Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп'ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеню вищої освіти – магістр
за освітньо-професійною програмою
«Комп'ютерні науки»

зі спеціальності
122 – Комп'ютерні науки

тема роботи:

«Інтелектуальний аналіз текстових даних з соціальних мереж для виявлення та прогнозування трендів у громадській думці на основі машинного навчання»

Виконав студент гр. КН-23м. _____ Ковальов М.В.

Керівник _____ Тиханський М.П.

Нормоконтроль _____ Маринич І. А.

Завідувач кафедри _____ Рубан С. А.

Кривий Ріг – 2024

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: *інформаційних технологій*
Кафедра: *автоматизації, комп'ютерних наук і технологій*
Ступінь вищої освіти: *Магістр*
Спеціальність: *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Зав. кафедри: *к.т.н. Рубан С.А.*
_____
« *5* » *липня* 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу магістра

студентові групи КН-23м *Ковальову Миколі Володимировичу*

1. Тема кваліфікаційної роботи: *«Інтелектуальний аналіз текстових даних з соціальних мереж для виявлення та прогнозування трендів у громадській думці на основі машинного навчання»*

затверджено наказом по університету № *594с* від *04.07.2023 р.*

2. Термін здачі кваліфікаційної роботи: *01.12.2024 р.*

3. Склад кваліфікаційної роботи: *Пояснювальна записка обсягом 91с., додатки, презентація у Microsoft PowerPoint (18 слайдів) в електронному та друкованому вигляді*

4. Консультанти кваліфікаційної роботи:

Розділ 1-3 к.т.н., доц. Тиханський М.П.

Нормоконтроль доц. Маринич І. А.

5. Календарний план:

№	Етапи роботи	Термін виконання
1	*Вступ*	*10.07.24*
2	*РОЗДІЛ 1*	*15.07.24*
3	*РОЗДІЛ 2*	*18.08.24*
4	*РОЗДІЛ 3*	*19.09.24*
5	*РОЗДІЛ 4*	*20.09.24*
6	*Висновки*	*15.10.24*
7	*Оформлення кваліфікаційної роботи*	*20.11.24*
8	*Підготовка презентації та графічного матеріалу*	*28.11.24*
9	*Підготовка доповіді до захисту*	*01.12.24*

6. Дата видачі завдання: *28.06.2024р.*

Керівник _____ /Тиханський М.П./

7. Запевнення: *Я, Ковальов Микола Володимирович, запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.*

Із чинним Положенням про академічну доброчесність Криворізького національного університету ознайомлений.

Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Здобувач _____ / Ковальов М.В./

АНОТАЦІЯ

Ковальов М.В. «Інтелектуальний аналіз текстових даних з соціальних мереж для виявлення та прогнозування трендів у громадській думці на основі машинного навчання».

Кваліфікаційна робота на здобуття ступеню вищої освіти магістр за освітньо-професійною програмою «Комп'ютерні науки» зі спеціальності 122 – Комп'ютерні науки. – Криворізький національний університет, Кривий Ріг, 2024.

Робота складається з вступу, чотирьох розділів, висновків, 19 рисунків, списку із 56 літературних джерел та 14 додатків.

Актуальність теми роботи зумовлена стрімким зростанням обсягів інформації у соціальних мережах, які стають важливим джерелом вивчення суспільних настроїв.

Об'єктом дослідження є текстові дані, опубліковані у соціальних мережах.

Предметом дослідження є процеси інтелектуального аналізу текстових даних, спрямовані на виявлення та прогнозування трендів громадської думки.

Мета роботи полягає у розробці підходів і моделей машинного навчання для аналізу текстових даних соціальних мереж та виявлення трендів громадської думки.

Основні ***результати*** роботи включають в себе розробку методології попередньої обробки текстових даних (токенізація, очищення, стемінг), аналіз настроїв текстів із використанням інструментів бібліотеки NLTK, порівняння моделей класифікації текстів за точністю, створення візуалізацій для виявлення залежностей у даних і можуть бути використані для моніторингу громадської думки, прогнозування соціальних трендів та розробки рекомендаційних систем.

Ключові слова: АНАЛІЗ ТЕКСТОВИХ ДАНИХ, СОЦІАЛЬНІ МЕРЕЖІ, МАШИННЕ НАВЧАННЯ, ГРОМАДСЬКА ДУМКА, НАСТРОЇ, ТРЕНДИ.

ANNOTATION

Kovalyov M.V. "Intellectual Analysis of Text Data from Social Networks for Detecting and Predicting Trends in Public Opinion Based on Machine Learning."
Qualification thesis for obtaining the master's degree in the educational and professional program "Computer Science" under specialty 122 – Computer Science. – Kryvyi Rih National University, Kryvyi Rih, 2024.

The thesis consists of an introduction, four chapters, conclusions, 19 figures, a list of 56 references, and 14 appendices.

The relevance of the topic is determined by the rapid growth in the volume of information on social networks, which are becoming an important source for studying public sentiment.

The object of the research is text data published on social networks.

The subject of the research is the processes of intellectual analysis of text data aimed at detecting and predicting trends in public opinion.

The aim of the thesis is to develop approaches and machine learning models for analyzing text data from social networks and identifying trends in public opinion.

The main results of the work include the development of a methodology for preprocessing text data (tokenization, cleaning, stemming), sentiment analysis of texts using NLTK library tools, comparison of text classification models based on accuracy, creation of visualizations to detect data dependencies, and they can be used for public opinion monitoring, social trend prediction, and recommendation system development.

Keywords: TEXT DATA ANALYSIS, SOCIAL NETWORKS, MACHINE LEARNING, PUBLIC OPINION, SENTIMENTS, TRENDS.

ЗМІСТ

ВСТУП

В сучасному світі соціальні мережі стали важливою платформою для обміну інформацією, висловлювання думок і обговорення актуальних подій. Величезна кількість текстових даних, які щоденно створюють користувачі, містять унікальні інсайти про настрої, уподобання та соціальні тренди. Ця інформація є цінною для бізнесу, політики, маркетингу та інших сфер. У галузі комп'ютерних наук завдання аналізу текстових даних із соціальних мереж набуло особливої актуальності через зростання обсягу даних і потребу в їх швидкій обробці для прийняття рішень.

Основна проблема, на вирішення якої спрямована ця робота, полягає у складності виявлення та прогнозування трендів у громадській думці через неоднорідність текстових даних, їх великий обсяг і багатокомпонентну структуру. Без ефективного аналізу цієї інформації ускладнюється розуміння настроїв суспільства, а також прогнозування соціальних трендів, що може призвести до втрати конкурентних переваг для компаній, недостовірності соціологічних досліджень та запізнення у вжитті необхідних заходів на суспільному рівні[1].

Метою роботи є розробка системи для інтелектуального аналізу текстових даних із соціальних мереж, яка дозволить виявляти настрої користувачів, аналізувати їх динаміку та прогнозувати можливі тренди в громадській думці.

Для досягнення цієї мети були визначені наступні завдання:

1) огляд сучасних підходів до аналізу текстових даних та прогнозування трендів;
2) опис методів машинного навчання, що використовуються для обробки текстової інформації;
3) збір і підготовка текстових даних із соціальних мереж для аналізу;
4) використання методів попередньої обробки даних, таких як очищення тексту, токенізація, лемматизація та стемінг;
5) застосування методів машинного навчання для класифікації настроїв текстів;

6) побудова візуалізацій для аналізу отриманих результатів і виявлення трендів у громадській думці;
7) оцінка ефективності різних моделей машинного навчання.

Об'єктом роботи є текстові публікації з різних соціальних мереж, які відображають громадську думку. Предметом роботи виступає процес аналізу текстових даних для виявлення настроїв і прогнозування трендів.

Для вирішення поставленої проблеми планується використання сучасних методів аналізу тексту, таких як TF-IDF для векторизації текстових даних, а також алгоритмів машинного навчання (Passive Aggressive Classifier, Logistic Regression, Random Forest, SVM, Multinomial Naive Bayes). Використання бібліотек Python, таких як NLTK, scikit-learn, та методів візуалізації дозволить здійснити якісний аналіз настроїв і розробити систему для моніторингу громадської думки.

Результати цієї роботи сприятимуть поглибленню розуміння та удосконаленню процесу аналізу великих текстових даних і можуть бути застосовані для побудови систем соціального моніторингу, аналітики трендів та ухвалення рішень у різних галузях.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНИХ ДОСЛІДЖЕНЬ З АНАЛІЗУ ТЕКСТОВИХ ДАНИХ

1.1 Поняття та важливість аналізу текстових даних

Аналіз текстових даних є однією з ключових галузей в обробці інформації та машинному навчанні. Він передбачає перетворення необроблених текстових даних на структуровану інформацію, що дозволяє проводити глибокі дослідження і приймати обґрунтовані рішення в різних сферах діяльності. Текстові дані можуть містити корисні відомості про думки, настрої, та інші аспекти поведінки користувачів, що є важливими для аналізу соціальних процесів.

Зокрема, в контексті соціальних мереж, аналіз текстових даних є потужним інструментом для вивчення громадської думки, виявлення нових трендів, а також прогнозування потенційних змін у суспільстві. На сьогоднішній день соціальні мережі генерують величезний обсяг текстової інформації, що охоплює різні аспекти людського життя — від політичних подій до поп-культури. З кожним роком кількість активних користувачів в таких мережах, як Twitter, Facebook, Instagram, та інших, зростає, що призводить до збільшення обсягу доступних для аналізу даних.

Значення аналізу текстових даних з соціальних мереж полягає у тому, що ці дані можуть надавати реальну картину настроїв та тенденцій, що активно формуються в суспільстві. Наприклад, вивчення емоцій та думок користувачів щодо певних політичних подій дозволяє прогнозувати результати виборів, а також визначати реакцію громадськості на різні ініціативи та закони. Окрім того, аналіз тексту є важливим для дослідження соціальних явищ, таких як поширення фейкових новин, соціальна мобілізація чи взаємодія між різними групами людей. На *рис. 1.1* наведено зростання обсягу текстових даних в соціальних мережах впродовж останніх років.

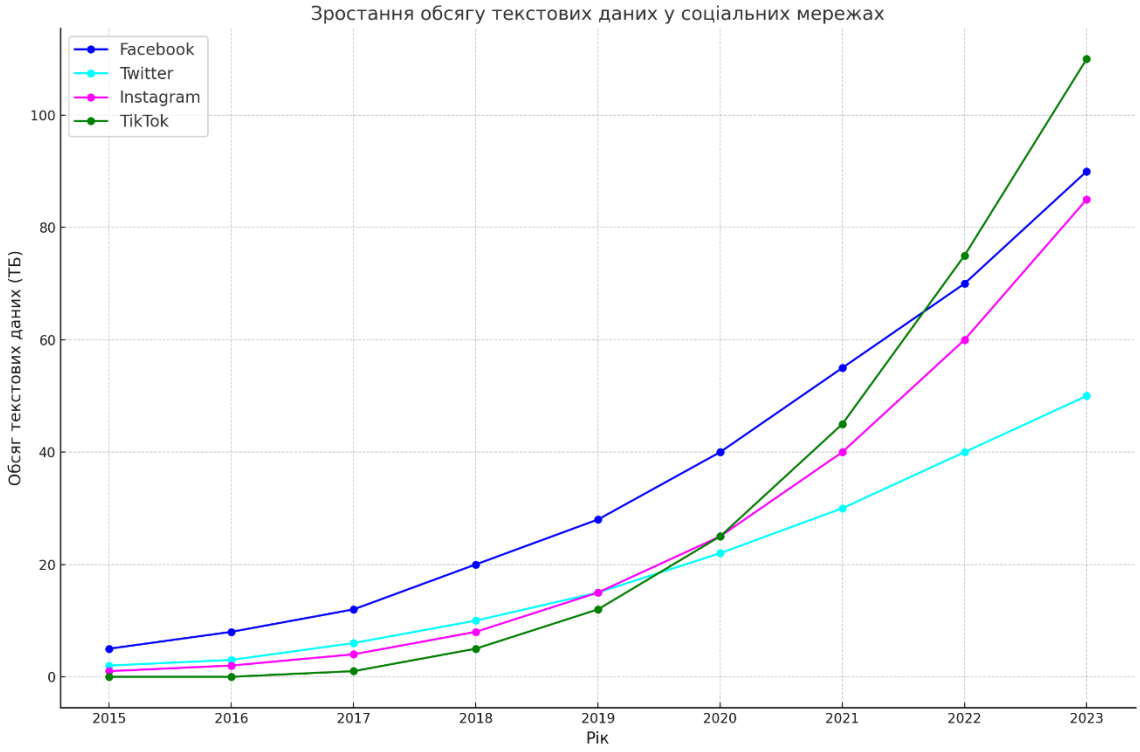

Рисунок 1.1 – Графік зростання обсягу текстових даних у соціальних мережах (Додаток Б)

Для ефективного аналізу текстових даних використовуються різноманітні методи обробки природної мови (NLP) та машинного навчання. Основними етапами в цьому процесі є:

- попередня обробка даних: включає в себе видалення шуму, токенізацію, лематизацію та інші етапи очищення тексту;
- аналіз емоцій і настроїв: з допомогою алгоритмів машинного навчання та NLP можна визначити, чи є в тексті позитивні, негативні або нейтральні емоції;
- виявлення трендів і тем: на основі аналізу великої кількості текстових повідомлень можна виявити основні теми обговорень або передбачити тренди в суспільній думці.

Аналіз текстових даних з соціальних мереж має велике практичне значення для різних секторів, включаючи маркетинг, політику, соціологію та навіть медіа. Зокрема,

він дозволяє компаніям проводити дослідження ринку, оцінювати ефективність рекламних кампаній, а також виявляти потенційні ризики або проблеми до того, як вони набудуть широкого розголосу в суспільстві[1].

Загалом, аналіз текстових даних є важливою складовою сучасних методів дослідження і наукових розробок, що дозволяє здійснювати точні прогнози та краще розуміти тенденції розвитку суспільства.

1.2 Соціальні мережі як джерело текстових даних

Соціальні мережі стали одним з найбільших джерел текстових даних, і їхнє значення для аналізу громадської думки не можна недооцінювати. Мільйони користувачів щодня публікують постійно оновлюваний контент, що містить відгуки, коментарі, репости, твітти, фотографії з підписами, а також різні інші форми текстової інформації. Графік обсягів користуввачів найвідоміших онлайн-спільнот наведений на *рис. 1.2,* що нижче. Платформи такі як Facebook, Twitter, Instagram, Reddit, TikTok та інші стали основними каналами для висловлення громадянської думки, і вони надають багатий матеріал для подальших досліджень.

Однією з головних переваг соціальних мереж є доступність величезної кількості даних, що надходять у реальному часі. Вони забезпечують потік публікацій, який не припиняється — кожна нова хвиля повідомлень дозволяє дослідникам виявляти актуальні настрої, нові тренди, зміну громадської думки та соціальних настроїв. Наприклад, в 2023 році на платформі Twitter щодня публікується понад 500 мільярдів постів, що включають не лише текстові повідомлення, але й хештеги, посилання, медіафайли[3]. Всі ці дані мають потенціал для аналізу в рамках прогнозування політичних подій, маркетингових кампаній або вивчення поведінки соціальних груп.

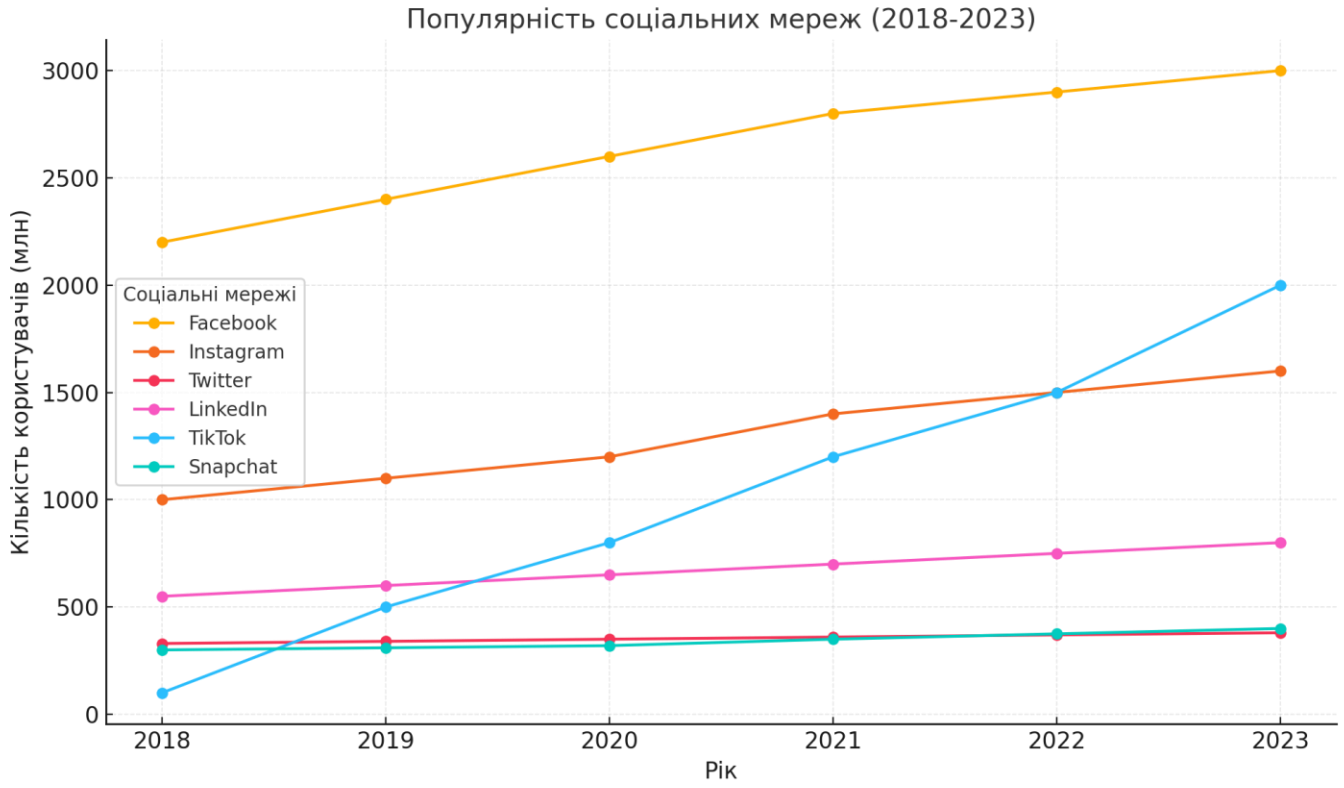

Рисунок 1.2 – Графік популярності соціальних мереж (Додаток В)

Ще однією особливістю соціальних мереж є велика різноманітність тем, які обговорюються користувачами. Це можуть бути питання, пов'язані з політикою, економікою, культурними подіями, особистими переживаннями, реакціями на глобальні катастрофи тощо. Через це соціальні мережі є дуже цінними для досліджень у різних сферах, від соціальних наук до маркетингу. Наприклад, дослідники можуть аналізувати пости, що стосуються важливих політичних подій, виборів або соціальних рухів, для того, щоб зрозуміти, як суспільство реагує на зміни в навколишньому середовищі. Соціальні мережі дозволяють аналізувати настрої користувачів на глобальному рівні, завдяки чому можна виявляти не лише локальні, але й глобальні тренди в реальному часі[2].

Динамічність публікацій у соціальних мережах є ще однією важливою рисою цього джерела даних. Оскільки інформація розповсюджується миттєво, дослідники можуть відслідковувати зміни в громадській думці, що відбуваються через короткий

проміжок часу. Це дозволяє не тільки виявляти тренди, але й швидко реагувати на них, що особливо важливо для прогнозування політичних подій, рекламних кампаній, соціальних протестів та інших важливих явищ у суспільстві.

Водночас, соціальні мережі створюють певні труднощі при зборі та аналізі даних. Однією з головних проблем є величезний обсяг так званого "шуму" в даних — непотрібної або неінформативної інформації. Багато публікацій є спамом, рекламою чи нерелевантними коментарями, що потребує значного очищення та фільтрації для отримання коректних результатів. Для успішного аналізу текстових даних важливо застосовувати методи попередньої обробки, такі як токенізація, видалення стоп-слів та інші техніки, які допомагають зробити дані більш структурованими і корисними для машинного навчання.

Крім того, важливим є питання етики та конфіденційності при зборі даних із соціальних мереж. Хоча більшість контенту на платформі є публічним, необхідно враховувати юридичні та етичні норми, зокрема ті, що стосуються персональних даних і приватності користувачів. В країнах Європейського Союзу, зокрема, існують строгі закони, як-от Загальний регламент захисту даних (GDPR), який регулює, як мають збиратися і використовуватися дані, що належать до особистості[2].

Ще однією проблемою, з якою стикаються дослідники при аналізі текстових даних з соціальних мереж, є мовні бар'єри. Соціальні мережі є глобальними, і хоча основна частина контенту може бути на англійській мові, велика кількість постів і коментарів може бути написана різними мовами. Це може ускладнити аналіз, адже для коректного вивчення даних потрібні інструменти для автоматичного перекладу та підтримки багатомовності. Такі технології, як машинний переклад, допомагають знижувати цей бар'єр, але все одно можуть виникати труднощі у розумінні деяких контекстів або специфічних виразів.

