Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу
Факультет інформаційних технологій
Кафедра комп'ютерних систем і мереж

Семків Юлія Богданівна

УДК 004.85

МАГІСТЕРСЬКА РОБОТА

Розробка та дослідження системи виявлення дезінформації на основі методів машинного навчання

Комп'ютерна інженерія
(назва освітньої програми)

123 – Комп'ютерна інженерія
(шифр і назва спеціальності)

Здобувач освітнього ступеня *Семків Ю.Б.*
(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник *Кропивницька Віталія Богданівна, к.т.н., доцент*
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту
Завідувач кафедри

д-р.т.н., проф /С. І. Мельничук/
(посада) (підпис) (дата) (ініціали та прізвище)

Рецензент

доцент / Мойсеєнко О. В./
(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2025 рік

Івано-Франківський національний технічний університет нафти і газу

Факультет *Інформаційних технологій*

Кафедра *Комп'ютерних систем і мереж*

Освітньо-кваліфікаційний рівень *магістр*

Спеціальність *123 – Комп'ютерна інженерія*

ЗАТВЕРДЖУЮ**:**

Зав. кафедрою КСМ

С.І. Мельничук

«____» _____ 2025 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Семків Юлії Богданівні

(прізвище, ім'я, по-батькові)

Тема проекту (роботи*) Розробка та дослідження системи виявлення дезінформації на основі методів машинного навчання*

керівник проекту (роботи) Кропивницька Віталія Богданівна, *к.т.н., доцент*

затверджені наказом вищого навчального закладу від *8 грудня 2025 р. № 246/12*

2. Строк подання студентом проекту (роботи) *10.12.2025р.*

3. Вихідні дані до роботи *Методичні вказівки, матеріали переддипломної практики, технічна література.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз існуючих підходів та систем виявлення дезінформації.

2. Розробка структури та архітектури системи виявлення дезінформації.

3. Реалізація та дослідження програмної системи виявлення дезінформації на основі методів машинного навчання.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів роботи

Розділ	Консультант	Підпис, дата
Нормоконтроль	*Мойсеєнко О. В.*	

7. Дата видачі завдання *12.03.2025 р.*

№ з/п	**Назва етапів роботи**	**Термін виконання етапів роботи**	**Примітка**
1	Визначення теми магістерської роботи	12.03.25 –17.03.25	Виконано
2	Збір інформації. Дослідження предметної області	03.04.25 –31.04.25	Виконано
3	Аналіз предметної області	01.05.25 –20.05.25	Виконано
4	Розробка модуля публікації та його місце в основній структурі системи	14.06.25 –28.08.25	Виконано
5	Реалізація модуля	08.09.25 –30.10.25	Виконано
6	Оформлення роботи	01.11.25 –29.11.25	Виконано

Студент _____ *Семків Ю.Б.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Кропивницька В.Б.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Тема магістерської роботи «Розробка та дослідження системи виявлення дезінформації на основі методів машинного навчання».

Магістерська робота складається зі вступу, трьох розділів, висновків, списку використаної літератури та додатків.

У першому розділі наведено аналіз предметної області виявлення дезінформації, розглянуто основні характеристики дезінформаційного контенту, а також виконано огляд сучасних підходів і методів виявлення дезінформації з використанням машинного навчання.

Другий розділ присвячено дослідженню моделей, методів та алгоритмів виявлення дезінформації. Описано процес збору, підготовки та попередньої обробки експериментальних даних, побудову та навчання моделей машинного навчання, а також формування архітектури системи та методів інтерпретації результатів.

У третьому розділі представлено реалізацію програмної системи виявлення дезінформації та проведення експериментальних досліджень. Описано архітектуру та програмну реалізацію системи, методику проведення експериментів, отримані результати та їх аналіз, а також оцінено працездатність і практичну значущість розробленого рішення.

Ключові слова: дезінформація, класифікація, структура, аналіз.

ABSTRACT

The topic of the master's thesis is "Development and research of a disinformation detection system based on machine learning methods".

The master's thesis consists of an introduction, three sections, conclusions, a list of used literature and appendices.

The first section provides an analysis of the subject area of disinformation detection, considers the main characteristics of disinformation content, and also provides a review of modern approaches and methods for detecting disinformation using machine learning.

The second section is devoted to the study of models, methods and algorithms for detecting disinformation. The process of collecting, preparing and pre-processing experimental data, building and training machine learning models, as well as forming the system architecture and methods for interpreting the results is described.

The third section presents the implementation of a software system for detecting disinformation and conducting experimental research. The architecture and software implementation of the system, the methodology for conducting experiments, the results obtained and their analysis, and the performance and practical significance of the developed solution are described.

Keywords: disinformation, classification, structure, analysis.

ЗМІСТ

ВСТУП

Актуальність теми. В умовах сучасних геополітичних протистоянь інформаційний простір став ключовою ареною боротьби, де маніпулятивні технології систематично використовуються як інструменти впливу на масову свідомість та соціальну поведінку.

Дезінформація створює не лише приховані, а й прямі загрози сучасному суспільству, підриваючи довіру до інституцій, дестабілізуючи демократичні процеси та ускладнюючи прийняття обґрунтованих рішень громадянами.

За таких обставин розробка та дослідження ефективних систем виявлення дезінформації на основі методів машинного навчання стає актуальною науково-практичною задачею.

Метою магістерської роботи є розробка та експериментальна оцінка системи виявлення дезінформації в текстовому контенті за допомогою методів машинного навчання. Основні завдання магістерської роботи включають:

- аналіз існуючих систем виявлення дезінформації;
- розробка системи виявлення дезінформації на основі машинного навчання, включаючи попередню обробку даних та розробку моделей;
- проведення експериментальних досліджень та оцінка ефективності розробленої системи.

Об’єктом дослідження є процес автоматичного аналізу та класифікації текстової інформації з метою виявлення дезінформації.

Предметом дослідження є методи та алгоритми на основі машинного навчання для виявлення дезінформації в текстовому контенті.

Методи дослідження, що використовувалися в роботі: методи аналізу та синтезу, методи машинного навчання, методи статистичного аналізу тексту, методи проектування та програмування програмного забезпечення.

Практична цінність полягає у тому, що розроблена система може бути використана як прототип інструменту для автоматичного виявлення дезінформації в текстових даних. Результати дослідження можуть бути застосовані в системах моніторингу інформації, платформах аналізу медіа та освітніх чи дослідницьких проектах, пов'язаних з інформаційною безпекою. Розроблена програмна реалізація, експериментальна методологія та отримані результати оцінювання можуть бути використані для подальшого розвитку та інтеграції в реальні інформаційні системи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дезінформація у сучасних інформаційних системах

Дезінформація – це не лише неправдива інформація, яка була навмисно створена, трансформована або спотворена, щоб примусити, завдати шкоди або обдурити окрему особу, соціальну групу, організацію чи державу; це також потужний інструмент впливу на масову свідомість. Як трансформована та часто спотворена форма комунікації, дезінформація підриває саму сутність міжособистісної та соціальної взаємодії, поступово розмиваючи демократичні принципи та довіру до інституцій. [1]

У сучасному інформаційному просторі, особливо в умовах гібридних та відкритих військових конфліктів, суспільство все частіше стикається з «жорсткою» формою інформаційної війни, в якій правда систематично замінюється брехнею, а потім агресивно просувається. Україна сьогодні є яскравим прикладом держави, яка стала об’єктом масштабних дезінформаційних кампаній, скоординованих по всіх інформаційних екосистемах та спрямованих на дестабілізацію суспільства, делегітимізацію державної влади, виправдання агресії та спотворення історичної та соціально-політичної реальності.

Довгий час вважалося, що в країнах з розвиненою або зростаючою економікою та широким доступом до глобальних інформаційних ресурсів масштабне спотворення реальності малоймовірне: здатність формувати власне бачення подій, використовуючи різні міжнародні джерела інформації, розглядалася як важливий прояв демократії та цивілізації. Однак еволюція мас-медіа та цифрових

платформ призвела до того, що багато членів суспільства виробили звичку споживати «готові відповіді»: замість пошуку, порівняння та критичної оцінки інформації користувачі часто покладаються на заздалегідь підготовлені наративи, інтерпретації та висновки. Ця звичка стала надзвичайно зручним інструментом цілеспрямованого впливу на суспільну свідомість, де дезінформація слугує ключовим елементом «жорсткої» інформаційної, насамперед ментальної війни та засобом спотвореного формування громадської думки у різних цільових аудиторіях.

