстах про освіту, потрібно **отримати та очистити тексти**, видаливши **зайві символи та непотрібні слова**. Далі необхідно сформувати **корпус даних** за допомогою **gensim.corpora.Dictionary**, що дозволить структурувати текстову інформацію. Наступним кроком є побудова **LDA-моделі**, що міститиме **п'ять тем**, кожна з яких відповідатиме певній категорії текстів. У підсумку слід **вивести п'ять ключових слів** для кожної з визначених тем.

Контрольні питання:

1. Як підготувати текстовий корпус для LDA?
2. Які етапи потрібно виконати перед навчанням моделі LDA?
3. Як можна **візуалізувати** теми, знайдені за допомогою LDA?
4. Що таке **topic coherence** і чому вона важлива?
5. Як працює `gensim.corpora.Dictionary`?
6. Чому потрібно **нормалізувати текст** перед тематичним аналізом?
7. Як **визначити ключові слова** в кожній темі?
8. Що означає **дистрибуція тем** в документах?
9. Як можна **перевірити якість моделі** без візуалізації?
10. Які альтернативи LDA існують у тематичному моделюванні?

Лабораторна робота №6

Тема: Класифікація тексту з використанням нейронних мереж

Варіант 1

Завдання:

1. Завантажити **10 текстових новин** з сайту pravda.com.ua (5 політичних і 5 економічних).
2. Створити **нейронну мережу на базі LSTM** для класифікації новин.
3. Оцінити **якість класифікації** та вивести **точність (accuracy)**.

Хід роботи:

Щоб побудувати модель для текстової класифікації, спочатку необхідно **завантажити тексти** та **очистити їх**, видаливши **зайві символи, пунктуацію** та перевівши весь текст у **нижній регістр** для стандартизації. Далі слід використати **TensorFlow/Keras** для створення та навчання **LSTM-моделі**, яка здатна обробляти послідовні текстові дані. Завершальним етапом є оцінка якості моделі за допомогою **метрики accuracy**, що дозволяє визначити точність класифікації.

Контрольні питання:

1. Що таке **рекурентні нейронні мережі (RNN)**?
2. Як працює **LSTM**?
3. Які переваги використання **LSTM** у класифікації текстів?
4. Що таке **метрика accuracy**?
5. Як підготувати **дані** для навчання нейронної мережі?
6. Що означає **епоха** у навчанні моделі?
7. Чому важливо розбивати **дані** на тренувальний і тестовий набори?
8. Що таке **переобучення (overfitting)**?
9. Як боротися з **overfitting**?
10. Як оцінювати **результати класифікації**?

Варіант 2

Завдання:

1. Завантажити **10 текстів новин** з сайту bbc.com/ukrainian (5 спортивних і 5 культурних).
2. Побудувати **нейронну мережу на основі LSTM** для класифікації.
3. Вивести значення **precision** та **recall**.

Хід роботи:

Щоб побудувати LSTM-класифікатор, необхідно **отримати та очистити тексти**, прибравши **шумові дані** (зайві символи, HTML-теги, спецсимволи тощо). Далі слід використати **TensorFlow/Keras** для створення **LSTM-мережі**, яка навчиться розпізнавати

закономірності у текстах. Оцінку якості класифікації необхідно виконати за допомогою **метрик precision та recall**, що дозволяють проаналізувати, наскільки добре модель розрізняє класи.

Контрольні питання:

1. Що таке **precision** і **recall**?
2. Як розраховуються **precision та recall** у класифікації?
3. Чим **precision** відрізняється від **accuracy**?
4. Яка роль **LSTM** у NLP?
5. Як працює **padding** при роботі з текстами у нейромережах?
6. Як розбити текст на **токени** для навчання моделі?
7. Що таке **dropout** і як він допомагає уникнути переобучення?
8. Чому важливо нормалізувати тексти перед навчанням моделі?
9. Як можна **оптимізувати LSTM-архітектуру**?
10. Як візуалізувати **результати класифікації**?

Варіант 3

Завдання:

1. Завантажити **10 новин** з сайту unian.ua (5 медичних і 5 технологічних).
2. Створити **нейронну мережу (LSTM)** для класифікації.
3. Використати **cross-validation** для оцінки якості класифікації.

Хід роботи:

Для оцінки ефективності моделі за допомогою **перехресної перевірки (cross-validation)**, спочатку слід **завантажити тексти** та провести їх **очищення**, видаливши зайві символи та нормалізувавши дані. Далі необхідно побудувати **LSTM-архітектуру** та навчити її на текстових даних. Після цього слід провести **перехресну перевірку**, що дозволить оцінити стабільність моделі на різних підмножинах даних.

Контрольні питання:

1. Що таке **cross-validation** і як його застосовують у класифікації?
2. Чому важливо використовувати **перехресну перевірку** у машинному навчанні?
3. Які особливості **LSTM** у порівнянні з традиційними RNN?
4. Як обрати **оптимальну кількість нейронів** у LSTM-шарі?
5. Що таке **loss function** у класифікаційних моделях?
6. Як працює **оптимізатор Adam** у нейромережах?