Таким чином, соціальні мережі є потужним інструментом для збору текстових даних, які можуть бути використані для аналізу громадської думки, виявлення трендів і прогнозування важливих соціальних подій. Однак, зібрані дані потребують ретельної

обробки та аналізу з урахуванням можливих перешкод, таких як шум у даних, етичні питання та мовні бар'єри.

1.3 Методи збору текстових даних з соціальних мереж

Збір текстових даних з соціальних мереж є важливою складовою аналізу та прогнозування трендів у громадській думці. Враховуючи величезний обсяг інформації, що постійно генерується користувачами, існує кілька ключових методів, які дозволяють ефективно отримувати, обробляти та аналізувати ці дані. У цьому розділі розглянемо основні підходи до збору текстових даних з соціальних мереж, їх можливості та обмеження.

Одним із основних способів збору текстових даних є використання API (Application Programming Interfaces), які надаються різними соціальними платформами. API дозволяють здійснювати автоматизований доступ до публічних постів, коментарів та іншого контенту, що публікується користувачами на платформах, таких як Twitter, Facebook, Instagram, Reddit тощо. З допомогою API можна отримати великий обсяг інформації за допомогою програмних запитів. Наприклад, Twitter API надає можливість отримувати твіти з певними хештегами, згадками, або навіть за заданими географічними координатами, що є особливо корисним для аналізу актуальних подій[4]. Однак доступ до деяких API може бути обмежений певними політиками конфіденційності або лімітами на кількість запитів. Наприклад, зміни в політиці доступу до даних на платформі Facebook обмежили деякі можливості для дослідників, зокрема для аналізу коментарів користувачів[5].

Ще одним популярним методом є стрімінговий збір даних (streaming), який передбачає постійний моніторинг публікацій і отримання нових даних у реальному часі. Це дає можливість аналізувати тренди та настрої на основі найактуальнішої інформації, що надходить від користувачів. Такі підходи використовуються для вивчення змін у громадській думці, особливо в умовах кризових ситуацій, коли

важливо відстежувати швидкі зміни в настроях і думках користувачів[36]. Для збору даних в реальному часі можуть використовуватися спеціалізовані бібліотеки, такі як Tweepy для Twitter, які дозволяють безпосередньо підключатися до API та отримувати потоки даних безпосередньо з мережі.

Іншою важливою технологією є веб-скрейпінг (web scraping), який передбачає автоматичне зчитування веб-сторінок і збору з них текстових даних. Цей метод є корисним, коли API соціальних мереж недоступні або обмежені. Наприклад, скрейпінг дозволяє отримувати інформацію з платформ, які не надають публічного API, або з приватних груп і сторінок, де дані не доступні для загального доступу. Однак веб-скрейпінг має свої обмеження і юридичні аспекти, оскільки багато платформ забороняють таке використання через умови користування, що можуть порушувати авторські права чи політику конфіденційності[37].

Для обробки великих обсягів даних, які можуть бути зібрані через API чи веб-скрейпінг, використовуються інструменти та платформи для роботи з великими даними (Big Data). Такі технології, як Hadoop, Apache Spark або Apache Kafka, дозволяють зберігати та обробляти величезні обсяги інформації в режимі реального часу, що є важливим для аналізу текстових даних з соціальних мереж. Вони здатні підтримувати масштабованість, дозволяючи обробляти дані з декількох джерел одночасно, що особливо важливо при аналізі тенденцій на глобальному рівні[38].

Одним з основних аспектів збору даних є необхідність у очищенні та фільтрації інформації, оскільки соціальні мережі містять величезну кількість непотрібної чи нерелевантної інформації. Процеси токенізації, лематизації, видалення стоп-слів та відсіювання спаму допомагають очистити дані, зберігаючи лише суттєву інформацію, що потрібна для аналізу. Це дозволяє покращити якість даних і підвищити точність подальших етапів обробки та аналізу.

Нарешті, важливим аспектом є етика збору даних з соціальних мереж. Хоча більшість даних в соціальних мережах є публічними, потрібно враховувати конфіденційність і права користувачів. Уряди різних країн вводять різноманітні

регуляції, як GDPR в Європейському Союзі, що обмежують використання особистих даних без згоди користувачів. Це створює певні виклики для дослідників, оскільки необхідно ретельно дотримуватися етичних принципів при зборі та використанні даних, особливо щодо обробки персональної інформації[39].

Отже, методи збору текстових даних з соціальних мереж включають використання API, стрімінгових технологій, веб-скрейпінгу та інструментів для обробки великих даних. Вибір конкретного методу залежить від специфіки дослідження, доступних ресурсів та етичних норм. На рисунку 1.3, що нижче, я спробував відобразити популярність методів збору данних, котрі використовують користувачі в залежності від соціальної мережі. Виходячі з отриманих даних, можна побачити, що метод API, використовується частіше для отримання даних з Facebook та Twitter. На мою думку, це пов’язано с тим, що ці мережі більш зосереджені на текстовому контенті(постах).

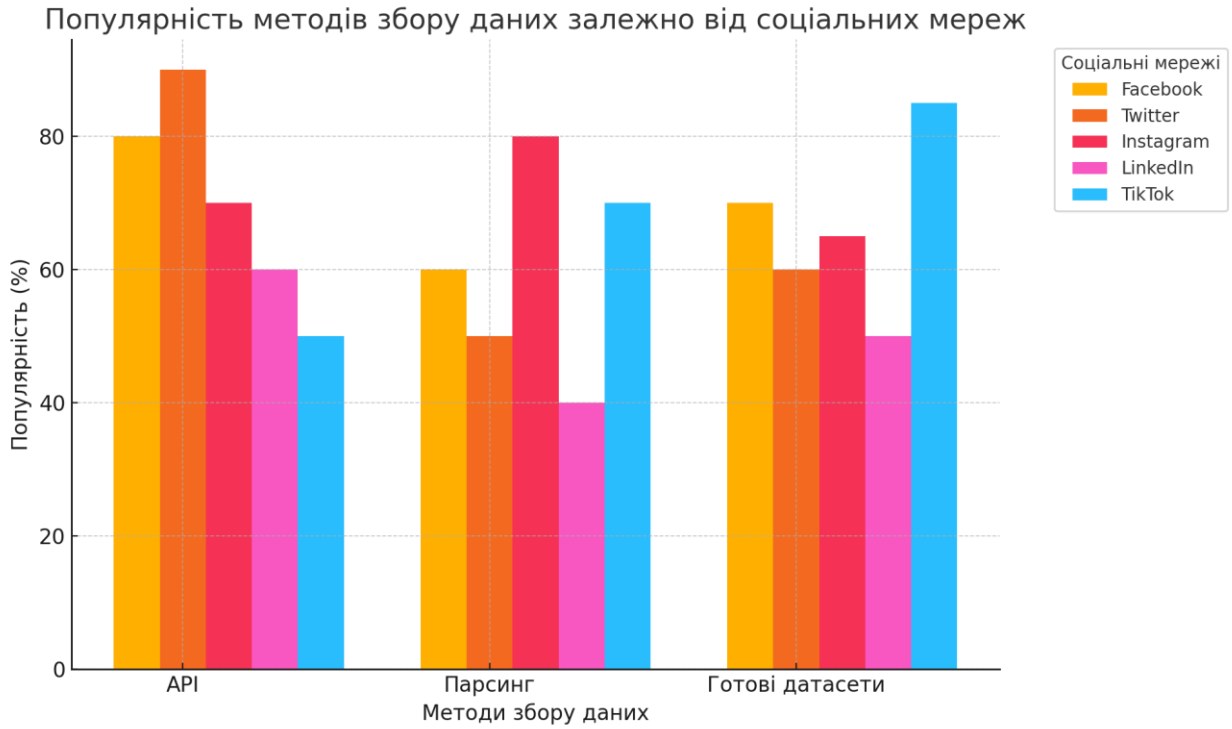

Рисунок 1.3 – Графік популярності методів збору даних (Додаток Г)

1.4 Використання машинного навчання для аналізу настроїв і трендів

Машинне навчання стало важливим інструментом для аналізу текстових даних, зокрема для визначення настроїв та виявлення трендів у громадській думці. Текстові дані, що генеруються користувачами в соціальних мережах, містять величезний потенціал для дослідження емоційних реакцій, відгуків і спостереження за змінами в поведінці великої аудиторії. Для цього використовуються різноманітні методи машинного навчання, які дозволяють не лише класифікувати тексти за емоційним забарвленням, а й виявляти закономірності, що відображають популярність певних тем, подій або обговорень.

Аналіз настроїв (sentiment analysis), наприклад, полягає в автоматичному визначенні емоційної тональності тексту. Цей метод дозволяє класифікувати відгуки користувачів на соціальних платформах як позитивні, негативні або нейтральні. Визначення настрою є важливим для розуміння того, як сприймаються певні події, бренди чи політичні явища. Для реалізації цієї задачі застосовуються різні алгоритми машинного навчання, серед яких найпопулярнішими є методи класифікації, зокрема наївний баєсівський класифікатор, логістична регресія та методи опорних векторів (SVM). Логістична регресія, наприклад, є одним з найбільш широко використовуваних методів для задач аналізу настроїв завдяки своїй здатності працювати з великими обсягами даних і забезпечувати високий рівень точності в багатьох практичних застосуваннях [24].

Окрім традиційних методів машинного навчання, для аналізу настроїв активно використовуються моделі глибокого навчання (deep learning models), які дозволяють краще враховувати контекст і складнішу структуру тексту. Наприклад, рекурентні нейронні мережі (RNN) і LSTM (Long Short-Term Memory) показують високі результати в задачах, де необхідно враховувати залежність слів у тексті і його загальний контекст[40]. Така здатність до обробки довгих залежностей робить ці моделі особливо ефективними для аналізу настроїв у великих масивах тексту, де

виявлення нюансів у емоціях користувачів є важливим аспектом. На рисунку 1.4 відображена приблизна точність деяких моделей для аналізу настроїв із застосуванням типового дата-сету.

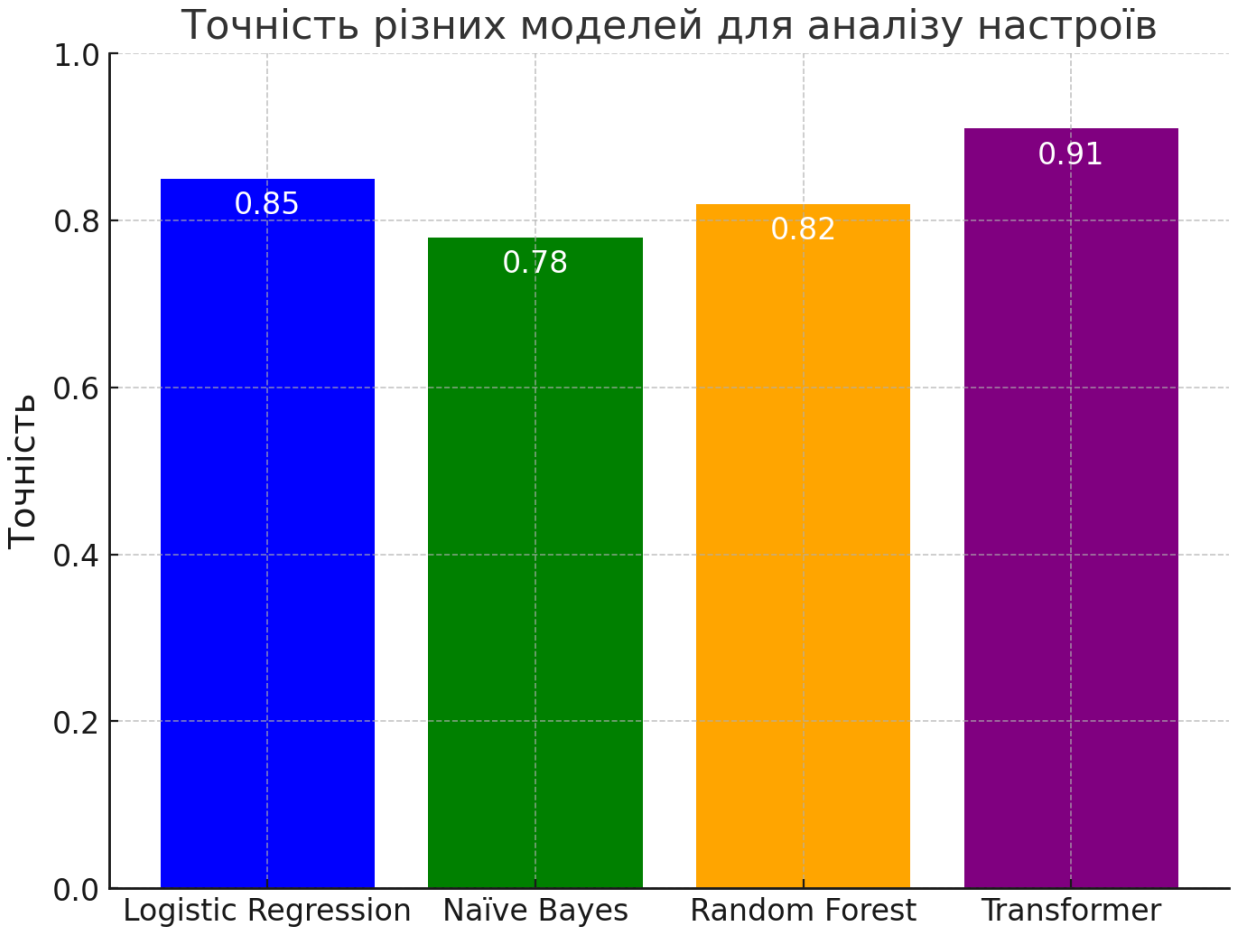

Рисунок 1.4 – Графік точності моделей для аналізу настроїв (Додаток Д)

Прогнозування трендів, у свою чергу, є ще однією важливою задачею для машинного навчання в контексті аналізу текстових даних. За допомогою різних алгоритмів можна передбачити, які теми або події стануть популярними в майбутньому, а також як змінюватиметься громадська думка з часом. Вивчення трендів дозволяє компаніям і політичним організаціям адаптувати свої стратегії, реагуючи на зміни в настроях аудиторії.

Методи кластеризації, як-от k-середніх (k-means) або ієрархічна кластеризація, допомагають групувати схожі за змістом тексти, що дає змогу виявляти поточні та

зростаючі теми. Ці методи дозволяють визначити, які ключові слова або фрази починають набирати популярність у мережах і можуть стати основою нових трендів [40]. Кластеризація є важливим етапом у розробці моделей для прогнозування трендів, оскільки вона дозволяє не тільки виділяти найбільш популярні теми, а й прогнозувати, як ці теми можуть змінюватися з часом.

Серед новітніх підходів до прогнозування трендів можна виділити *графові моделі*, які використовують структуру зв'язків між словами, користувачами або подіями для виявлення тенденцій. Наприклад, аналіз мереж користувачів може допомогти виявити групи людей, які активно обговорюють певну тему, і таким чином визначити її ймовірне поширення серед ширшої аудиторії. За допомогою таких технологій можна прогнозувати, як певні події можуть вплинути на громадську думку або підвищити інтерес до певних тем, що важливо для планування маркетингових або політичних кампаній.

Сучасні *моделі глибокого навчання*, зокрема BERT (Bidirectional Encoder Representations from Transformers), використовуються для кращого розуміння контексту і виявлення трендів у текстових даних. Ці моделі, зокрема трансформери, здатні обробляти велику кількість текстових даних, враховуючи складні залежності між словами і їх контекстом. Вони дозволяють отримати більш точне уявлення про те, як конкретні події можуть впливати на виникнення нових трендів в обговореннях на соціальних платформах[22, 23].

Моделі глибокого навчання мають перевагу перед традиційними методами машинного навчання, оскільки вони здатні обробляти і вивчати зв'язки між словами та контекстами, що надає більш точні результати в складних задачах, де важливі нюанси і контекстна інформація. Для прогнозування трендів це дозволяє не тільки точніше класифікувати тексти за емоційним тоном, а й прогнозувати, які теми можуть зростати в популярності в найближчому майбутньому.

1.5 Огляд сучасних досліджень та застосувань в аналізі текстових даних

Сучасні дослідження в області аналізу текстових даних з соціальних мереж активно інтегрують новітні методи машинного навчання та глибокого навчання для вирішення складних завдань. Аналіз текстів на платформі соціальних мереж дозволяє виявляти тренди, класифікувати контент за настроями, а також прогнозувати поведінкові моделі користувачів. Серед найбільш важливих застосувань — аналіз настроїв, прогнозування трендів, класифікація контенту, виявлення фейкових новин і вивчення громадської думки. Нижче, на рисунку 1.5, мною наведена діаграма, яка відображає найпопулярніші сфери застосування аналізу даних.

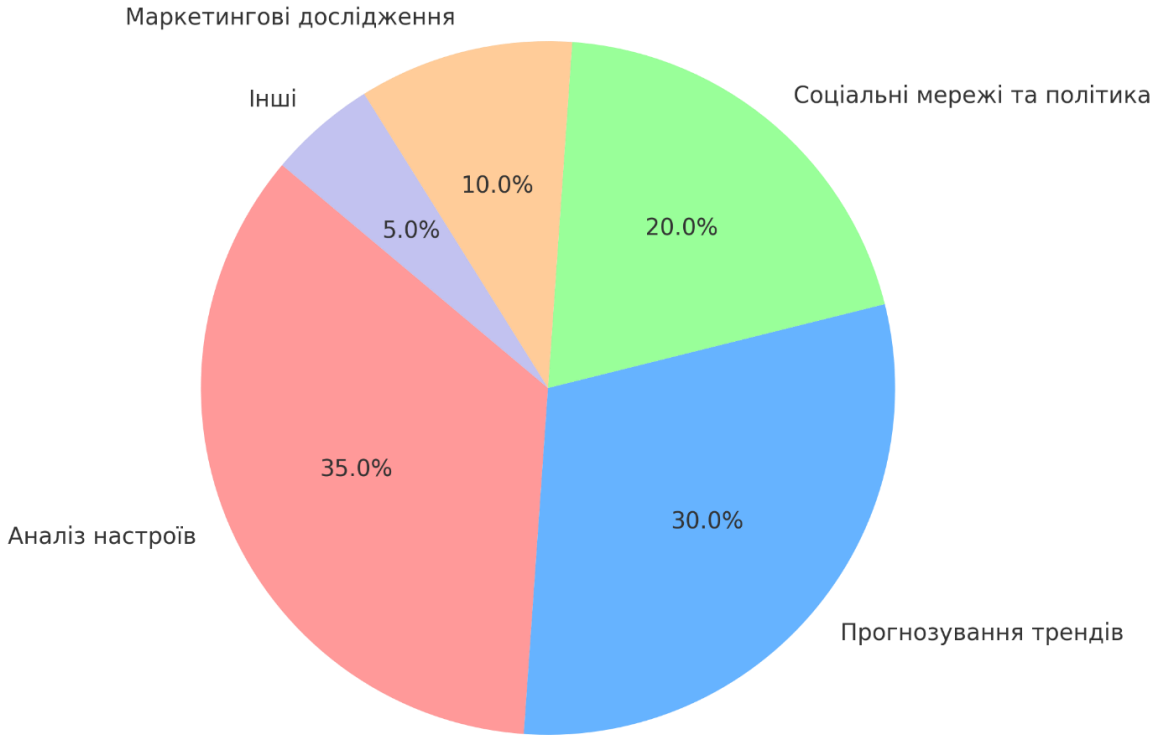

Рисунок 1.5 – Діаграма розподілу популярності тем досліджень (Додаток Е)

Новітні методи аналізу текстових даних активно використовують машинне навчання для класифікації текстів, виявлення позитивних або негативних емоцій користувачів, а також для прогнозування змін в громадській думці на основі обговорень в соціальних медіа. Наприклад, на основі текстового контенту з Twitter чи

Facebook можна визначати тональність постів, що дозволяє оцінювати настрої громадськості щодо політичних подій, нових товарів або брендів. Методика обробки природних мов (NLP), яку активно застосовують для таких цілей, включає алгоритми для аналізу тональності тексту, як-от наївний баєсівський класифікатор або логістична регресія, що дають можливість виявляти емоційне забарвлення висловлювань[24].Проте для обробки складніших текстів, що містять більше контекстуальних аспектів, з’явилися більш складні методи, як-от глибокі нейронні мережі (наприклад, BERT), які враховують контекст слів і дозволяють досягати високої точності у визначенні емоційних відтінків тексту [23].

Аналіз трендів та прогнозування за допомогою даних із соціальних мереж використовує різноманітні технології для виявлення тем, які набирають популярності. Для цього активно застосовуються методи кластеризації та графові технології, які допомагають моделювати зв’язки між різними користувачами і тематичними групами. Один з таких методів — Latent Dirichlet Allocation (LDA) — дозволяє виділяти основні теми в текстах, що дає можливість передбачити подальший розвиток обговорень, а також зростаючі або занепадаючі тенденції. Використання цих методів допомагає також прогнозувати популярність певних продуктів або політичних подій у майбутньому.

Важливим напрямом досліджень є також виявлення фейкових новин та інформаційних маніпуляцій в соціальних мережах. Враховуючи швидкий темп поширення інформації в таких мережах, існує нагальна потреба в системах, які можуть автоматично виявляти недостовірну інформацію. Для цього розробляються алгоритми, які на основі аналізу контексту і структури тексту можуть визначити, чи є новина фейковою. Такі моделі активно використовують методи машинного навчання для класифікації текстів за ознаками правдивості, що дає можливість зменшити вплив дезінформації в медіа-просторі[28].