Особливо небезпечним для держави та суспільства є контент, який:

1) прямо чи опосередковано посягає на національні інтереси, територіальну цілісність та демократичні основи;

2) навмисно спотворює реальні соціально-політичні процеси та історичні факти;

3) підміняє поняття, створює міфи та пропагує ворожі наративи та цінності зовнішніх акторів;

4) використовує популізм та маніпулятивні політичні кліше для створення негативного або спотвореного образу політичних лідерів та високопосадовців.

Не менш загрозливим є навмисне поширення оманливого, шкідливого, неправдивого, перекрученого та маніпулятивного контенту, спрямованого проти окремих осіб (як елемент кібербулінгу), певних соціальних чи мовних груп, релігійних чи гендерних спільнот або цілих країн. Такий контент може провокувати не лише депресивні чи суїцидальні тенденції, але й радикалізацію, соціальну дестабілізацію та формування хибних, але емоційно переконливих ідей, спрямованих на виправдання насильства та війни. [2]

У цьому контексті дезінформація стає ефективним інструментом ведення сучасної війни, вплив якої виходить далеко за межі цифрової сфери та призводить до відчутних політичних, соціальних та безпекових наслідків. Початковий етап розвитку дезінформації зазвичай включає спотворення контенту та створення великих обсягів псевдоіндустріалізованих маніпуляцій у рамках кіберкампаній, які

навмисно забруднюють цифрову екосистему та формують громадську думку. На наступних етапах дезінформація переростає від розпалювання етнічної ненависті, гніву та ворожості до інших груп до легітимізації ідей масштабного насильства та агресивної війни.

У контексті сучасних інформаційних систем дезінформація може набувати різних форм та використовувати різні технічні та комунікаційні канали. Для цілей розробки та дослідження системи виявлення дезінформації на основі методів машинного навчання важливо визначити чітку, практично орієнтовану класифікацію, яка відображає як дезінформація створюється, представляється та поширюється в цифровому середовищі.

1. Класифікація за відношенням до правди:

- сфабрикований контент (повна вигадка) – цілком вигадана інформація, яка не має фактичної основи (фейкові новини, сфабриковані «документи», вигадані події чи цитати). Релевантність для систем машинного навчання: схильна демонструвати специфічні лексичні, стилістичні та структурні закономірності, які можна вивчити за допомогою моделей класифікації;
- маніпульований контент (часткове спотворення) – інформація, що базується на реальних фактах, але вибірково відредагована, доповнена, обрізана або візуально змінена (наприклад, відредаговані зображення чи відео, вибірково скорочені цитати). Релевантність для систем машинного навчання: вимагає моделей, здатних фіксувати невідповідності, пропуски чи суперечності відносно відомих фактів або типових наративних структур;
- хибний контекст – справжній контент (фото, відео, цитата) представлений в оманливому контексті, тобто неправильний час, місце, джерело або причинно-наслідковий зв'язок (наприклад, стара фотографія, показана так, ніби вона з поточної події). Релевантність для систем машинного навчання: особливо складно, оскільки сам контент може бути автентичним, виявлення часто вимагає зовнішніх баз знань та аналізу метаданих;

- оманливе формулювання та інтерпретація – фактично правильні твердження, оточені маніпулятивним формулюванням, оціночною мовою або упередженим вибором деталей, що призводить до спотвореного загального враження. Релевантність для систем машинного навчання: пов'язана з аналізом настроїв, виявленням позицій та ідентифікацією маніпулятивних риторичних прийомів.

2. Класифікація за способом комунікації в інформаційних системах:

- текстова дезінформація – новинні статті, дописи в соціальних мережах, коментарі, записи в блогах, миттєві повідомлення та підписи. Релевантність: основний фокус для більшості систем виявлення на основі машинного навчання: текст легше збирати, маркувати та обробляти;

- візуальна дезінформація – статичні зображення, меми, інфографіка та відредаговані фотографії (наприклад, діпфейки у формі зображення, неправдиві графіки). Релевантність: вимагає моделей комп'ютерного зору та спільного аналізу тексту та зображення;

- аудіовізуальна дезінформація – відеоконтент та аудіодоріжки (включаючи діпфейкове відео/аудіо, відредаговані промови, неправильні субтитри). Релевантність: виявлення потребує складних архітектур, здатних обробляти синхронізовані аудіо, відео та текстові потоки;

- мультимодальна дезінформація – комбіноване використання тексту, зображення, відео та метаданих в одному інформаційному об'єкті (наприклад, допис з емоційно зарядженим підписом та фейковим зображенням). Релевантність: важливий напрямок досліджень систем виявлення дезінформації, які інтегрують різні сигнали.

3. Класифікація за каналом та мережею поширення:

- дезінформація в мас-медіа – дезінформація, що поширюється через онлайн-портали новин, веб-сайти телеканалів та офіційні медіа-платформи. Релевантність: часто має відносно структуровані формати (заголовки, основний

текст, розділи), які можуть бути використані моделями;

- дезінформація в соціальних мережах та месенджерах – контент, що поширюється через платформи соціальних мереж (наприклад, Facebook, X, Instagram, TikTok), а також месенджери (наприклад, Telegram, WhatsApp, Viber). Релевантність: великий обсяг, короткі формати, вірусні репости та коментарі; виявлення має обробляти шумний текст, створений користувачами та швидкозмінні наративи;

- кампанії, керовані ботами – дезінформація, посилена мережами автоматизованих або напівавтоматизованих облікових записів (боти, тролі, скоординована неавтентична поведінка). Релевантність: вимагає поєднання аналізу контенту з графічними та поведінковими ознаками (моделі публікацій, мережі взаємодії);

- цільова мікродезінформація – повідомлення, адаптовані до певних демографічних або зацікавлених груп, з використанням механізмів таргетування (наприклад, цільова реклама, групові канали). Релевантність: може включати нішеву лексику та місцевий контекст; моделі повинні узагальнюватися на невеликі підспільноти.

4. Класифікація за наміром та стратегічною метою:

- пропаганда та ідеологічна дезінформація – спрямована на просування конкретних ідеологічних, політичних або геополітичних цілей, формування довгострокових установок;

- дестабілізуюча дезінформація – орієнтована на підрив довіри до інституцій, розпалювання соціальних конфліктів та провокування заворушень у громадах;

- дезінформація, що шкодить репутації та характеру – спрямована на конкретних осіб, організації чи групи (наприклад, кампанії з дискредитації, кібербулінг, наклеп);

- економічна та комерційна дезінформація - пов'язана з фінансовими

маніпуляціями, шахрайськими схемами, неправдивою рекламою та впливом на ринок;

- дезінформація, пов'язана з безпекою – спрямована на критичну інфраструктуру, оборону, громадську безпеку та реагування на кризи (наприклад, дезінформація під час військових операцій або надзвичайних ситуацій).

З точки зору виявлення на основі машинного навчання ці класифікації, що формуються на намірах, висвітлює різні профілі ризику та може вимагати окремих таксономій позначень та стратегій оцінки для моделей навчання та тестування.

5. Класифікація за часовою динамікою та адаптивністю:

- статична дезінформація – відносно стабільні наративи та матеріали, які суттєво не змінюються з часом (наприклад, давні історичні міфи);

- адаптивна та еволюціонуюча дезінформація – наративи та повідомлення, які швидко змінюють формулювання, формат або платформу у відповідь на модерацію, перевірку фактів або реакцію громадськості;

- дезінформація, що базується на кампаніях – скоординовані та обмежені в часі зусилля, пов'язані з певними подіями (вибори, референдуми, військова ескалація, кризи). [3]

Для системи машинного навчання врахування часових аспектів є надзвичайно важливим: моделі необхідно оновлювати або налаштовувати, щоб вони залишалися ефективними проти наративів, що розвиваються та мовних або візуальних моделей, що змінюються.

Ці класифікації дозволяють розробляти та досліджувати систему виявлення дезінформації на основі методів машинного навчання, спираючись на структуроване розуміння того, які типи дезінформації існують в інформаційних системах, чим вони відрізняються технічно та семантично, а також які комбінації методів, комп'ютерного зору, аналізу графів та поведінкового моделювання є найбільш доцільними для кожного класу.

1.2 Існуючі підходи та методи виявлення дезінформації

Для України проблема дезінформації має яскраво виражений безпековий характер. Численні дослідження описують, як український інформаційний простір став мішенню скоординованих дезінформаційних кампаній та пропагандистських наративів, зокрема в контексті російської агресії. Ці роботи надають емпіричні приклади типових повідомлень, каналів та наративів, що використовуються для спотворення сприйняття подій, послаблення соціальної згуртованості та підриву опору зовнішньому впливу. Теоретичні висновки з цих джерел формують важливу основу для розробки автоматизованих систем, здатних виявляти та класифікувати дезінформацію в реальних інформаційних потоках.

