ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «ШТУЧНИЙ ІНТЕЛЕКТ ДЛЯ УДОСКОНАЛЕННЯ АВТОМАТИЧНОЇ ОБРОБКИ МОВИ ТА ТЕКСТІВ»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ *Богдан БУХАНЧУК*

Виконав:
здобувач вищої освіти Богдан БУХАНЧУК
група ІСД-42

Керівник: Оксана ТКАЛЕНКО
науковий ступінь, вчене звання к.т.н., доцент

Рецензент:
науковий ступінь, вчене звання _____
Ім’я, ПРІЗВИЩЕ

Київ – 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій

Ступінь вищої освіти Бакалавр

Спеціальність Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІСТ

_____Каміла СТОРЧАК
«____» _____20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ СТУДЕНТУ
Буханчуку Богдану Вікторовичу
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: «Штучний інтелект для удосконалення автоматичної обробки мови та текстів»

керівник кваліфікаційної роботи Оксана ТКАЛЕНКО, к.т.н., доцент
(ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «24» лютого 2025 року № 56.

2. Строк подання кваліфікаційної роботи: 30 травня 2025 року.

3. Вихідні дані до кваліфікаційної роботи: датасет "movie_reviews";
бібліотеки Python: NLTK, Scikit-learn;
test_size = 0.3;
тип ознак екземплярів датасету "movie_reviews" - текстові;
науково-технічна література з питань, пов'язаних з AI, ML.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз сучасних підходів до обробки мови та тексту.

2. Застосування машинного навчання для розуміння та використання тексту.
3. Моделі AI з використанням технології NLP.

5. Перелік ілюстративного матеріалу: *презентація*
1. Технологія NLP в області штучного інтелекту.
2. Задачі NLP.
3. Структура роботи з текстом.
4. Етапи аналізу та обробки даних в NLP.
5. Датасет.
6. Мови Python та R у контексті NLP.
7. Вибір середовища розробки ML-моделі. Вибір мови програмування для розробки ML-моделі.
8. Розробка моделі машинного навчання з використанням технології NLP.
9. Оцінка ефективності ML-моделі.

6. Дата видачі завдання: 24 лютого 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	24.02 – 28.02.25	
2	Дослідження методів автоматичної обробки текстів на основі AI	03.03 – 07.03.25	
3	Дослідження методів машинного навчання	10.03 – 14.03.25	
4	Підходи для розробки ML-моделей з використанням NLP	17.03 – 28.03.25	
5	Робота з текстом на основі даних	31.03 – 10.04.25	
6	Розробка моделі AI з використанням технології NLP	11.04 – 24.04.25	
7	Оформлення роботи: вступ, висновки, реферат	25.04 – 14.05.25	
8	Розробка демонстраційних матеріалів	15.05 – 30.05.25	

Здобувач вищої освіти _____ Богдан БУХАНЧУК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник роботи
кваліфікаційної роботи _____ Оксана ТКАЛЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 62 стор., 20 рис., 6 табл., 20 джерел.

Мета роботи – аналіз методів штучного інтелекту для автоматичної обробки природної мови (NLP), розробка моделі машинного навчання.

Об'єкт дослідження – процес розробки та навчання моделі ML, використовуючи методи обробки природної мови (NLP).

Предмет дослідження – алгоритми машинного навчання, що використовуються для задач NLP.

Короткий зміст роботи: Досліджені сучасні підходи до обробки мови та текстової інформації, методи автоматичної обробки текстів на основі штучного інтелекту. Обгрунтовано застосування машинного навчання для удосконалення автоматичної обробки мови та текстів. Здійснено обробку і аналіз даних з використанням засобів ML в інтерактивному середовищі Jupyter Notebook, яке є популярним інструментом серед дослідників, програмістів та аналітиків даних. Обгрунтовано вибір мови програмування та середовища розробки моделі ML. Визначені особливості роботи з датасетом. Здійснено вибір алгоритму навчання моделі ML. Розроблено модель ML для автоматичної класифікації відгуків про фільми на позитивні та негативні, використовуючи методи обробки природної мови (NLP) та машинного навчання. Виконано оцінку якості роботи моделі.

КЛЮЧОВІ СЛОВА: ДАНІ, МАШИННЕ НАВЧАННЯ, ІНФОРМАЦІЯ, МОВА ПРОГРАМУВАННЯ PYTHON, ОБРОБКА, ПРИРОДНА МОВА, АЛГОРИТМ, ТЕКСТ, ПРОТОКОЛ, ІНТЕРАКТИВНЕ СЕРЕДОВИЩЕ.

ЗМІСТ

ВСТУП……..8
РОЗДІЛ 1. АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО ОБРОБКИ МОВИ ТА ТЕКСТУ……..10
1.1 Роль штучного інтелекту у сфері обробки природної мови (NLP)…….10
1.2 Використання NLP у різних галузях……..14
1.3 Методи та алгоритми автоматичної обробки текстів на основі AI……....18
РОЗДІЛ 2. ЗАСТОСУВАННЯ МАШИННОГО НАВЧАННЯ ДЛЯ РОЗУМІННЯ ТА ВИКОРИСТАННЯ ТЕКСТУ…….…23
2.1 Дослідження підходів для розробки моделей машинного навчання з використанням NLP……...23
2.2 Робота з текстом на основі даних……..28
2.3 Виділення ознак. Дослідження способів представлення слів…….32
2.4 Ембеддінг-спосіб представлення об’єктів…….....36
РОЗДІЛ 3. МОДЕЛЬ AI З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ NLP……...42
3.1 Вибір бібліотеки для розробки моделі……..42
3.2 Аналіз та обробка даних…….....46
3.3 Вибір методу обробки текстових даних…….....49
3.4 Оцінка ефективності моделі AI……..54
ВИСНОВКИ……...59
ПЕРЕЛІК ПОСИЛАНЬ…….61
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)……63

ВСТУП

Актуальність теми. У сучасному світі, де обсяги текстової інформації стрімко зростають, автоматична обробка природної мови (Natural Language Processing, NLP) стає критично важливою технологією для різних сфер діяльності: бізнесу, медицини, освіти, юриспруденції, аналітики та комунікацій. Значний прогрес у галузі штучного інтелекту, зокрема у машинному та глибокому навчанні, та трансформерних моделях (BERT, GPT) дозволяє значно підвищити ефективність обробки текстової інформації, автоматизувати рутинні процеси та покращити якість взаємодії людини з технологіями.

Зростання обсягів текстових даних забезпечує необхідність автоматизації їхньої обробки для швидкого аналізу та прийняття рішень. Підвищення вимог до якості обробки тексту, точність аналізу тональності, машинного перекладу та чат-ботів стає критично важливою. На теперішній час важливим аспектом є впровадження штучного інтелекту у бізнес-процеси для автоматизації підтримки клієнтів, аналітики, генерації контенту та персоналізації взаємодії.

Також, актуальність теми бакалаврської кваліфікаційної роботи визначається постійним розвитком нових технологій штучного інтелекту та соціальною значущістю NLP-технології. Дослідження методів штучного інтелекту для удосконалення автоматичної обробки мови та текстів є вкрай важливим і має велике практичне значення для подальшого розвитку інформаційних технологій.

Метою роботи є аналіз методів штучного інтелекту для автоматичної обробки природної мови (NLP), розробка моделі машинного навчання.

Для досягнення поставленої мети необхідно виконати наступні *завдання*:

- дослідити технологію NLP в області штучного інтелекту;
- дослідити методи обробки інформації в області NLP;
- дослідити алгоритми автоматичної обробки текстів на основі штучного інтелекту;
- розробити модель машинного навчання з використанням технології NLP;
- оцінити якість роботи моделі.

Об'єкт дослідження - процес розробки та навчання моделі ML, використовуючи методи обробки природної мови (NLP).

Предметом дослідження є алгоритми машинного навчання, що використовуються для задач NLP.

Методи дослідження: огляд існуючих технологій, емпіричні методи; метод комп'ютерного моделювання, методи машинного навчання, аналіз літературних джерел та огляд існуючих рішень.

Наукова новизна отриманих результатів: розроблено та проаналізовано модель AI, використовуючи методи обробки природної мови (NLP) та машинного навчання з урахуванням сучасних алгоритмів та методів оптимізації.

Практична значущість кваліфікаційної бакалаврської роботи. Результати дослідження можуть бути використані для розробки та впровадження моделей машинного навчання у різних сферах, де важливе використання методів обробки та аналізу природної мови. Також, результати можуть бути використані як навчальний матеріал для студентів, дослідників і розробників, які працюють у сфері штучного інтелекту для удосконалення автоматичної обробки мови та текстової інформації.

Апробація результатів бакалаврської роботи:

1. Буханчук Б. В. «Оптимальний алгоритм виявлення сигналу з невідомою амплітудою у некоррельованому шумі». Стаття у науковому журналі «Телекомунікаційні та інформаційні технології», м.Київ - №1 (74) 2022. – С.47-53.

2. Буханчук Б. В. «IoT для «розумних міст» та промисловості». Тези доповіді на Всеукраїнській науково-технічній конференції «Сучасний стан та перспективи розвитку IoT». – Київ, 7 квітня 2023 року. – С.184-185.

1 АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО ОБРОБКИ МОВИ ТА ТЕКСТУ

1.1 Роль штучного інтелекту у сфері обробки природної мови (NLP)

Мовлення людини відрізняється від мовлення машини. Мова машин проста, логічна і структурована, у свою чергу людська мова є різноманітною з точки зору побудови словесних конструкцій, використання багатозначних слів та контекстного значення розмови в цілому.

Люди спілкуються словами, а машини – числами. Будь-яке звернення людини до машини голосом або текстом потребує його перекладу на машинну мову чисел. Далі машина повинна зрозуміти, ідентифікувати та розпізнати сутність наміру людини, з яким вона до неї звертається. І після цього – прийняти певне рішення, виконати якусь функцію або відповісти. Цей процес ідентичний спілкуванню людей між собою без придавання значення, завдяки чому та яким чином ми можемо вирізнити той або інший зміст у нашому спілкуванні. Машина ж потребує детальних пояснень на зрозумілій їй логічній мові. Для цього всього і призначена технологія NLP (Natural language processing).

NLP означає «Natural language processing» (природну обробку мови) і відноситься до галузі штучного інтелекту, яка займається взаємодією між комп’ютерами та людьми, використовуючи природну мову. NLP досліджує та розвиває методи обробки, розуміння та генерації людської мови, зокрема писемного та усного тексту.

Основна мета NLP полягає в тому, щоб комп’ютери могли ефективно розуміти, аналізувати, інтерпретувати та відтворювати людську мову у своїх різноманітних формах. Це включає розпізнавання та розуміння природних мовних виразів, переклад тексту з однієї мови на іншу, створення систем діалогу між комп’ютером та користувачем, аналіз відгуків та настроїв людей на основі тексту, автоматичну генерацію тексту.

Усе, що ми висловлюємо письмово або усно, переносить велику кількість інформації. Тема, яку ми обираємо, наш тон, наш вибір слів – все це добавляє деяку інформацію, яку можна інтерпретувати, витягаючи з неї певний зміст.

Теоретично ми можемо зрозуміти і навіть передбачити поведінку людини, використовуючи цю інформацію. Але одна людина може згенерувати декларацію обсягом у сотні або навіть тисячі слів, що складається з речень різної складності. Якщо вас цікавлять великі масштаби і нам потрібно проаналізувати кілька сотень, тисяч або навіть мільйонів людей або декларацій для певного регіону, то в деякий момент ця задача може стати абсолютно невирішуємою [1-3].

Дані, які отримані з розмов, декларацій та інших джерел є типовим прикладом неструктурованих даних. Неструктуровані дані не вписуються у традиційну структуру рядків і стовпців реляційних баз даних і представляють переважну більшість даних, доступних у реальному світі. Такі дані невпорядковані і їх важко обробляти. Але завдяки прогресу у таких дисциплінах, як машинне навчання, на теперішній час у цій сфері відбувається революція. Сьогодні йдеться вже не про спроби інтерпретувати текст або мову на основі ключових слів, а про розуміння значення цих слів (когнітивний спосіб). Сучасні розробки дозволяють ідентифікувати жанри мови, наприклад, іронію або навіть проводити аналіз тональності тексту.

Технології AI (Artificial Intelligence, штучний інтелект), ML (Machine Learning, машинне навчання), DL (Deep Learning, глибоке навчання) і DS (Data Science, наука про дані) мають певні спільні риси, коли їх застосовують в NLP (рис. 1.1, рис. 1.2). Всі ці технології сприяють розвитку і вдосконаленню задач в NLP, але мають свої особливості в підходах до вирішення завдань. AI є широким поняттям, яке включає всі технології, спрямовані на створення систем, що можуть виконувати завдання, які вимагають "інтелекту" людини, зокрема в контексті NLP. AI використовується для загальних задач, таких як обробка мови, переклад, генерація тексту, розпізнавання мови. Штучний інтелект включає всі методи від класичних до новітніх, наприклад, використання алгоритмів пошуку, логічного виведення і навіть експертних систем.

Машинне навчання є підмножиною штучного інтелекту і в його основі лежить ідея навчання на даних. У ML системи автоматично «вчаться» з даних, не маючи чітких правил.

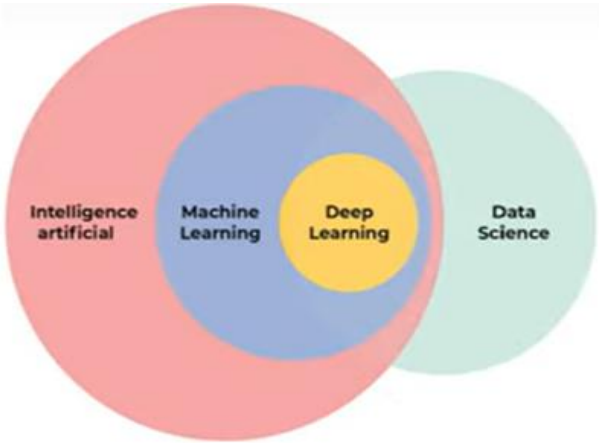

Рис. 1.1 Роль штучного інтелекту в аналізі та обробці даних

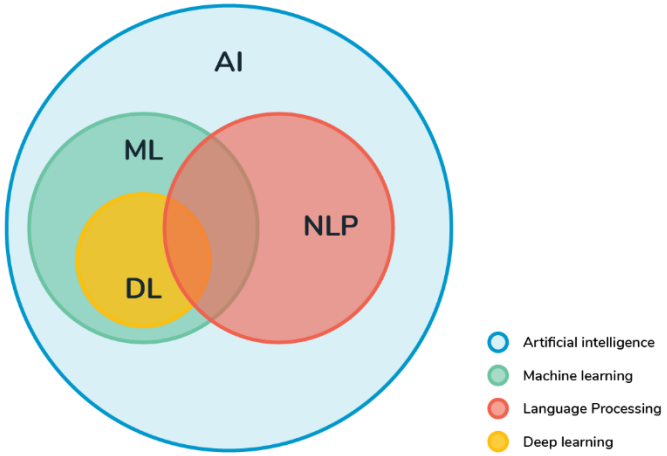

Рис. 1.2 Технологія NLP в області штучного інтелекту

Досліджуючи застосування машинного навчання в NLP, слід виділити такі задачі для його застосування як класифікація тексту (наприклад, класифікація відгуків як позитивних або негативних), аналіз настроїв, розпізнавання іменованих сутностей (NER), переклад тексту, в основі яких часто лежать алгоритми машинних навчальних моделей [5-8].