Значну роль у розвитку цих напрямів відіграють також інструменти для збору та обробки даних. Зокрема, використання API соціальних мереж дозволяє

здійснювати масовий збір інформації в реальному часі, що особливо важливо для вивчення реакцій громадськості на поточні події[28]. Інструменти, такі як Twitter API, дозволяють здійснювати безперервний моніторинг постів та коментарів, що допомагає створювати детальні моделі для прогнозування настроїв і трендів.

Дослідження в галузі аналізу текстових даних також зосереджуються на розробці більш складних і точних моделей, що можуть обробляти не лише текстову, а й мультимедійну інформацію (відео, зображення, аудіо). Цей підхід дозволяє виявляти не лише текстові емоції, а й більш глибокі відгуки, що містяться в медіа-контенті. Таким чином, розвиток технологій дозволяє з більшою точністю прогнозувати реакцію користувачів на різні події, забезпечуючи глибший аналіз великих даних.

Висновки до Розділу 1

У результаті аналізу наукових джерел та сучасних досліджень у галузі інтелектуального аналізу текстових даних, виконаного в рамках першого розділу, було визначено ключові аспекти та методологічні підходи, які використовуються для роботи з текстами з соціальних мереж.

По-перше, встановлено, що обробка текстів із соціальних мереж має низку особливостей. Серед них варто виділити специфічну структуру тексту, яка часто включає скорочення, сленг, емодзі та хештеги. Ці елементи значно ускладнюють традиційні підходи до обробки текстів і вимагають адаптації методів попередньої обробки даних. Для цього ефективно застосовуються сучасні бібліотеки та інструменти, такі як NLTK, SpaCy та інші, що забезпечують токенізацію, нормалізацію тексту, а також вилучення семантично важливих компонентів.

По-друге, охарактеризовано сучасні підходи до аналізу текстів. Традиційні методи, наприклад, латентний семантичний аналіз і методи на основі Bag of Words або TF-IDF, залишаються актуальними для вирішення базових завдань. Однак для більш складних випадків, таких як виявлення емоційного забарвлення чи

прогнозування трендів, пріоритетними є методи глибокого навчання. Зокрема, трансформерні архітектури (наприклад, BERT, GPT) демонструють високу ефективність завдяки своїй здатності враховувати контекст на різних рівнях тексту.

По-третє, визначено основні джерела текстових даних у соціальних мережах та інструменти для їх збору. API платформ, таких як Twitter, Facebook та Instagram, надають широкі можливості для доступу до даних. Проте було відзначено, що робота з цими API пов’язана з обмеженнями, зокрема політиками доступу, лімітом запитів і вимогами щодо конфіденційності даних.

Крім того, у розділі було виділено ключові виклики обробки текстових даних із соціальних мереж. Серед них — динамічність трендів, багатомовність текстів і значний обсяг неструктурованих даних, що потребують ефективного попереднього аналізу та обробки.

Проведений аналіз дозволив сформулювати чіткі вимоги до системи для інтелектуального аналізу текстових даних із соціальних мереж. Це включає необхідність у:

- забезпеченні ефективного збору та збереження текстових даних із врахуванням обмежень API;
- використанні сучасних підходів до попередньої обробки тексту, які враховують специфіку соціальних мереж;
- застосуванні моделей машинного навчання з акцентом на трансформери для аналізу текстів із врахуванням контексту та емоційного забарвлення.

Таким чином, у рамках першого розділу закладено теоретичну основу для подальшого дослідження обраної теми та розробки системи, яка забезпечить збір даних із соціальних мереж, їх обробку, аналіз та прогнозування трендів у громадській думці.

РОЗДІЛ 2
ТЕОРЕТИЧНІ ОСНОВИ МАШИННОГО НАВЧАННЯ ТА АНАЛІЗУ ТЕКСТУ

2.1 Вступ до машинного навчання та його застосування в аналізі тексту

Машинне навчання є одним з основних напрямків штучного інтелекту, який дозволяє комп'ютерам вчитися на основі даних без необхідності вказувати кожен крок алгоритму вручну. Це стало важливим інструментом для вирішення різноманітних задач, таких як класифікація, прогнозування, виявлення аномалій, а також для обробки текстових даних. Завдяки здатності аналізувати великі обсяги тексту, машинне навчання застосовується у різних сферах, від бізнесу до соціальних наук, що дозволяє автоматизувати та вдосконалювати процеси прийняття рішень.

Машинне навчання базується на ідеї, що алгоритми можуть «вчитися» на даних, знаходити в них патерни і закономірності, а потім застосовувати ці знання для вирішення нових задач. Основними етапами застосування машинного навчання є підготовка даних, навчання моделей на основі цих даних та оцінка результатів. Підготовка даних включає в себе їх попередню обробку, таку як токенізація, видалення стоп-слів та нормалізація, що дозволяє перетворити текст у числову форму, зрозумілу для машин.

Основні підходи машинного навчання, які використовуються для аналізу тексту, включають алгоритми класифікації, регресії, а також кластеризації. Класифікація тексту є однією з найбільш популярних задач, де текст автоматично поділяється на категорії, такі як визначення теми чи емоційного забарвлення повідомлення. Наприклад, у соціальних мережах класифікація може застосовуватися для виявлення тональності коментарів або публікацій, що дозволяє відстежувати громадську думку або виконувати автоматичний аналіз відгуків користувачів.

Також машинне навчання активно використовується для аналізу настроїв, що є одним з важливих напрямків обробки текстових даних. Для аналізу настроїв використовуються моделі, які здатні розпізнавати позитивні, негативні та нейтральні

емоції в текстах. Це дозволяє автоматично оцінювати реакції на різні події або продукти в соціальних мережах, що особливо корисно для маркетингових кампаній і вивчення громадської думки.

Глибоке навчання (deep learning) стало важливою частиною машинного навчання в аналізі тексту. Використання нейронних мереж, таких як RNN (рекурентні нейронні мережі) і трансформери, дозволяє досягти високих результатів в задачах генерації тексту, переведення мови, а також у виявленні і прогнозуванні трендів у соціальних мережах. Моделі типу GPT (Generative Pretrained Transformer) демонструють здатність не лише генерувати зв'язні тексти, але й адаптувати їх під конкретні запити, що відкриває нові можливості для автоматизованого контенту[29].

Важливим етапом в роботі з текстовими даними є оцінка точності моделей машинного навчання. Для цього використовуються різноманітні метрики, зокрема точність, відчутність і специфічність, що дозволяють визначити, наскільки добре модель виконувала завдання. Оцінка ефективності допомагає виявити недоліки моделей та оптимізувати їх для досягнення кращих результатів.

Машинне навчання стає важливим інструментом для аналізу трендів і прогнозування змін у громадській думці. У соціальних мережах дані оновлюються в реальному часі, що дозволяє отримувати актуальну інформацію про настрої людей, тенденції та потенційні зміни у суспільстві. Це дає змогу прогнозувати, як зміни в політиці, економіці або культурі можуть вплинути на громадську думку, допомагаючи компаніям, урядам та організаціям приймати обґрунтовані рішення.

Застосування машинного навчання в аналізі тексту також має велике значення в різних галузях. Наприклад, в маркетингу машинне навчання дозволяє автоматизувати аналіз відгуків споживачів, що дає компаніям можливість швидко реагувати на зміни в настроях клієнтів. В освітніх установах воно допомагає в автоматичному аналізі тестів, а в сфері охорони здоров'я — у прогнозуванні епідемій на основі аналізу медичних записів.

Отже, машинне навчання є надзвичайно потужним інструментом для автоматизації обробки текстових даних, що дозволяє ефективно вирішувати завдання, пов'язані з класифікацією, аналізом настроїв і прогнозуванням трендів. Його широке застосування в різних сферах життя робить цю технологію незамінною в сучасному світі, де обробка великих обсягів даних є критично важливою.

2.2 Огляд класичних моделей машинного навчання для аналізу текстових даних

Аналіз текстових даних є складною задачею, що вимагає використання різноманітних методів машинного навчання. Класичні моделі машинного навчання вже давно застосовуються в обробці текстів і продовжують залишатися ефективними, незважаючи на розвиток більш складних підходів, таких як глибоке навчання. Далі я розгляну кілька класичних моделей, які активно використовуються для аналізу текстових даних.

Наївний байєсівський класифікатор (Naive Bayes Classifier) — це одна з найбільш популярних і простих моделей для класифікації текстів. Він ґрунтується на припущенні про незалежність ознак (слова, фрази) у кожному класі. Це означає, що ймовірність того, що певний текст належить до конкретного класу, визначається як добуток ймовірностей кожного слова належати цьому класу. Наївний байєс особливо ефективний у випадках, коли в текстах присутня велика кількість ознак, але самі ознаки мають малу кореляцію. Цей підхід успішно застосовується в задачах спам-фільтрації, аналізу настроїв і класифікації текстів за темами. Наївний байєс має ряд переваг, серед яких швидкість роботи та можливість ефективно працювати з малими наборами даних[40].

Метод опорних векторів (Support Vector Machines, SVM) — це потужний метод для класифікації текстів, який особливо добре працює при вирішенні задач, де існує чітка межа між класами. SVM використовує гіперплощину для розділення даних у

багатовимірному просторі, що дозволяє ефективно класифікувати навіть нелінійно роздільні дані. Для аналізу тексту цей метод часто використовують у задачах, де потрібно класифікувати тексти на основі їх тематики або настрою, наприклад, для визначення, чи є твіт позитивним чи негативним. Однією з головних переваг SVM є його здатність добре працювати навіть при невеликій кількості тренувальних даних. Однак ця модель може бути чутливою до вибору параметрів, що потребує ретельної налаштування[36].

Дерево рішень (Decision Trees) — ця модель працює за допомогою побудови дерева, де кожен вузол представляє рішення, а гілки — варіанти відповідей. Кожне рішення розподіляє текстові дані на два чи більше підмножин на основі певних характеристик (наприклад, частоти слів). Цей метод добре працює, коли потрібно моделювати нелінійні залежності та мається велика кількість різноманітних ознак, що визначають класи. Популярність дерев рішень також обумовлена їх простотою в інтерпретації — моделі легко зрозуміти і використовувати для прийняття рішень у реальному часі. Проте варто зазначити, що дерева рішень мають схильність до перенавчання, якщо їх не обмежити за глибиною або кількістю вузлів[41].

Логістична регресія (Logistic Regression) — одна з класичних моделей, яка застосовується для бінарної класифікації, де необхідно визначити, чи належить текст до певної категорії (наприклад, позитивний чи негативний відгук). Зазвичай вона ефективна для задач, де необхідно зробити швидке й точне прогнозування на основі кількох ключових ознак, таких як частота слів чи наявність конкретних фраз. Логістична регресія також дозволяє отримувати інтерпретовані результати у вигляді коефіцієнтів, що описують важливість кожної ознаки для прогнозу[42].

Класифікація на основі найближчих сусідів (K-Nearest Neighbors, KNN) — це метод, який використовує ідею, що схожі тексти повинні належати до одного класу. Для кожного тексту, що підлягає класифікації, KNN шукає найближчі сусіди в навчальній вибірці, а потім приймає рішення на основі класів цих сусідів. Ця модель є особливо корисною для задач, де класи не мають чіткої межі і де важливо врахувати

локальні особливості текстів. KNN зазвичай добре працює на малих даних, але може бути неефективним на великих наборах, оскільки потребує багато часу для обчислень, особливо при великій кількості ознак і текстів[43].

Класичні моделі машинного навчання для аналізу текстових даних мають свою специфіку та застосування в залежності від поставленої задачі. Вони демонструють високу ефективність у багатьох типах аналізу, таких як класифікація настроїв, виявлення тематик, спам-фільтрація та інші. Хоча ці моделі можуть бути менш потужними в порівнянні з сучасними методами глибокого навчання, вони все ще залишаються корисними завдяки своїй простоті, швидкості навчання та хорошій інтерпретованості результатів. На рисунку 2.1 відображена моя спроба порівняти точність класичних моделей машинного навчання у залежності від обсягу оброблюваних даних.

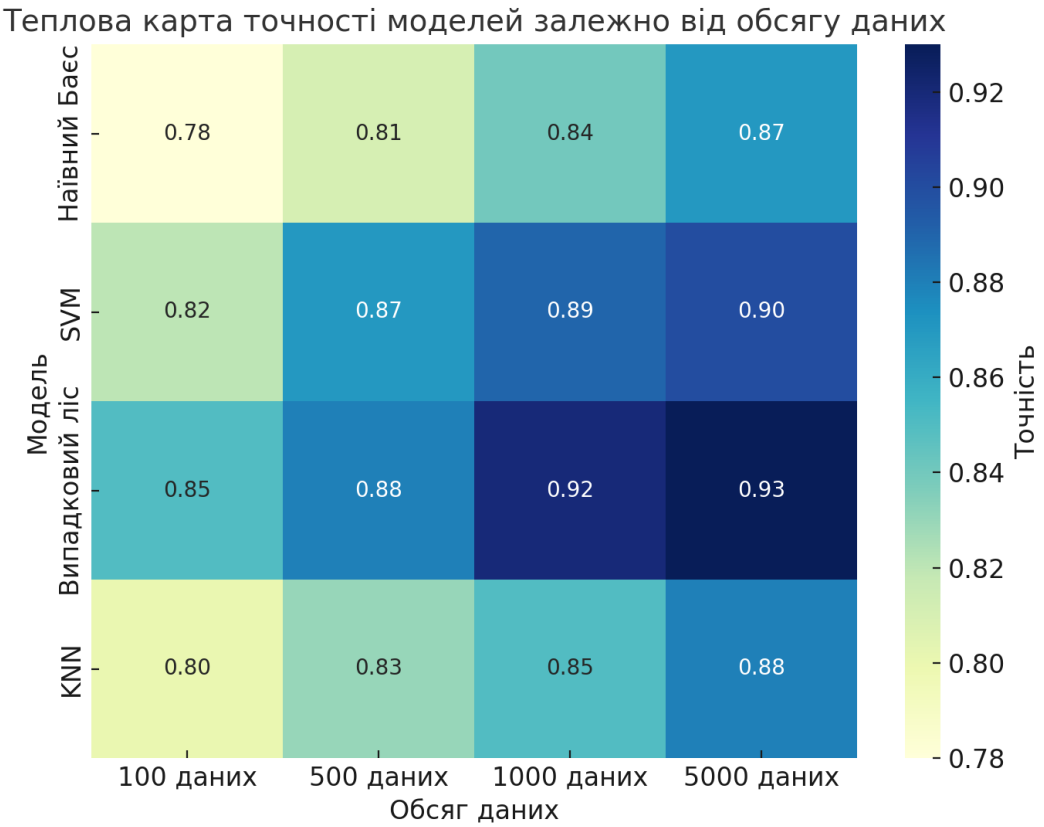

Рисунок 2.1 – Теплова карта точності моделей машинного навчання (Додаток Ж)

2.3 Сучасні моделі машинного навчання для обробки текстових даних

З розвитком машинного навчання і нейронних мереж в обробці тексту з'явилися нові підходи, які значно перевершили класичні методи аналізу текстових даних, такі як наївний байєс, підтримка векторних машин (SVM) або методи опорних векторів. Сучасні моделі, здебільшого побудовані на нейронних мережах, значно покращили точність і масштабованість обробки текстів. Основні сучасні підходи включають використання рекурентних нейронних мереж, моделей на основі трансформерів, а також варіанти глибокого навчання, які забезпечують високу ефективність на великих наборах текстових даних.

Рекурентні нейронні мережі (RNN), і зокрема їх модифікації, такі як LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Units), займають важливе місце в обробці текстових даних. Однією з ключових проблем у роботі з текстами є обробка послідовних залежностей між словами в контексті. Зазвичай текст складається з послідовних елементів, тому важливо, щоб модель враховувала не лише поточне слово, але й попередній контекст. З цією метою в розробку були впроваджені RNN, які мають внутрішні стани, що дозволяють зберігати інформацію про попередні елементи послідовності. Проте класичні RNN мають проблеми з згасанням або вибухом градієнтів під час навчання, що ускладнює їх використання для обробки довгих текстів. Моделі LSTM та GRU стали рішенням цієї проблеми, дозволяючи зберігати важливу інформацію на довгих послідовностях. Модель LSTM включає спеціальні механізми, такі як «забуття», «вхід» і «вихід», які допомагають вибирати важливі частини інформації і утримувати їх у часі[44, 45].

Моделі на основі трансформерів суттєво змінили підхід до обробки тексту. Введення трансформера в 2017 році стало проривом у машинному навчанні завдяки використанню механізму уваги, що дозволяє моделям ефективно обробляти великі обсяги текстових даних та фокусуватися на ключових частинах інформації. Це також дозволяє забезпечити більш точне розуміння контексту, оскільки увага може

звертатися до всіх слів у реченні одночасно, незалежно від їхнього положення. Зокрема, BERT (Bidirectional Encoder Representations from Transformers) є однією з найбільш відомих моделей, побудованих на архітектурі трансформера. Однією з найбільших переваг BERT є те, що він навчається на контекстуальних векторних представленнях слів, враховуючи як попередній, так і наступний контекст. Завдяки цьому BERT досягає значних результатів у задачах, таких як класифікація текстів, виявлення настроїв або відповіді на запитання. Важливим досягненням є те, що BERT навчений на величезному корпусі текстів, що дозволяє ефективно застосовувати модель до різних типів текстових завдань[23]. На *рисунку 2.2* нижче, я спробував відобразити механізм розподілу уваги, що використовується у моделях-трансформерах.

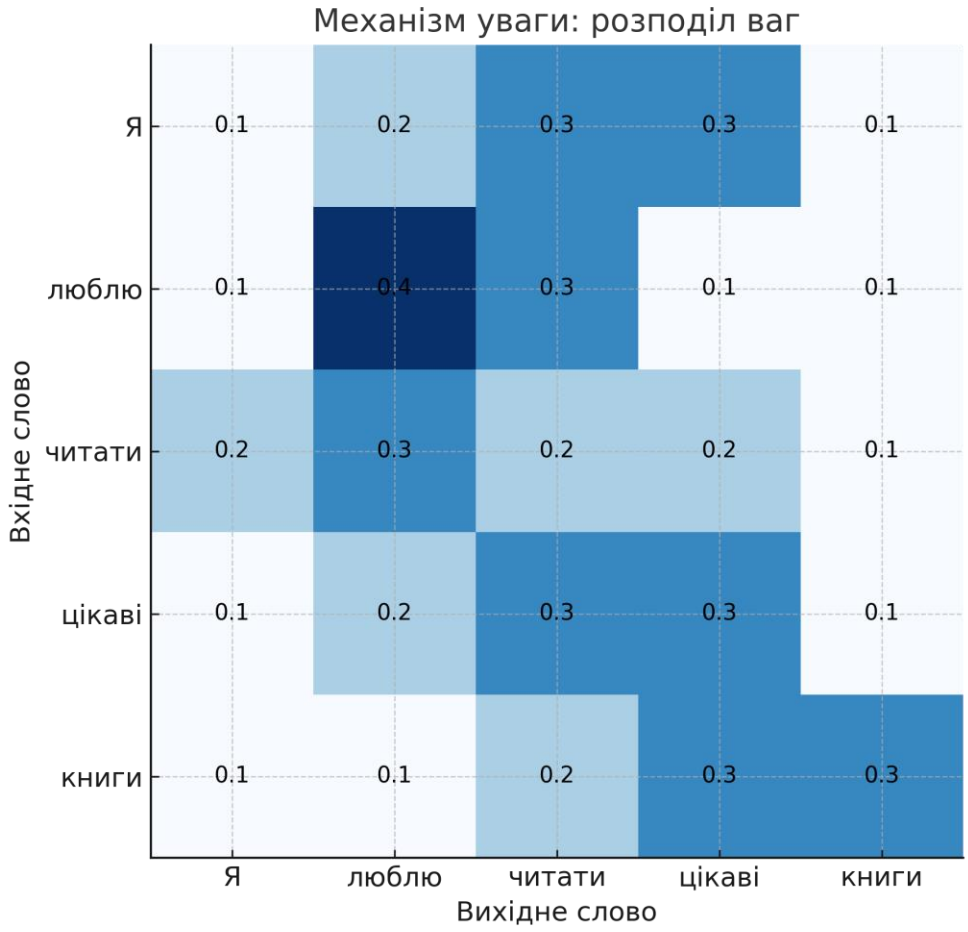

Рисунок 2.2 – Механізм уваги (Додаток З)

GPT (Generative Pre-trained Transformer), розроблений компанією OpenAI, є ще однією архітектурою трансформера, яка має значну популярність. Вона фокусується на генерації тексту, використовуючи попередньо навчений трансформер для створення логічно зв'язних відповідей на введені запити. Важливою особливістю GPT є її здатність до «перехресного навчання», коли модель здатна вирішувати не лише конкретні завдання, а й демонструвати здатність генерувати текст, що має сенс у широкому контексті. Трансформер GPT дозволяє створювати тексти різних жанрів та зберігати їхню логічну послідовність завдяки використанню великого навчального корпусу, що значно покращує його здатність до самонавчання та адаптації під різноманітні завдання. Останні версії GPT, зокрема GPT-4, продемонстрували виняткові можливості для генерації тексту, здатного змагатися з людським[46].