Технічна література з автоматизованого виявлення дезінформації спочатку зосереджувалася на традиційних методах машинного навчання, що застосовувалися до текстового контенту. Ранні підходи до виявлення фейкових новин та дезінформації використовували такі алгоритми, як наївний баєсівський класифікатор, методи опорних векторів та логістична регресія. Ці моделі спиралися на вручну створені текстові ознаки, такі як частота слів, n-грами та стилістичні показники, і продемонстрували задовільну продуктивність на контрольованих експериментальних наборах даних.

Для підтримки досліджень у цій галузі було запропоновано кілька еталонних наборів даних, включаючи набір даних LIAR, який містить короткі політичні заяви з перевіреними позначками правдивості, а також колекції новинних заголовків та статей, позначених як «фейкові» або «реальні». Експериментальні дослідження, проведені на цих наборах даних показують, що навіть відносно прості моделі машинного навчання можуть досягати прийнятних результатів у завданнях бінарної та багатокласової класифікації. [4]

Водночас, було розроблено та застосовано низку практичних систем для виявлення дезінформації та перевірки фактів у реальних середовищах. Міжнародні

платформи, такі як PolitiFact, FactCheck.org та Snopes, в основному покладаються на експертний аналіз, але все частіше використовують автоматизовані інструменти для фільтрації та визначення пріоритетів контенту для перевірки. Великі технологічні компанії, включаючи Google та Meta (Facebook), використовують моделі машинного навчання для аналізу текстового контенту, поведінки користувачів та моделей поширення інформації, щоб виявляти оманливий контент та зменшувати його поширення. Ці системи зазвичай поєднують класифікацію на основі тексту з аналізом метаданих та функціями мережевого рівня.

Дослідницькі платформи, такі як FakeNewsNet, надають інтегровані середовища, що поєднують текстовий контент, інформацію про джерела та соціальний контекст для виявлення фейкових новин. Такі системи часто використовуються для оцінки та порівняння підходів машинного навчання в уніфікованих експериментальних умовах. Їхні модульні архітектури, які включають збір даних, попередню обробку, вилучення ознак, класифікацію та інтерпретацію результатів, концептуально узгоджуються з підходом, запропонованим у цій роботі.

Однак обмеження ранніх підходів машинного навчання стали очевидними відносно швидко. Моделі, засновані переважно на поверхневих лексичних ознаках та простих стилістичних індикаторах, часто не враховують глибші семантичні зв'язки, довгостроковий контекст та мовні моделі, що розвиваються. Як наслідок, їхня здатність узагальнювати по різних темах, часових періодах та інформаційних кампаніях обмежена. Крім того, багато ранніх підходів ігнорують метадані, такі як достовірність джерела або поведінка користувачів, а також мережеві характеристики, що описують, як поширюється інформація, що є важливими факторами надійного виявлення дезінформації.

Ці обмеження спонукали до переходу до більш просунутих текстових представлень та використання методів глибокого навчання. Згорткові та рекурентні нейронні мережі, включаючи архітектури LSTM та BiLSTM, були введені для моделювання послідовних залежностей та контекстних зв'язків у тексті. Пізніше

були запропоновані механізми уваги та ієрархічні моделі для кращого фіксування структури на рівні документа та зосередження на найбільш інформативних частинах текстового контенту. Паралельно дослідники почали включати соціальні та поширені ознаки для моделювання динаміки поширення дезінформації.

Значним проривом у цій галузі стала поява мовних моделей на основі Transformer, таких як BERT, RoBERTa та їх багатомовних варіантів. Ці моделі попередньо навчені на великомасштабних текстових корпусах і здатні вивчати багаті контекстуальні представлення, які фіксують як синтаксичну, так і семантичну інформацію. При точному налаштуванні на наборах даних дезінформації або фейкових новин такі моделі досягають найсучаснішої продуктивності в таких завданнях, як перевірка фактів, виявлення чуток, класифікація позицій та ідентифікація пропаганди. Спеціалізовані архітектури, такі як FakeBERT, додатково адаптують моделі Transformer спеціально для виявлення дезінформації. [5]

Незважаючи на ці досягнення, залишається кілька важливих проблем. Сучасним моделям часто бракує інтерпретованості, що ускладнює пояснення, чому конкретне повідомлення класифікується як дезінформація. Їхня стійкість до маніпуляцій з боку суперника та зміни наративів обмежена, а моделі, навчені на одній мові, темі чи платформі, часто погано працюють при застосуванні до інших. Крім того, існує нестача великих, високоякісних маркованих наборів даних багатьма мовами, включаючи українську.

Вітчизняні дослідження, пов'язані з дезінформацією в українському інформаційному просторі, в основному зосереджені на якісному та напівкількісному аналізі пропагандистських наративів, ворожих інформаційних операцій та маніпуляцій у ЗМІ. Українські дослідники аналізують домінуючі теми, стратегії повідомлень та лінгвістичні маркери, що використовуються в традиційних та онлайн-медіа, особливо в контексті гібридної війни. Хоча ці дослідження дають цінні знання, кількість робіт, які безпосередньо застосовують сучасні методи машинного навчання та глибокого навчання для виявлення дезінформації

українською мовою, залишається обмеженою.

Крім того, існуючі українські ініціативи, такі як StopFake та Центр протидії дезінформації при Раді національної безпеки і оборони України, значною мірою спираються на експертний аналіз, що підтримується базовими автоматизованими інструментами. Відсутність великих загальнодоступних наборів даних українською мовою та обмежене застосування передових моделей машинного навчання обмежують масштабованість та автоматизацію цих систем.

Аналіз вітчизняних та іноземних джерел виявляє кілька ключових прогалин у дослідженнях: обмежене мовне охоплення, окрім англійської, недостатня доступність україномовних наборів даних, відсутність систематичної адаптації сучасних моделей до української дезінформації, обмежений розгляд мультимодального контенту та недостатня увага до інтерпретації та практичної інтеграції в реальні робочі процеси інформаційної безпеки.

Ці недоліки обґрунтовують необхідність розробки та дослідження системи виявлення дезінформації на основі машинного навчання, адаптованої до українського інформаційного середовища. Така система може сприяти зміцненню інформаційної стійкості, підвищенню ефективності моніторингу та підтримці своєчасного реагування на дезінформаційні кампанії в цифровому просторі.

1.3 Вибір методів теоретичних та експериментальних досліджень

Розробка та дослідження системи виявлення дезінформації на основі методів машинного навчання вимагає обґрунтованого вибору як теоретичних, так і експериментальних методів дослідження. Характер проблеми - обробка великих обсягів текстових даних, виявлення прихованих закономірностей та оцінка моделей класифікації, що визначає необхідність поєднання підходів з різних галузей.

З теоретичної точки зору використовуються такі методи:

1. Аналіз наукових джерел.

Проведено огляд вітчизняних та зарубіжних публікацій з дезінформації, інформаційної війни, виявлення фейкових новин, обробки природної мови та машинного навчання. Це дозволяє узагальнити існуючі визначення, типології та концептуальні моделі, а також визначити характерні риси дезінформаційного контенту та сучасний стан досліджень у сфері автоматизованих систем виявлення.

2. Формалізація та класифікація.

На основі огляду літератури уточнено ключові поняття та запропоновано практично орієнтовану класифікацію видів дезінформації в інформаційних системах. Ця класифікація згодом частково буде використана для визначення структури класів для моделей машинного навчання.

3. Порівняльний аналіз методів та моделей.

Існуючі підходи до машинного та поглибленого навчання (традиційні класифікатори, згорткові та рекурентні нейронні мережі) порівнюються з точки зору їхньої придатності для завдань класифікації тексту, обчислювальної складності, вимог до даних та потенціалу адаптації до україномовного та багатомовного контенту. Результати цього аналізу обґрунтовують вибір підмножини моделей для подальшого емпіричного дослідження.

З експериментальної точки зору, дослідження базується на таких методах та процедурах:

1. Збір даних та побудова експериментального набору даних.

Створюється репрезентативний набір текстових матеріалів, який може включати новинні статті, публікації в соціальних мережах та інші повідомлення, що містять як достовірну інформацію, так і дезінформацію. По можливості ці матеріали маркуються з використанням існуючих ресурсів перевірки фактів, експертних суджень або чітких правил анотування. Цей крок забезпечує маркований набір даних, необхідний для навчання моделей машинного навчання та об'єктивної оцінки їхньої ефективності.