DL - це підгалузь машинного навчання, яка використовує нейронні мережі з багатьма шарами для навчання на великих обсягах даних. У порівнянні з традиційним ML, DL здатне вирішувати складніші задачі завдяки великим обсягам даних та обчислювальним потужностям. Технологія NLP використовує наступне застосування методів глибокого навчання:

- трансформери та нейронні мережі для задач, таких як переклад тексту, генерація тексту, розпізнавання мови;

- BERT, GPT, T5 - сучасні моделі для аналізу тексту, які використовують глибоке навчання;
- аналіз контексту у тексті за допомогою контекстуальних представлень слів, що є дуже ефективним способом у порівнянні з класичними методами.

Наука про дані займається обробкою та аналізом великих масивів даних. В NLP DS використовує техніки з обробки даних для підготовки, очищення, візуалізації даних, а також для отримання корисних інсайтів. DS в NLP використовується для попередньої обробки даних (токенізації, лематизації, видалення стоп-слів); аналізу трендів у текстах (наприклад, частота появи певних слів); візуалізації даних (наприклад, для відображення тематики текстів або класифікації) та аналізу великих даних (Big Data) у контексті обробки тексту (рис. 1.3).

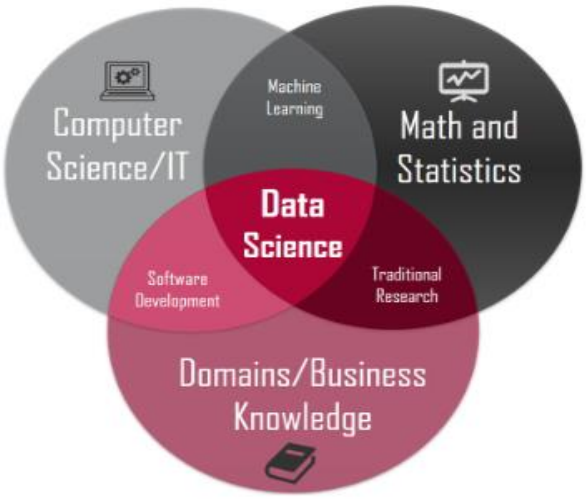

Рис. 1.3 Data Science

Ключовими особливостями Data Science є: робота з великими обсягами даних, аналіз та обробка даних, використання штучного інтелекту та машинного навчання, візуалізація даних, прогнозування та прийняття рішень, автоматизація процесів. Data Science включає аналіз Big Data - великих масивів інформації, які традиційні методи не можуть ефективно обробляти. Data Science застосовує ML та DL.

Спільні риси між технологіями AI, ML, DL, DS в NLP приведені у табл. 1.1.

Таблиця 1.1

Спільні риси між технологіями AI, ML, DL, DS в NLP

№ п/п	Спільні риси	Технології AI, ML, DL, DS, NLP
1	Обробка та аналіз текстових даних	Усі технології використовують текстові дані для навчання, аналізу та генерування результатів.
2	Навчання на даних	Кожна з цих технологій (особливо в ML та DL) вимагає великих обсягів даних для створення моделей.
3	Автоматизація та поліпшення точності	Всі підходи використовуються для автоматизації завдань NLP, наприклад, таких як аналіз тексту, класифікація.
4	Моделі	Всі технології створюють моделі, які можна застосовувати для прогнозування або класифікації текстових даних, навіть якщо ці моделі мають різні складності та підходи.
5	Оптимізація	Всі технології сприяють покращенню ефективності та точності обробки природної мови через різні підходи до моделювання та аналізу даних.

Всі технології в контексті NLP сприяють вирішенню задач через обробку текстових даних. AI є базовою технологією, яка охоплює всі інші, тоді як ML і DL застосовуються для побудови точніших моделей на основі великих обсягів даних. DS, в свою чергу, є важливою складовою для аналізу та підготовки даних для подальшого застосування в NLP.

1.2 Використання NLP у різних галузях

NLP (Natural language processing) – це задачі з області аналізу даних, які використовують дані, що представлені у вигляді тексту. Найрозповсюдженішим прикладом задач NLP є задача класифікації документів. Наприклад, ми маємо якийсь датасет статей і необхідно віднести їх до тієї або іншої категорії (наприклад, статті про спорт, про технології, про моду).

NLP пов’язана з іншими суміжними областями аналізу даних, наприклад, задача перекладу мови у текст. З одного боку ми маємо аудіосигнал, з іншого боку ми повинні отримувати текст. Ця задача більш комплексна та складна з

огляду на те, що дані не тільки використовуються у вигляді тексту, але й у вигляді сигналу (це задача транскодування звуку).

Рис. 1.4 Датасет статей для розбиття на категорії

Актуальним є використання чат-ботів. Чат-боти впроваджуються у великі фірми та корпорації, заміняють при цьому ручну роботу. Аудіо-боти заміняють контакт-центри, автоматизують спілкування з клієнтами – це великий напрямок, який затребуваний у бізнесі.

Рис. 1.5 Chat-bot, Text summarization

Цікавий напрямок – це Text summarization – виділення головного у тексті. Може бути корисним для аналізу великої кількості статей або документів, коли необхідно виділити тільки головну суть кожного документу.

Важливим є переклад з однієї мови на іншу. Ця задача вирішується досить давно і за останній час отримані значні просування в цьому напрямку за допомогою нейронних мереж. Для цього потрібно знати основні архітектури, яким чином здійснюється переклад і задачі, які відносяться до задач перекладу.

Важливою задачею є Named Entity Recognition – виділення сутності у тексті, коли важливо зрозуміти інформацію про якусь персону. Це цікава задача, затребувана у бізнесі і треба знати підходи до її рішення (рис. 1.6).

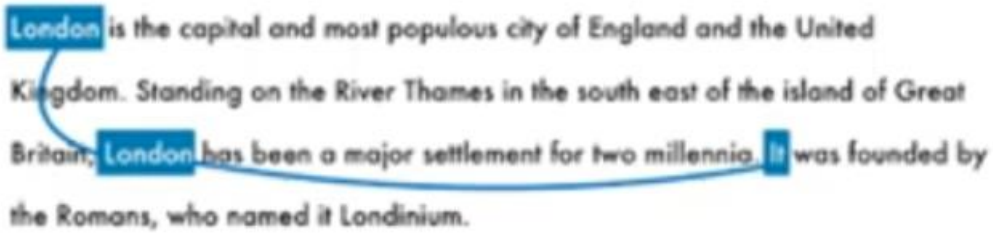

Рис. 1.6 Задача Named Entity Recognition

Провівши дослідження, було визначено та надані рекомендації, у якій послідовності доцільно вивчати технологію NLP. Спочатку потрібно ознайомитися з векторним представленням слів Embeddings. Для цього є алгоритми. Потрібно знати застосування Embeddings для вирішення класичних задач NLP. Потрібно знати застосування згорткових нейронних мереж в NLP, як правильно будувати згорткові нейронні мережі [9, 10].

На базі рекурентних нейронних мереж можна вирішувати задачі про звук. Для цього потрібно знати основні архітектури рекурентних нейронних мереж та їх застосування для вирішення ряду задач.

Робота із звуковою інформацією також є важливою задачею. Потрібно вміти робити транскрипатори звуку у текст.

Отже, слід зазначити, що NLP (Natural Language Processing) - це галузь штучного інтелекту, яка займається взаємодією між комп'ютерами та людськими мовами. Основна мета NLP полягає в тому, щоб дати комп'ютерам можливість розуміти, інтерпретувати і генерувати людську мову таким чином, щоб це було корисно.

Основні завдання NLP включають:

1. *Розпізнавання мови*: виявлення та перетворення голосу в текст.

2. *Аналіз тексту:* аналіз граматики та структури тексту.

3. *Семантичний аналіз:* розуміння значення слів і речень.

4. *Синтез мови:* генерація людською мовою відповідей на основі текстового вводу.

5. *Машинний переклад:* автоматичний переклад тексту з однієї мови на іншу.

6. *Сентимент-аналіз:* визначення настрою або емоцій, виражених у тексті.

7. *Витягування інформації:* витягування корисної інформації з тексту.

8. *Відповіді на запитання:* автоматичне надання відповідей на запитання на основі текстової інформації.

NLP використовує методи комп'ютерної лінгвістики, статистики, машинного навчання та глибокого навчання для виконання цих завдань.

Вивчення технології NLP можна розділити на етапи, кожен з яких охоплює певні аспекти цієї галузі. До цих етапів відносяться наступні: основи програмування та математики; основи обробки тексту; основи Natural Language Processing; машинне навчання для NLP; розширені методи NLP; глибоке навчання для NLP; практичні проєкти та кейси; розширені теми; ресурси для навчання. Послідовне вивчення цих тем допоможе отримати глибоке розуміння технології обробки природної мови та тексту [11].

Визначивши етапи для вивчення технології NLP, можна виконати їх структурування. Аналізуючи етап програмування та математики, потрібно розпочати вивчення мови програмування Python, оскільки так як ця мова є найбільш популярною у сфері NLP. При цьому потрібно мати математичні знання з таких напрямків як лінійна алгебра, ймовірності та статистики, оскільки вони потрібні для розуміння алгоритмів машинного навчання.

Для забезпечення обробки текстової інформації потрібно вивчити основи роботи з текстом за допомогою регулярних виразів (Regex); вміти здійснювати розділення тексту на слова або речення (токенізація); вміти здійснювати перетворення тексту до стандартизованого формату (наприклад, приведення до нижнього регістру, видалення пунктуації), тобто нормалізацію.

Потрібно знати деякі NLP бібліотеки, такі як NLTK, SpaCy та Gensim, фреймворки TensorFlow та PyTorch, знати методи приведення слів до їх кореневих форм – стемінг і лематизацію, прості моделі представлення тексту - Bag of Words (BoW) і TF-IDF.

Що стосується машинного навчання для NLP, потрібно знати основи машинного навчання, тобто ознайомитися з бібліотеками Scikit-learn та TensorFlow/PyTorch. Знати, за якими принципами відбувається класифікація

тексту, вивчивши такі методи класифікації як логістична регресія, наївний Байєс, SVM. Знати методи кластеризації, такі як k-середні та ієрархічна кластеризація.

Для того, щоб бути грамотним спеціалістом в області NLP можна рекомендувати вивчення розширених методів NLP. До таких методів відносяться наступні:

- *Word Embeddings:* концепції Word2Vec, GloVe, FastText;
- *рекурентні нейронні мережі (RNN) та LSTM:* для роботи з послідовностями тексту;
- *Seq2Seq-моделі:* для задач перекладу та генерації тексту.

Глибоке навчання для NLP передбачає використання трансформерів з архітектурами BERT, GPT, T5 та Fine-tuning - налаштування попередньо натренованих моделей на конкретні задачі.

Важливими розширеними темами в області NLP є семантичний пошук, а саме методи пошуку інформації в тексті; розуміння тексту - розпізнавання іменованих сутностей (NER), резюмування тексту. Важлимим є етичні аспекти, тобто вивчення питань етики, які пов'язані з NLP, такі як упередження в моделях.

Послідовність вищевказаного допоможе отримати глибоке розуміння технологій NLP та їх застосування.

1.3 Методи та алгоритми автоматичної обробки текстів на основі AI

Автоматична обробка текстів використовує різні методи та алгоритми для аналізу, розуміння та генерації текстових даних. Основні методи можна розділити на *правильні (лінгвістичні), статистичні* та *машинного навчання*. Методи автоматичної обробки тексту постійно вдосконалюються завдяки розвитку AI та машинного навчання. Поєднання традиційних лінгвістичних методів, статистики та нейронних мереж дає найкращі результати в NLP.

Лінгвістичні (правильні) методи базуються на правилах, розроблених експертами та лінгвістичних моделях мови. До них відносяться 6 методів, які приведені у табл. 1.1.

Таблиця 1.1

Лінгвістичні (правильні) методи

№ п/п	Назва методу	Призначення
1	Токенізація	Поділ тексту на слова, речення або символи
2	Морфологічний аналіз	Визначення частин мови
3	Лематизація	Приведення слів до початкової форми
4	Стеммінг	Усічення слів до кореня
5	Синтаксичний аналіз	Визначення структури речення та взаємозв’язків між словами
6	Розпізнавання іменованих сутностей	Виявлення імен, дат, локацій у тексті

Статистичні методи – це методи, які використовують ймовірності та частоти слів. До них відносяться:

- TF-IDF (Term Frequency - Inverse Document Frequency) – метод оцінки значущості слів у тексті;
- N-грамна модель - аналіз послідовностей слів для прогнозування наступного слова;
- латентне семантичне аналізування (LSA, Latent Semantic Analysis) - виявлення зв’язків між словами та темами тексту [12, 13].

Обробка природної мови, є областю штучного інтелекту, яка зосереджена на здатності машин читати, розуміти та витягувати значення з людської мови. Це дисципліна, яка зосереджена на розробці та застосуванні сучасних підходів з Data Science до людської мови та знаходить своє практичне застосування у все більшій кількості різних галузей.

NLP представляє собою групу технік автоматичної обробки природної людської мови у формі усних або текстових даних. Хоча сама концепція цікава, справжня цінність цієї технології полягає у її практичному застосуванні. NLP може допомогти з широким спектром завдань.

NLP дозволяє розпізнавати та прогнозувати хвороби на основі електронної медичної документації та усної мови пацієнта. На теперішній час проводиться багато досліджень, які спрямовані розкрити потенціал цієї технології для різних

станів здоров’я, які варіюються від серцево-судинних захворювань до депресії і навіть шизофренії. Прикладом напрацювань у цій сфері є Amazon Comprehend Medical - сервіс, який використовує NLP для вилучення даних про хвороби, медичні препарати і результати лікування з історій хвороб, звітів про клінічні дослідження та іншої електронної медичної документації.