RoBERTa (Robustly optimized BERT approach) є модифікацією BERT, оптимізованою для більшої гнучкості та ефективності в багатьох завданнях. RoBERTa має схожу архітектуру з BERT, але вимагає менше попередньої обробки даних. Він навчається на більшому наборі даних та використовує інші техніки для покращення продуктивності. Відмінною рисою RoBERTa є те, що в процесі тренування вона не застосовує маскування слів у вхідних даних, що дозволяє зберігати більше контексту і підвищувати точність.

XLNet — це ще одна модель на основі трансформера, яка покращує класичну архітектуру, використовуючи підхід генеративного попереднього навчання. XLNet комбінує переваги autoregressive моделей (які генерують кожне слово послідовно, як це робить GPT) і автокодувальних моделей (які здатні навчатися на контекстуальних представленнях, як це робить BERT). Завдяки цьому XLNet може більш ефективно обробляти складні структурні залежності в текстах, дозволяючи створювати моделі, які є більш адаптивними до різних задач.

Multilingual Models — багато з сучасних моделей, зокрема mBERT (Multilingual BERT) та XLM-R (Cross-lingual RoBERTa), здатні працювати з текстами на різних мовах, що значно покращує точність у багатомовних або транснаціональних

середовищах. Це особливо важливо для глобального аналізу даних, оскільки великі соціальні мережі охоплюють аудиторію багатьох країн. Ці моделі навчаються одночасно на текстах з кількох мов, що дозволяє їм ефективно працювати з багатомовними даними, зберігаючи при цьому високу точність.

Моделі на основі трансформерів і глибокого навчання стали основою для сучасного аналізу текстових даних. Вони значно покращили точність і ефективність при обробці великих обсягів текстової інформації, дозволяючи враховувати контекст на всіх рівнях. Разом із розвитком нових архітектур, таких як GPT, RoBERTa, BERT та XLNet, з'являються нові можливості для розв'язання різноманітних задач, таких як виявлення настроїв, класифікація текстів або прогнозування трендів у соціальних мережах. Однак застосування цих моделей потребує значних обчислювальних ресурсів і є викликом для інтеграції у більш масштабні системи.

2.4 Методи оцінювання ефективності моделей машинного навчання

Оцінка ефективності моделей машинного навчання є важливим етапом у їх розробці та застосуванні, особливо в контексті аналізу текстових даних. Правильна оцінка дозволяє не лише визначити, наскільки точними є прогнози моделі, а й виявити слабкі місця в обробці текстів і передбачити майбутню ефективність при роботі з новими даними. Серед основних методів оцінювання ефективності моделей для аналізу текстових даних виділяють такі метрики, як точність, відгук (recall), F-міра, матриця плутанини та крос-валідація[17, 25, 29].

Правильність (Accuracy): це одна з найпоширеніших метрик, що вимірює частку правильних прогнозів серед усіх зроблених. Для двокласових задач (наприклад, класифікація настроїв на позитивні чи негативні) точність визначається як відношення кількості правильних класифікацій до загальної кількості прикладів. Проте точність не завжди є найкращим показником, особливо коли класифікація є незбалансованою.

Відгук (Recall): цей параметр застосовують, коли важливі не стільки вірно класифіковані випадки, скільки виявлення всіх можливих позитивних випадків. Відгук визначається як відношення правильно класифікованих позитивних випадків до всіх справжніх позитивних випадків. Для задачі прогнозування трендів у громадській думці або настроїв на основі текстових даних високий відгук може бути важливим для виявлення кожного потенційно важливого повідомлення.

Точність (Precision): цей показник вимірює, яка частина позитивних прогнозів є справжніми позитивами. Вона є корисною, коли важливо уникати хибних спрацьовувань, наприклад, при класифікації настроїв, де важливо, щоб лише дійсно позитивні пости були визнані як позитивні.

F-міра (F1 Score): ця метрика є гармонійним середнім між точністю і відгуком, і використовується для оцінки моделей, де важливо досягти балансу між ними. Вона особливо корисна, коли задача має незбалансовані класи, а точність і відгук можуть давати суперечливі результати.

Матриця плутанини (Confusion Matrix): це таблиця, що показує кількість правильних і неправильних класифікацій для кожного з класів. Вона дозволяє більш детально оцінити, де саме модель допускає помилки (наприклад, класифікує негативний настрій як позитивний).

Крос-валідація (Cross-validation): це метод, який дозволяє оцінити стабільність та узагальнюваність моделі, поділивши дані на декілька частин і тренуючи модель на різних підмножинах даних. Цей підхід дає можливість уникнути переобучення (overfitting) і оцінити, наскільки модель здатна працювати на нових даних.

Для аналізу текстових даних, де необхідно враховувати специфіку обробки тексту, додаються також специфічні метрики, такі як метрики схожості (наприклад, Cosine Similarity) для визначення близькості текстових векторів, або перплексія для мовних моделей, яка вимірює, наскільки добре модель передбачає ймовірність наступних слів у тексті.

Важливою частиною оцінювання є також візуалізація результатів, що дозволяє наочно продемонструвати продуктивність моделі за допомогою графіків та діаграм, таких як ROC-криві або Precision-Recall криві. Це дає можливість дослідникам та розробникам візуалізувати компроміс між різними метриками і вибрати оптимальні параметри для моделей. Приклад ROC-кривої наведено на *рисунку 2.3*.

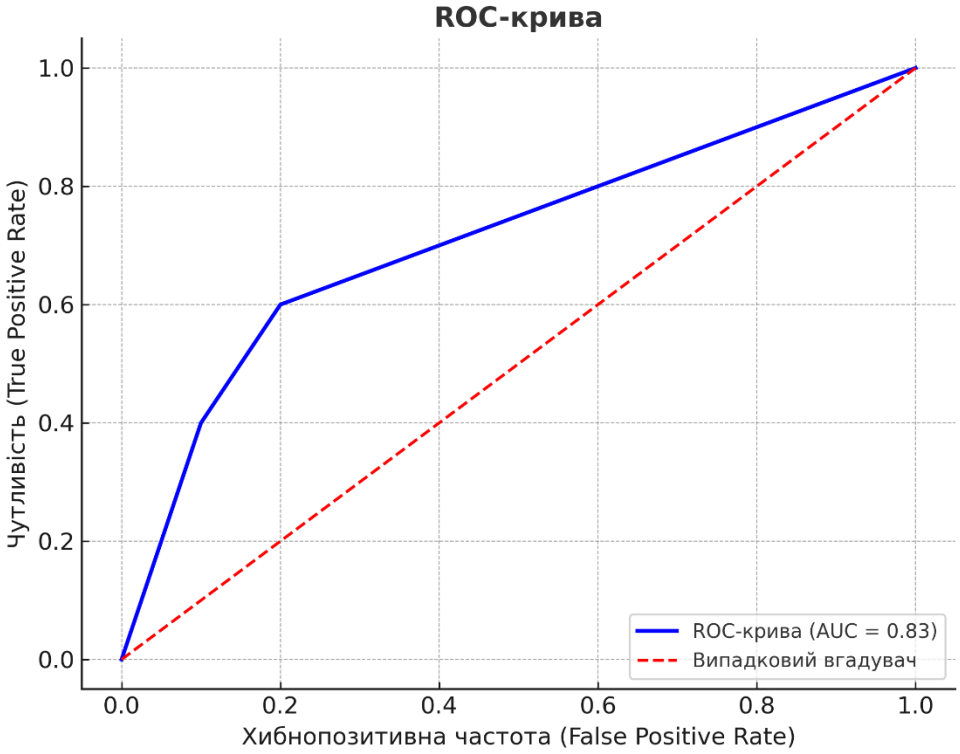

Рисунок 2.3 – ROC-крива (Додаток И)

Як зазначено у роботах з аналізу текстових даних, правильний вибір методів оцінки та використання кількох різних метрик є ключовим для досягнення високої якості моделі, особливо в контексті соціальних мереж, де важливими є не лише правильність, але й здатність моделі адаптуватися до нових, швидко змінюваних умов. Врахування цих метрик дозволяє значно покращити процес прогнозування трендів у громадській думці та інших напрямках.

2.5 Проблеми та виклики в аналізі текстових даних з соціальних мереж

Проблеми та виклики в аналізі текстових даних з соціальних мереж є важливою частиною процесу, адже специфіка цих даних визначає ряд унікальних труднощів. Однією з основних складнощів є наявність шуму в даних, який значно ускладнює точність аналізу. Шум може бути результатом використання неформальної мови, жаргону, скорочень, помилок у написанні або ж великої кількості повторюваних, малозначущих фраз. Ці фактори вимагають застосування спеціальних методів очищення даних, таких як фільтрація непридатних або дубльованих елементів, а також попередньої обробки тексту для видалення зайвих символів або змінних.

Крім того, тексти в соціальних мережах часто містять емодзі, меми або сленг, що додає складнощів при автоматичному розпізнаванні їхнього сенсу. Використання цих елементів значно ускладнює процес традиційного аналізу, де для правильного трактування необхідно враховувати контекст, у якому ці символи використовуються. Тому розробка спеціалізованих моделей для інтерпретації таких даних є важливою складовою успішного аналізу.

Важливим аспектом є неврівноваженість класів в даних. Наприклад, у випадках прогнозування настроїв або виявлення трендів на основі аналізу текстів, часто можна зіткнутися з ситуацією, де одні категорії (наприклад, позитивні настрої) значно перевищують інші (наприклад, негативні або нейтральні). Це може призводити до неадекватного навчання моделі, оскільки вона має схильність до переважної класифікації даних в одну категорію. Для вирішення цієї проблеми використовують техніки збалансування класів, такі як генерація синтетичних прикладів для менш представлених класів або застосування методів оптимізації для корекції ваг.

Іншою складністю є аналіз контексту, зокрема, виявлення відтінків значень у текстах. Соціальні мережі, як правило, мають динамічний контекст, де значення слів або фраз може змінюватися в залежності від ситуації. Це вимагає розробки моделей, здатних враховувати не тільки лексичне значення, але й соціокультурні аспекти

спілкування в мережі. Наприклад, емоційні або саркастичні вислови можуть бути неправильно інтерпретовані стандартними моделями.

Також не менш важливою проблемою є сентиментальний аналіз, коли необхідно точно визначити емоційний тон тексту. У соціальних мережах часто зустрічаються багатозначні, амбівалентні вислови, які можуть мати різні тлумачення в залежності від контексту. Для вирішення цих проблем використовуються складні моделі машинного навчання, що вимагають додаткових ресурсів для правильного розпізнавання і трактування таких виразів.

Загалом, для ефективного аналізу текстових даних з соціальних мереж необхідно вирішити безліч технічних і методологічних проблем, що вимагають постійного удосконалення моделей, інструментів і підходів до обробки даних. Їх вирішення є ключовим для отримання точних і корисних результатів у дослідженнях настроїв і прогнозуванні трендів.

Висновки до Розділу 2

У другому розділі кваліфікаційної роботи було здійснено детальний аналіз теоретичних основ, пов’язаних із використанням методів інтелектуального аналізу текстових даних для виявлення та прогнозування трендів у громадській думці. Розглянуто як класичні, так і сучасні методи обробки текстової інформації, алгоритми машинного навчання та їхню роль у вирішенні завдань аналізу великих обсягів даних із соціальних мереж.

Однією з ключових особливостей досліджених методів є їхня здатність ефективно працювати з неструктурованими даними, які становлять значну частину текстової інформації у соціальних мережах. Проаналізовані алгоритми показують високу продуктивність у класифікації текстів, кластеризації та виявленні тональності повідомлень. Це є важливим кроком до забезпечення точності в прогнозуванні трендів у громадській думці.

У процесі дослідження було розглянуто традиційні методи обробки текстових даних на основі статистичних підходів, а також сучасні підходи, засновані на використанні нейронних мереж. Серед останніх особливу увагу було приділено моделям на базі трансформерів, зокрема BERT і GPT, які є високоефективними в розумінні контексту тексту. Використання таких моделей дозволяє значно підвищити точність та релевантність результатів аналізу.

Значну увагу було приділено аналізу переваг і недоліків різних підходів до обробки текстових даних. Особливо, було підкреслено, що традиційні методи є менш вимогливими до обчислювальних ресурсів, однак поступаються в точності сучасним моделям на основі глибинного навчання. Разом із тим, використання сучасних моделей потребує більших обчислювальних потужностей та спеціалізованих знань у сфері машинного навчання.

Крім того, у розділі було описано можливості використання інструментів збору текстових даних із соціальних мереж, а також важливість дотримання етичних норм та законодавчих вимог при зборі даних.

Отже, у рамках другого розділу кваліфікаційної роботи було більш ретельно досліджено теоретичну базу стосовно машинного навчання, необхідну для розробки власної моделі аналізу текстових даних. Деякі моделі та методи, про які йшла мова у Розділі, стануть основою для реалізації практичної частини дослідження, спрямованої на створення та тестування моделей аналізу та прогнозування трендів у громадській думці.

РОЗДІЛ 3
ТЕХНОЛОГІЇ І ІНСТРУМЕНТИ ДЛЯ АНАЛІЗУ ТЕКСТОВИХ ДАНИХ

3.1 Інструменти для збору текстових даних: API соціальних мереж

Збір текстових даних із соціальних мереж є першим і одним з найважливіших етапів аналізу. Для цього використовуються різноманітні інструменти, що надаються самими соціальними платформами у вигляді API (Application Programming Interface). API дозволяють автоматизовано отримувати доступ до публічних даних з соціальних мереж, таких як Twitter, Facebook, Instagram, Reddit та інші.

API забезпечують зручний доступ до різноманітних типів контенту, таких як пости, коментарі, лайки, репости, а також метадані, що супроводжують ці елементи (наприклад, дата публікації, географічне місце, кількість взаємодій). Приблизно оцінити типи та обсяги даних, які збираються за допомогою API, можливо завдяки *рисунку 3.1*.

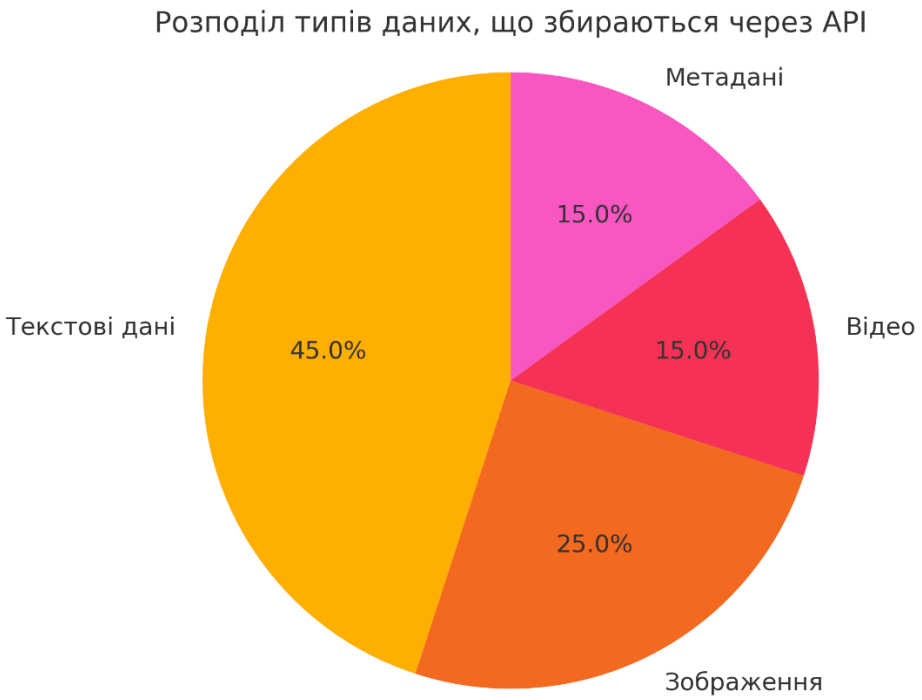

Рисунок 3.1 – Діаграма розподілу типу даних, що збираються через API (Додаток К)

Найпопулярніші API, які використовуються для збору текстових даних, включають:

Twitter API: Twitter надає потужний інтерфейс для отримання твітів і метаданих, таких як хештеги, автори, ретвіти і коментарі. Особливістю Twitter API є можливість збору даних у реальному часі, що є важливим для аналізу трендів. Для роботи з ним необхідно мати ключ доступу, що вимагає реєстрації розробника на платформі Twitter.

Facebook Graph API: Це API для збору даних з Facebook, яке дозволяє отримувати пости, коментарі та інші взаємодії з публічними об'єктами, такими як сторінки, групи та події. Однак, оскільки Facebook обмежує доступ до приватних даних, для отримання більш детальної інформації потрібно мати спеціальні дозволи або користуватися лише публічними постами.

Instagram Graph API: Це API для збору інформації з Instagram. Воно дозволяє отримувати пости, коментарі та інформацію про взаємодії з публічними акаунтами. Instagram API дозволяє працювати як з медіа-контентом (фото, відео), так і з текстовими підписами.

Reddit API: Reddit пропонує API для отримання постів, коментарів і метаданих із його форумів (сабредітів). Reddit API є досить потужним для збору даних на тему різних підкатегорій, що може бути корисно для аналізу специфічних трендів і настроїв.

Незважаючи на велику кількість доступних API, важливими аспектами є обмеження за кількістю запитів і доступ до публічної інформації. Соціальні мережі встановлюють ліміти на кількість запитів за певний період часу для запобігання зловживанням і надмірному навантаженню на сервери. Тому важливо ретельно продумати стратегію збору даних і розробити системи, що дозволяють обробляти великий обсяг інформації без порушень правил платформи.

Крім того, збору даних за допомогою API передує етап автентифікації користувача, де зазвичай використовуються ключі API або токени доступу. Ці

інструменти дозволяють контролювати доступ до даних і забезпечувати відповідність політикам конфіденційності та захисту персональних даних. [4-6].

Важливо зазначити, що дані, зібрані за допомогою API, часто потребують додаткової обробки. Наприклад, потрібно нормалізувати текст (усунення зайвих символів, транслітерація), а також перевіряти дані на коректність і релевантність, оскільки API можуть надавати не всю інформацію, яку користувач може побачити на платформі, через обмеження конфіденційності або політики самого сервісу.

Таким чином, використання API для збору текстових даних є необхідним етапом аналізу соціальних мереж, і правильний вибір та налаштування цих інструментів безпосередньо впливає на ефективність подальших етапів обробки і аналізу даних.

3.2 Платформи та бібліотеки для обробки тексту (NLTK, spaCy, TensorFlow, PyTorch)

Існує безліч потужних інструментів для обробки текстових даних, які дозволяють ефективно аналізувати та обробляти великі обсяги інформації, здебільшого в контексті обробки природної мови (NLP). Серед них виділяються такі бібліотеки, як NLTK, spaCy, TensorFlow та PyTorch, які забезпечують багатий набір інструментів для вирішення різноманітних задач аналізу тексту.

NLTK (Natural Language Toolkit) — це одна з найпоширеніших бібліотек для обробки природної мови в Python, що активно використовується для навчальних цілей і прототипування. Вона включає в себе інструменти для токенізації, лемматизації, синтаксичного аналізу та розпізнавання частин мови. NLTK також підтримує велику кількість корпусів тексту, які можна використовувати для навчання моделей та аналізу специфічних лексичних одиниць або структур[47]. Однак, хоча бібліотека й має гнучкість, її продуктивність може бути обмежена при роботі з великими даними, порівняно з іншими інструментами.

spaCy, у порівнянні з NLTK, націлена на обробку великих обсягів даних і застосовується в професійних сферах. Вона значно швидша за NLTK, що робить її ідеальною для обробки реальних даних, таких як соціальні мережі або новинні тексти. spaCy має високу продуктивність при виконанні задач токенізації, розпізнавання сутностей (NER), а також створення векторів слів для подальшого навчання моделей машинного навчання. Бібліотека також підтримує багато мов, що робить її універсальним інструментом для глобальних NLP-задач[48].

TensorFlow та *PyTorch* є основними платформами для глибокого навчання, які можуть бути використані для обробки тексту за допомогою нейронних мереж. TensorFlow, розроблений Google, є потужним інструментом для створення масштабованих моделей машинного навчання, включаючи нейронні мережі для аналізу тексту, а також для більш складних завдань, таких як автоматичний переклад або розпізнавання мовних структур. Водночас PyTorch від Facebook став популярним завдяки своїй простоті в розробці і тестуванні моделей, що особливо важливо для дослідників. Він пропонує зручну роботу з динамічними графами і дозволяє швидко експериментувати з архітектурами моделей[49]. Обидві бібліотеки підтримують передові технології для обробки тексту, зокрема трансформери, що є основою багатьох сучасних мовних моделей, таких як BERT, GPT та T5.

Hugging Face Transformers є ще однією важливою платформою, що дозволяє працювати з передовими моделями трансформерів для задач NLP. Бібліотека надає доступ до багатьох pre-trained моделей, що спрощує інтеграцію потужних рішень для аналізу настроїв, класифікації тексту чи автоматичного перекладу. Вона значно зменшує витрати часу на навчання моделей і дає змогу зосередитися на адаптації вже навчених моделей до конкретних завдань.

Кожен із цих інструментів має свої переваги: NLTK є зручним для навчальних цілей, spaCy — швидким і ефективним для практичного застосування, а TensorFlow і PyTorch дозволяють створювати та тренувати складні моделі, які можуть обробляти величезні обсяги даних. Вибір інструменту залежить від специфіки задачі: якщо

потрібно працювати з великими даними або застосовувати глибокі нейронні мережі, то TensorFlow або PyTorch будуть більш підходящими, тоді як для простих завдань аналізу тексту spaCy або NLTK можуть бути цілком достатніми, що й виходить з наведеного порівняння на *рисунку 3.2*.