2. Попередня обробка та представлення текстових даних.

Зібрані текстові дані проходять процедури попередньої обробки, включаючи очищення, нормалізацію та, за потреби, ідентифікацію мови. Згодом застосовуються різні методи представлення тексту, отримані з попередньо навчених мовних моделей. Вибір цих представлень обґрунтований їх широким використанням у сучасній обробці мови та їх доведеною ефективністю у вирішенні задач класифікації тексту.

3. Навчання та оцінка моделей машинного навчання.

Набір моделей навчається та емпірично порівнюється, включаючи базові традиційні класифікатори та розширені архітектури поглибленого навчання. Якість моделей оцінюється за допомогою стандартних метрик, таких як точність, прецизійність (ступінь близькості та узгодженості результатів повторних вимірювань однієї й тієї ж величини), повнота та F1-оцінка (середнє гармонійне). Для оцінки здатності моделей до узагальнення та зменшення впливу випадкових факторів використовуються такі методи, як поділ на тренувальні/валідаційні/тестові набори або перехресна валідація.

4. Аналіз помилок та уточнення рішень моделювання.

Неправильно класифіковані приклади потім аналізуються, щоб знайти типові причини помилок (наприклад, сарказм, тексти різними мовами, складні пропагандистські наративи або нечіткі позначення). Висновки цього аналізу використовуються для покращення попередньої обробки даних, текстових представлень та архітектур моделей, а також для кращого розуміння обмежень запропонованого підходу.

5. Порівняльний експериментальний аналіз.

Результати, отримані для різних моделей та текстових представлень, порівнюються та інтерпретуються. На цій підставі обґрунтовується остаточний вибір моделі (або комбінації моделей), враховуючи точність, надійність, обчислювальні витрати та практичне застосування до завдання виявлення дезінформації в сучасних інформаційних системах. [6]

1.4 Постановка задачі

Завданням цієї роботи є розробка, впровадження та експериментальна оцінка системи на основі машинного навчання для автоматичного виявлення дезінформації в текстових даних. Запропонований підхід базується на контрольованій двійковій класифікації тексту, де кожен вхідний текст відноситься до одного з двох класів: дезінформація або достовірна інформація.

Завдання також охоплює створення набору даних, попередню обробку тексту, перетворення текстових даних у числові представлення ознак та навчання моделі класифікації машинного навчання. Ефективність розробленої системи оцінюється за допомогою стандартних показників продуктивності, включаючи точність, прецизійність, повноту та F1-оцінку, а потім проводиться аналіз отриманих результатів для виявлення сильних сторін та обмежень застосованого підходу.

2 ДОСЛІДЖЕННЯ МЕТОДІВ ТА АЛГОРИТМІВ ВИЯВЛЕННЯ ДЕЗІНФОРМАЦІЇ

2.1 Аналіз характеристик та вимог до даних

Дезінформація є специфічною категорією текстового контенту, яка відрізняється від достовірної інформації своєю структурою, використанням мови та семантичними властивостями. Визначення цих відмінностей є важливим для розробки системи на основі машинного навчання, яка може автоматично виявляти дезінформацію.

Однією з ключових характеристик є лексичні ознаки. Дезінформаційні тексти часто містять емоційно-експресивну лексику, сенсаційні формулювання та суб'єктивну мову, спрямовану на вплив на читача. Ці тексти можуть містити повторювані ключові слова, спрощені вирази та емоційно заряджені терміни. З алгоритмічної точки зору, такі характеристики можна ефективно фіксувати за допомогою текстових представлень на основі частоти, включаючи Bag of Words та TF-IDF, які кількісно визначають важливість слів у документах та між ними. [7]

Ще одна важлива характеристика складається з синтаксичних ознак. Дезінформаційний контент часто використовує короткі речення, повторювані конструкції та нестандартну пунктуацію. Ці властивості можна виміряти за допомогою простих статистичних параметрів, таких як довжина речення, кількість лексем та частота пунктуації. Включення таких ознак допомагає моделям машинного навчання розрізняти стилістичні відмінності між дезінформацією та достовірним текстом.

На семантичному рівні дезінформаційні тексти можуть містити розпливчасті твердження, слабкі логічні зв'язки або оманливі асоціації між поняттями. Хоча класичні методи машинного навчання не повністю розуміють значення тексту, вони можуть виявляти спільні семантичні закономірності, аналізуючи, як слова виглядають разом та в подібних контекстах.

На основі цього аналізу визначено основні вимоги до даних. Набір даних повинен містити текстові зразки, позначені як дезінформація або достовірна інформація. Він має бути достатньо великим, щоб підтримувати надійне навчання моделі, а кількість зразків у кожному класі має бути збалансованою або скоригованою за необхідності. Усі текстові дані повинні пройти попередньо редаговані та бути представлені в числовій формі для введення в алгоритми машинного навчання. [8]

2.2 Збір, підготовка та попередня обробка експериментальних даних

Збір та підготовка експериментальних даних є важливим етапом у розробці системи виявлення дезінформації на основі машинного навчання. Якість набору даних безпосередньо впливає на точність, стабільність та здатність до узагальнення навчених моделей.

Для цілей цього дослідження набір даних було сформовано з використанням загальнодоступних текстових джерел, які надають позначені приклади дезінформації та достовірної інформації. Зразки дезінформації були зібрані з відкритих платформ перевірки фактів та дослідницьких наборів даних, що містять перевірені неправдиві або оманливі твердження. Зокрема, були використані загальнодоступні колекції фейкових новинних статей та коротких тверджень, опублікованих на платформах відкритих даних. Зразки достовірної інформації були зібрані з офіційних новинних веб-сайтів, урядових порталів та перевірених ЗМІ, які публікують фактичні звіти та оголошення. [9]

Як конкретний приклад, дезінформаційні тексти включали заголовки новин та короткі статті, визначені як неправдиві організаціями з перевірки фактів, тоді як достовірні тексти включали офіційні прес-релізи та новинні повідомлення, опубліковані довіреними джерелами. Кожному зібраному зразку тексту було присвоєно мітку класу (дезінформація або достовірність) на основі анотації, наданої джерелом.

Усі зібрані дані зберігаються у структурованому CSV-файлі. Кожен запис у наборі даних містить два поля: текстове поле, що містить повний текстовий вміст, та поле мітки, що вказує на клас тексту. Цей формат спрощує обробку даних та дозволяє зручно використовувати набір даних під час експериментальних досліджень.

Після збору даних було проведено початковий етап підготовки даних. Під час цього кроку було видалено дублікати записів, відфільтровано неповні або надто короткі тексти, а також виключено зразки, які не відповідали попередньо визначеним вимогам якості. Це гарантувало, що набір даних містив лише змістовну та релевантну текстову інформацію.

Наступний етап включав попередню обробку текстових даних для досягнення узгодженого формату. Усі тексти були модифіковані на малі літери, а неінформативні елементи, такі як URL-адреси, числа, розділові знаки та спеціальні символи, були видалені. Також було виключено загальновживані слова, які не несуть значної семантичної інформації.

Крім того, було застосовано нормалізацію слів шляхом зведення їх до базових форм. Цей крок зменшує кількість різних варіантів слів у наборі даних.

Після попередньої обробки тексти були перетворені на числові представлення ознак за допомогою методу TF-IDF. Цей підхід надає значення словам на основі їхньої важливості в окремих текстах та в усьому наборі даних, що дозволяє обробляти текстову інформацію алгоритмами машинного навчання.

На завершальному етапі підготовлений набір даних був розділений на

навчальну та тестову підмножини. Навчальна підмножина використовувалася для побудови моделей машинного навчання, тоді як тестова підмножина була зарезервована для оцінки продуктивності. Такий поділ забезпечує об'єктивну оцінку розробленої системи та зменшує ризик перенавчання.

2.3 Побудова та навчання моделей машинного навчання для виявлення дезінформації

Побудова та навчання моделей машинного навчання є важливим етапом розробки системи виявлення дезінформації. Цей етап виконується після завершення збору, підготовки та попередньої обробки даних, оскільки якість та структура вхідних даних безпосередньо впливають на ефективність алгоритмів машинного навчання. У цій роботі завдання виявлення дезінформації сформульовано як бінарну задачу класифікації тексту, де кожне текстове повідомлення має бути віднесено до одного з двох класів: дезінформація або достовірна інформація.

Формулювання проблеми як завдання бінарної класифікації дозволяє застосовувати стандартні методи машинного навчання та спрощує аналіз отриманих результатів. Кожен текст перетворюється на числовий вектор ознак та йому присвоюється мітка класу, що дозволяє моделям вивчати закономірності, що відрізняють дезінформацію від достовірного контенту.