За допомогою NLP компанії можуть визначати, що клієнти говорять про їхні послуги або продукти, ідентифікуючи та витягуючи інформацію з таких джерел, як соціальні мережі. Такий аналіз тональності дописів може надавати багато корисної інформації про вибір клієнтів і фактори, які впливають на їх рішення.

Винахідник з IBM розробив когнітивного помічника, який працює як персоналізована пошукова система, яка може вивчати все про нас, а потім нагадати нам ім’я, пісню або що-небудь інше, що ми не можемо згадати саме тоді, коли нам це потрібно. Компанія Google фільтрує і класифікує наші електронні листи за допомогою NLP, аналізуючи текст в електронних листах, які проходять через їх сервери, завдяки чому відбувається фільтрація спаму до того, як він потрапить до нашої поштової скриньки [14].

Щоб допомогти ідентифікувати фейкові новини, команда NLP з Массачусетського технологічного інституту розробила нову систему для оцінки того, є джерело надійним або політично упередженим, допомагаючи визначити чи можна довіряти певному джерелу новин. Alexa від Amazon і Siri від Apple є яскравими прикладами інтелектуальних голосових інтерфейсів. Вони використовують NLP, щоб реагувати на голосові команди та виконувати на їх основі низку ряд завдань, наприклад, знаходити певний магазин, повідомляти нам прогноз погоди, запропонувати найкращий маршрут до офісу або ввімкнути світло вдома.

Розуміння того, що відбувається у світі та що зараз обговорюють люди, може бути дуже цінним для фінансових трейдерів. NLP використовується для відслідковування новин, звітів, коментарів щодо можливого злиття між компаніями - все це потім може бути подано в алгоритм для біржової торгівлі з метою максимізації прибутку. NLP також використовується на етапах пошуку та

відбору перспективних кадрів, визначення навичок потенційних співробітників, а також виявлення потенційних клієнтів до їх активності на ринку праці.

Компанія LegalMation розробила платформу для автоматизації судових задач на основі технології NLP. IBM Watson, яка допомагає юридичним відділам економити час, скорочувати витрати та змінювати свій стратегічний фокус.

NLP має важливе значення застосування у сфері охорони здоров'я. Ця технологія допомагає покращити надавання медичної допомоги, діагностику захворювань та знижує витрати. Особливо цьому сприяє той факт, що організації охорони здоров'я масово переходять на електронні способи обліку медичних документів. Той факт, що клінічна документація може бути покращена, означає й те, що пацієнти можуть бути краще зрозумілими та отримують більш якісне медичне обслуговування. Однією з головних цілей є оптимізація їх досвіду і декілька серйозних організацій вже працюють над цим.

Такі компанії, як Winterlight Labs, досягають значних успіхів у лікуванні хвороби Альцгеймера, відстежуючи зниження когнітивних функцій за допомогою усної мови, а також підтримують клінічні випробування та дослідження широкого спектру інших розладів центральної нервової системи. Дотримуючись подібного підходу, Стенфордський університет розробив Woebot – бота-терапевта, який призначений для допомоги людям із тривогою та іншими розладами.

Однак навколо цієї теми все ще відбуваються серйозні суперечки. Кілька років тому компанія Microsoft продемонструвала, що, аналізуючи великі вибірки пошукових запитів, вони можуть ідентифікувати інтернет-користувачів з раком підшлункової залози ще до того, як їм був поставлений діагноз цього захворювання. Але потрібно розуміти, як користувачі відреагують на такий діагноз і що може статися, якщо наш тест виявиться хибнопозитивним, тобто, що у нас можуть діагностувати хворобу, хоча насправді її немає. Це схоже на випадок Google Flu Trends, який у 2009 році був оголошений як здатний передбачати спалахи грипу, але пізніше зник через низьку точність і невідповідність прогнозованим показникам.

Основні виклики, з якими приходиться стикатися в NLP сьогодні, пов'язані з тим, що мова NLP дуже складна. Процес розуміння та обробки мови надзвичайно складний, у зв'язку з чим для вирішення різних задач потрібно використовувати різні методи до того, як все буде пов'язано воєдино. Для реалізації цих технік доцільно використовувати такі мови програмування, як Python та R, концепції в їх основі та NLP-алгоритми, які найчастіше застосовуються.

Python та R є двома популярними мовами програмування для роботи з аналізом даних та машинним навчанням, але вони мають різні підходи та можливості у сфері обробки природної мови NLP (табл. 1.2).

Таблиця 1.2

Мови Python та R у контексті NLP

№ п/п	Параметр	Python	R
1	*Призначення*	Загальне програмування, AI, ML, NLP	Статистичний аналіз, візуалізація, NLP
2	*Популярність*	Дуже популярний у AI/ML/NLP	Використовується у статистиці та аналітиці
3	*Простота синтаксису*	Легкий для читання та написання	Більш складний синтаксис для ML/NLP
4	*Продуктивність*	Висока завдяки оптимізованим бібліотекам	Менш оптимізований для великих NLP-задач
5	*Масштабованість*	Добре масштабується для великих NLP-проєктів	Менш ефективний для великих обсягів даних

Мова програмування Python є найкращим вибором для NLP, особливо якщо йдеться про штучний інтелект, трансформери, глибоке навчання та комерційні проєкти. Мова R є корисною для статистичних NLP-досліджень та аналітики.

2 ЗАСТОСУВАННЯ МАШИННОГО НАВЧАННЯ ДЛЯ РОЗУМІННЯ ТА ВИКОРИСТАННЯ ТЕКСТУ

2.1 Дослідження підходів для розробки моделей машинного навчання з використанням технології NLP

У першому розділі бакалаврської кваліфікаційної роботи були досліджені розповсюджені приклади задач NLP, основні завдання, етапи вивчення NLP-технології, область застосування якої є різноманітною та великою. Але на сьогоднішній день відсутні повноцінні рекомендації стосовно того, яким чином виконувати та справлятися ефективно із завданнями NLP. Для того, щоб ефективно справлятися з завданнями з використанням технології NLP в інформаційних системах обов'язково потрібно використовувати машинне навчання (Machine Learning, ML) у цій сфері. Основою машинного навчання є дані, які потрібно зібрати, виконати їх очистку, здійснити предсталення даних, можна виконати їх класифікацію, інспектування, обов'язково врахувати структуру словника для NLP-задач, розібратися із застосуванням семантики, використанням синтаксису для того, щоб ефективно вміти розробляти моделі машинного навчання для роботи з природною мовою та текстовими даними.

Дані є критично важливими у машинному навчанні, тому що вони визначають якість та ефективність моделі ML. Якість, кількість та репрезентативність даних є ключовими факторами для створення точних, надійних та ефективних моделей машинного навчання.

Якість моделі залежить від якості даних. Якщо дані містять помилки або шум, модель може навчитися хибним залежностям і давати неправильні прогнози. Чим більше релевантних даних, тим краще модель може виявити закономірності та узагальнювати інформацію для нових випадків. Перед навчанням важливо очистити дані, обробити відсутні значення, нормалізувати та перекодувати їх у зручний формат, щоб модель могла ефективно навчатися.

У табл. 2.1 приведені основні типи даних, які використовуються для реалізації моделей ML з використанням технології NLP.

Таблиця 2.1

Типи даних

Табличні дані	**Зображення**	**Текст**	**Звук**
Оцінка кредитоздатності	Виявлення дефектів на виробництві	Виявлення спаму	Ідентифікація по голосу
Ймовірність переходу на інший тарифний план	Самокеровані автомобілі	Перекладачі	Переведення голосу в текст
Контекстна реклама	Ідентифікація по обличчю	Автодоповнення	Видалення шумів

Виконавши дослідження типів даних, визначено типи наборів вищевказаних даних та ознаки, за якими їх класифікують (рис.2.1).

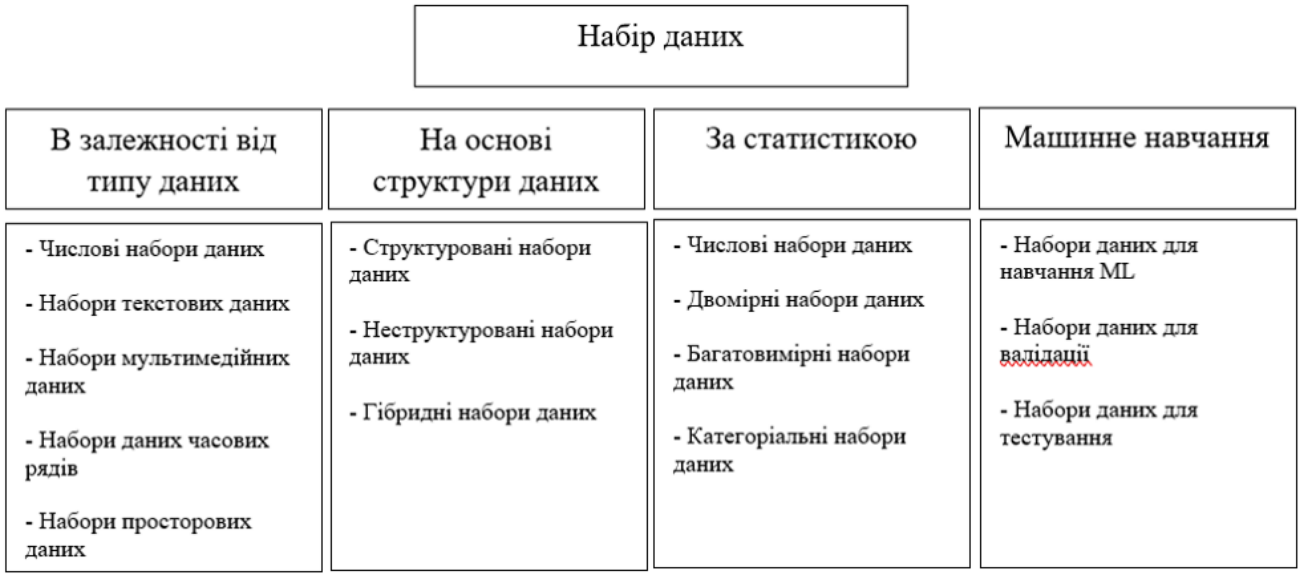

Рис. 2.1 Набори даних

Табличні дані (таблиці, структуровані набори даних) відіграють важливу роль у обробці природної мови (NLP), оскільки допомагають організувати, аналізувати та використовувати текстову інформацію у вигляді структурованих записів.

Табличні дані в NLP застосовуються для:

- зберігання та обробки текстових даних (наприклад, набір відгуків клієнтів у форматі CSV);
- аналізу метаданих (наприклад, дата створення документа, автор, категорія тексту);
- формування навчальних вибірок для ML-моделей;
- зв'язку текстової інформації з іншими структурованими даними (наприклад, співвідношення текстів із числовими характеристиками).

Таблиця може містити тексти (наприклад, відгуки клієнтів) та мітки (позитивний або негативний відгук). Дані можуть включати текст та додаткові параметри, такі як рейтинг або тип користувача. Лог розмови між користувачем і ботом може зберігатися у вигляді таблиці. Табличні дані часто використовуються для підготовки моделей машинного навчання. Таблиці можуть зберігати TF-IDF, Word2Vec, BERT embeddings як числові вектори для моделювання. Наприклад, у таблиці кожен текст може мати набір числових характеристик (табл. 2.2).

Таблиця 2.2

Табличні дані

	Документ 1	Документ 2	Документ 3
high	1	0	0
five	1	0	1
i	0	1	0
am	0	1	0
old	0	1	0
she	0	0	1
is	0	0	1

Частота слів, довжина тексту, кількість унікальних слів можуть бути колонками в таблиці. Також є можливість поєднання NLP з іншими видами даних. Часто текстові дані аналізуються разом із числовими або категоріальними даними, наприклад, аналіз тексту відгуків разом із оцінками користувачів [15].

Для обробки табличних даних у NLP найчастіше використовують:
- Pandas (Python) - *pd.DataFrame* для роботи з CSV/Excel;
- SQL - зберігання текстових даних у реляційних базах;
- Spark DataFrame - розподілена обробка великих текстових масивів.

Табличні дані у NLP допомагають структурувати текстову інформацію, полегшують аналітику та машинне навчання. Вони використовуються для класифікації текстів, аналізу настроїв, чат-ботів, інформаційного пошуку. Поєднання текстових та числових характеристик покращує якість моделей NLP.

Хоча обробка природної мови традиційно працює з текстовими даними, зображення також можуть відігравати важливу роль у цій сфері. Поєднання тексту та зображень часто використовується у задачах мультимодального навчання, де моделі аналізують і взаємодіють з різними типами даних.

Моделі автоматично підписують зображення текстовими описами, що знаходить широке застосування в асистивних технологіях для людей з вадами зору, автоматичному створенні описів товарів в e-commerce, аналізі вмісту відео та фотографій.

Зображення та текст часто використовуються разом для покращення розуміння контенту в NLP. Завдяки мультимодальному навчанню сучасні моделі можуть аналізувати текст у контексті візуальної інформації, що відкриває нові можливості в обробці природної мови.

Текстовий тип даних є основним у Natural Language Processing, оскільки всі алгоритми та методи NLP працюють із текстовою інформацією. NLP дозволяє аналізувати, інтерпретувати, генерувати та змінювати текст, використовуючи різні методи та моделі машинного навчання.

Текст у NLP використовується для:

- розпізнавання, обробки та розуміння мови (наприклад, аналіз новин, соціальних медіа, повідомлень);
- перетворення тексту у векторні представлення для роботи моделей машинного навчання;
- взаємодії людини з комп’ютером (чат-боти, віртуальні асистенти);
- інформаційного пошуку та витягування знань (Google Search, пошук у базах даних);
- перекладу, автоматичної генерації та резюмування текстів.

Текст обробляється та аналізується за допомогою методів NLP, таких як токенізація, векторизація, аналіз настроїв, класифікація. Моделі машинного навчання та нейромережі використовують текст для навчання, наприклад, Word2Vec, BERT, GPT. Текстові дані можна конвертувати у числові вектори, що дозволяє використовувати їх у машинному навчанні. Текст є центральним компонентом NLP і його обробка відкриває широкі можливості для створення розумних систем, чат-ботів, автоматичних перекладачів та аналізаторів тексту.

Звуковий тип даних є важливим у NLP, оскільки мова, яку ми чуємо - це аудіосигнал, який потрібно обробляти для розуміння, аналізу та перетворення в текст. Звук у NLP використовується у таких технологіях, як розпізнавання мовлення (Speech Recognition), синтез мовлення (Text-to-Speech, TTS), діалогові системи та голосові асистенти.