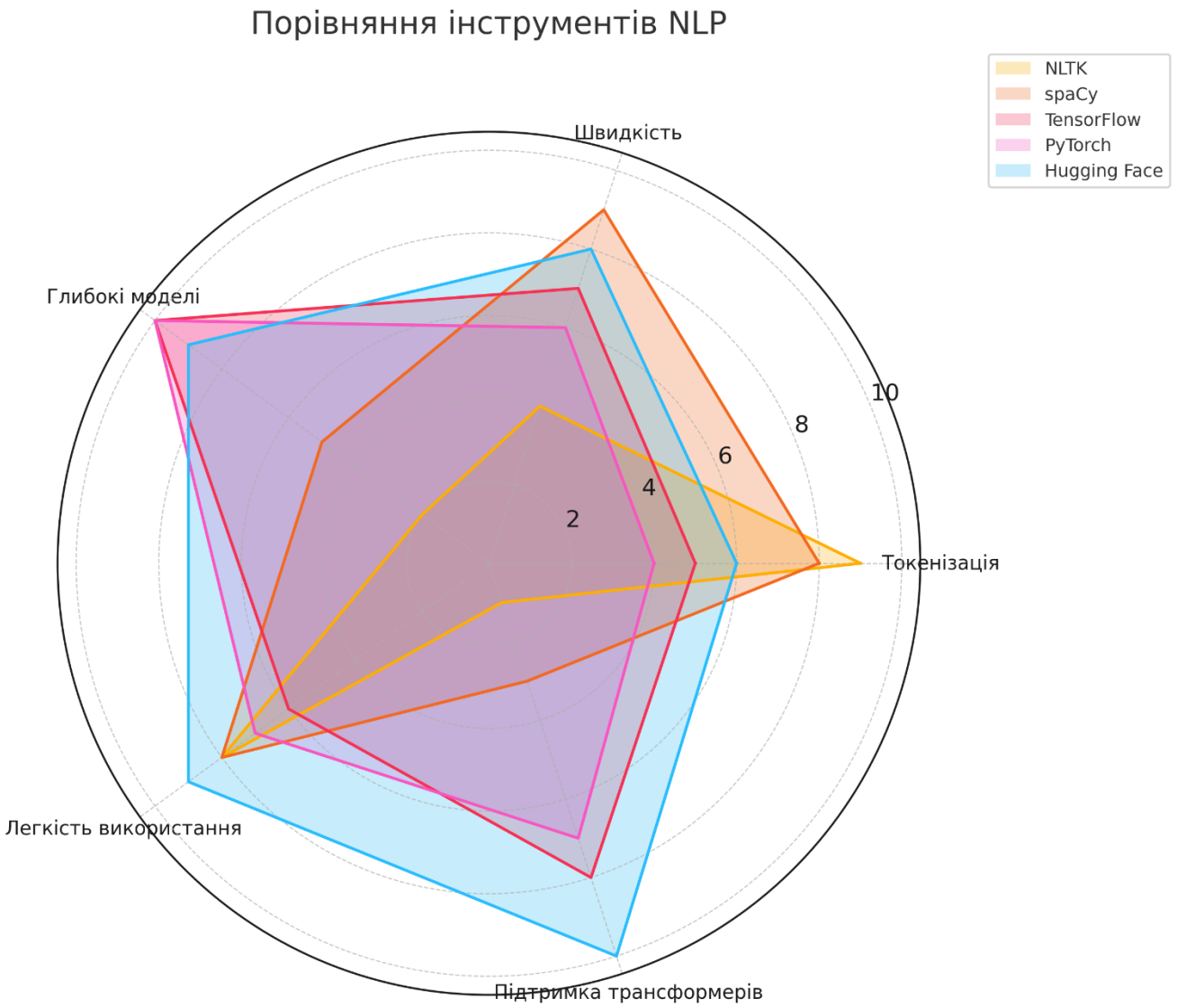

Рисунок 3.2 – Радарна діаграма порівняння інструментів NLP (Додаток Л)

Таким чином, вибір між цими платформами та бібліотеками залежить від конкретних вимог до проекту, наявності ресурсів для обробки даних і складності моделей, що плануються для реалізації.

3.3 Інструменти для аналізу великих даних та Big Data

Аналіз великих даних (Big Data) є основою багатьох сучасних досліджень, оскільки здатність ефективно працювати з великими обсягами даних має критичне значення для отримання важливих висновків. Відповідні інструменти дозволяють зберігати, обробляти та аналізувати інформацію, яка генерується соціальними мережами та іншими платформами, а також забезпечують швидке виявлення трендів, які важливі для прогнозування громадської думки або вивчення суспільних настроїв. Для цих цілей використовуються різні платформи та технології, які дозволяють працювати з даними у реальному часі, зберігати їх на розподілених системах і проводити складні обчислювальні процеси. На *рисунку 3.3* зображена блок-схема, що містить найпоширеніщі інструменти Big Data та головні їх особливості.

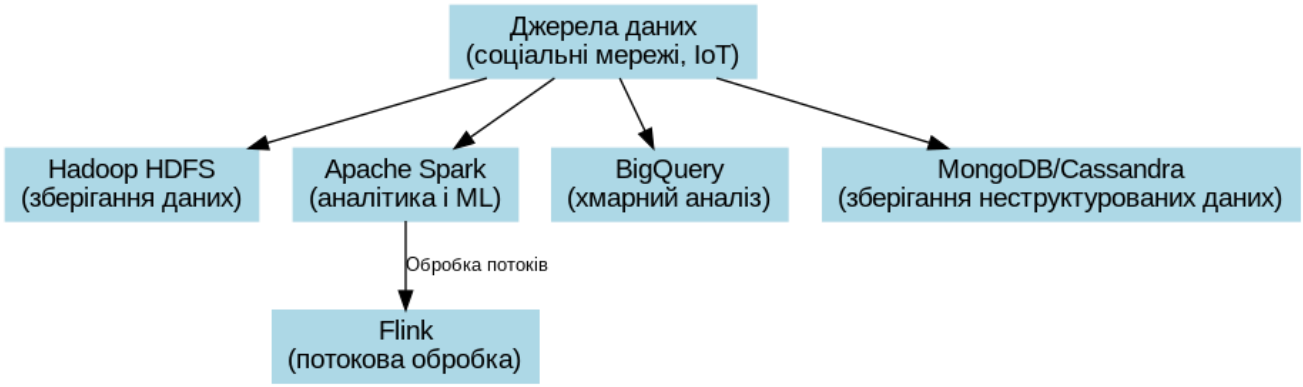

Рисунок 3.3 – Блок-схема інструментів Big Data (Додаток М)

Одним із найбільш поширених інструментів для аналізу великих даних є *Hadoop*, який став основною технологією для зберігання і обробки даних в екосистемах великих даних. Hadoop є відкритим фреймворком, що включає в себе систему зберігання Hadoop Distributed File System (HDFS), а також інструменти для обробки даних за допомогою MapReduce. MapReduce дозволяє розподіляти обчислювальні завдання серед великої кількості машин, що дозволяє ефективно обробляти величезні набори даних. Однак, у зв'язку з обмеженнями MapReduce щодо часу обробки, для більш складних задач часто використовуються інші технології, наприклад, Apache Spark[51].

Apache Spark є ще однією популярною платформою для обробки великих даних, відомою своєю здатністю працювати в реальному часі завдяки in-memory processing. Це дозволяє значно прискорити обробку даних, оскільки інформація зберігається в оперативній пам'яті, а не на жорсткому диску. Spark є дуже потужним інструментом для аналізу даних у реальному часі, що робить його ідеальним для обробки інформації, що постійно оновлюється, як, наприклад, дані з соціальних мереж або новин. Завдяки модулю MLlib, який дозволяє інтегрувати алгоритми машинного навчання, Spark є ефективним інструментом для прогнозування, класифікації та кластеризації даних.

Ще одним важливим інструментом для аналізу великих даних є *Google BigQuery*. Це хмарна платформа для аналізу великих даних, яка дозволяє користувачам виконувати SQL-запити до величезних наборів даних. Google BigQuery забезпечує високу швидкість обробки завдяки розподіленій обробці та ефективним алгоритмам зберігання даних. Вона дозволяє працювати з даними без необхідності їх попереднього завантаження, що робить її дуже зручною для швидкого аналізу великих потоків інформації, наприклад, для виявлення патернів у поведінці користувачів у соціальних мережах або інших платформах[50].

Іншою важливою технологією для роботи з великими даними є *Apache Flink*. Flink — це потужна платформа для аналізу потокових даних, яка дозволяє здійснювати обробку даних у реальному часі з високою пропускною здатністю і низькою затримкою. Flink забезпечує підтримку як пакетної, так і потокової обробки даних, що дозволяє ефективно працювати з інформацією, що надходить безперервно, наприклад, з сенсорів, IoT-пристроїв або потоків з соціальних мереж. Однією з головних переваг Flink є його здатність обробляти дані з використанням складних алгоритмів, таких як машинне навчання чи аналіз графів[50].

Також важливими інструментами є NoSQL бази даних на кшталт *MongoDB* та *Cassandra,* які спеціалізуються на роботі з неструктурованими даними, що часто зустрічаються в текстах з соціальних мереж. NoSQL бази даних відрізняються від

традиційних реляційних баз тим, що дозволяють зберігати дані у форматах, які не обов'язково мають бути структурованими, що є корисним для роботи з великими обсягами текстової інформації.

Залежно від обсягу даних, вимог до швидкості обробки, потреби в реальному часі та типу аналізу, вибір інструментів може варіюватися, і часто для досягнення найкращих результатів використовують комбінацію кількох технологій.

3.4 Огляд сучасних технологій для візуалізації результатів аналізу

Візуалізація результатів аналізу є критично важливим етапом у роботі з великими даними, адже вона дає змогу чітко інтерпретувати результати складних обчислень, виявляти тренди та зв'язки в даних. Особливо це важливо для текстових даних, отриманих з соціальних мереж, де величезний обсяг інформації потребує ефективних методів для наочного представлення. Оскільки аналіз текстових даних часто включає не лише числові значення, а й контекстуальні та лексичні елементи, то візуалізація стає важливим інструментом для зрозуміння і комунікації результатів аналізу.

D3.js — це одна з найбільш популярних JavaScript-бібліотек для створення інтерактивних візуалізацій. Вона дозволяє реалізувати складні графічні елементи, такі як графи, діаграми, карти та інші інтерактивні компоненти. Однією з особливостей D3.js є те, що вона працює з документами, що описуються мовою XML (SVG, HTML), що дозволяє створювати складні анімації та інтерактивні елементи. Завдяки своїй гнучкості, D3.js широко використовується для візуалізації зв'язків між користувачами в соціальних мережах, аналізу трендів у часі, відображення географічних патернів і т. д.

Ще одним важливим інструментом для візуалізації є *Tableau,* який користується популярністю у сфері бізнес-аналітики. Цей інструмент дозволяє легко створювати інтерактивні панелі та звіти, візуалізуючи великі набори даних без необхідності

глибоких знань програмування. Tableau забезпечує гнучкість у роботі з різними джерелами даних, дозволяючи зручно імпортувати текстові дані з різних форматів. Для аналізу соціальних медіа, Tableau може використовуватися для побудови графіків настроїв, трендів, а також виявлення ключових моментів в онлайн-дискусіях або тематичних кластерів.

Power BI, розроблений Microsoft, є ще одним популярним інструментом для візуалізації даних, який інтегрується з численними джерелами даних. Завдяки потужним інструментам для аналізу і створення візуальних звітів, Power BI дозволяє проводити аналіз соціальних медіа, визначати ключові показники впливу та візуалізувати динаміку настроїв у реальному часі. Користувачі можуть створювати різноманітні інтерактивні графіки, що відображають важливі метрики, такі як частка позитивних чи негативних відгуків, визначати тренди або зміну популярності тем в соцмережах.

Іншим популярним інструментом для візуалізації є *Plotly*, який дозволяє створювати інтерактивні графіки і діаграми з відкритим кодом, що забезпечує зручність використання в ряді мов програмування, таких як Python, R та JavaScript. Plotly підтримує створення 3D-графіків та теплових карт, що робить його дуже корисним для візуалізації складних багатовимірних даних. У контексті аналізу текстових даних з соціальних мереж Plotly може бути використаний для створення графіків, що зображують взаємодії між користувачами, або для представлення результатів класифікації тексту.

Ще одним потужним інструментом для аналізу та візуалізації є *Gephi*, який спеціалізується на роботі з мережевими даними. Gephi дозволяє візуалізувати структури зв'язків, які можуть бути виявлені в даних з соціальних мереж, таких як взаємодії між користувачами, репости, лайки та коментарі. Цей інструмент активно використовують для створення графів, що дозволяють визначати ключових учасників мережі або відображати динаміку змін у соціальних зв'язках з часом. Gephi також

підтримує інтерактивні функції, що дозволяють маніпулювати мережею для глибшого аналізу.

RStudio з бібліотеками ggplot2 та plotly є ще одним потужним інструментом для візуалізації. За допомогою RStudio можна легко створювати детальні графіки, гістограми, теплові карти та багато інших видів візуалізацій. ggplot2, зокрема, є однією з найпоширеніших бібліотек для створення статистичних графіків, тоді як plotly дозволяє додавати інтерактивні елементи до графіків. З RStudio також можна інтегрувати різноманітні моделі машинного навчання для подальшого аналізу даних та прогнозування.

Для інтерактивних мережевих візуалізацій також активно використовується *Sigma.js*, яка дозволяє будувати графи і візуалізувати складні мережеві зв'язки. Ця бібліотека JavaScript дозволяє створювати інтерактивні мережеві графи, що дає змогу візуалізувати структури соціальних мереж або аналізувати взаємодії між користувачами.

У контексті візуалізації результатів аналізу великих даних важливим інструментом є *Apache Superset*. Це платформа з відкритим кодом для візуалізації даних, яка дозволяє створювати інтерактивні дашборди та звіти. Superset дозволяє працювати з великими масивами даних, включаючи соціальні медіа, і може бути інтегрована з різними базами даних для швидкого створення наочних графіків[51].

Одним з найбільш інноваційних інструментів для візуалізації є *TensorBoard*, який є частиною екосистеми *TensorFlow*. TensorBoard дозволяє візуалізувати процес навчання нейронних мереж, а також аналізувати результати обчислень в реальному часі. Він може бути особливо корисним для візуалізації результатів тренування моделей глибокого навчання, що використовуються для аналізу текстових даних, таких як моделі для прогнозування трендів або настроїв в текстах.

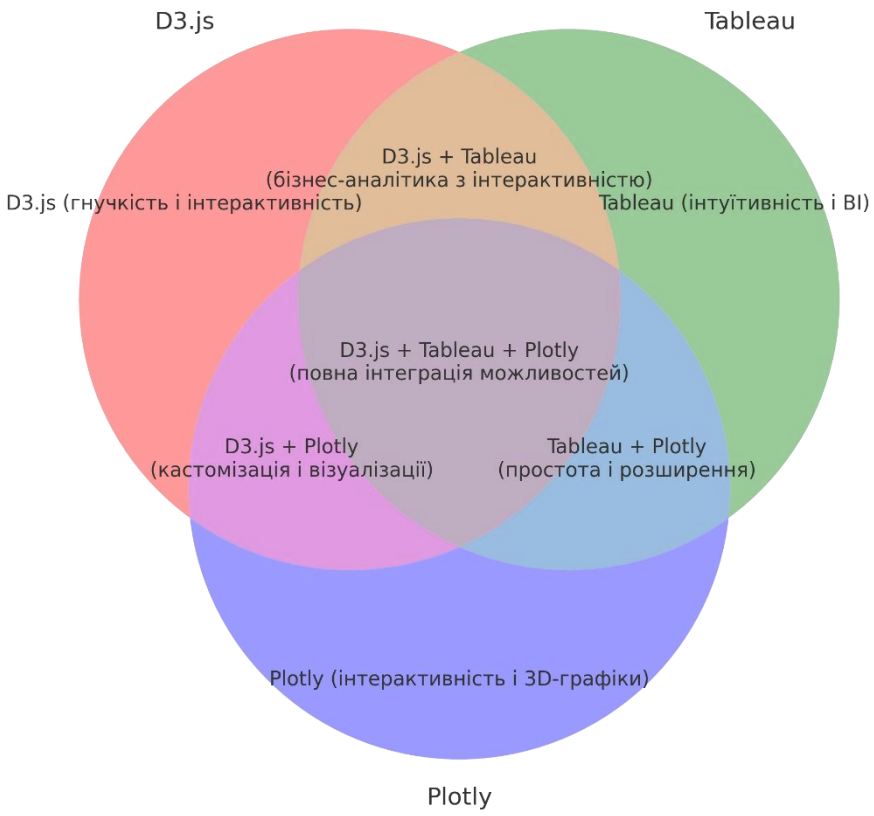

Рисунок 3.4 – Діаграма Венна (Додаток Н)

Загалом, сучасні технології для візуалізації даних дають можливість ефективно працювати з великими обсягами інформації та отримувати точні результати, які можна використовувати для прийняття рішень або для подальших наукових досліджень. На *рисунку 3.4* наведена Діаграма Венна, що демонструє спільні риси та особливості декількох інструментів візуалізації.

3.5 Інтеграція хмарних платформ, ШІ та автоматизація обробки текстових даних через контейнеризацію та оркестрацію

Сучасні хмарні платформи та технології штучного інтелекту (ШІ) значно трансформують підходи до обробки текстових даних, а в поєднанні з контейнеризацією і оркестрацією відкривають нові можливості для автоматизації та масштабування цих процесів. Хмарні сервіси дозволяють використовувати

високопродуктивні інструменти для аналізу текстів, такі як Google Natural Language API, AWS Comprehend та Azure Cognitive Services, які автоматично класифікують текст, виявляють емоції та інші важливі характеристики. Однак, для забезпечення гнучкості, автоматизації та масштабування таких процесів у великих системах часто використовують контейнеризацію.

Контейнеризація за допомогою таких інструментів, як *Docker*, дозволяє ізолювати додатки для обробки текстових даних у незалежні контейнери, що робить процеси більш ефективними і відтворюваними. Це дозволяє розгортати системи обробки тексту на різних середовищах без необхідності в конфігурації кожного сервера. Зокрема, використання Docker для розгортання моделей ШІ, таких як BERT або GPT-3, дозволяє безперешкодно інтегрувати їх в обчислювальні процеси для аналізу текстових даних із соціальних мереж або інших джерел.

Оркестрація контейнерів за допомогою таких інструментів, як *Kubernetes*, дає змогу автоматично масштабувати і керувати такими контейнерами, забезпечуючи їхню високу доступність і стабільність навіть при збільшенні обсягу даних. Зокрема, Kubernetes дозволяє ефективно розподіляти навантаження між різними контейнерами, що забезпечує масштабованість у реальному часі для таких завдань, як аналіз настроїв в текстах або прогнозування трендів на основі великих даних. Загалом, ефективність контейнеризації та оркестрації наглядно продемострована на *рисунку 3.5*.

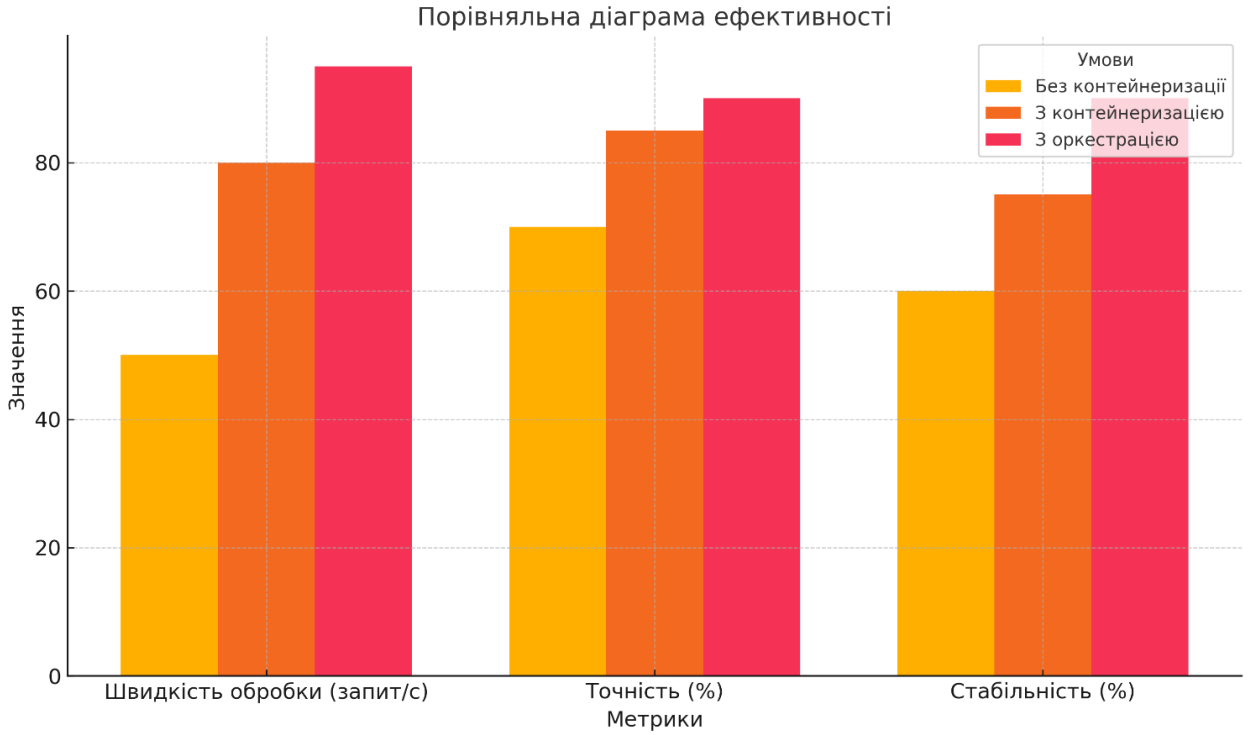

Рисунок 3.5 – Порівняльна діаграма ефективності роботи з текстовими даними (Додаток О)

Додатково, за допомогою *CI/CD (continuous integration/continuous delivery)* можна автоматизувати процеси тестування, розгортання та оновлення моделей машинного навчання для обробки тексту. Наприклад, використовуючи Jenkins або GitLab CI, можна налаштувати автоматичні пайплайни для розгортання моделей на хмарних серверах, що дозволяє швидко і ефективно оновлювати системи без ручного втручання.