Вибір алгоритмів машинного навчання

Для вирішення задачі класифікації обрано класичні алгоритми машинного навчання. Вибір таких методів мотивований їхньою ефективністю в задачах класифікації тексту, відносно низькою обчислювальною складністю та можливістю пояснення процесу прийняття рішень. Зокрема, для експериментальної оцінки обрано два алгоритми: поліноміальний наївний баєсівський алгоритм та логістична регресія.

Обидва алгоритми широко використовуються в задачах обробки мови та добре підходять для багатовимірних ознак, що генеруються методами векторизації

тексту, такими як TF-IDF. Їх використання також дозволяє провести порівняння між ймовірнісними та дискримінативними підходами до класифікації. [10]

Процес навчання

Загальний процес побудови моделі, навчання, налаштування та оцінки проілюстровано на блок-схемі (рис. 2.1).

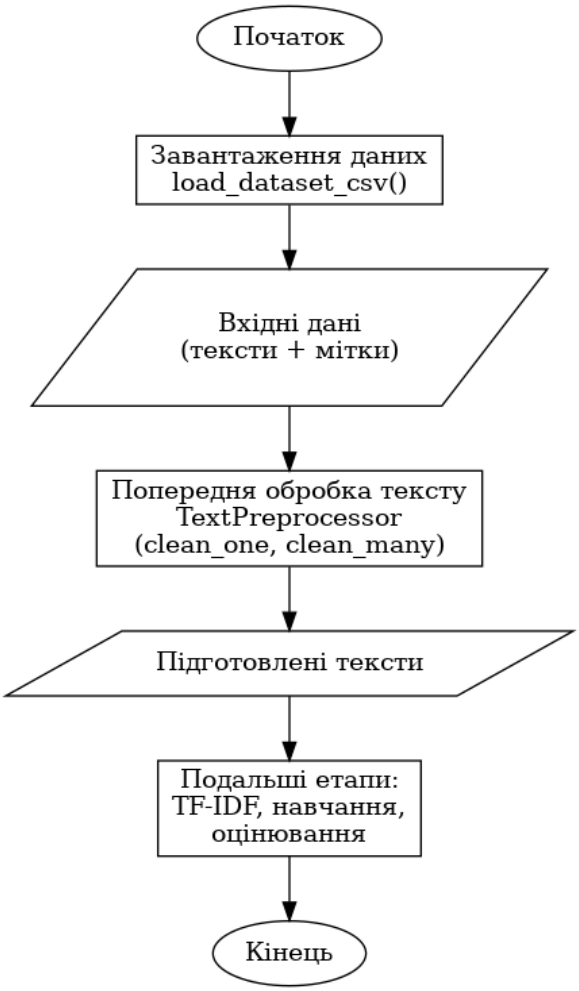

Рисунок 2.1 – Блок-схема початкових етапів процесу навчання моделі

Ця блок-схема відображає експериментальний конвеєр, що використовується в реалізації та показує послідовний та ітеративний характер процесу навчання. Кожен блок на схемі безпосередньо відповідає певному компоненту або методу в програмній реалізації.

Процес починається із завантаження експериментального набору даних за допомогою функції load_dataset_csv(). Ця функція зчитує CSV-файл, що містить зразки тексту та відповідні мітки класів і повертає їх у структурованій формі, придатній для подальшої обробки. Цей крок відповідає блоку вхідних даних на блок-схемі.

Наступний етап – попередня обробка тексту, яка реалізується класом TextPreprocessor. Методи clean_one() та clean_many() відповідають за застосування правил попередньої обробки до окремих текстів або набору текстів. Цей етап відповідає блоку попередньої обробки тексту на схемі та гарантує, що всі тексти будуть перетворені в узгоджений формат перед вилученням ознак.

Вилучення ознак та підготовка набору даних

Після попередньої обробки текстові дані перетворюються на числові вектори ознак за допомогою методу TF-IDF. Ця операція реалізується компонентом TfidfVectorizer, який створюється всередині методу _build_pipeline() класу DisinfoDetectionSystem. TF-IDF дозволяє представляти тексти як вектори, що відображають важливість слів у документах та по всьому набору даних.

На цьому етапі набір даних розділяється на навчальні та тестові підмножини за допомогою функції train_test_split() з бібліотеки scikit-learn. Цей крок відповідає блоку розділення набору даних на блок-схемі процесу навчання та забезпечує те, що оцінка моделі виконується на даних, які не використовувалися під час навчання.

Реалізація та аналіз алгоритмів класифікації

Етап класифікації відповідає блокам вибору алгоритму та навчання моделі, показаним на рисунку 2.1. Створення класифікаторів обробляється функцією build_classifier(model_type), яка створює екземпляр моделі LogisticRegression або MultinomialNB залежно від конфігурації, зазначеної в класі SystemConfig.

Внутрішня логіка процесу роботи алгоритмів класифікації проілюстрована на блок-схемі (рисунок 2.2).

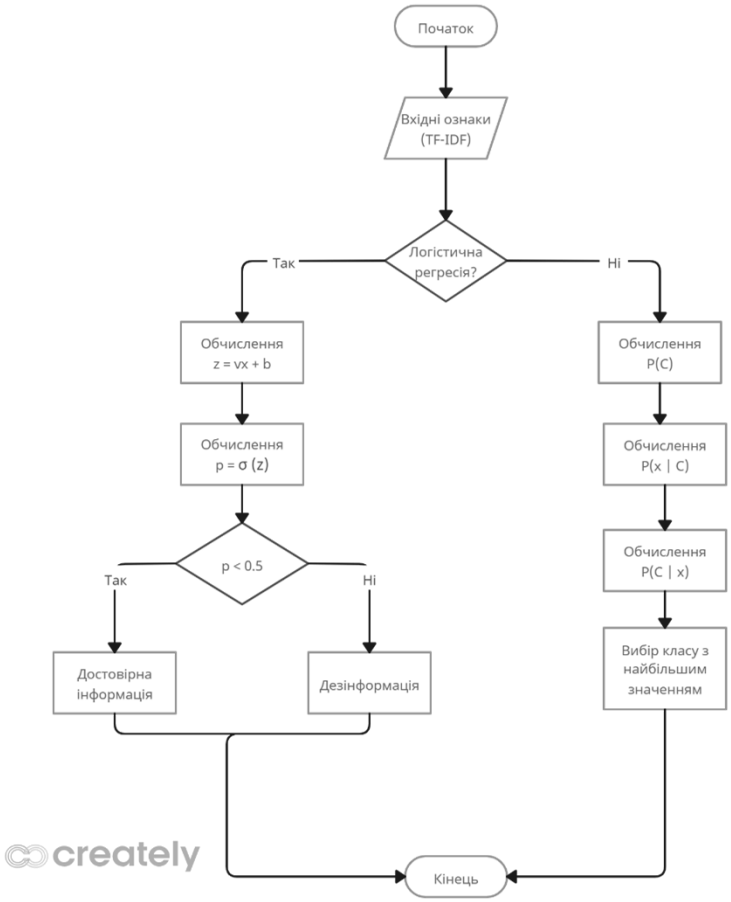

Рисунок 2.2 - Блок-схема реалізації та роботи алгоритмів класифікації

Ця блок-схема представляє етап класифікації системи та показує, як різні алгоритми працюють в одній і тій самій структурі обробки. Як логістична регресія, так і поліноміальний наївний байєсівський алгоритм отримують ідентичні вхідні дані та відрізняються лише внутрішніми механізмами прийняття рішень, що забезпечує справедливе та методологічно коректне порівняння.

У гілці наївного баєсівського методу блок-схеми на рисунку 2.2 рішення про класифікацію базується на ймовірнісному моделюванні. Алгоритм спочатку оцінює апріорні ймовірності для кожного класу, використовуючи навчальні дані. Потім він оцінює ймовірність спостереження заданих значень ознак TF-IDF для кожного класу за припущення умовної незалежності між ознаками.

Ці ймовірності об'єднуються для обчислення апостериорних балів для

кожного класу і клас з найвищим балом вибирається як результат класифікації. Як результат, багатономіальний наївний баєсівський метод забезпечує швидке навчання та стабільну базову продуктивність.

Однак, через спрощене припущення незалежності, алгоритм може не повністю враховувати зв'язки між ознаками. Це може обмежити якість класифікації у випадках, коли шаблони дезінформації є більш складними та залежать від комбінацій слів, а не від окремих термінів.