Звук - це аналоговий сигнал, який для обробки в NLP потрібно перевести у цифровий формат (рис. 2.2).

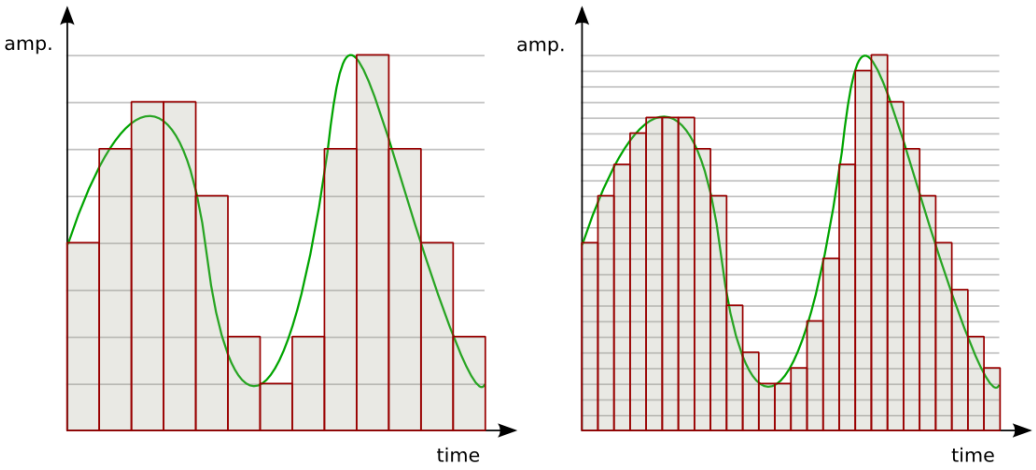

Рис. 2.2 Цифрове представлення аналогового аудіосигналу

Тип даних "Звук" є одним із ключових у NLP, оскільки багато взаємодій людини з комп'ютером відбувається голосом. Розпізнавання мовлення та синтез голосу активно використовуються у чат-ботах, голосових асистентах, розумних пристроях. Звук є важливим типом даних у NLP, оскільки він забезпечує голосову комунікацію між людиною та машиною, що робить технології AI більш природними та доступними.

2.2 Робота з текстом на основі даних

При дослідженні штучного інтелекту для удосконалення автоматичної обробки мови та текстів, доцільно визначити основні аспекти роботи з текстом і як можна вирішувати задачі, виходячи з того, які дані ми маємо.

NLP = linguistic + machine_learning

Якщо приводити аналогію з комп'ютерним зором, то у комп'ютерному зорі є картинки. Потрібно знати як працювати з картинками, як вони представляються у комп'ютері, які операції з ними можливі, яка обробка іноді потрібна для картинок для розуміння як з ними працювати.

Так само потрібно розуміти як працювати з текстами. У текстах інші види даних і тому необхідно розуміти, що з ними робити. Якщо розуміти, що робити з цими даними, не буде проблем, застосовуючи певний алгоритм машинного навчання, можна вирішити задачу автоматичної обробки мови та текстів.

Якщо говорити про структуру роботи з текстом, то тут можна все розділити на 2 процеси. Спочатку потрібно виділити аналіз тексту (Understanding – розуміння), потім - процес генерації тексту (Generation – генерація). Представимо цю структуру у вигляді піраміди, яка складається з блоків (рис. 2.3). Виконуючи аналіз тексту – виконуємо дослідження знизу вгору. Для розуміння тексту нам потрібна морфологія, синтаксис, семантика, прагматика та зміст [16-18].

Для початку необхідне розуміння того, що текст – це слова. Розуміючи форму слів, розуміючи зв'язок між словами, можна розуміти взаємодію між ними. Після того, як ми зрозуміємо взаємодію, ми можемо перейти до значень словосполучень, до значень деяких фрагментів тексту. Прагматика – як ми використовуємо текст, для чого використовуємо, для якої задачі. Далі ми переходимо до самого змісту тексту. Тобто, опираючись на морфологію, синтаксис, семантику і прагматику, ми розуміємо зміст всього сказаного, всього написаного. Зміст ми маємо закодувати. І виходячи з деякого представлення ми

вже можемо генерувати текст. Генерувати можна по різному і різне, все залежить від наших цілей.

Якщо розглядати розуміння і генерацію, то бачимо цикл. Тобто на вхід подається текст, ми його обробляємо, розуміємо, спираючись на всі частини мови (морфологію, синтаксис, семантику, прагматику) і після цього ми виділяємо деякі фрагменти у тексті для того, щоб виділити важливе. Виходячи з цього важливого, ми робимо якийсь висновок про текст.

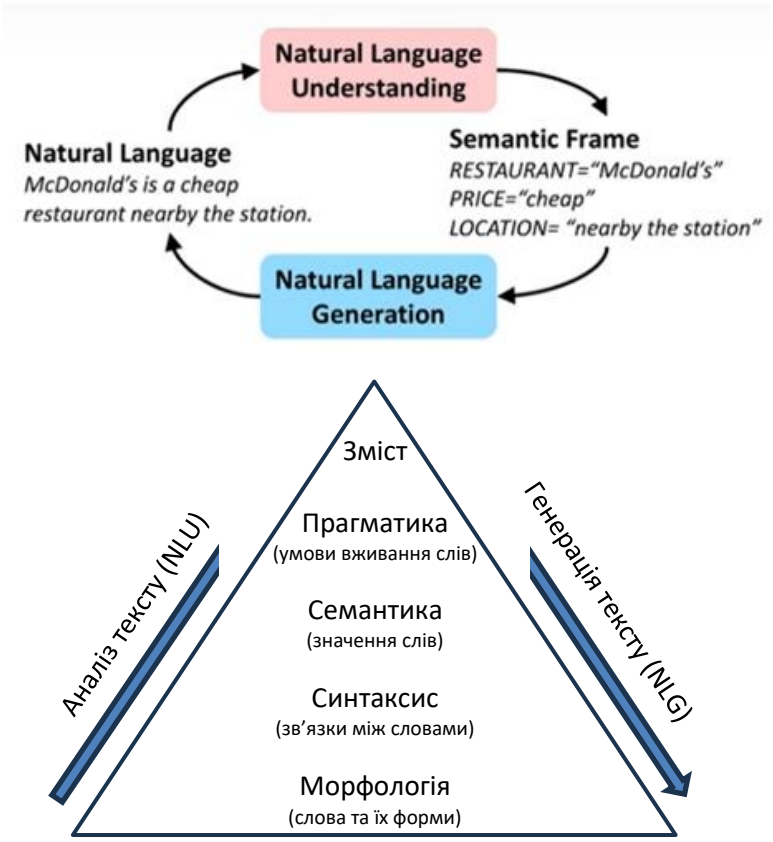

Рис. 2.3 Структура роботи з текстом

Наприклад, ми виходячи з речення, виявляємо що є ресторан, що є ціна і що є місцеположенням. Ми можемо інтерпретувати дану схему як діалог з чат-ботом. Наприклад, голосовий помічник. Ми їй задаємо якесь питання, після цього алгоритм AI розуміє про що ми говоримо, підрозділяє наш текст на логічні фрагменти, після цього він може відповісти – згенерувати текст, який є відповіддю на наше питання. Процес всього NLP – є розуміння деякої генерації. Тобто для того, щоб щось навчати, отримувати якісь моделі, нам потрібно розібратися у самій мові. Без кожного вищевказаного кроку (морфологія,

синтаксис, семантика, прагматика) буде проблематично вирішувати поставлену задачу.

Наприклад, у нас є текст і нам потрібно виконати всі ці етапи. Для цього нам потрібно зробити деякі кроки у нашому процесі. Процес передобробки тексту складається з таких етапів як: токенізація, нормалізація слів (стеммінг, лематизація), видалення слів (стоп-слова; неінформативні слова, шаблони).

Всі кроки виконувати не обов'язково, але є приблизна послідовність рішення, обробки, дослідимо пункти, які будуть входити в цю послідовність. Спочатку здійснюється токенізація. Токенізація – це розбиття тексту на токени. Токен – це не обов'язково слово, токеном може бути пунктуація, пробіл, все що завгодно може бути токеном. Коли ми розбиваємо речення по елементарним елементам, можна сказати, що токенізація відбувається по словам. А можемо розбивати і по реченням в залежності від нашої задачі.

Після того, як ми розбили текст на токени, ми отримали деякий словник – словник всіх токенів. Цей словник описує весь наш датасет. Цей словник може бути до сить великим – досягати мільйону, а то й мільярду слів. Потрібно визначитися, що з цим робити. Для того, щоб зменшити словник, потрібно виконати нормалізацію слова або видалення слів. Ці пункти можна змінювати місцями. Спочатку дослідимо принцип нормалізації.

Є словник. У кожного слова є множина форм – число, рід, відмінок. Іноді ми хочемо це привести до однієї форми для того, щоб зменшити розмір нашого словника. У цьому допомагає нормалізація слова. У нормалізації є 2 підвиди – стеммінг, лематизація. Процес стеммінга – це процес видалення останніх символів, тобто ми робимо стемм – обрізання від всього слова. За рахунок видалення ми намагаємося зробити так, щоб багато слів прийшли до одного слова. Процес лематизації є більш складним. Лематизація – це приведення слова до початкової форми. Для іменника це називний відмінок і однина. Лематизація бореться з тим, що у деяких слів початок співпадає. При стеммінгу можна слова, що мають зовсім різні значення, привести до одного і того ж слова (рис. 2.4).

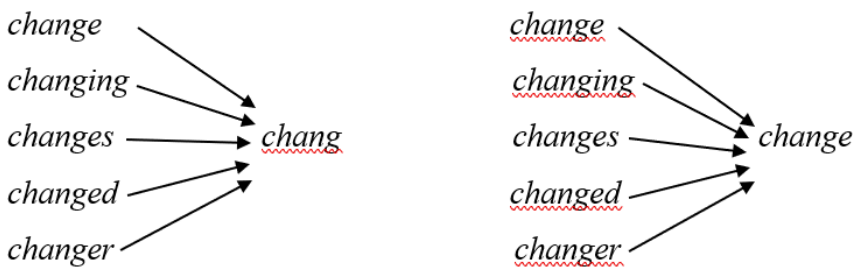

Рис. 2.4 Стеммінг та лематизація

Отже, стеммінг - це процес знаходження основи (кореня) слова шляхом відсікання афіксів (суфіксів, префіксів, закінчень). Лематизація - це процес приведення слова до його основної (словникової) форми, тобто до леми (наприклад, "бігти" замість "біжу", "будинок" замість "будинку").

В цілому, за допомогою нормалізації можна зменшити словник**.** Тепер всі слова стоять у початковій формі. Але навіть якщо слово стоїть у початковій формі, воно може бути для нашого тексту зайвим. Зайвим у якому плані – воно не несе ніякого змісту і не важливе для нашої задачі. Тут можна видаляти стоп-слова - службові слова нашої мови такі як прийменники, дієслова зв’язки, займенники та інші. Стоп-слова можна видаляти не всі, а якусь їх частину. Тобто потрібно видаляти службові слова, на які не падає змістовне логічне навантаження.

Також є неінформативні слова. Вони можуть бути не стоп-словами, але вони нам не потрібні. До неінформативних слів можна віднести різні шаблони. Наприклад, якщо взяти того ж самого листа, то там завжди буде звідки, куди, в кінці буде якась переписка, тобто у кожного листа є свій шаблон. Якщо включити шаблон у словник, то він не буде нести ніякого змістовного навантаження, але збільшить наш словник.

Все, що людина робила раніше, зараз може робити AI: класифікувати тексти за жанром, авторством та іншими критеріями, виправляти помилки, виконувати ранжування пошукової видачі, генерацію тексту, картинок по тексту, машинний переклад, діалогові системи, сумаризація тексту.

Ми розглянули словник, який описує весь наш датасет. Слова з речень, які входять у наш словник, є ознаками даного тексту.

2.3 Виділення ознак. Дослідження способів представлення слів

Необхідно дослідити, яким може бути комп'ютерне представлення слів, адже з ознаками тексту потрібно працювати. Найкращим рішенням є числові ознаки, тому важливо розуміти, яке комп'ютерне представлення слів можливе, як їх зберігати.

Дослідимо способи представлення слів. Перший спосіб називається *One-Hot encoding* (рис. 2.4). Потрібно представляти слова як One-Hot-вектори. Зробили токенізацію, лематизацію, стеммінг, видалили неінформативні слова і тим самим отримали якийсь словник, який описує наш датасет. Припустимо, що розмір словника 500 тисяч слів Кожне слово буде представлятися вектором розмірністю 500 тисяч. Нехай у нашому словнику слово *cat* буде першим. Перша координата вектору буде 1, всі інші 0. Припустимо, що слово *mother* є *i*-тим у нашому словнику, тому *i*-та координата дорівнюватиме 1-ці, всі інші будуть 0. Це перший найпростіший спосіб кодування. Розглянемо, які виникають проблеми [19, 20].

Дослідження показали, що ми не можемо встановити міру близькості між словами. Якщо ми візьмемо скалярне множення між векторами, воно буде нульове, так як ці вектори перпендикулярні один до одного. Розмір векторів дуже великий (500 тисяч), а інформація, яка для нас важлива, тільки одне число, яке говорить нам про зміст слів. 1/500000 – це відсоток корисної інформації серед всієї інформації і цього треба уникнути.

One-hot вектор розміром довжини словника (наприклад, 500000).

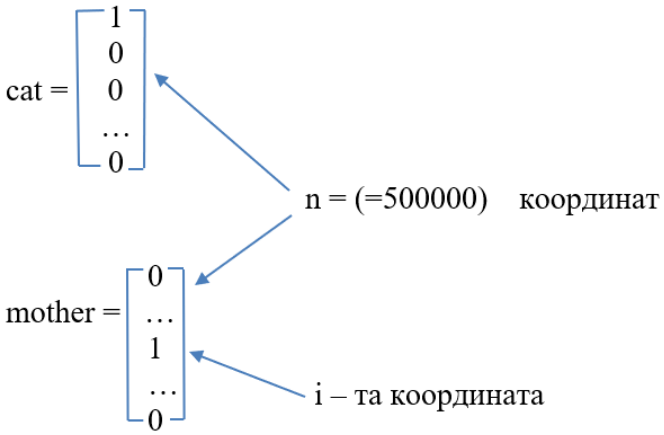

Рис. 2.4 One-Hot encoding

У результаті досліджень, були визначені наступні проблеми даного способу представлення слів:

- вектори не відображають значення слів;
- не можливо встановити міру близькості між словами;
- великий розмір векторів при малій кількості інформації.