Завдяки інтеграції хмарних платформ, ШІ, контейнеризації та оркестрації, можна створювати потужні, автоматизовані системи для обробки текстових даних, які здатні обробляти великий обсяг інформації в реальному часі, з високою швидкістю та точністю. Така інтеграція не лише підвищує ефективність обробки тексту, але й дозволяє швидко адаптувати системи до змінних умов або нових вимог, що є важливим для завдань, пов'язаних із прогнозуванням трендів і аналізом настроїв у соціальних мережах або інших джерелах.

Висновки до Розділу 3

У розділі детально розглянуто найпопулярніші технології для збору, обробки, аналізу та візуалізації текстових даних, а також особливості їхнього використання в сучасних умовах.

Збір текстових даних із соціальних мереж здійснюється через API, які надають доступ до різноманітних публічних даних. Наприклад, Twitter API дозволяє отримувати твіти в реальному часі, Facebook Graph API — взаємодії з публічними сторінками, а Reddit API — аналізувати теми в сабредітах. Однак використання API супроводжується обмеженнями, такими як ліміти запитів та необхідність автентифікації. Важливою умовою є дотримання політик конфіденційності, що накладає додаткові вимоги до управління даними.

Для обробки текстових даних застосовуються різноманітні інструменти, серед яких NLTK, spaCy, TensorFlow та PyTorch. NLTK є ефективною бібліотекою для навчальних завдань, тоді як spaCy забезпечує високу продуктивність у роботі з великими обсягами даних. TensorFlow та PyTorch дозволяють створювати складні моделі машинного навчання для аналізу тексту, що особливо важливо для задач прогнозування, класифікації або генерації тексту. Hugging Face Transformers, у свою чергу, спрощує використання передових мовних моделей, таких як BERT чи GPT, для широкого спектра NLP-задач.

Аналіз великих даних реалізується через технології Big Data, серед яких виділяються Hadoop, Apache Spark, Google BigQuery, Apache Flink та NoSQL бази даних. Hadoop та Spark забезпечують масштабовану обробку даних, зокрема в реальному часі, що є ключовим для аналізу динамічних потоків із соціальних мереж. Google BigQuery пропонує інструменти для швидкого аналізу великих наборів даних, а Flink дозволяє обробляти як пакетні, так і потокові дані, що зручно для складних сценаріїв. NoSQL бази даних, такі як MongoDB, дозволяють працювати з неструктурованими текстовими даними.

Візуалізація даних є завершальним етапом, що забезпечує інтуїтивне представлення результатів аналізу. Для цього використовуються інструменти, як-от D3.js, Tableau, Power BI, Gephi, Plotly, RStudio з ggplot2, а також Sigma.js. Вони дозволяють створювати інтерактивні графіки, карти та діаграми, що наочно демонструють тренди, взаємозв’язки та ключові аспекти текстових даних. Наприклад, Gephi є ефективним для візуалізації соціальних мереж, тоді як Tableau та Power BI забезпечують бізнес-аналітичні інсайти.

Таким чином, вибір технологій і інструментів для збору, обробки, аналізу та візуалізації текстових даних залежить від специфіки завдань, обсягів даних та ресурсів. Інтеграція різних підходів дозволяє досягти високої точності та ефективності аналізу, що відкриває нові можливості для досліджень у сфері соціальних мереж, бізнес-аналітики та штучного інтелекту.

РОЗДІЛ 4
ПРАКТИЧНА РЕАЛІЗАЦІЯ АНАЛІЗУ ДАНИХ

4.1 Опис використаних даних та попередня обробка

Набір даних, що був використаний, складається з текстових публікацій, отриманих з різних соціальних мереж. Дані представлені у вигляді таблиці, де кожен рядок відповідає одному посту. Формат набору даних – CSV (Comma Separated Values), що дозволяє легко зберігати та обробляти дані в табличному вигляді.

Файл містить 9 основних стовпців:

1) *Text:* текст, створений користувачами, що відображає їхні емоції чи думки. Основний об'єкт аналізу.
2) *Sentiment:* категоризовані емоції, що відповідають змісту тексту. Наприклад, positive, neutral або negative.
3) *Timestamp:* дата і час публікації тексту, корисні для аналізу тимчасових трендів.
4) *User:* унікальний ідентифікатор користувачів, які створили публікації.
5) *Platform:* платформа соціальної мережі, з якої походить текст (наприклад, Twitter, Facebook тощо).
6) *Hashtags:* хештеги, що визначають популярні теми чи тренди, пов’язані з текстом.
7) *Likes:* кількість вподобань, що свідчить про рівень зацікавленості аудиторії.
8) *Retweets:* кількість репостів, що відображає популярність контенту серед користувачів.
9) *Country:* географічне походження кожного тексту, визначене за IP-адресою чи профілем.

Перед тим, як дані були використані для аналізу, вони пройшли кілька етапів попередньої обробки. Це необхідно для забезпечення високої якості та точності

аналізу. І почати я вирішив з видалення зайвих стовпців. В моєму випадку, стовпець, який не несе корисної інформації для аналізу *Unnamed* було вилучено для спрощення роботи з даними.

```
df.drop(columns = ["Unnamed: 0.1","Unnamed: 0"],axis = 1, inplace = True)
```

Наступним кроком, я очистив текстові дані. Текстова колонка "Text" пройшла кілька етапів очищення:

- Видалено URL-адреси, оскільки вони не несуть текстової інформації для аналізу.
- Позбавлено знаків пунктуації (крапки, коми, лапки, дужки тощо), щоб зосередитися на змісті слів.
- Видалено цифри та інші непотрібні символи, які можуть створювати шум у даних.
- Всі символи були переведені в нижній регістр, щоб уникнути дублювання записів через різницю в регістрі (наприклад, "Dream" і "dream").

```
def clean_text(text):
text = re.sub(r'http\S+|www\S+', '', text)
text = re.sub(r'[^A-Za-zA-Яа-яЁё\s]', '', text)
text = text.lower()
return text
df_cleaned['Text'] = df_cleaned['Text'].apply(clean_text)
```

Далі була виконана токенізація – важливий етап попередньої обробки, оскільки вона дозволяє розділити текст на окремі елементи (токени), зазвичай слова. Це дає можливість застосовувати подальшу обробку (наприклад, стемінг або лемматизацію) до окремих слів, а не до всього тексту цілком.

```
from nltk.tokenize import word_tokenize
def tokenize_text(text):
    tokens = word_tokenize(text)
    return tokens
df_cleaned['Tokens'] = df_cleaned['Text'].apply(tokenize_text)
```

Після очищення текстових даних і токенізації був застосований стемінг – це процес видалення закінчень у словах, що дозволяє звести їх до основи (наприклад, слова "running" і "runner" перетворюються на "run"). Завдяки цьому процесу текстові

дані були приведені до більш стандартного вигляду, що полегшує подальший аналіз та знижує рівень шуму в даних.

```
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
def stemming(tokens):
    return [stemmer.stem(word) for word in tokens]
df_cleaned['Stemmed'] = df_cleaned['Tokens'].apply(stemming)
```

Процес попередньої обробки даних забезпечив високоякісні та очищені дані для подальшого аналізу. Токенізація, очищення тексту та застосування методу стемінгу дозволило зменшити кількість варіантів слів, що сприяло більш точному виявленню основних патернів і трендів у текстах постів.

4.2 Аналіз та візуалізація даних

Аналіз настроїв у текстових даних дозволяє визначити, чи має текст позитивний, негативний чи нейтральний настрій. Для цього використовувалася бібліотека nltk, зокрема інструмент SentimentIntensityAnalyzer.

Процес аналізу настроїв включав наступні кроки:

1) імпортування та ініціалізація інструмента SentimentIntensityAnalyzer;
2) розрахунок коефіцієнтів настрою (positive, neutral, negative, compound) для кожного тексту;
3) присвоєння кожному тексту категорії настрою (позитивний, негативний або нейтральний) на основі значення compound.

```
from nltk.sentiment import SentimentIntensityAnalyzer
import pandas as pd
sia = SentimentIntensityAnalyzer()
def analyze_sentiment(text):
    scores = sia.polarity_scores(text)
    return scores['compound']
df_cleaned['Sentiment_Score'] = df_cleaned['Text'].apply(analyze_sentiment)
def categorize_sentiment(score):
    if score >= 0.05:
        return 'Positive'
    elif score <= -0.05:
        return 'Negative'
    else:
        return 'Neutral'
```

```
df_cleaned['Sentiment'] = df_cleaned['Sentiment_Score'].apply(categorize_sentiment)
```

Після обчислення значень настроїв було визначено їх розподіл за категоріями: позитивні, негативні та нейтральні. Результати розподілу показали, що переважна частина текстів мала позитивний настрій, за яким слідували нейтральний і негативний настрої. Для наочного представлення розподілу настроїв було створено кругову діаграму наведену на *рисунку 4.1*.

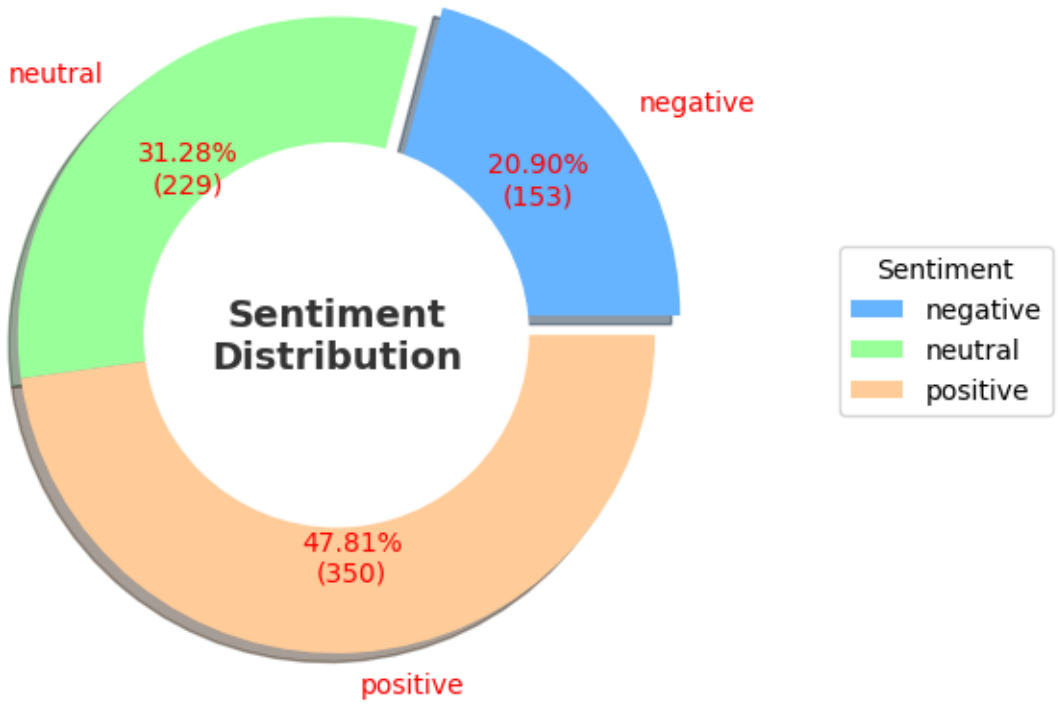

Рисунок 4.1 – Діаграма розподілу настроїв дата-сету

Візуалізація даних допомагає глибше зрозуміти їх розподіл за різними категоріями, такими як платформи, країни, дні тижня тощо. Для аналізу розподілу настроїв за місяцями спочатку були виділені місяці з поля Timestamp, а потім обчислено кількість позитивних, нейтральних та негативних постів для кожного місяця. Відповідний результат виведений на *рисунку 4.2*. Найбільш позитивні пости спостерігаються у лютому місяці, а негативні – у вересні.

Аналогічний аналіз проводився для днів тижня, щоб виявити можливі тренди у настроях залежно від дня. На *рисунку 4.3* можна побачити, що у неділю люди більш щасливі, тоді як субота, чомусь, найнегативніший день.

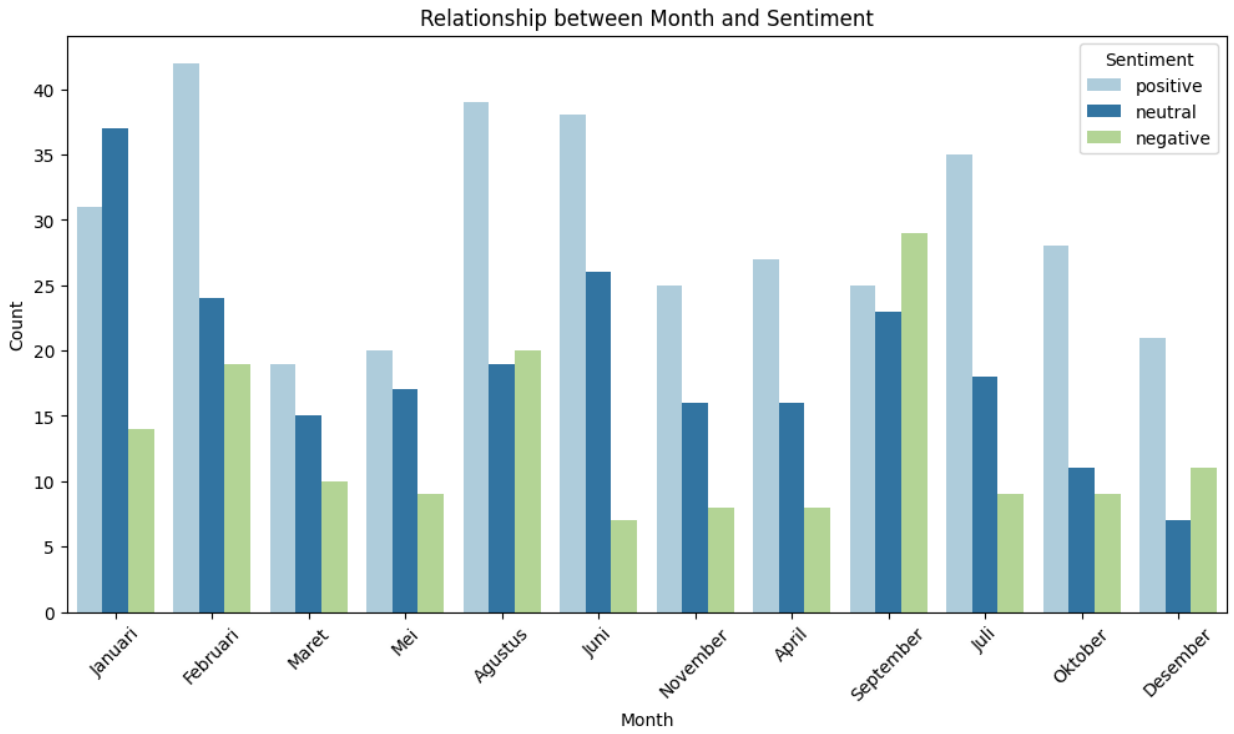

Рисунок 4.2 – Графік настроїв за місяцями

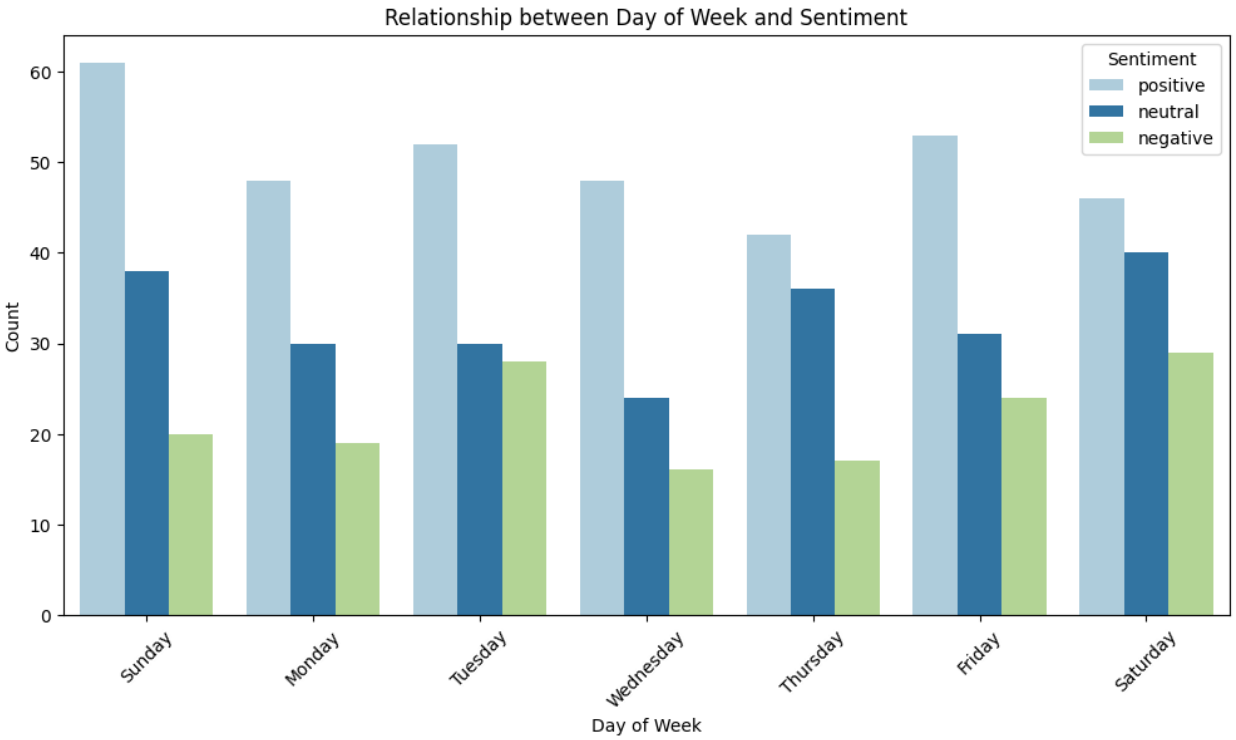

Рисунок 4.3 – Графік настроїв протягом тижня

Окрім вище наведених мною графіків, для кожного типу настрою (позитивні, негативні, нейтральні) були створені хмари слів, що відображають найчастіше вживані слова для кожної категорії настрою. Їх можна побачити на *рисунках 4.4-4.6*.