Гілка логістичної регресії на блок-схемі дотримується іншої логіки прийняття рішень. Замість моделювання розподілів ймовірностей, алгоритм вивчає набір значень для окремих ознак. Під час класифікації він обчислює лінійну комбінацію ознак TF-IDF, яка потім перетворюється на значення ймовірності за допомогою логістичної (сигмоїдної) функції.

До цієї ймовірності застосовується поріг прийняття рішення для визначення остаточної мітки класу. Такий підхід дозволяє логістичній регресії моделювати складніші межі прийняття рішень та враховувати відносну важливість різних ознак. Як результат, вона часто забезпечує вищу точність класифікації та кращий баланс між хибнопозитивними та хибнонегативними помилками.

Додатковою перевагою логістичної регресії є наявність оцінок ймовірностей та важливості ознак, що покращує інтерпретованість та підтримує аналіз рішень щодо класифікації. [11]

Наївний байєсівський метод спирається на ймовірнісні міркування та припущення незалежності, пропонуючи швидкість та простоту, тоді як логістична регресія використовує важливість ознак та лінійне розділення, забезпечуючи вищу точність та краще узагальнення у складних сценаріях класифікації.

Налаштування параметрів та ітеративне вдосконалення

Для покращення продуктивності класифікації параметри моделі налаштовуються експериментально. Цей етап відповідає блокам налаштування параметрів та оцінки якості на блок-схемі процесу навчання. Налаштування

параметрів виконується шляхом зміни налаштувань класифікатора та TF-IDF, що зберігаються в класі SystemConfig, таких як сила регуляризації, діапазон n-грамів та кількість ознак.

Якщо результати оцінювання не відповідають бажаному рівню якості, процес навчання повторюється з оновленими параметрами. Цей ітераційний цикл дозволяє поступово покращувати продуктивність моделі та зменшує ризик перенавчання.

Оцінювання, порівняння та вибір моделі

Оцінювання моделі реалізовано в методі evaluate() класу DisinfoDetectionSystem. Цей метод обчислює стандартні метрики оцінювання, включаючи точність, прецизійність, повноту, F1-оцінку та матрицю невідповідності. Ці метрики відповідають блоку оцінювання на рисунку 2.1 та забезпечують кількісну оцінку якості класифікації.

Як логістична регресія, так і наївна байєсівська моделі оцінюються за однакових експериментальних умов, що дозволяє справедливо порівнювати їхню продуктивність. Логістична регресія загалом демонструє вищу точність класифікації та більш збалансовані значення точності та повноти завдяки своїй здатності присвоювати важливість окремим ознакам. Наївна байєсівська модель, хоча й створює дещо нижчі метрики продуктивності, пропонує швидше навчання та простіші обчислення.

На основі отриманих результатів вибирається найефективніша модель для інтеграції в систему виявлення дезінформації. Вибрана модель разом із правилами попередньої обробки та параметрами конфігурації зберігається за допомогою бібліотеки joblib для забезпечення відтворюваності та подальшого практичного використання.

2.4 Структура системи

На основі експериментальних результатів, отриманих на попередніх етапах

дослідження, було проаналізовано застосовані моделі машинного навчання та методи обробки даних з метою оцінки їхньої ефективності у виявленні дезінформації. Аналіз базується на кількісних метриках ефективності, а також на якісному дослідженні помилок класифікації. Такий підхід дозволяє визначити, як окремі компоненти системи впливають на кінцевий результат класифікації, та визначити напрямки покращення загальної якості рішення.

Експериментальна оцінка продемонструвала, що продуктивність моделі залежить не лише від вибору алгоритму класифікації, але й від конфігурації етапів попередньої обробки та вилучення ознак. В результаті було впроваджено покращення як в обробці даних, так і в конфігурації моделі і ці уточнені компоненти були використані як основа для проектування кінцевої програмної системи для виявлення дезінформації.

Розроблена система виявлення дезінформації дотримується модульної конвеєрної структури. Цей підхід обраний тому, що проблема виявлення дезінформації природно складається з послідовності етапів обробки, де вихід одного етапу служить вхідними даними для наступного. Розділення цих етапів на незалежні модулі підвищує чіткість реалізації, спрощує налагодження та дозволяє змінювати або замінювати окремі компоненти, не впливаючи на всю систему.

Загальна структура системи проілюстрована на схемі (рис. 2.3).

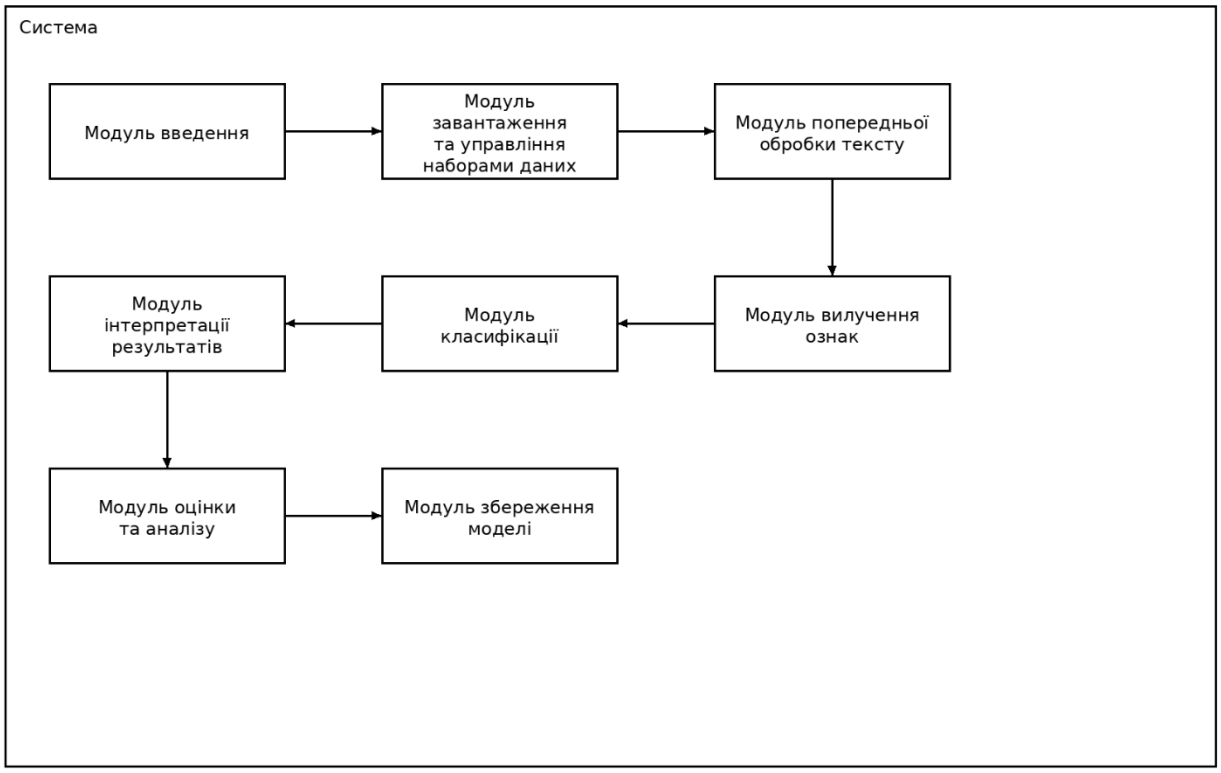

Рисунок 2.3 – Схема структури системи

Ця схема представляє основні функціональні модулі системи та напрямок потоку даних між ними, від початкового введення текстових даних до кінцевого виведення результатів класифікації.

Структура системи виявлення дезінформації складається з кількох взаємопов'язаних модулів, кожен з яких відповідає за певну функцію обробки.

Модуль введення

Модуль введення відповідає за отримання текстових даних, які будуть аналізуватися системою. Система підтримує два основні режими роботи: експериментальний режим, у якому набори даних завантажуються з CSV-файлів та режим логічного висновку, у якому для класифікації надаються окремі тексти. Ця варіативність дозволяє використовувати систему як для дослідницьких експериментів, так і для практичного тестування окремих повідомлень.

Модуль завантаження та управління наборами даних

Цей модуль зчитує структуровані набори даних із зовнішніх джерел та готує їх до подальшої обробки. Він перевіряє наявність обов'язкових полів, витягує зразки тексту та відповідні мітки класів, а також гарантує, що дані надаються у форматі, сумісному з наступними етапами. Централізуючи обробку наборів даних в окремому модулі, система забезпечує узгодженість між експериментами та спрощує адаптацію до різних форматів даних.