Другим способом представлення слів є *Bag-of-Words.* Bag-of-Words – це безпосереднє кодування документа або речення/тексту. Bag-of-Words – це сума всіх векторів слів. Наприклад, є речення і словник. Ми відмічаємо слова, які зустрілися, співставляємо їм одиничку і сумуємо всі ці вектори. Тим самим отримали такий вектор, який описує наше речення. Якщо слово зустрічається декілька разів, то при сумі у нас зустрічається не 1, а 2. Приклад способу представлення слів Bag-of-Words приведений на рис. 2.5.

Сума one-hot векторів слів

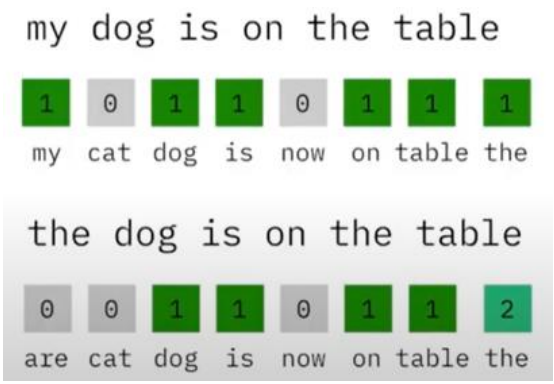

Рис. 2.5 Bag-of-Words

До недоліків даного способу предсталення слів слід віднести відсутність інформації про порядок слів. Bag-of-Words основане на One-Hot Encoding, У нас немає ніякої інформації про значення слів, а є певна кількісна міра. Якщо у нас словник великий, а речення маленьке, то кількість значимих координат не дуже збільшиться. Є мало інформації. Добавляється проблема, що ми маємо текст (речення), а в тексті є порядок слів, так як речення – це послідовність слів. І в даному кодуванні ми ніяк не враховуємо їх порядок.

Тобто у цих двох методах замість позиції під кожне слово ми використовували скільки разів слово зустрічається у цьому тексті – таке число стоїть у цій координаті. Тому, замість цих числових значень ми можемо використовувати таку метрику як *TF-IDF*. TF-IDF визначає, на скільки слово важливе для документа.

Ймовірність зустріти слово *w* у документі *d*:

$$p(w, d) = N_w / N \ , \qquad (2.1)$$

де n_{dw} – кількість входжень слова w у документ d;

N_w – кількість документів, які містять w;

N – кількість документів.

Чим менша ймовірність, тим важливіше слово для документа або навпаки. Наприклад, слово зустрічається у малій кількості тексту і показник $(N_w/N)^{n_{dw}}$ буде маленьким і взагалі вся ймовірність буде маленькою. Якщо слово зустрічалося в 0.01 тексту, то ймовірність буде маленькою. Якщо слово зустрічалося мало, то для цього тексту воно є важливою характеристикою. Чим більше слово зустрілося у цьому документі, тим ймовірність буде меншою, так як чим більший показник $^{n}{}_{dw,}$ тим менша ймовірність. Якщо слово дуже часто зустрічало в документах, то наша основа (N_w/N) велика і ймовірність буде наближена до 1. Якщо наше слово зустрічалося у всіх документах, то не важливо, скільки разів це слово зустрілося у цьому документі, все рівно ймовірність буде 1.

Модель Bag-of-Words не враховує, скільки слово зустрічається в інших текстах. Приклад використання даного способу представлення слів приведений на рис. 2.6.

$$\mathrm{tf}(\text{"this"}, d_1) = \frac{1}{5} = 0.2$$
$$\mathrm{tf}(\text{"this"}, d_2) = \frac{1}{7} \approx 0.14$$
$$\mathrm{idf}(\text{"this"}, D) = \log\left(\frac{2}{2}\right) = 0$$

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

Рис. 2.6 TF-IDF

$$tfidf(«this», d_1, D) = 0.2 \times 0 = 0$$

$$tfidf(«this», d_2, D) = 0.14 \times 0 = 0$$

У результаті аналізу визначено, що слово «this» не є інформативним.

У нас є документ 1 і документ 2. Можна вважати, що всередині кожного документу значення відповідають Bag-of-Words. Порахуємо TF для слова «this» для двох випадків: для документа 1 і для документа 2. У чому перевага підрахунку частоти, а не кількості. Слово «this» зустрічається 1 раз для 1 документу і 1 раз для другого документу, і важко зрозуміти на скільки воно важливіше для документу 1 і для документу 2. Але коли ми вже виконуємо нормалізацію, тобто ділимо на кількість слів, ми розуміємо, що у першому документі на слово «this» потрібно звертати більше уваги, ніж на слово «this» в іншому документі. Порахуємо idf. Idf говорить нам, що це слово зустрічається всюди, тому воно не важливе ні для документу 1, ні для документу 2. Тому метрика tfidf=0 у двох випадках.

Отже, замість Bag-of-Words можна застосовувати значення TF-IDF. Документ буде представлятися вектором, в якому буде зберігатися TF-IDF-слово для даного документу.

Ми рахували TF-IDF, рахували зустрічальність тільки слів, тобто окремих токенів. Ми можемо використовувати N-грами або колокацію. Можемо обчислювати TF-IDF для n-грами, а можемо обчислювати TF-IDF для колокації.

N-грама – послідовність з N слів, які розміщуються поряд.

Колокація – поєднання слів, які не обов’язково розміщуються поряд (рис. 2.7).

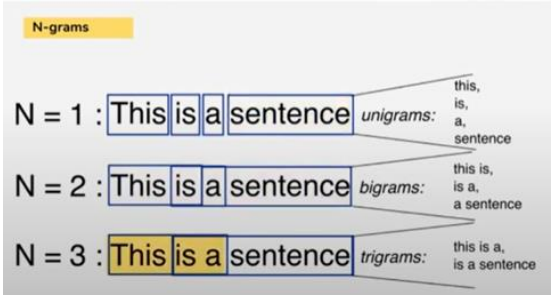

Рис. 2.7 N-грама

Для поєднання слів замість TF-IDF ми використовуємо метрику Pointwise mutual information. PMI - це метрика, яка використовується для оцінки асоціації між двома подіями або словами в тексті. Вона вимірює, наскільки ймовірніше два слова зустрічаються разом, ніж якщо б вони були незалежними.

Формула PMI:

$$PMI(x,y) = \log \frac{P(x,y)}{P(x)P(y)} \quad , \tag{2.2}$$

де P(x, y) – ймовірність одночасної появи слів x і y;

P(x) – ймовірність появи слова x;

P(y) – ймовірність появи слова y.

Pointwise mutual information говорить про те, наскільки невипадково зустрілися 2 слова разом. Така метрика характеризує наскільки слова зустрічаються разом, наскільки їх зустріч невипадкова. Ця метрика враховує те, як часто слово взагалі зустрічається. Буває таке, що слова зустрічаються часто і в силу випадковості вони також зустрічаються разом часто. Чим більше PMI, тим важливіше словосполучення (табл. 2.2).

Таблиця 2.2

Метрика PMI

Word 1	Word 2	Count Word 1	Count Word 2	Count of co-occurrences	PMI
puerto	rico	1938	1311	1159	10.0349081703
hong	kong	2438	2694	2205	9.72831972408
los	angeles	3501	2808	2791	9.56067615065
carbon	dioxide	4256	1353	1032	9.09852946116
prize	laureate	5131	1676	1210	8.85870710982
san	francisco	5237	2477	1779	8.83305176711
nobel	prize	4098	5131	2498	8.68948811416

2.4 Ембеддінг-спосіб представлення об'єктів

Embedding – це коли якимось чином закодували наше слово, виходячи з тексту, якогось сенсу. В цілому це *приховане представлення* (рис. 2.8).

Vocabulary	One-hot	Embedding
1. red	[1, 0, 0, 0, 0, 0, 0]	[1.0, 0.0, 0.0]
2. green	[0, 1, 0, 0, 0, 0, 0]	[0.0, 1.0, 0.0]
3. blue	[0, 0, 1, 0, 0, 0, 0]	[0.0, 0.0, 1.0]
4. yellow	[0, 0, 0, 1, 0, 0, 0]	[1.0, 1.0, 0.0]
5. cyan	[0, 0, 0, 0, 1, 0, 0]	[1.0, 0.0, 1.0]
6. magenta	[0, 0, 0, 0, 0, 1, 0]	[0.0, 1.0, 1.0]
7. gray	[0, 0, 0, 0, 0, 0, 1]	[0.2, 0.2, 0.2]

Рис. 2.8 Embedding

Проходимо вікном по тексту і обчислюємо, скільки разів слово *i* і слово *j* зустрілися разом. Вектор, який описує слово $word_1$, є якраз із зустрічаємістю з іншими словами. Такий вектор розраховуємо для кожного слова. І в даному випадку наш вектор відображає вже деяке значення слова – у цьому векторі зберігається зв'язок з іншими словами.

Після того, як ми отримали представлення, воно до цих пір велике. Наприклад, якщо у нас 500 тисяч слів, то у нас буде вектор 500 000 – 1, так як ми не рахуємо, що слово *i* зустрічалося із словом *i*. 500 000 – це дуже багато і тому можна зробити зменшення розмірності (рис. 2.9, рис. 2.10).

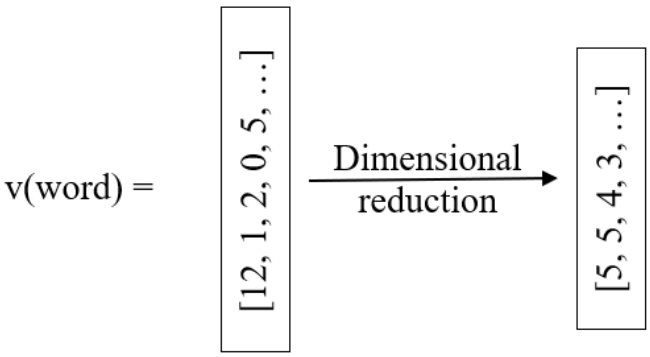

Рис. 2.9 Контекстний ембедінг

У результаті дослідження були визначені недоліки, які має Embedding-представлення слів. До них відносяться:

- слова, які вживаються не часто;
- потрібно багато обчислювальних ресурсів;
- при зміні датасету потрібні нові обчислення.

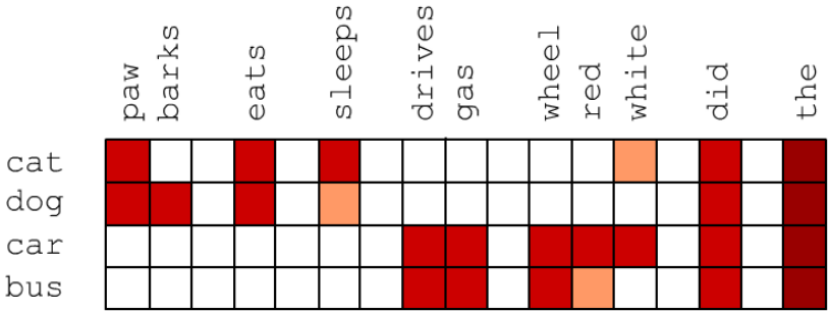

Рис. 2.10 Вектори контексту

Щоб усунути вищевказані недоліки, рекомендовано використовувати SVD-метод представлення слів (англ. Singular Value Decomposition, сингулярне розкладання матриці) (рис. 2.11).

Є вектор представлення для кожного слова. Їх потрібно сконкатенувати і отримати матрицю. Кожний рядок є представлення нашого слова. Наше слово у приведено стовпці є контекстом, ми дивимося скільки разів воно з кожним з'являлося. Після цього ми робимо SVD-розкладання цієї матриці. Якщо матриця квадратна (nxn), то SVD-розкладання також буде nxn. Але для зменшення розмірності потрібно вибирати r-перших стовпців і r-перших рядків. І матриця стає розміром r x r. Тобто ми вибираємо r-стовпців і r-рядків. Тим самим ми скорочуємо розмірність представлення слова з 500000 до розмірності r.

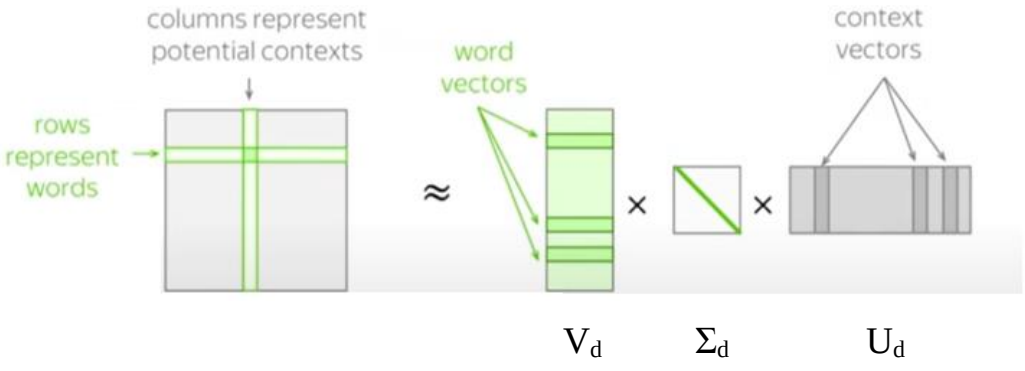

Рис. 2.11 Singular Value Decomposition

Коли ми використовуємо конкатенацію всіх цих векторів, ми використовуємо їх зустрічальність. Зустрічальність обчислюється за допомогою вікна. Можна

використовувати метрику Pointwise mutual information, тобто ми використовуємо не просто скільки разів вони зустрілися, а потрібно подивитися на скільки ця зустріч не випадкова. З цієї точки зору метрика Pointwise mutual information більш досконала і якість ембедінгу виходить кращою при SVD-розкладанні, коли наша матриця складається з чисел Pointwise mutual information.

Замість Counts можемо використовувати метрику Pointwise mutual information, тобто ми використовуємо не просто скільки разів вони зустрілися, а потрібно подивитися, на скільки ця зустріч не випадкова. З цієї точки зору метрика Pointwise mutual information більш досконала і якість ембедінгу виходить кращою при SVD-розкладанні, коли наша матриця складається з чисел Pointwise mutual information.

Якщо v – це центральне слово, а u – це контекст, переходимо до того, що документ – це центральне слово, а весь інший текст цього документу – це контекст. Матриця може складатися з Bag-of-Words. Але ми можемо використовувати більш досконалу метрику TF-IDF, тобто значимість слова u для документу d і будемо виконувати розкладання.