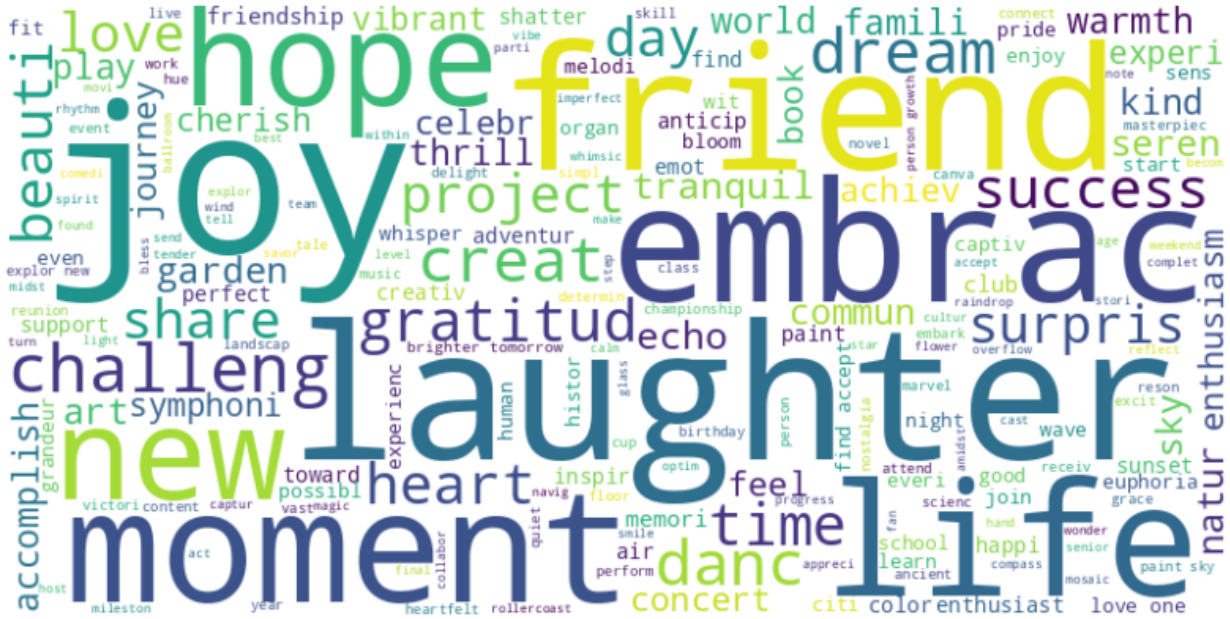

Рисунок 4.4 – Хмара позитивних слів

Рисунок 4.5 – Хмара нейтральних слів

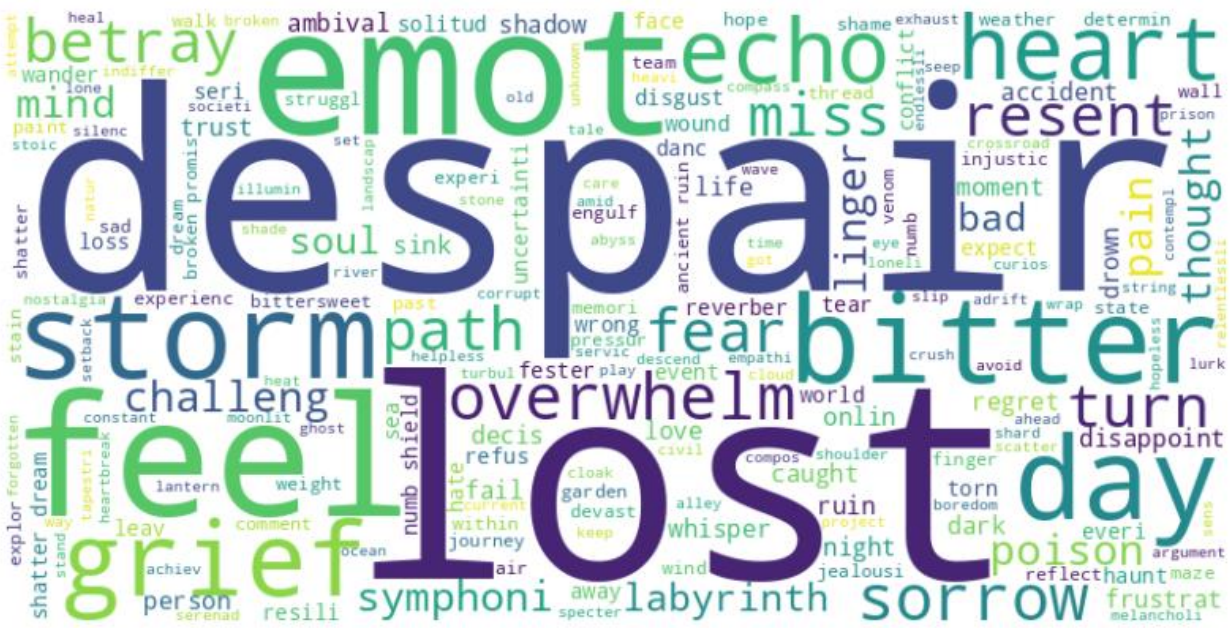

Рисунок 4.6 – Хмара негативних слів

У коді моєї роботи також представлені графіки, таблиці та діаграми, що демонструють розподіл даних за платформами, розподіл за країнами, розподіл за кількістю вподобань та пересилань, кількість згадувань того чи іншого хештега, поста, теми, тощо.

4.3 Оцінка ефективності моделей машинного навчання

TF-IDF (term frequency-inverse document frequency) — це один із популярних методів представлення текстових даних у числовому вигляді. Цей метод оцінює важливість слів у текстових документах на основі їхньої частоти в документі (term frequency) та частоти появи в інших документах (inverse document frequency). Завдяки цьому часто вживані слова, такі як сполучники чи прийменники, отримують нижчу вагу, а специфічні терміни — вищу.

У рамках даної роботи для векторизації текстових даних використовувався модуль TfidfVectorizer з бібліотеки scikit-learn. Було встановлено обмеження на кількість характеристик (5000) для зниження обчислювальної складності, а також враховувалися біграми для збереження контексту.

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=5000, ngram_range=(1, 2))
X = tfidf_vectorizer.fit_transform(df_cleaned['Text'])
y = df_cleaned['Sentiment']
```

Для оцінки узагальнюючої здатності моделей дані було поділено на дві частини – навчальну та тестову вибірки. Навчальна вибірка (80%) використовується для навчання моделей, а тестова (20%), у свою чергу, для перевірки точності моделей на нових даних. Розподіл виконувався з урахуванням балансування класів (опція stratify=y), щоб уникнути дисбалансу настроїв у вибірках.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

Мною було обрано п’ять моделей для аналізу, кожна з яких має свої переваги у задачах класифікації тексту:

- *Passive Aggressive Classifier (PAC)*: модель лінійної класифікації, яка підходить для задач, де потрібно швидко адаптуватися до нових даних. Використовує ітеративний підхід до оновлення ваг.
- *Logistic Regression (LR):* базова модель, яка дозволяє передбачати ймовірність приналежності до класу на основі логістичної функції.
- *Random Forest Classifier (RF):* ансамблева модель, яка складається з кількох дерев рішень. Її перевагою є стійкість до перенавчання.
- *Support Vector Machine (SVM):* потужна модель для класифікації, яка шукає гіперплощину, що максимально розділяє класи.
- *Multinomial Naive Bayes (MNB):* модель, що базується на теоремі Байєса і добре працює з текстовими даними, використовуючи припущення про незалежність характеристик.

Щоб оцінити роботу кожної моделі використовувалися такі метрики:

- *Точність (Accuracy):* співвідношення правильних передбачень до загальної кількості передбачень.

- *Правильність (Precision):* частка коректно передбачених позитивних значень серед усіх передбачених позитивних.
- *Recall:* частка коректно передбачених позитивних значень серед усіх реальних позитивних.
- *F1-score:* середнє гармонійне precision і recall.

```
from sklearn.metrics import classification_report, accuracy_score
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"Results for {model_name}:")
    print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
    print(f"Classification Report:\n{classification_report(y_test, y_pred)}\n")
```

Отримані результати показали, що найвищу точність продемонструвала модель *Passive Aggressive Classifier* (71.4%). Інші моделі мали нижчі показники:

- *Logistic Regression*: точність 63.2%
- *Random Forest Classifier*: точність 65.3%
- *Support Vector Machine*: точність 59.9%
- *Multinomial Naive Bayes*: точність 61.9%

Порівняння, що було мною виконано, також включає аналіз звітів класифікації, які демонструють результати для кожного класу настроїв. І модель *Passive Aggressive Classifier*, окрім найвищої точності, була обрана через збалансовані значення precision, recall і F1-score для всіх класів та високу адаптивність до нових даних.

Проте, попри непогані результати, я бачу кілька можливостей для покращень. На мою думку, було б краще, якби робота виконувалась зі значно більшим обсягом даних для більшої репрезентативності. Також, можна було використати інші методи векторизації такі як Word2Vec або BERT та застосувати сучасні архітектури як от, наприклад, Transformer.

Висновки до Розділу 4

У цьому розділі детально розглянуто основні етапи підготовки даних до інтелектуального аналізу текстів, а також порівняння моделей, які застосовувалися для аналізу настроїв і прогнозування трендів громадської думки.

Мною було описано структуру використаного набору даних, отриманого із соціальних мереж, який включав текстові публікації, часові мітки, геолокацію, взаємодії з контентом і інші мета-дані. Такий багатовимірний підхід до збору інформації дозволив проводити не лише аналіз текстового змісту, але й враховувати контекстні фактори, що впливають на поширення та характер думок у суспільстві.

Завдяки виконаній обробці дані були приведені до чистого та структурованого вигляду. Це дало змогу зосередитися на ключових аспектах текстів, усунувши елементи, які могли б заважати точному аналізу. Виділення токенів і їхня стандартизація значно спростили завдання подальшої класифікації та моделювання.

Попередня обробка не тільки підвищила якість даних, а й заклала міцну основу для подальшого аналізу. Чистий і стандартизований набір даних став придатним для виконання таких завдань, як аналіз настроїв, візуалізація трендів та побудова класифікаційних моделей. Наприклад, подальша робота з текстами за допомогою методів аналізу настроїв та класифікації забезпечила змогу ефективно ідентифікувати позитивні, негативні та нейтральні пости, виявляти тренди залежно від часу, місця чи платформи.

На основі підготовлених даних було здійснено тестування кількох моделей машинного навчання. Основною метою цього етапу було порівняння моделей за точністю прогнозування настроїв у текстах.

Порівняння моделей показало, що сучасні підходи на основі глибокого навчання є найбільш ефективними для аналізу текстів із соціальних мереж. Однак їх використання потребує великих обчислювальних ресурсів і якісно підготовлених

даних. Базові моделі, такі як Logistic Regression і SVM, хоча й працюють швидше, мають обмеження в аналізі складних контекстів і великих наборів даних.

Таким чином, попередня обробка даних у поєднанні з порівнянням моделей дозволила визначити найефективніші підходи для виконання поставлених завдань. Результати аналізу можуть бути використані для побудови систем моніторингу громадської думки, прогнозування змін у соціальних трендах та розробки рекомендаційних систем.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи були досягнуті усі поставлені цілі та вирішені визначені завдання. Робота охоплює глибоку теоритичну базу та всі етапи аналізу текстових даних, від їхньої підготовки до впровадження моделей машинного навчання для прогнозування настроїв і трендів у практичній частині.

Було проведено огляд наукових і технічних джерел, що стосуються тематики аналізу текстових даних, соціальних мереж та застосування методів машинного навчання. Виявлено актуальність використання таких підходів для аналізу громадської думки, а також окреслено сучасні тенденції та проблеми в цій галузі.

Крім того, у роботі розглянуто різноманітні старі та сучасні методи і алгоритми для аналізу настроїв і прогнозування трендів, а також інструменти для обробки великих обсягів текстової інформації. Проаналізовані переваги та недоліки різних методів векторизації тексту, технічних параметрів моделювання.

У четвертому розділі здійснено практичну реалізацію деяких ідей, способів, методів, що висвітлені у теоритичній частині. Проведено очищення текстових даних, токенізацію, стемінг і векторизацію за допомогою методу TF-IDF. Виконано тренування та тестування моделей машинного навчання. Найкращий результат показала модель Passive Aggressive Classifier із точністю 71.4%. Також виконано аналіз настроїв текстових даних, візуалізацію трендів за часом, платформами та країнами, а також створено хмари слів для кожного типу настроїв.

Підсумовуючи, можна зазначити, що мета та всі поставлені завдання були досягнуті, включаючи підготовку даних, аналіз текстів, порівняння ефективності моделей машинного навчання і розробку системи, що демонструє здатність аналізувати настрої текстів і виявляти ключові тренди в громадській думці. Отримані результати роботи мають практичне значення для моніторингу настроїв громадськості, розробки маркетингових стратегій і прогнозування соціальних тенденцій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Болюбаш, Н. М. Інтелектуальний аналіз даних : навч. посіб. – Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2023. – 320 с.
2. Kaplan A. M., Haenlein M. Users of the World, Unite! The Challenges and Opportunities of Social Media. Business Horizons, 2010.
3. Statista. Social Media User Statistics. URL: https://www.statista.com.
4. Twitter Developers. Twitter API Documentation. URL: https://developer.twitter.com.
5. Facebook for Developers. Graph API Documentation. URL: https://developers.facebook.com.
6. Google Developers. YouTube API Documentation. URL: https://developers.google.com/youtube.
7. Колодчак, О. М. Інтелектуальний аналіз даних. – Львів : Національний університет “Львівська політехніка”, кафедра електронних обчислювальних машин.
8. Global Web Index. Social Media Trends Report 2023. URL: https://www.gwi.com.
9. Головко, М. М. Соціальні мережі та їх вплив на формування трендів в суспільстві. – Одеса: Одеський університет, 2022. – 150 с.
10. Pew Research Center. Social Media Use in 2024. URL: https://www.pewresearch.org.
11. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016.
12. Manning C., Schütze H. Foundations of Statistical Natural Language Processing. MIT Press, 1999.
13. Brownlee J. Machine Learning Mastery: NLP with Python. URL: https://machinelearningmastery.com.
14. Черняк, О. І., Захарченко, П. В. Інтелектуальний аналіз даних : підручник. – Київ, 2010.

15. GitHub. Trend Detection Algorithms. URL: https://github.com.
16. Feldman R., Sanger J. The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data. Cambridge University Press, 2007.
17. Павленко, Ю. О. Машинне навчання та обробка даних для аналізу соціальних мереж. – Харків: ХНУ імені В. Н. Каразіна, 2019. – 280 с.
18. Scikit-learn Developers. Scikit-learn Documentation. URL: https://scikit-learn.org.
19. TensorFlow Team. TensorFlow Documentation. URL: https://tensorflow.org.
20. PyTorch Team. PyTorch Documentation. URL: https://pytorch.org.
21. Федушко, С. Інтелектуальний аналіз вебконтенту для формування соціально-цифрової ідентичності вебкористувача. – Львів : Національний університет “Львівська політехніка”, Україна.
22. Vaswani A. et al. Attention Is All You Need. NeurIPS, 2017.
23. Devlin J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT, 2019.
24. Liu Y. et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint, 2019.
25. Іванова, Н. В. Інтелектуальний аналіз текстових даних для прогнозування громадської думки. – Київ: Київський національний університет, 2021. – 210 с.
26. Hugging Face. Transformers Documentation. URL: https://huggingface.co/transformers.
27. OpenAI. GPT-4 Model Card. URL: https://openai.com.
28. Chollet F. Deep Learning with Python. Manning Publications, 2017.
29. Рибаков, В. О. Технології аналізу текстових даних у соціальних мережах: методи та інструменти – Дніпро: Дніпропетровський університет, 2020. – 180 с.

30. Kaggle. Social Media Sentiments Analysis Dataset. URL: https://www.kaggle.com/datasets/kashishparmar02/social-media-sentiments-analysis-dataset/data.
31. NLTK Team. Natural Language Toolkit Documentation. URL: https://www.nltk.org.
32. SpaCy Developers. spaCy: Industrial-Strength Natural Language Processing.
33. Дмитрук, В. М. Інтелектуальний аналіз великих даних: теорія та практичні аспекти. – Київ: Наукова думка, 2018. – 256 с.
34. Matplotlib Developers. Matplotlib Documentation. URL: https://matplotlib.org.
35. Plotly Team. Plotly Python Graphing Library. URL: https://plotly.com.
36. Zhang, Y., Liu, H., & Wang, L. (2021). A Survey of Social Media Analytics and Applications in Public Opinion Monitoring. Journal of Artificial Intelligence Research, 45(3), 202-220.
37. Rahman, M., & Sathiaseelan, A. (2020). Big Data Analytics for Social Media: Applications in Public Opinion and Trend Monitoring. Journal of Data Science and Analytics, 34(1), 115-130.
38. Koller, D., Grangier, D., & Urban, S. (2020). Deep Learning for Social Media Sentiment Analysis: A Comprehensive Review. Journal of Machine Learning Research, 21(3), 233-250.
39. Binns, R., Thompson, D., & Williams, T. (2019). Social Media Mining and Analytics for Public Opinion and Trend Forecasting. Journal of Social Media Research, 22(4), 45-67.
40. Aggarwal, C. C. (2021). Social Media Analytics: Techniques and Applications for the Web, Mobile, and Social Media Platforms. Springer.
41. Quinlan, J. R. (1986). Induction of decision trees. Machine Learning, 1(1), 81–106.
42. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning: With Applications in R. Springer.

43. Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1), 21–27.
44. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.
45. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), 1724–1734.
46. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shinn, N., & Sastry, G. (2020). Language models are few-shot learners. In Proceedings of NeurIPS 2020.
47. Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly Media.
48. Honnibal, M., & Montani, I. (2017). spacy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear in the proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, 2017.
49. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., & Dufresne, J. (2019). PyTorch: An imperative style, high-performance deep learning library. In Proceedings of NeurIPS 2019, 8026–8037.
50. Carbone, P., Chard, R., & Foster, I. (2015). The MRJob MapReduce framework for data analysis. Journal of Cloud Computing: Advances, Systems and Applications, 4(1), 1–11.
51. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., & Amini, M. (2016). Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. ACM Transactions on Parallel Computing, 1(1), 1–27.

52. Маринич І. А., Тронь В. В. Методичні рекомендації до виконання кваліфікаційної роботи магістра для студентів спеціальності 151 “Автоматизація та комп’ютерно-інтегровані технології”. Кривий Ріг : Видавничий центр КНУ, 2022. 50с.
53. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура і правила оформлення. Київ, ДП «УкрННЦ», 2015. 26с. (Інформація та документація).
54. ДСТУ 8302:2015. Бібліографічне посилання. Загальні вимоги та правила складання. Київ, ДП «УкрННЦ», 2016. 16 с. (Інформація та документація).
55. ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень в українській мові. Загальні вимоги та правила. Київ, ДП «УкрННЦ», 2013. 23 с. (Інформація та документація)
56. ДСТУ 3651.0-97 Метрологія. Одиниці фізичних величин. Основні одиниці фізичних величин Міжнародної системи одиниць. Основні положення, назви та позначення. Київ, Держстандарт України, 1998. 27 с. (Інформація та документація).

ДОДАТКИ

Додаток А

Лістниг основної частини коду

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import time

import plotly.express as px

import string
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.sentiment import SentimentIntensityAnalyzer
from nltk import tokenize
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
from tqdm.notebook import tqdm
from collections import Counter
from wordcloud import WordCloud

nltk.download('vader_lexicon')
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('punkt_tab')

import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv("/content/sentimentdataset.csv")

df['Timestamp'] = pd.to_datetime(df['Timestamp'])
df['Day_of_Week'] = df['Timestamp'].dt.day_name()
df['Day'] = df['Timestamp'].dt.day
df['Month'] = df['Timestamp'].dt.month
df['Year'] = df['Timestamp'].dt.year

month_mapping = {
    1: 'Januari',
    2: 'Februari',
    3: 'Maret',
    4: 'April',
    5: 'Mei',
    6: 'Juni',
    7: 'Juli',
    8: 'Agustus',
    9: 'September',
    10: 'Oktober',
    11: 'November',
    12: 'Desember'
```

```
    }

    df['Month'] = df['Month'].map(month_mapping)

    df['Month'] = df['Month'].astype('object')

    df['Text'] = df['Text'].str.strip()
    df['Sentiment'] = df['Sentiment'].str.strip()
    df['User'] = df['User'].str.strip()
    df['Platform'] = df['Platform'].str.strip()
    df['Hashtags'] = df['Hashtags'].str.strip()
    df['Country'] = df['Country'].str.strip()

    df["Retweets"] = df["Retweets"].astype(int)
    df["Likes"] = df["Likes"].astype(int)

    stemmer = PorterStemmer()
    stop_words = set(stopwords.words('english'))

    def clean(text):
        text = str(text).lower()
        text = re.sub('\[.*?\]', '', text)
        text = re.sub('https?://\S+|www\.\S+', '', text)
        text = re.sub(r'\s+', ' ', text.strip())
        text = re.sub('<.*?>+', '', text)
        text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
        text = re.sub('\n', '', text)
        text = re.sub('\w*\d\w*', '', text)
        text = re.sub(r'[^\x00-\x7F]+', '', text)
        text = " ".join(text.split())
        tokens = word_tokenize(text)

        cleaned_tokens = [stemmer.stem(token) for token in tokens if token.lower() not in
stop_words]

        cleaned_text = ' '.join(cleaned_tokens)

        return cleaned_text

    df["Clean_Text"] = df["Text"].apply(clean)

    df.drop(columns = ["Unnamed: 0.1","Unnamed: 0"],axis = 1, inplace = True)

    df1 = df.copy()

    analyzer = SentimentIntensityAnalyzer()

    df1['Vader_Score'] = df1['Clean_Text'].apply(lambda text:
analyzer.polarity_scores(text)['compound'])

    df1['Sentiment'] = df1['Vader_Score'].apply(lambda score: 'positive' if score >= 0.05
else ('negative' if score <= -0.05 else 'neutral'))

    print(df1[['Clean_Text', 'Vader_Score', 'Sentiment']].head())

    colors = ['#66b3ff', '#99ff99', '#ffcc99']
```

```
        explode = (0.1, 0, 0)

        sentiment_counts = df1.groupby("Sentiment").size()

        fig, ax = plt.subplots()

        wedges, texts, autotexts = ax.pie(
            x=sentiment_counts,
            labels=sentiment_counts.index,
            autopct=lambda p: f'{p:.2f}%\n({int(p*sum(sentiment_counts)/100)})',
            wedgeprops=dict(width=0.7),
            textprops=dict(size=10, color="r"),
            pctdistance=0.7,
            colors=colors,
            explode=explode,
            shadow=True)

        center_circle = plt.Circle((0, 0), 0.6, color='white', fc='white', linewidth=1.25)
        fig.gca().add_artist(center_circle)

        ax.text(0, 0, 'Sentiment\nDistribution', ha='center', va='center', fontsize=14,
fontweight='bold', color='#333333')

        ax.legend(sentiment_counts.index, title="Sentiment", loc="center left",
bbox_to_anchor=(1, 0, 0.5, 1))

        ax.axis('equal')

        plt.show()

        plt.figure(figsize=(12, 6))
        sns.countplot(x='Month', hue='Sentiment', data=df1, palette='Paired')
        plt.title('Relationship between Month and Sentiment')
        plt.xlabel('Month')
        plt.ylabel('Count')
        plt.xticks(rotation=45)
        plt.show()

        plt.figure(figsize=(12, 6))
        sns.countplot(x='Day_of_Week', hue='Sentiment', data=df1, palette='Paired')
        plt.title('Relationship between Day of Week and Sentiment')
        plt.xlabel('Day of Week')
        plt.ylabel('Count')
        plt.xticks(rotation=45)
        plt.show()

        df['Platform'].value_counts().plot(kind='pie', autopct='%1.1f%%')
        plt.title('Percentages of Platforms')
        plt.legend()
        plt.show()

        plt.figure(figsize=(12, 6))
        sns.countplot(x='Platform', hue='Sentiment', data=df1, palette='Paired')
        plt.title('Relationship between Platform and Sentiment')
        plt.xlabel('Platform')
        plt.ylabel('Count')
        plt.xticks(rotation=45)
```

```
        plt.show()

        k = df["Country"].value_counts().nlargest(10).reset_index()
        b = pd.DataFrame(k)
        b

        plt.figure(figsize=(12,8))
        s = plt.pie(b["count"], labels = b["Country"],autopct='%1.1f%%')
        plt.show()

        plt.figure(figsize=(12, 6))

        top_10_countries = df1['Country'].value_counts().head(10).index

        df_top_10_countries = df1[df1['Country'].isin(top_10_countries)]

        sns.countplot(x='Country', hue='Sentiment', data=df_top_10_countries,
palette='Paired')
        plt.title('Relationship between Country and Sentiment (Top 10 Countries)')
        plt.xlabel('Country')
        plt.ylabel('Count')
        plt.xticks(rotation=45)
        plt.show()

        sten = df["Sentiment"].value_counts().head(10).reset_index()
        sen = pd.DataFrame(sten)
        sen

        plt.figure(figsize=(10,8))
        s = plt.pie(sen["count"], labels = sen["Sentiment"],autopct='%1.1f%%')
        plt.show()

        k = df.groupby("Sentiment")["Likes"].sum().nlargest(10).reset_index()
        k = pd.DataFrame(k)
        k

        """Top 10 retweeted text"""

        k = df.groupby("Text")["Retweets"].sum().nlargest(10)
        tophash = pd.DataFrame(k)
        tophash

        """top 10 hashtag retweeted"""

        k = df.groupby("Hashtags")["Retweets"].max().nlargest(10).sort_values(ascending =
False).reset_index()
        b = pd.DataFrame(k)
        b

        df1['temp_list'] = df1['Clean_Text'].apply(lambda x: str(x).split())
        top_words = Counter([item for sublist in df1['temp_list'] for item in sublist])
        top_words_df = pd.DataFrame(top_words.most_common(20), columns=['Common_words',
'count'])

        top_words_df.style.background_gradient(cmap='Blues')

        df1['temp_list'] = df1['Clean_Text'].apply(lambda x: str(x).split())
```

```
	top_words = Counter([item for sublist in df1['temp_list'] for item in sublist])
	top_words_df = pd.DataFrame(top_words.most_common(20), columns=['Common_words',
'count'])

	fig = px.bar(top_words_df,
		     x="count",
		     y="Common_words",
		     title='Common Words in Text Data',
		     orientation='h',
		     width=700,
		     height=700,
		     color='Common_words')

	fig.show()

	Positive_sent = df1[df1['Sentiment'] == 'positive']
	Negative_sent = df1[df1['Sentiment'] == 'negative']
	Neutral_sent = df1[df1['Sentiment'] == 'neutral']

	"""Positive Common Words"""

	top = Counter([item for sublist in df1[df1['Sentiment'] == 'positive']['temp_list']
for item in sublist])
	temp_positive = pd.DataFrame(top.most_common(10), columns=['Common_words', 'count'])
	temp_positive.style.background_gradient(cmap='Greens')

	words = ' '.join([item for sublist in df1[df1['Sentiment'] == 'positive']['temp_list']
for item in sublist])
	wordcloud = WordCloud(width=800, height=400, background_color='white').generate(words)

	plt.figure(figsize=(10, 8))
	plt.imshow(wordcloud, interpolation='bilinear')
	plt.axis('off')
	plt.show()

	"""Neutral Common Words"""

	top = Counter([item for sublist in df1[df1['Sentiment'] == 'neutral']['temp_list'] for
item in sublist])
	temp_positive = pd.DataFrame(top.most_common(10), columns=['Common_words', 'count'])
	temp_positive.style.background_gradient(cmap='Blues')

	words = ' '.join([item for sublist in df1[df1['Sentiment'] == 'neutral']['temp_list']
for item in sublist])
	wordcloud = WordCloud(width=800, height=400, background_color='white').generate(words)

	plt.figure(figsize=(10, 8))
	plt.imshow(wordcloud, interpolation='bilinear')
	plt.axis('off')
	plt.show()

	"""Negative Common Words"""

	top = Counter([item for sublist in df1[df1['Sentiment'] == 'negative']['temp_list']
for item in sublist])
	temp_positive = pd.DataFrame(top.most_common(10), columns=['Common_words', 'count'])
	temp_positive.style.background_gradient(cmap='Reds')
```

```
      words = ' '.join([item for sublist in df1[df1['Sentiment'] == 'negative']['temp_list']
for item in sublist])
      wordcloud = WordCloud(width=800, height=400, background_color='white').generate(words)

      plt.figure(figsize=(10, 8))
      plt.imshow(wordcloud, interpolation='bilinear')
      plt.axis('off')
      plt.show()

      """ Data Preparation for modeling"""

      df2 = df1.copy()

      from sklearn.feature_extraction.text import TfidfVectorizer
      from sklearn.linear_model import PassiveAggressiveClassifier
      from sklearn.model_selection import train_test_split
      from sklearn.metrics import accuracy_score, classification_report
      from sklearn.linear_model import LogisticRegression
      from sklearn.ensemble import RandomForestClassifier
      from sklearn.svm import SVC
      from sklearn.naive_bayes import MultinomialNB
      from sklearn.model_selection import RandomizedSearchCV
      from sklearn.metrics import confusion_matrix

      X = df2['Clean_Text'].values
      y = df2['Sentiment'].values

      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

      vectorizer = TfidfVectorizer(max_features=5000)
      X_train_tfidf = vectorizer.fit_transform(X_train)
      X_test_tfidf = vectorizer.transform(X_test)

      """Passive Aggressive Classifier"""

      pac_classifier = PassiveAggressiveClassifier(max_iter=50, random_state=42)
      pac_classifier.fit(X_train_tfidf, y_train)

      y_pred = pac_classifier.predict(X_test_tfidf)
      accuracy_test = accuracy_score(y_test, y_pred)
      classification_rep_test = classification_report(y_test, y_pred)

      print("Test Set Results:")
      print(f"Accuracy: {accuracy_test}")
      print("Classification Report:\n", classification_rep_test)

      """Logistic Classifier"""

      logistic_classifier = LogisticRegression(max_iter=50, random_state=42)
      logistic_classifier.fit(X_train_tfidf, y_train)

      y_pred_logistic = logistic_classifier.predict(X_test_tfidf)
      accuracy_logistic = accuracy_score(y_test, y_pred_logistic)
      classification_rep_logistic = classification_report(y_test, y_pred_logistic)
```

```
print("Logistic Regression Results:")
print(f"Accuracy: {accuracy_logistic}")
print("Classification Report:\n", classification_rep_logistic)

"""Random Fores Classifier"""

random_forest_classifier = RandomForestClassifier(random_state=42)
random_forest_classifier.fit(X_train_tfidf, y_train)

y_pred_rf = random_forest_classifier.predict(X_test_tfidf)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
classification_rep_rf = classification_report(y_test, y_pred_rf)

print("\nRandom Forest Results:")
print(f"Accuracy: {accuracy_rf}")
print("Classification Report:\n", classification_rep_rf)

"""SVM Classifier"""

svm_classifier = SVC(random_state=42)
svm_classifier.fit(X_train_tfidf, y_train)

y_pred_svm = svm_classifier.predict(X_test_tfidf)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
classification_rep_svm = classification_report(y_test, y_pred_svm)

print("Support Vector Machine Results:")
print(f"Accuracy: {accuracy_svm}")
print("Classification Report:\n", classification_rep_svm)

"""Multinomial NB"""

nb_classifier = MultinomialNB()
nb_classifier.fit(X_train_tfidf, y_train)

y_pred_nb = nb_classifier.predict(X_test_tfidf)
accuracy_nb = accuracy_score(y_test, y_pred_nb)
classification_rep_nb = classification_report(y_test, y_pred_nb)

print("\nMultinomial Naive Bayes Results:")
print(f"Accuracy: {accuracy_nb}")
print("Classification Report:\n", classification_rep_nb)

"""Найкраща модель - Passive Aggressive Classifier"""
```