Модуль попередньої обробки тексту

Модуль попередньої обробки тексту виконує нормалізацію та очищення необроблених текстових даних. Його метою є зменшення непотрібної змінюваності текстуального введення, зберігаючи при цьому інформацію, що має значення для класифікації. Правила попередньої обробки налаштовуються та включають такі операції, як нормалізація регістру літер, видалення посилань та числових символів, фільтрація нерелевантних символів та виключення дуже коротких слів. В результаті всі тексти перетворюються на узгоджене представлення, що покращує стабільність та надійність вилучення ознак.

Модуль вилучення ознак

Після попередньої обробки текстові дані передаються до модуля вилучення ознак, який перетворює текст на числові представлення. Система використовує векторизацію TF-IDF для представлення кожного тексту як вектора ознак. Цей метод відображає важливість окремих слів у документі та в усьому наборі даних, що робить його придатним для класичних алгоритмів машинного навчання. Виходом цього модуля є багатовимірний, але розріджений простір ознак, який фіксує ключові лексичні закономірності в даних.

Модуль класифікації

Модуль класифікації є ядром системи та відповідає за призначення міток класів вхідним текстам. Система підтримує дві моделі машинного навчання: логістичну регресію та поліноміальну наївну байєсівську. Обидві моделі працюють

на векторах ознак TF-IDF та інтегровані в єдиний конвеєр обробки. Вибір моделі можна скоригувати за допомогою конфігурації системи, що дозволяє безпосередньо порівнювати їхню продуктивність за ідентичних умов.

Модуль інтерпретації результатів

Модуль інтерпретації результатів бере вихідні дані, отримані за допомогою моделі класифікації та перетворює їх у форму, зручну для розуміння та використання. Він надає остаточний прогнозований клас тексту та, якщо обрана модель дозволяє, додаткове значення достовірності, яке показує, наскільки достовірним є прогноз. Ці вихідні дані використовуються як для оцінки системи під час експериментів, так і для класифікації окремих текстів на практиці.

Модуль оцінки та аналізу

Для оцінки продуктивності системи модуль оцінки обчислює стандартні показники класифікації, включаючи точність, прецизійність, повноту, F1-оцінку та матрицю невідповідності. Ці показники дозволяють проводити детальний аналіз різних типів помилок класифікації та підтримують об'єктивне порівняння моделей та конфігурацій. Результати оцінки безпосередньо пов'язані з архітектурним проектом, оскільки вони забезпечують зворотний зв'язок для вдосконалення окремих модулів.

Модуль збереження моделі

Система включає модуль для збереження та завантаження навчених моделей разом з їх параметрами попередньої обробки та конфігурації. Цей модуль забезпечує відтворюваність експериментальних результатів та дозволяє повторно використовувати навчену систему без перенавчання.

Взаємодія між системними модулями відбувається у визначеній послідовності, що також відображається на блок-схемі архітектури системи. Текстові дані проходять від вхідного модуля через етапи попередньої обробки, вилучення ознак та класифікації і досягають модуля інтерпретації результатів. Зворотній зв'язок від модуля оцінки може впливати на попередні етапи, мотивуючи

зміни в правилах попередньої обробки або параметрах моделі.

Цей чітко визначений порядок взаємодії забезпечує стабільну поведінку системи та запобігає невідповідностям між етапами навчання та логічного висновку. Оскільки кожен модуль виконує одну чітко визначену функцію, система залишається прозорою та легкою для аналізу.

Типові сценарії використання системи проілюстровано на діаграмі прецедентів (Рис. 2.4).

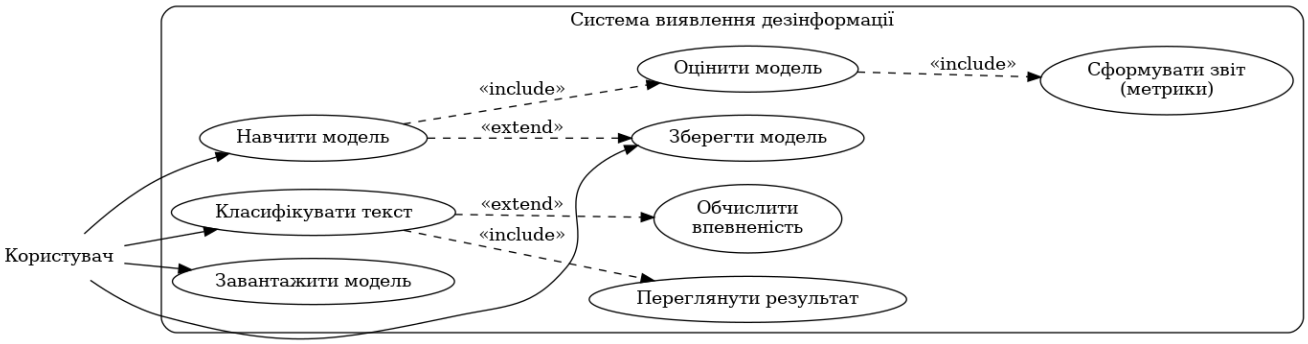

Рисунок 2.4 – Use Case діаграма

Ця діаграма представляє систему з точки зору користувача та показує, як доступ до різних функцій системи здійснюється під час роботи. Вона включає одного основного актора – Користувача, який може ініціювати навчання моделі, класифікувати нові тексти, завантажувати існуючі моделі та зберігати навчені моделі для подальшого використання.

Варіант використання «Навчити модель» представляє процес навчання моделі класифікації з позначених даних. Цей варіант використання включає «Оцінити модель», оскільки після навчання моделі завжди йде оцінка продуктивності. Результати оцінки узагальнюються за допомогою варіанту використання «Сформувати звіт», який надає стандартні показники класифікації.

Варіант використання «Класифікувати текст» представляє практичне застосування системи для аналізу окремих текстових повідомлень. Він включає

«Переглянути результат», який гарантує, що результати класифікації будуть представлені користувачеві. Додаткове розширення «Обчислити впевненість» виконується, коли вибрана модель підтримує оцінку ймовірності.

Варіанти використання «Зберегти модель» та «Завантажити модель» підтримують повторне використання навчених моделей та забезпечують роботу системи без повторного навчання.

Загалом, діаграма прецедентів демонструє, що система підтримує як експериментальні дослідницькі робочі процеси, так і практичні сценарії класифікації тексту чітким та структурованим чином.

Загалом, запропонована архітектура системи формує чітку та адаптивну основу для застосування методів машинного навчання до завдання виявлення дезінформації. Розділення системи на окремі компоненти робить етапи обробки зрозумілими, спрощує аналіз результатів та забезпечує можливість відтворення експериментальних результатів. Така організація також дозволяє безпосередньо пов'язувати експериментальні результати з певними компонентами системи, що допомагає покращити ефективність класифікації послідовним та контрольованим способом. [12]

Використання модульної конструкції разом із систематичною експериментальною оцінкою показує, що розроблена система підходить для ефективного виявлення дезінформації. Водночас, архітектура залишається відкритою для подальшого розвитку, включаючи додавання нових моделей класифікації, використання інших джерел даних або розширення інструментів аналізу та оцінки.

3 РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ СИСТЕМИ ВИЯВЛЕННЯ ДЕЗІНФОРМАЦІЇ

3.1 Реалізація програмного забезпечення

Система виявлення дезінформації реалізована на Python з використанням конвеєрної архітектури. Програма зберігається як один файл Python для зручності запуску та відтворення проєкту, але всередині цього файлу код організовано в окремі логічні частини: конфігурація, попередня обробка, завантаження набору даних, побудова моделі, навчання/оцінка, прогнозування та інтерфейс командного рядка.

Використані технології та бібліотеки

Програмна реалізація системи виявлення дезінформації базується на широко використовуваних бібліотеках Python, які підтримують завдання обробки даних та машинного навчання.

```
# Data handling
import pandas as pd
# Text processing
import re
# Machine learning and evaluation
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
# Model persistence
import joblib
# Command-line interface
import argparse
```

Бібліотека pandas використовується для завантаження та керування наборами даних, що зберігаються у форматі CSV, надаючи зручні інструменти для структурованої обробки даних. Операції попередньої обробки тексту, такі як зіставлення зі зразком та видалення небажаних елементів, реалізуються за допомогою модуля re, який підтримує регулярні вирази.

Функціональність машинного навчання реалізована за допомогою бібліотеки scikit-learn. Вона використовується для перетворення текстових даних у числові представлення за допомогою методу TF-IDF, навчання та застосування моделей класифікації, розділення даних на навчальні та тестові підмножини, а також розрахунку стандартних метрик оцінки. Крім того, використовуються конвеєри scikit-learn для об'єднання вилучення ознак та класифікації в єдиний, узгоджений робочий процес обробки.