В результаті отримаємо, що перша матриця складатиметься з рядків, які є векторним представленням документів, а остання матриця складатиметься із стовпців, які є вектором представлення слів при розмірності слова r.

Метод SVD використовується в NLP для аналізу тексту та знаходження прихованих зв’язків між словами. SVD знаходить своє застосування у рекомендаційних системах. Netflix, Amazon застосовують SVD для знаходження латентних (прихованих) факторів у даних користувачів та товарів. Використовуючи даний метод при стисненні зображень, визначено, що зображення можна представити у вигляді матриці пікселів, а SVD дозволяє зменшити розмір, зберігши лише важливі компоненти. У системі лінійних рівнянь даний метод використовується для знаходження найкращих апроксимацій при вироджених або погано обумовлених матрицях.

Ембеддінг є способом представлення слів у векторному просторі, спосіб представлення слів у вигляді числових векторів у багатовимірному просторі. Це

дозволяє алгоритмам машинного навчання краще розуміти семантичні та синтаксичні зв’язки між словами.

На відміну від One-Hot encoding (де кожне слово представлено як бінарний вектор), ембеддінги дозволяють моделювати семантичну подібність між словами:
- схожі за значенням слова мають близькі вектори;
- відношення між словами можна виражати математичними операціями.

Якщо потрібно порівняти ембеддінги з традиційними методами представлення тексту (Bag of Words або TF-IDF), то в останніх втрачається контекст слів. Ембеддінги вирішують цю проблему, перетворюючи слова на багатовимірні вектори, що відображають їхній сенс.

Популярними методами створення ембеддінгів є: Word2Vec, GloVe, FastText, BERT.

Word2Vec навчає вектори слів за допомогою нейронної мережі. Використовує дві архітектури:
- CBOW (Continuous Bag of Words), яка прогнозує слово за контекстом.
- Skip-gram прогнозує контекст за словом.

GloVe використовує статистичну матрицю співзустрічей слів у великому корпусі текстів і є більш точним для моделювання семантичних відношень.

FastText є поліпшеною версією Word2Vec. Враховує частини слова (N-грамні підрядки), що допомагає обробляти незнайомі слова, є дуже корисним для морфологічно багатих мов, таких як українська та німецька.

BERT використовує контекстне представлення слів (тлумачить слово залежно від контексту). Підходить для завдань, де важливий контекст речення (переклад, аналіз тональності).

Порівняння методів ембеддінгу приведено у табл. 2.3.

У результаті проведених досліджень визначено, що ембеддінги – це потужний спосіб представлення слів у вигляді векторів, який зберігає семантичні зв’язки між словами. Популярні методи Word2Vec, GloVe, FastText, BERT мають свої переваги і застосовуються залежно від задачі. Сучасні нейромережеві методи, такі як трансформери (BERT, GPT), покращують розуміння контексту і сенсу тексту.

Таблиця 2.3

Порівняння методів ембеддінгу

№ п/п	Метод	Особливості	Задачі
1	Word2Vec	Враховує контекст, швидке навчання	Загальні завдання NLP
2	GloVe	Використовує матрицю співзустрічей	Семантичний аналіз, тематичні моделі
3	FastText	Розбиває слова на підрядки (n-грами), добре працює з морфологічно багатими мовами	Українська, німецька, турецька мови
4	BERT	Контекстне представлення, сильний в аналізі сенсу	Складні завдання NLP (чати, переклад)

Використання ембеддінгів - основа для більшості сучасних NLP-завдань, таких як чат-боти, переклад, аналіз тональності та пошукові системи. Навчання векторних представлень слів з нуля вимагає великих обсягів текстових даних і обчислювальних ресурсів. Ембеддінги можуть комбінуватися з графовими нейромережами, рекомендаційними системами, пошуковими алгоритмами, що розширює їх застосування у реальних задачах.

Майбутнє ембеддінгів - контекстні моделі та мультимодальні представлення. Останні дослідження в NLP рухаються у бік контекстних представлень слів, генеративних моделей та мультимодальних ембеддінгів (де текст поєднується із зображеннями, аудіо та відео). Це відкриває нові можливості для створення інтелектуальних чат-ботів, голосових асистентів, машинного перекладу та інших AI-застосунків.

3 МОДЕЛЬ AI З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ NLP

3.1 Вибір бібліотеки для розробки моделі

Методи машинного навчання, які дозволяють здійснювати автоматичну обробку мови та текстів можна класифікувати на наступні типи: класичне машинне навчання; глибоке навчання та гібридні методи. Якщо аналізувати методи класичного машинного навчання, до основних з них відносяться:

- Байєсівський класифікатор (Naïve Bayes Classifier) - використовується для класифікації текстів (наприклад, спам-фільтри);
- методи опорних векторів (SVM, Support Vector Machine) - ефективний для категоризації текстів;
- кластеризація текстів (K-Means, DBSCAN) - групування схожих текстів.

Застосування нейронних мереж дозволяє значно покращити точність NLP-задач. До методів глибокого навчання належать:

- рекурентні нейронні мережі (RNN, LSTM, GRU) – використовуються для аналізу послідовностей слів (машинний переклад, генерація тексту);
- трансформери (Transformer) - основа сучасних NLP-моделей (GPT, BERT, T5);
- Word Embeddings (Word2Vec, GloVe, FastText) - перетворення слів у векторний вигляд для врахування їхнього контексту.

Сучасні системи обробки текстів комбінують різні сучасні підходи та гібридні методи:

- генеративні моделі (GPT, ChatGPT, BARD) - створення текстів на основі глибокого навчання;
- розпізнавання емоцій та аналіз сентименту – визначення тональності тексту (позитивний, негативний, нейтральний);
- автоматичне резюмування – скорочення текстів із збереженням змісту;
- втоматичний переклад (Google Translate, DeepL) - використання трансформерів для перекладу.

Перейдемо до задачі класифікації текстів. Класифікація текстів використовується для спам-фільтрації, для виявлення негативних або позитивних

коментарів, фейкових новин. Для оцінки тональності тексту скоріш за все використовується регресія. Також класифікація текстів може використовуватися як частина іншої задачі NLP, наприклад, для фільтрації навчальної вибірки, вибору сценарію у діалоговій системи.

Класичними методами класифікації тексту є: Bag of Words, Tf-Idf, Embeddings.

Якщо у нас є текст, спочатку його потрібно токенізувати, для цього потрібно отримати представлення кожного слова, деякий ембеддінг і далі ці ембеддінги потрібно якимось чином акумулювати, роблячи якусь операцію – або усереднення, або сума, або якась більш хитра операція. Наприклад, що ми можемо зробити. Ми можемо враховувати слова з рівною вагою. Ми хочемо змінити вагу. Ми можемо додавати всі ці вектори з коефіцієнтом TF-IDF. Отримаємо приховане представлення всього тексту і вже за допомогою будь-якої нейронної мережі ми можемо вирішувати задачу класифікації. Нейромережа може бути будь-якою. Можна застосовувати згорткову нейронну мережу.

Практичною частиною бакалаврської кваліфікаційної роботи є розробка моделі ML для автоматичної класифікації, наприклад, відгуків про фільми на позитивні та негативні, використовуючи методи обробки природної мови (NLP) та машинного навчання.

У даній задачі вхідними даними є набір текстових відгуків із корпусу NLTK movie_reviews, де кожен відгук має мітку:

- "pos" (позитивний відгук);
- "neg" (негативний відгук).

Вихідними даними є:

- прогноз для нового відгуку, тобто модель повинна визначити є відгук позитивним (1) або негативним (0);
- точність моделі, яка вимірюватиметься за допомогою метрики accuracy.

У задачі буде використовуватися модель "Bag of Words" (BoW) для векторизації тексту. Для навчання рекомендовано використовувати наївний байєсівський класифікатор (Multinomial NB). Датасет розбивається на

тренувальну (70%) і тестову (30%) вибірки. Програма має навчити модель та оцінити її якість, вивівши точність (accuracy) на тестових даних.

У наведеному коді використовується датасет "movie_reviews" з бібліотеки NLTK. Цей датасет містить 1000 позитивних і 1000 негативних рецензій на фільми, взятих з IMDB. Кожен відгук збережений у вигляді текстового файлу та доступний через *movie_reviews.fileids().*

Мова програмування Python має потужну екосистему бібліотек для NLP:

- NLTK - класична бібліотека для базової обробки тексту;
- spaCy - швидка та ефективна бібліотека для сучасного NLP;
- TextBlob - простий у використанні інструмент для аналізу настроїв;
- Scikit-learn використовується для класичних ML-моделей у NLP (наприклад, класифікація тексту);
- Transformers (Hugging Face) - сучасні нейронні моделі (BERT, GPT, T5) для роботи з текстами;
- Keras/Tensorflow - підтримка DL-моделей.

Мова програмування R теж має бібліотеки для NLP, але їх кількість менша:

- tm (Text Mining) - основна бібліотека для обробки текстів;
- quanteda - потужний інструмент для обробки текстів, класифікації, аналізу настроїв;
- tidytext - інтеграція NLP в екосистему tidyverse;
- text2vec - векторизація тексту, робота з Word2Vec.

Для розробки моделі AI з використанням технології NLP була вибрана мова програмування Python. Python є найкращою мовою програмування для задач Natural Language Processing завдяки своїм перевагам, які спрощують обробку текстів, аналіз даних та розробку моделей машинного навчання. Python має зрозумілий синтаксис, що дозволяє легко працювати з текстом без зайвої складності, легко інтегрується з TensorFlow, PyTorch, Scikit-learn, що робить його ідеальним для створення моделей NLP (класичних ML-методів та глибоких нейронних мереж). Python чудово працює з табличними (Pandas, NumPy) та неструктурованими (JSON, XML, CSV) даними, що важливо для NLP.

Завдяки NumPy, Cython, JIT-компіляції Python дозволяє обробляти великі текстові дані з високою швидкістю. Python підтримується хмарними сервісами Google AI, OpenAI, AWS NLP, Azure AI, що полегшує розгортання NLP-рішень у реальних додатках.

Python є гарним вибором для NLP, оскільки він має розвинуту екосистему бібліотек, підтримку ML/DL, гнучкість у роботі з текстом та велику спільноту. Він дозволяє швидко та ефективно створювати рішення для обробки природної мови, починаючи від простого аналізу текстів і закінчуючи складними нейромережами, такими як GPT чи BERT.

Для розробки моделі доцільно використати бібліотеки NLTK та Scikit-learn, так як вони забезпечують ефективність, простоту використання та широкий набір функцій для роботи з текстами.

На першому етапі розробки ML-моделі необхідно виконати імпорт потрібних бібліотек.

```
In [20]: import nltk
         from nltk.corpus import movie_reviews
         from sklearn.feature_extraction.text import CountVectorizer
         from sklearn.naive_bayes import MultinomialNB
         from sklearn.metrics import accuracy_score
```

NLTK (Natural Language Toolkit) - бібліотека для обробки природної мови (NLP), використовується для роботи з текстовими даними, зокрема для завантаження датасету movie_reviews. Далі завантажуємо корпус (dataset) movie_reviews - набір текстових рецензій на фільми, який містить 1000 позитивних та 1000 негативних відгуків у текстових файлах і який потрібно буде використати його для тренування моделі класифікації тексту.

NLTK допомагає підготувати текстові дані для моделей ML, виконуючи токенізацію, нормалізацію та фільтрацію.

Scikit-learn - це потужний інструмент для класичного машинного навчання, який містить алгоритми для класифікації, регресії, кластеризації та перетворення текстових даних. Scikit-learn дозволяє швидко побудувати, навчити та оцінити NLP-модель за допомогою класичних ML-алгоритмів.

NLTK та Scikit-learn покривають усі етапи роботи з текстом: від попередньої обробки до навчання та оцінки моделі. NLTK допомагає очистити та структурувати текстові дані, а Scikit-learn використовується для їх векторизації, навчання моделей та оцінки якості. Таке поєднання дозволяє швидко та ефективно створювати NLP-рішення.

3.2 Аналіз та обробка даних

Основними етапи аналізу та обробки даних в NLP є:

- збір та підготовка текстових даних;
- попередня обробка тексту;
- перетворення тексту у числовий формат;
- аналіз тексту;
- навчання ML/DL-моделей для NLP;
- оцінка та тестування NLP-моделей.

Дані можуть бути взяті із соціальних мереж, статей, чатів, блогів, книг, а також коментарів. Важливо отримати достатньо велику, якісну та збалансовану вибірку для побудови моделі. У межах бакалаврської кваліфікаційної роботи джерелом даних є датасет "movie_reviews" з бібліотеки NLTK.

Текстові дані незручні для безпосередньої обробки, тому їх потрібно привести до структурованого вигляду, використовуючи такі кроки як токенізація, видалення стоп-слів, приведення слів до базової форми, видалення пунктуації, цифр та спеціальних символів, уніфікація тексту.

Оскільки ML та DL-алгоритми не працюють з текстом напряму, його потрібно представити у вигляді чисел. На етапі аналізу тексту відбувається статистичний та лінгвістичний аналіз текстів. Для розв’язання задач NLP використовуються моделі машинного навчання (ML) та глибокого навчання (DL). Після навчання необхідно оцінити якість моделі, використовуючи метрики якості.

Перетворення тексту у числовий формат використовує наступні методи векторизації:

- Bag of Words (BoW) відображає, які слова входять у текст (без урахування порядку);
- TF-IDF (Term Frequency-Inverse Document Frequency) визначає вагу слів у текстах;
- Word Embeddings (Word2Vec, GloVe, FastText) перетворює слова у багатовимірні вектори, що зберігають семантичний зміст;
- Transformer Embeddings (BERT, GPT) - сучасний метод, який враховує контекст слова в реченні.

Bag of Words - модель, яка широко використовується і дозволяє підраховувати всі слова у фрагменті тексту. Створюється матриця входжень для речення або всього документа, ігноруючи граматику та порядок слів. Ця частота появи або входження слів потім використовуються в якості ознак для навчання класифікатора.

	words	rain	a	paper	they	slip	the	universe	...
Words are flowing out like endless rain into a paper cup,	1	1	1	1	0	0	0	0	...
They slither while they pass, they slip away across the universe	0	0	0	0	3	1	1	1	...

Рис. 3.1 Приклад підрахунку слів в моделі Bag of Words

Метод TF-IDF (Term Frequency-Inverse Document Frequency) - це статистична міра, яка використовується в NLP для оцінки важливості слова у документі відносно всього корпусу документів. Цей метод широко застосовується для інформаційного пошуку, класифікації текстів, тематичного моделювання та інших завдань обробки тексту.