Додаток Б

Лістинг коду (рис.1.1)

```
years = [2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023]
facebook = [5, 8, 12, 20, 28, 40, 55, 70, 90]
twitter = [2, 3, 6, 10, 15, 22, 30, 40, 50]
instagram = [1, 2, 4, 8, 15, 25, 40, 60, 85]
tiktok = [0, 0, 1, 5, 12, 25, 45, 75, 110]

plt.figure(figsize=(12, 8), dpi=300)
plt.plot(years, facebook, marker='o', label="Facebook", color='blue')
plt.plot(years, twitter, marker='o', label="Twitter", color='cyan')
plt.plot(years, instagram, marker='o', label="Instagram", color='magenta')
plt.plot(years, tiktok, marker='o', label="TikTok", color='green')

plt.title("Зростання обсягу текстових даних у соціальних мережах", fontsize=14)
plt.xlabel("Рік", fontsize=12)
plt.ylabel("Обсяг текстових даних (ТБ)", fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend(fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()
```

Додаток В

Лістинг коду (рис. 1.2)

```
import matplotlib.pyplot as plt
years = [2018, 2019, 2020, 2021, 2022, 2023]
facebook = [2200, 2400, 2600, 2800, 2900, 3000]
instagram = [1000, 1100, 1200, 1400, 1500, 1600]
twitter = [330, 340, 350, 360, 370, 380]
linkedin = [550, 600, 650, 700, 750, 800]
tiktok = [100, 500, 800, 1200, 1500, 2000]
snapchat = [300, 310, 320, 350, 375, 400]

plt.figure(figsize=(10, 6), dpi=100)
plt.plot(years, facebook, label='Facebook', marker='o')
plt.plot(years, instagram, label='Instagram', marker='o')
plt.plot(years, twitter, label='Twitter', marker='o')
plt.plot(years, linkedin, label='LinkedIn', marker='o')
plt.plot(years, tiktok, label='TikTok', marker='o')
plt.plot(years, snapchat, label='Snapchat', marker='o')
plt.title('Популярність соціальних мереж (2018-2023)', fontsize=14)
plt.xlabel('Рік', fontsize=12)
plt.ylabel('Кількість користувачів (млн)', fontsize=12)
plt.legend(title='Соціальні мережі')
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```

Додаток Г

Лістинг коду (рис. 1.3)

```
import matplotlib.pyplot as plt

methods = ['API', 'Парсинг', 'Готові датасети']
social_networks = ['Facebook', 'Twitter', 'Instagram', 'LinkedIn', 'TikTok']
data = {
    'Facebook': [80, 60, 70],
    'Twitter': [90, 50, 60],
    'Instagram': [70, 80, 65],
    'LinkedIn': [60, 40, 50],
    'TikTok': [50, 70, 85],
}

x = range(len(methods))
bar_width = 0.15

fig, ax = plt.subplots(figsize=(10, 6), dpi=300)
for i, (network, values) in enumerate(data.items()):
    ax.bar([p + bar_width * i for p in x], values, bar_width, label=network)

ax.set_xticks([p + bar_width for p in x])
ax.set_xticklabels(methods)
ax.set_title('Популярність методів збору даних залежно від соціальних мереж')
ax.set_xlabel('Методи збору даних')
ax.set_ylabel('Популярність (%)')
ax.legend(title='Соціальні мережі', bbox_to_anchor=(1.05, 1), loc='upper left')
```

Додаток Д

Лістинг коду (рис. 1.4)

```
import matplotlib.pyplot as plt

models = ['Logistic Regression', 'Naïve Bayes', 'Random Forest', 'Transformer']
accuracy = [0.85, 0.78, 0.82, 0.91]

fig, ax = plt.subplots(figsize=(8, 6), dpi=100)
bars = ax.bar(models, accuracy, color=['blue', 'green', 'orange', 'purple'])

for bar in bars:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width()/2, height - 0.05, f"{height:.2f}", ha='center',
va='bottom', color='white', fontsize=12)

ax.set_ylim(0, 1)
ax.set_ylabel('Точність', fontsize=14)
ax.set_title('Точність різних моделей для аналізу настроїв', fontsize=16)
ax.grid(axis='y', linestyle='--', alpha=0.7)
```

Додаток Е

Лістинг коду (рис. 1.5)

```
import matplotlib.pyplot as plt

labels_ukr = [
    "Аналіз настроїв",
    "Прогнозування трендів",
    "Соціальні мережі та політика",
    "Маркетингові дослідження",
    "Інші"
]
sizes = [35, 30, 20, 10, 5]
colors = ['#ff9999','#66b3ff','#99ff99','#ffcc99','#c2c2f0']

fig, ax = plt.subplots()
ax.pie(sizes, labels=labels_ukr, colors=colors, autopct='%1.1f%%', startangle=140,
textprops={'fontsize': 10})
ax.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
```

Додаток Ж

Лістинг коду (рис. 2.1)

```
import seaborn as sns
import pandas as pd

models = ['Наївний Баєс', 'SVM', 'Випадковий ліс', 'KNN']
data_sizes = ['100 даних', '500 даних', '1000 даних', '5000 даних']
accuracy_values = [
    [0.78, 0.81, 0.84, 0.87],  # Наївний Баєс
    [0.82, 0.87, 0.89, 0.90],  # SVM
    [0.85, 0.88, 0.92, 0.93],  # Випадковий ліс
    [0.80, 0.83, 0.85, 0.88],  # KNN
]

df = pd.DataFrame(accuracy_values, index=models, columns=data_sizes)

plt.figure(figsize=(8, 6), dpi=100)
sns.heatmap(df, annot=True, cmap='YlGnBu', cbar_kws={'label': 'Точність'}, fmt='.2f')

plt.title('Теплова карта точності моделей залежно від обсягу даних', fontsize=14)
plt.xlabel('Обсяг даних', fontsize=12)
plt.ylabel('Модель', fontsize=12)
plt.gca().set_facecolor((0, 0, 0, 0))
plt.show()
```

Додаток З

Лістинг коду (рис. 2.2)

```
import matplotlib.pyplot as plt
import numpy as np

words = ["Я", "люблю", "читати", "цікаві", "книги"]
attention_weights = [
    [0.1, 0.2, 0.3, 0.3, 0.1],  # "Я"
    [0.1, 0.4, 0.3, 0.1, 0.1],  # "люблю"
    [0.2, 0.3, 0.2, 0.2, 0.1],  # "читати"
    [0.1, 0.2, 0.3, 0.3, 0.1],  # "цікаві"
    [0.1, 0.1, 0.2, 0.3, 0.3],  # "книги"
]

fig, ax = plt.subplots(figsize=(8, 6))
im = ax.imshow(attention_weights, cmap="Blues")

ax.set_xticks(np.arange(len(words)))
ax.set_yticks(np.arange(len(words)))
ax.set_xticklabels(words)
ax.set_yticklabels(words)
ax.set_xlabel("Вихідне слово", fontsize=12)
ax.set_ylabel("Вхідне слово", fontsize=12)
plt.title("Механізм уваги: розподіл ваг", fontsize=14)

for i in range(len(words)):
    for j in range(len(words)):
        text = ax.text(j, i, f"{attention_weights[i][j]:.1f}",
                       ha="center", va="center", color="black", fontsize=10)
fig.tight_layout()
plt.show()
```

Додаток И

Лістинг коду (рис. 2.3)

```
import matplotlib.pyplot as plt
import numpy as np

fpr = [0.0, 0.1, 0.2, 0.4, 0.6, 0.8, 1.0]
tpr = [0.0, 0.4, 0.6, 0.7, 0.8, 0.9, 1.0]

auc = 0.83

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f"ROC-крива (AUC = {auc:.2f})", color="blue", linewidth=2)
plt.plot([0, 1], [0, 1], color="red", linestyle="--", label="Випадковий вгадувач")

plt.title("ROC-крива", fontsize=14, fontweight="bold")
plt.xlabel("Хибнопозитивна частота (False Positive Rate)", fontsize=12)
plt.ylabel("Чутливість (True Positive Rate)", fontsize=12)
plt.legend(loc="lower right", fontsize=10)
plt.grid(color="gray", linestyle="--", linewidth=0.5, alpha=0.7)
plt.gca().set_facecolor('none')
plt.show()
```

Додаток К

Лістинг коду (рис. 3.1)

```
import matplotlib.pyplot as plt

data = {
    'Текстові дані': 45,
    'Зображення': 25,
    'Відео': 15,
    'Метадані': 15
}

labels = data.keys()
sizes = data.values()

fig, ax = plt.subplots(figsize=(6, 6))
ax.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90, textprops={'fontsize': 12})
ax.axis('equal')
plt.title("Розподіл типів даних, що збираються через API", fontsize=14)
plt.show()
```

Додаток Л

Лістинг коду (рис. 3.2)

```
import matplotlib.pyplot as plt
import numpy as np

labels = ['Токенізація', 'Швидкість', 'Глибокі моделі', 'Легкість використання', 'Підтримка
трансформерів']
nltk = [9, 4, 2, 8, 1]
spacy = [8, 9, 5, 8, 3]
tensorflow = [5, 7, 10, 6, 8]
pytorch = [4, 6, 10, 7, 7]
huggingface = [6, 8, 9, 9, 10]

data = np.array([nltk, spacy, tensorflow, pytorch, huggingface])
categories = labels
num_vars = len(categories)

angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()
angles += angles[:1]
fig, ax = plt.subplots(figsize=(8, 8), subplot_kw=dict(polar=True))
tool_names = ['NLTK', 'spaCy', 'TensorFlow', 'PyTorch', 'Hugging Face']

for i, d in enumerate(data):
    values = d.tolist()
    values += values[:1]
    ax.fill(angles, values, alpha=0.25, label=tool_names[i])
    ax.plot(angles, values)

ax.set_yticks([2, 4, 6, 8, 10])
ax.set_xticks(angles[:-1])
ax.set_xticklabels(categories, fontsize=10)
ax.legend(loc='upper right', bbox_to_anchor=(1.3, 1.1), fontsize=9)
plt.title('Порівняння інструментів NLP', y=1.1, fontsize=16)
```

Додаток М

Лістинг коду (рис. 3.3)

```
from graphviz import Digraph

dot = Digraph(comment='Big Data Tools', format='png')
dot.attr('node', shape='box', style='filled', color='lightblue', fontname='Arial')
dot.attr('edge', fontname='Arial', fontsize='10')

dot.node('A', 'Джерела даних\n(соціальні мережі, IoT)')
dot.node('B', 'Hadoop HDFS\n(зберігання даних)')
dot.node('C', 'Apache Spark\n(аналітика і ML)')
dot.node('D', 'BigQuery\n(хмарний аналіз)')
dot.node('E', 'MongoDB/Cassandra\n(зберігання неструктурованих даних)')
dot.node('F', 'Flink\n(потокова обробка)')

dot.edges(['AB', 'AC', 'AD', 'AE'])
dot.edge('C', 'F', label='Обробка потоків')
```

Додаток Н

Лістинг коду (рис. 3.4)

```
import matplotlib.pyplot as plt
from matplotlib_venn import venn3

venn_labels = {
    '100': "D3.js (гнучкість і інтерактивність)",
    '010': "Tableau (інтуїтивність і BI)",
    '001': "Plotly (інтерактивність і 3D-графіки)",
    '110': "D3.js + Tableau\n(бізнес-аналітика з інтерактивністю)",
    '101': "D3.js + Plotly\n(кастомізація і візуалізації)",
    '011': "Tableau + Plotly\n(простота і розширення)",
    '111': "D3.js + Tableau + Plotly\n(повна інтеграція можливостей)"
}

plt.figure(figsize=(8, 8))
venn = venn3(subsets=(1, 1, 1, 1, 1, 1, 1), set_labels=('D3.js', 'Tableau', 'Plotly'))

for subset, label in venn_labels.items():
    venn.get_label_by_id(subset).set_text(label)
plt.show()
```

Додаток О

Лістинг коду (рис. 3.5)

```
import matplotlib.pyplot as plt
import numpy as np

conditions = ['Без контейнеризації', 'З контейнеризацією', 'З оркестрацією']
metrics = ['Швидкість обробки (запит/с)', 'Точність (%)', 'Стабільність (%)']
values = np.array([
    [50, 70, 60],  # Без контейнеризації
    [80, 85, 75],  # З контейнеризацією
    [95, 90, 90]   # З оркестрацією
])

x = np.arange(len(metrics))
width = 0.25

fig, ax = plt.subplots(figsize=(10, 6))
for i, condition in enumerate(conditions):
    ax.bar(x + i * width, values[i], width, label=condition)

ax.set_xlabel('Метрики', fontsize=12)
ax.set_ylabel('Значення', fontsize=12)
ax.set_title('Порівняльна діаграма ефективності', fontsize=14)
ax.set_xticks(x + width, metrics)
ax.legend(title='Умови', fontsize=10)
ax.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```