7. Чому важливо балансувати вибірку при класифікації?
8. Як можна покращити якість моделі за допомогою **регуляризації**?
9. Що означає **cross-entropy loss** у задачах класифікації?
10. Як оцінювати якість класифікації за допомогою **перехресної перевірки**?

Варіант 4

Завдання:

1. Завантажити **10 текстів новин** з сайту sport.ua (5 про футбол і 5 про баскетбол).
2. Побудувати **класифікатор** на основі нейронної мережі **LSTM**.
3. Провести оцінку результатів за допомогою **confusion matrix**.

Хід роботи:

Щоб виконати класифікацію текстів про **футбол та баскетбол**, спочатку потрібно **завантажити тексти** та виконати **токенізацію**, перетворюючи слова у числові представлення. Далі слід створити **LSTM-мережу** для текстової класифікації, яка здатна аналізувати послідовності слів. Для оцінки якості класифікації необхідно **побудувати та візуалізувати confusion matrix**, що допоможе проаналізувати помилки моделі та визначити її сильні та слабкі сторони.

Контрольні питання:

1. Що таке **confusion matrix** і як її використовувати у класифікації?
2. Які основні **компоненти** confusion matrix?
3. Як розраховуються **TP, FP, FN, TN**?
4. Що таке **recall** і як він залежить від **confusion matrix**?
5. Чому важливо оцінювати якість моделі на **незалежних тестових даних**?
6. Як змінюється **точність моделі**, якщо вона тренується на невеликому наборі даних?
7. Як збільшити ефективність **LSTM** при роботі з довгими текстами?
8. Чим відрізняється **одношарова LSTM** від багатошарової?
9. Що таке **batch size** і як він впливає на навчання моделі?
10. Як можна використовувати **confusion matrix** для пошуку помилок у класифікації?

Варіант 5

Завдання:

1. Завантажити **10 текстів** з сайту itc.ua (5 про смартфони і 5 про комп’ютери).
2. Використати **нейронну мережу (LSTM)** для класифікації.
3. Вивести **метрики F1-score** та **accuracy**.

Хід роботи:

Щоб оцінити ефективність LSTM-моделі за **F1-score та accuracy**, необхідно **отримати тексти**, виконати **очищення** та **векторизацію**, перетворивши їх у числовий формат для подальшої обробки. Далі слід навчити **LSTM-мережу**, яка здійснюватиме класифікацію текстів. Після навчання моделі потрібно провести її оцінку, використовуючи **метрики F1-score** (що враховує як precision, так і recall) та **accuracy** для загальної точності класифікації.

Контрольні питання:

1. Що таке **F1-score** і як він розраховується?
2. Чому **F1-score** краще підходить для **незбалансованих вибірок**, ніж accuracy?
3. Як обрати **кількість шарів** у нейронній мережі для класифікації тексту?
4. Які функції активації найчастіше використовуються у **LSTM**?
5. Що таке **word embedding**, і як він допомагає покращити класифікацію тексту?
6. Чому важливо використовувати **pretrained word embeddings** (наприклад, Word2Vec або GloVe)?
7. Як працює **attention mechanism** і чи можна його використовувати разом із LSTM?
8. Як змінюється **навчання моделі**, якщо використати більшу кількість даних?
9. Що таке **validation set** і як він використовується у процесі навчання?
10. Як можна зменшити час навчання LSTM-моделі?

Варіант 6

Завдання:

1. Завантажити **10 текстів новин** з сайту nv.ua (5 про освіту і 5 про науку).
2. Побудувати **модель класифікації** за допомогою **LSTM**.
3. Провести **візуалізацію навчання моделі** (графік зміни **accuracy** і **loss**).

Хід роботи:

Щоб виконати навчання **LSTM-класифікатора** та проаналізувати процес його тренування, необхідно **завантажити та очистити тексти**. Далі слід побудувати **LSTM-модель** у **TensorFlow/Keras** та навчити її на підготовлених даних. У процесі навчання потрібно **зберігати графіки accuracy та loss**, що дозволить візуалізувати, як змінюється точність і помилка моделі протягом тренувальних епох.

Контрольні питання:

1. Що означає **loss function** у нейромережах?
2. Як змінюється **accuracy** під час навчання моделі?

3. Чому потрібно **візуалізувати** зміну accuracy і loss?
4. Як визначити момент, коли модель починає **перенавчатися (overfitting)**?
5. Як працює **early stopping**, і як він допомагає уникнути переобучення?
6. Що таке **batch normalization** і як вона впливає на швидкість навчання?
7. Як працює **gradient clipping**, і коли його слід використовувати?
8. Які є методи **різних варіантів LSTM** (наприклад, Bidirectional LSTM)?
9. Чому важливо правильно обирати **learning rate**?
10. Як можна використовувати **TensorBoard** для моніторингу навчання моделі?