TF (Term Frequency, частота терміну) визначає, як часто слово зустрічається у документі. IDF (Inverse Document Frequency, обернена частота документа) показує, наскільки слово рідкісне у всьому корпусі. TF-IDF = TF × IDF - комбінація обох показників, яка дозволяє визначити найбільш значущі слова у документі, відкидаючи загальні (часто вживані) слова, які зустрічаються всюди.

TF визначає, наскільки часто слово зустрічається у конкретному документі:

$$TF(t,d) = \frac{n_t}{N} \quad , \tag{3.1}$$

де t – слово (термін);

d – визначений документ;

n_t – кількість появ слова t у документі d;

N – загальна кількість слів у документі d.

IDF визначає, наскільки рідкісне слово у всьому корпусі документів. Якщо слово зустрічається у багатьох документах, його важливість зменшується:

$$IDF(t) = \log \frac{D}{D_t} \quad , \tag{3.2}$$

де D - загальна кількість документів у корпусі;

Dt - кількість документів, у яких зустрічається слово t;

логарифм використовується для зменшення масштабів значень.

Обчислення TF-IDF (зважене значення важливості слова):

$$TF - IDF(t,d) = TF(t,d) \times IDF(t) \tag{3.3}$$

TF-IDF допомагає визначати, наскільки документ відповідає пошуковому запиту, використовується для аналізу важливих термінів у документах, дозволяє виявляти ключові слова у відгуках клієнтів.

Word2Vec - метод представлення слів у вигляді векторів у багатовимірному просторі. Його основна ідея полягає в тому, щоб слова, які мають схожий контекст, розташовувалися ближче одне до одного у векторному просторі.

Слова з подібним значенням мають схожі векторні представлення. Модель навчається на великих текстових корпусах та дізнається, як часто слова з'являються поруч. Після навчання кожне слово представлено у вигляді багатовимірного вектора (зазвичай 100–300 вимірів).

Transformer Embeddings - спосіб представлення слів або речень у вигляді багатовимірних векторів за допомогою трансформерних моделей. На відміну від методів Word2Vec або TF-IDF, Transformer Embeddings враховують контекст

всього речення, а не тільки локальні залежності між словами. Кожне слово або речення кодується у векторний простір, де значення векторів залежать від контексту всього тексту. Transformer Embeddings є сучасним підходом до обробки природної мови, який дозволяє отримувати високоточні та контекстно-залежні представлення тексту.

Порівняння TF-IDF з іншими методами представлене у табл. 3.1.

Таблиця 3.1

Порівняння TF-IDF з методами Word2Vec, Transformer Embeddings

№ п/п	Метод	Переваги	Недоліки
1	TF-IDF	Простий, швидкий, добре працює на невеликих текстах	Не враховує порядок слів, не розуміє семантику
2	Word2Vec	Враховує значення слів, працює з контекстом	Вимагає великих даних для навчання
3	BERT, GPT	Використовує контекст слова в реченні	Високі обчислювальні витрати

Аналіз та обробка текстових даних в NLP - це складний процес, який включає збір даних, очищення, векторизацію, аналіз, навчання моделей та оцінку їхньої ефективності. Python-бібліотеки NLTK, Scikit-learn, TensorFlow, Hugging Face Transformers допомагають ефективно вирішувати NLP-завдання, а сучасні моделі GPT, BERT, LSTM дозволяють створювати розумні системи аналізу та генерації тексту.

3.3 Вибір методу обробки текстових даних

На другому етапі потрібно виконати імпортування інструментів машинного навчання. CountVectorizer - перетворює текстові дані у числову матрицю за допомогою методу "Bag of Words" (BoW). Це означає, що кожне слово з тексту стає окремим виміром у векторному представленні. MultinomialNB - це наївний Байєсівський класифікатор, який використовується для класичного NLP. Він добре працює з текстовими даними, де потрібно передбачити категорію

тексту, наприклад, позитивний або негативний відгук. Accuracy_score - метрика оцінки точності моделі. Вона вимірює частку правильних передбачень у тестовій вибірці.

У цілому за допомогою двох вищевказаних етапів потрібно завантажити корпус текстових рецензій з NLTK, перетворити текст у векторну форму, навчити Байєсівський класифікатор на текстах, оцінити точність моделі ML. Цей код є основою для автоматичної класифікації текстів у категорії (позитивний/негативний).

```
In [20]: import nltk
         from nltk.corpus import movie_reviews
         from sklearn.feature_extraction.text import CountVectorizer
         from sklearn.naive_bayes import MultinomialNB
         from sklearn.metrics import accuracy_score

         nltk.download('movie_reviews')
```

Команда *nltk.download('movie_reviews')* завантажує датасет "movie_reviews" з NLTK Corpora - набору текстових корпусів, які використовуються для обробки природної мови. Після завантаження файли зберігаються локально, щоб їх можна було використовувати без підключення до мережі Інтернет. Цю команду потрібно виконувати, якщо ми вперше використовуємо NLTK або не маєте цього корпусу у своїй системі. Після одноразового завантаження корпус зберігається і більше його завантажувати не потрібно.

```
# Завантаження даних
documents = [(list(movie_reviews.words(fileid)), category)
             for category in movie_reviews.categories()
             for fileid in movie_reviews.fileids(category)]
random.shuffle(documents)
```

List comprehension – генератор списків, який створює список кортежів. Кожен кортеж містить: токенізований текст (список слів з відгуку) і мітку категорії ('pos' або 'neg'). *movie_reviews.categories()* повертає список категорій відгуків: позитивний або негативний. Цикл проходить двічі, спочатку для позитивних відгуків, потім для негативних. *Movie_reviews.fileids(category)* повертає список ідентифікаторів файлів у вибраній категорії. Другий цикл проходить по всіх файлах кожної категорії. *Movie_reviews.words(fileid)* повертає

список слів із файлу. *list(movie_reviews.words(fileid)), category* формує кортеж. Після завершення циклів ми отримаємо список всіх рецензій разом із їх мітками.

Команда *random.shuffle(documents)* забезпечує перемішування даних випадковим чином для того, щоб уникнути впливу порядку завантаження даних, а також зробити розподіл тренувальної та тестової вибірки більш випадковим. Виконавши дослідження, чому це є важливим, потрібно зазначити, що у датасеті спочатку йдуть всі негативні відгуки, а потім позитивні і якщо їх не перемішати, модель може навчитися неправильно, оскільки вона буде бачити лише один клас за один раз.

Вищевказаний блок коду завантаження даних створює список усіх рецензій у форматі, який відображений на рис. 3.2, потім перемішує список, щоб навчання моделі було збалансованим.

```
documents = [
        (['слово1', 'слово2', ..., 'словоN'], 'pos'),
        (['слово1', 'слово2', ..., 'словоN'], 'neg'),
        ...
]
```

Рис. 3.2 Блок коду завантаження даних

```
# Підготовка даних
texts = [' '.join(doc) for doc, _ in documents]
labels = [1 if category == 'pos' else 0 for _, category in documents]
```

Етап підготовки даних є важливим у процесі розробки моделі машинного навчання. У задачі на цьому етапі відбувається створення списку *texts*, у якому кожен текстовий відгук представлений як один рядок (об'єднані слова через пробіл). Далі відбувається створення списку *labels,* який містить мітки класів:

- 1 для позитивних рецензій ('pos');
- 0 для негативних рецензій ('neg').

Ці два списки будуть використані для навчання класифікатора текстів. *Texts* - список текстових рецензій у вигляді суцільного рядка. *Labels* - список міток (1 – позитивні, 0 – негативні).

Слід зазначити важливу роль датасетів при розробці моделі ML. Датасети є основою машинного навчання (ML), оскільки без якісних даних модель не зможе навчитися ефективно розв'язувати поставлені завдання. Якість, розмір та структура даних визначають точність, продуктивність і здатність моделі узагальнювати знання.

Відомими датасетами у ML для обробки природної мови (NLP) є:

- IMDB Reviews - відгуки про фільми для аналізу настроїв;
- Common Crawl - корпус текстів з мережі Інтернет.

Для роботи з усіма моделями ML потрібні дані, тобто датасети. Треба розуміти як датасети добувати, де їх шукати, звідки їх брати, як з Інтернету викачувати дані, проводити парсінг (аналіз структури даних з метою вилучення необхідної інформації) сайтів у доступний нам спосіб, як із сайту взяти певну інформацію і яку потім можна буде використовувати в якості датасету. Будь-які бібліотеки для машинного навчання і для глибокого навчання мають у своєму наборі спеціальні пекетжі, де можна підкачати якийсь датасет для експериментів. Бібліотеки для глибокого навчання, для побудови нейронних мереж теж містять в собі початкові дані. Там є спеціальні пекетжі, як правило, які починаються із слова data, у бібліотеках TensorFlow, Pytorch їх можна підтягувати і звідти брати дані для попереднього тренування нейронних мереж. Крім того, для цих бібліотек розроблені ще додаткові бібліотеки сторонніми виробниками, які теж надають певні набори даних для більш широкого навчання, якщо, наприклад, TensorFlow має близько 40 можливих наборів навчальних даних, то TensorData (є такий додаток) пропонує ще в якості доповнення близько 200 різних датасетів для комп'ютерного зору, для побудови класифікаційних, рекомендаційних систем. Тобто все це знаходить у нас в доступі і не вимагає додаткових витрат.

Готові датасети можна знаходити у мережі Інтернет. Все що потрібно нам зробити - знайти у пошукову систему, той самий Google.

Наприклад, ресурс Kaggle є онлайн-платформою для змагань у сфері машинного навчання та аналізу даних. Він надає доступ до датасетів, хмарного

середовища для виконання коду, навчальних матеріалів та ком'юніті фахівців з Data Science та штучного інтелекту (рис. 3.3).

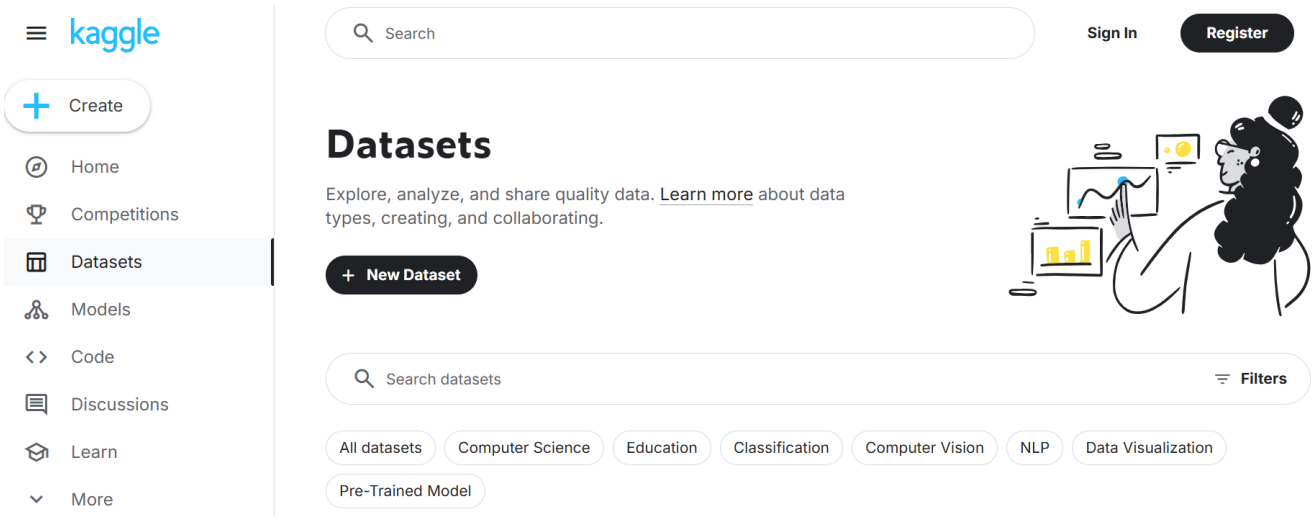

Рис. 3.3 Онлайн-платформа Kaggle

Kaggle містить тисячі відкритих датасетів у різних сферах: фінанси, медицина, NLP, комп’ютерний зір, а дослідники можуть використовувати ці дані для власних проєктів. На Kaggle, крім датасетів, є вже готові моделі, готові ділянки коду, навчальні курси різної тривалості, різної направленості, їх тут дуже багато і курси дуже непогані. Єдина умова – володіння англійською мовою. І вони здебільшого написані не англомовними програмістами, тому вони зрозумілі для нас. Щоб цим всім скористатися, потрібно пройти безкоштовну реєстрацію, що не займає багато часу.

Датасети на Kaggle можна фільтрувати за певними ознаками, вибирати розміри, формат, вибирати ліцензію, по якій вони розповсюджуються, і ще ряд характеристик.

Крім вищевказаного ресурсу, є ще й інші ресурси – менш відомі, але тим не менше можна знайти більш унікальні матеріали. Один із таких ресурсів є *SushTend.com*. Тут ми теж можемо знайти цікаві матеріали для себе. Iris – відомий нам датасет, Titanic – датасет, Bank Marketing. Якщо пройтися по цьому сайту, можна побачити, що більша частина цих датасетів включена у бібліотеку Sciket-learn, з якою ми працюємо.

Третє джерело, де можна шукати інформацію – це та організація, де ми працюємо. Якщо там є відділ статистики, який веде збір інформації, у них ця інформація повинна бути і повинна бути в електронному вигляді. Тому, потрібно звертатися туди.

Якщо ми не знайшли готових датасетів у фреймворках, на існуючих ресурсах і не отримали їх від замовника – нам їх потрібно шукати самостійно. І у нас для цього є мережа Інтернет. Інформацію потрібно шукати на тематичних ресурсах.

Важливо чистити, балансувати та правильно розподіляти дані перед навчанням моделі. Без якісних датасетів навіть найскладніші алгоритми не працюватимуть ефективно, адже гарна модель машинного навчання починається з хороших даних.

3.4 Оцінка ефективності моделі

Наступний етап - це створення ML-моделі та оцінка її ефективності.

```
# Створення BoW моделі
vectorizer = CountVectorizer()
bow = vectorizer.fit_transform(texts)
```

Bag of Words (мішок слів) - метод представлення тексту у вигляді векторів, які містять частоти появи слів у документі. Кожен документ (рецензія) перетворюється на вектор, де: стовпці - це унікальні слова у всіх текстах (словниковий запас), а числа в матриці - це кількість появ кожного слова у документі.

На етапі створення BoW моделі необхідно створити об’єкт CountVectorizer, який конвертуватиме текстові дані у вектори частот слів. CountVectorizer перетворює текст у матрицю частот термінів (TF, Term Frequency), будує словниковий запас - набір унікальних слів у корпусі, ігнорує пунктуацію, великі літери та слова, що зустрічаються дуже рідко або дуже часто, не враховує порядок слів (на відміну від нейронних мереж або word embeddings).

Далі потрібно перетворити список текстів *texts* у числову матрицю BoW. *fit_transform(texts)* виконує дві дії:

- fit() - аналізує всі тексти та створює словниковий запас (унікальні слова);
- transform() - конвертує кожен текст у вектор частот слів на основі словника.

Після дослідження етапу створення BoW моделі, визначено, що: *CountVectorizer()* створює BoW-модель, *fit_transform(texts)* перетворює текст у матрицю частот слів. BoW не враховує порядок слів, лише їх наявність та частоту. Отриманий *bow* - це розріджена матриця, яку можна використовувати для навчання ML-моделей.

На наступному етапі відбувається розділення даних на навчальну та тестову вибірки:

```
# Розділення даних на навчальну та тестову вибірки
X_train, X_test, y_train, y_test = train_test_split(bow, labels, test_size=0.3)
```

Після виконання команди отримуємо значення, які представлені у табл. 3.1.

Таблиця 3.1

Навчальні та тестові дані

№ п/п	Тип набору даних	Відсоток текстів і міток
1	X_train	70% текстів у форматі BoW (навчальні дані)
2	X_test	30% текстів у форматі BoW (тестові дані)
3	y_train	70% міток (відповідні категорії для X_train)
4	y_test	30% міток (відповідні категорії для X_test)

train_test_split() розділяє дані на тренувальну (70%) та тестову (30%) вибірки. *X_train, y_train* – навчальні дані та мітки; *X_test, y_test* - тестові дані та мітки. Використовується для оцінки точності моделі.

```
# Навчання класифікатору
classifier = MultinomialNB()
classifier.fit(X_train, y_train)
```

Вищевказаний код створює і навчає байєсівський класифікатор *MultinomialNB* на тренувальних даних *X_train, y_train.* Він використовується для класифікації тексту, зокрема для аналізу тональності (позитивний або негативний відгук). Саме *MultinomialNB* підходить для тексту, тому що він враховує частоту появи слів (що

важливо у BoW), ефективний навіть при невеликій вибірці, а також простий у реалізації та швидкий.

Класифікатор використовує формулу Байєса:

$$P(\text{клас}|\text{документ}) = \frac{P(\text{документ}|\text{клас})P(\text{клас})}{P(\text{документ})} \quad (3.1)$$

Ймовірність класу (апріорна ймовірність):

$$P(\text{клас}) = \frac{\text{кількість документів цього класу}}{\text{загальна кількість документів}} \quad (3.2)$$

Наприклад, якщо у нас 1000 рецензій: 500 позитивних → P(pos)=0.5P(pos) = 0.5P(pos)=0.5; 500 негативних → P(neg)=0.5P(neg) = 0.5P(neg)=0.5.

Ймовірність слів у класі (правдоподібність):

$$P(\text{слово}|\text{клас}) = \frac{\text{кількість разів, коли слово зустрілося в класі} + \alpha}{\text{загальна кількість слів у класі} + \alpha \cdot V} \quad (3.3)$$

де α - параметр Лапласового згладжування (щоб уникнути нулів);

V - кількість унікальних слів у словнику.

Класифікатор для нового документа рахує добуток ймовірностей слів:

$$P(pos|\text{документ}) \propto P(pos) \cdot P(\text{слово}_1|pos) \cdot P(\text{слово}_2|pos) \cdots \quad (3.4)$$

$$P(neg|\text{документ}) \propto P(neg) \cdot P(\text{слово}_1|neg) \cdot P(\text{слово}_2|neg) \cdots \quad (3.5)$$

Клас з більшою ймовірністю стає передбаченим.

classifier = MultinomialNB() створює наївний байєсівський класифікатор.

classifier.fit(X_train, y_train) навчає модель на текстових даних.

Multinomial Naïve Bayes використовує ймовірності слів у класах для передбачення, добре працює для класичного аналізу тексту, наприклад, аналізу тональності.

```
# Оцінка класифікатору
predictions = classifier.predict(X_test)
print("Accuracy:", accuracy_score(y_test, predictions))
```

```
[nltk_data] Downloading package movie_reviews to
[nltk_data]     C:\Users\ASUS\AppData\Roaming\nltk_data...
[nltk_data]   Package movie_reviews is already up-to-date!
```

```
Accuracy: 0.8183333333333334
```

Цей код використовує навчений класифікатор для передбачення класів на тестових даних, а потім оцінює точність моделі за допомогою метрики Accuracy.

Класифікатор використовує навчені параметри (ймовірності, які він обчислив на етапі навчання) для того, щоб передбачити клас для кожного документа в тестовому наборі. Виходить масив predictions, де кожен елемент - це передбачений клас для кожного документа:

- 1 - позитивний клас (наприклад, позитивний відгук);
- 0 - негативний клас (наприклад, негативний відгук).

classifier.predict(X_test) бере кожен текст із тестової вибірки *X_test* і передбачає чи є він позитивним (1) або негативним (0). Рішення приймається на основі ймовірностей для кожного класу, які були обчислені при навчанні.

Accuracy - метрика, яка вимірює, скільки передбачених класів збіглися з реальними і обчислюється за формулою:

$$\text{Accuracy} = \frac{\text{Кількість правильних передбачень}}{\text{Загальна кількість передбачень}} \tag{3.6}$$

y_test - це реальні мітки класів для тестових даних.

predictions – це класи, які передбачила модель.

Функція *accuracy_score()* порівнює ці два масиви:

- якщо елемент з *y_test* дорівнює елементу з *predictions*, то це правильне передбачення;
- якщо вони не співпадають, то це неправильне передбачення.

Функція повертає відсоток правильних передбачень.

Оцінка ефективності моделі машинного навчання є критично важливим етапом, який дозволяє зрозуміти, наскільки добре модель виконує своє завдання та чи може вона бути використана на реальних даних. Без оцінки неможливо зрозуміти, чи модель дає правильні результати. Оцінки ефективності моделі

допомагає визначити, наскільки добре модель узагальнює знання і застосовує їх до нових даних. Також вона дозволяє вибрати найкращу модель серед кількох варіантів, наприклад, можна порівняти логістичну регресію, дерево рішень та нейромережу за точністю, швидкістю роботи та іншими показниками.

У задачі, яка була поставлена в рамках бакалаврської кваліфікаційної роботи, значення метрики Accuracy становить 0.82, що є непоганим результатом роботи моделі машинного навчання.

Метрика Accuracy має простоту розрахунку та інтерпретації. Якщо у датасеті кількість позитивних і негативних прикладів приблизно рівна, Accuracy надійно відображає якість моделі. Використовується у комп'ютерному зорі для розпізнавання об'єктів, обробці природної мови для класифікації текстів, а також у медицині (діагностика хвороб).

Дана метрика дозволяє швидко оцінити, яка з моделей працює краще, при схожих умовах і збалансованих класах можна вибрати модель із вищою Accuracy.

ВИСНОВКИ

У бакалаврській роботі було розроблено модель машинного навчання для автоматичної класифікації відгуків про фільми на позитивні та негативні, використовуючи методи обробки природної мови (NLP) та машинного навчання. У сучасну епоху цифрової трансформації обсяги текстової інформації зростають експоненційно. Технології Natural Language Processing (NLP) та машинного навчання (ML) відіграють ключову роль у автоматизації аналізу, обробки та генерації текстів, що значно покращує та покращуватиме у майбутньому ефективність комунікації між людьми та машинами.

У сучасному інформаційному середовищі створюється величезна кількість текстових матеріалів – соціальні мережі, новини, наукові статті, електронні листи тощо. NLP-алгоритми дозволяють швидко аналізувати, структурувати та витягувати корисну інформацію. У наукових дослідженнях технологію доцільно використовувати для автоматизованого аналізу наукових статей, резюмування текстів та виявлення нових дослідницьких напрямів.

У дипломній роботі вхідними даними був набір текстових відгуків із корпусу NLTK movie_reviews. Для розробки моделі машинного навчання було вибране середовище Jupyter Notebook. Jupyter Notebook є одним із найпопулярніших середовищ для розробки моделей машинного навчання (ML) з використанням технологій NLP. Його популярність обумовлена зручністю, інтерактивністю та широкими можливостями для експериментів із даними. Під час виконання поставлених завдань Jupyter Notebook дозволяв виконувати код блоками, що забезпечило швидку розробку NLP-моделі. Також дане середовище забезпечувало гнучкість у роботі з NLP-інструментами.

Крім бібліотеки NLTK, під час написання ML-моделі була використана бібліотека Scikit-learn. NLTK та Scikit-learn покривали всі етапи роботи з текстом: від попередньої обробки до навчання та оцінки моделі. NLTK забезпечила структурування текстових даних, а Scikit-learn була використана для їх векторизації, навчання моделі та оцінки якості. Таке поєднання дозволило швидко

та ефективно створити NLP-рішення. Також, для розробки моделі доцільним було використання бібліотек NLTK та Scikit-learn, так як вони забезпечують ефективність, простоту використання та широкий набір функцій для роботи з текстами.

Основними етапи аналізу та обробки даних в NLP є: збір та підготовка текстових даних; попередня обробка тексту; перетворення тексту у числовий формат; аналіз тексту; навчання ML/DL-моделей для NLP; оцінка NLP-моделі.

Основні виклики, з якими приходиться стикатися в NLP сьогодні, пов’язані з тим, що мова NLP дуже складна. Процес розуміння та обробки мови надзвичайно складний, у зв’язку з чим для вирішення різних задач потрібно використовувати різні методи, до того, як все буде пов’язано воєдино. Для реалізації цих технік широко використовуються такі мови програмування, як Python та R, концепції в їх основі та NLP-алгоритми, які найчастіше використовуються. У дипломній роботі була використана мова програмування Python.

У роботі для задачі обробки природної мови бу використаний метод Bag of Words (BoW), який є корисним для швидкого прототипування та розуміння основних концепцій обробки тексту. У задачі, яка була поставлена в рамках бакалаврської кваліфікаційної роботи, значення метрики Accuracy становить 0.82, що є непоганим результатом роботи моделі машинного навчання.

NLP та машинне навчання є фундаментальними технологіями, що дозволяють обробляти, аналізувати та взаємодіяти з текстовими даними на абсолютно новому рівні. Їхнє впровадження у бізнес, науку, медицину та повсякденне життя дозволятиме автоматизувати процеси, підвищувати продуктивність та покращувати взаємодію між людьми та технологіями. У майбутньому розвиток NLP та ML відкриватиме ще більше можливостей для покращення та вдосконалення аналізу інформації та комунікацій.

ПЕРЕЛІК ПОСИЛАНЬ

1. https://uk.shaip.com/blog/15-nlp-dataset-for-nlp/
2. https://dou.ua/forums/topic/44792/
3. https://www.python.org/
4. https://www.nltk.org/
5. https://scikit-learn.org/stable/

6. Wang, Y., Ru, Z.-Y., Wang, K., Huang, P.-Q. Joint deployment and task scheduling optimization for large-scale mobile users in multi-UAV-enabled mobile edge computing. IEEE Trans. Cybern., 50(9), 2020. — 3984-3997 с.

7. Nguyen, V., Khanh, T. T., Nam, P. V., Thu, N. T., Hong, C. S., Huh, E.-N. Towards flying mobile edge computing. Proc. Int. Conf. Inf. Netw. (ICOIN), Jan. 2020. — 723-725 с.

8. Zhou, F., Hu, R. Q., Li, Z., Wang, Y. Mobile edge computing in unmanned aerial vehicle networks. IEEE Wireless Commun., 27(1), 2020. — 140-146 с.

9. Mishra, D., Natalizio, E. A survey on cellular-connected UAVs: Design challenges enabling 5G/B5G innovations and experimental advancements. Comput. Netw., 182, Dec. 2020.

10. W. Yu, W. Cheng, C. Aggarwal, K. Zhang, H. Chen, and Wei Wang. NetWalk: A flexible deep embedding approach for anomaly Detection in dynamic networks, ACM KDD Conference, 2018.

11. Jake VanderPlas Python Data Science Handbook: Essential Tools for Working with Data, 2022, 551p.

12. Heller, Nicholas et al. (2020). The KiTS19 Challenge Data: 300 Kidney Tumor Cases with Clinical Context, CT Semantic Segmentations, and Surgical Outcomes. arXiv: 1904. 00445 [q-bio.QM].

13. Hollo, Kaspar (2019). Exploring the Value of Weakly-Supervised Deep Learning Approaches for Artefact Segmentation in Brightfield Microscopy Images. URL: https://comserv. cs . ut . ee / home / files / hollo _ softwareengineering _ 2021.

14. Wang, Haofan et al. (2020). Score-CAM: Score-Weighted Visual Explanations for Convo-lutional Neural Networks. arXiv: 1910.01279 [cs.CV].

15. Yang, Guanyu et al. (2020). “Weakly-supervised convolutional neural networks ofrenal tumor segmentation in abdominal CTA images”. In: BMC Medical Imaging 20.1, p. 37. ISSN: 1471-2342. DOI: 10.1186/s12880- 020- 00435- w. URL: https://doi.org/10.1186/s12880-020-00435-w.

16. Selvaraju, Ramprasaath R. et al. (2019). “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization”. In: International Journal of ComputerVision 128.2, pp. 336–359. DOI: 10.1007 / s11263 - 019 - 01228 - 7. URL: https://doi.org/10.1007\%2Fs11263-019-01228-7.

17. Лосєв М.Ю. Програмування мовою Python: навчальний посібник / М. Ю. Лосєв, В. М. Федорченко. – Харків, - Львів: Видавництво ПП «Новий Світ - 2000», 2023. – 178 с.

18. Золотухіна О.А., Негоденко О.В., Резник С.Ю., Разіна С.Я. Якість та тестування інформаційних систем. Навчальний посібник підготовлено до друку для самостійної роботи студентів вищих навчальних закладів. Київ: ННІІТ ДУТ, 2020. – 128 с.

19. Зінченко О.В., Фесенко М.А., Березівський М.Ю., Кисіль Т.М. Технології смарт-систем. – Навчальний посібник. – К.: ДУІКТ, 2023. – 162 с.

20. Нікольський Ю. В., Пасічник В. В., Щербина Ю. М. Системи штучного інтелекту: навчальний посібник. – Львів: «Магнолія-2006», 2024. – 279 с.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)