Бібліотека joblib використовується для збереження та відновлення навчених моделей разом з їх конфігурацією, що гарантує можливість відтворення експериментальних результатів та повторного використання навчених систем без перенавчання. Взаємодія з системою організована через інтерфейс командного рядка, реалізований за допомогою модуля argparse, який дозволяє виконувати систему в режимі навчання або прогнозування.

Структура коду (основні частини програми)

1. Об'єкти конфігурації (PreprocessConfig, SystemConfig) – це невеликі структури, які зберігають параметри системи в одному місці.

PreprocessConfig визначає, як очищується необроблений текст (наприклад:

перетворення нижнього регістру, видалення URL-адрес, видалення цифр, мінімальна довжина слова).

SystemConfig визначає експериментальні та модельні налаштування (наприклад: яку модель використовувати, розмір тестового розділення, налаштування TF-IDF, випадкове початкове значення).

2. Попередня обробка тексту (TextPreprocessor) – цей блок готує необроблений текст, щоб він став узгодженим та придатним для вилучення ознак.

clean_one(text) обробляє одне повідомлення.

clean_many(texts) обробляє цілий список повідомлень (що використовуються під час навчання).

3. Завантаження набору даних (load_dataset_csv) – ця функція зчитує набір даних з CSV-файлу та повертає список зразків тексту і список міток (класів). Вона також перевіряє наявність необхідних стовпців.

4. Вибір моделі (build_classifier) – цей блок створює алгоритм машинного навчання, який використовується для класифікації.

Якщо model_type="logreg", він створює логістичну регресію.

Якщо model_type="nb", він створює поліноміальну наївну баєсівську регресію.

5. Основний системний клас (DisinfoDetectionSystem) - це основний клас, який об'єднує все в одну робочу систему.

Його обов'язки включають:

- побудову повного конвеєра (TF-IDF + класифікатор);
- навчання (train());
- оцінку (evaluate());
- прогнозування (predict());
- додатковий розрахунок достовірності (якщо підтримується).

6. Точка входу CLI (main()) – це та частина, яка робить програму придатною для використання з командного рядка. Вона підтримує два режими:

- train: завантажує набір даних, навчає модель, оцінює, зберігає модель;
- predict: завантажує збережену модель, класифікує один новий текст.

Рівень конфігурації (керує поведінкою системи)

1. PreprocessConfig – параметри попередньої обробки. Ця структура зберігає правила попередньої обробки, щоб їх можна було змінювати без перезапису логіки.

```
from dataclasses import dataclass

@dataclass
class PreprocessConfig:
lowercase: bool = True
remove_urls_emails: bool = True
remove_digits: bool = True
keep_punctuation: bool = False
min_word_len: int = 2
```

Цей об'єкт визначає, що має робити попередня обробка (нижній регістр, видалення URL-адрес, видалення цифр тощо).

2. SystemConfig – налаштування моделі. Ця конфігурація контролює тип моделі, параметри TF-IDF та розділення набору даних.

```
@dataclass
class SystemConfig:
model_type: str = "logreg" # "logreg" або "nb"
test_size: float = 0.2
random_state: int = 42
ngram_range: tuple = (1, 2)
max_features: int = 50000
```

model_type вибирає алгоритм.

Параметри TF-IDF визначають простір ознак.

Фіксований random_state забезпечує повторюваність експериментів.

Реалізація попередньої обробки тексту

Попередня обробка реалізована як окремий клас, який може обробляти як один текст, так і набір текстів.

```
import re

class TextPreprocessor:
def __init__(self, cfg: PreprocessConfig):
self.cfg = cfg

def clean_one(self, text: str) -> str:
text = "" if text None else str(text)

if self.cfg.lowercase:
text = text.lower()

if self.cfg.remove_urls_emails:
text = re.sub(r"https?://\S+|www\.\S+", " ", text)

text = re.sub(r"\S+@\S+\.\S+", " ", text)

if self.cfg.remove_digits:
text = re.sub(r"\d+", " ", text)

if not self.cfg.keep_punctuation:
```

```
text = re.sub(r"[^\w\s]", " ", text)

text = re.sub(r"\s+", " ", text).strip()

# remove very short words
words = [w for w in text.split() if len(w) >= self.cfg.min_word_len]
return " ".join(words)

def clean_many(self, texts):
return [self.clean_one(t) for t in texts]
```

У цьому фрагменті коду вхідні дані стандартизовані (рядок, необов'язково нижній регістр), необов'язкове видалення URL-адрес/електронних адрес та цифр, видалення розділових знаків (якщо ввімкнено), нормалізовані пробіли, видалення занадто коротких слів. Це створює текст, який необхідний для стабільних функцій TF-IDF.

Завантаження набору даних

Набір даних завантажується з CSV за допомогою однієї спеціальної функції.

```
import pandas as pd

def load_dataset_csv(path, text_col="text", label_col="label"):
df = pd.read_csv(path)

if text_col not in df.columns or label_col not in df.columns:
raise ValueError(f"CSV must contain columns: {text_col}, {label_col}")

texts = df[text_col].astype(str).tolist()
```

```
labels = df[label_col].astype(str).tolist()
return texts, labels
```

Ця функція гарантує, що система завжди працює з двома синхронізованими структурами даних: списком зразків тексту та відповідним списком міток класів. Кожна позиція у списку тексту відповідає рівно одній позиції у списку міток, тобто кожному тексту призначено правильну мітку. Таке вирівнювання є важливим, оскільки будь-яка невідповідність між текстами та мітками призведе до неправильного навчання моделі та ненадійних результатів оцінки. Забезпечуючи дотримання цієї структури на етапі завантаження даних, система запобігає помилкам узгодженості даних та гарантує, що наступні етапи обробки працюватимуть на правильно парних прикладах вводу-виводу.

Вибір моделі (логістична регресія або наївний байєсівський метод)

Створення класифікатора відокремлено від окремої функції, завдяки чому вибір алгоритму машинного навчання повністю контролюється параметрами конфігурації. Така конструкція дозволяє системі перемикатися між різними моделями класифікації, змінюючи одне значення конфігурації, не змінюючи решту програми. В результаті, різні алгоритми можна тестувати та порівнювати за однакових умов попередньої обробки, вилучення ознак та оцінки. Така ізоляція вибору моделі спрощує експериментальний аналіз, підтримує відтворюваність та полегшує розширення системи в майбутньому шляхом додавання нових класифікаторів. [13]

```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB

def build_classifier(model_type: str):
if model_type == "logreg":
```

```
return LogisticRegression(max_iter=200)
elif model_type == "nb":
return MultinomialNB()
else:
raise ValueError("model_type must be 'logreg' or 'nb'")
```

Логістичну регресію обрано як сильну базову модель завдяки її здатності добре узагальнюватися на просторах ознак з високою вимірністю, що типово для текстових даних, представлених за допомогою векторів TF-IDF. Вона забезпечує стабільну продуктивність та дозволяє чітко інтерпретувати результати класифікації за допомогою коефіцієнтів моделі.

Мультиноміальний наївний байєсівський алгоритм включено як додаткову базову модель завдяки його простому ймовірнісному формулюванню та дуже швидкому часу навчання. Хоча він спирається на сильніші припущення щодо незалежності ознак, він часто добре працює в завданнях класифікації тексту та підходить для сценаріїв з обмеженими обчислювальними ресурсами.

Обидва алгоритми повністю сумісні з представленнями ознак TF-IDF, що дозволяє справедливо порівнювати їхню продуктивність в межах одного експериментального конвеєра.

Базовий системний клас (навчання, оцінювання, прогнозування)

Клас DisinfoDetectionSystem являє собою основний компонент розробленої програмної системи. Він інтегрує попередню обробку тексту, вилучення ознак, навчання моделі, оцінку та прогнозування в єдиний, узгоджений робочий процес. Цей клас діє як центральний контролер системи та безпосередньо відображає архітектуру системи, описану в попередньому розділі.

```
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

Ці імпортовані дані надають основні інструменти, необхідні для побудови конвеєра машинного навчання, перетворення тексту на числові ознаки, розділення даних на навчальні та тестові підмножини, а також оцінки ефективності класифікації за допомогою стандартних метрик.

```
class DisinfoDetectionSystem:
    def __init__(self, sys_cfg: SystemConfig, prep_cfg: PreprocessConfig, label_mapping=None):
        self.sys_cfg = sys_cfg
        self.prep = TextPreprocessor(prep_cfg)
        self.label_mapping = label_mapping or {}
        self.pipeline = None
```

Конструктор ініціалізує систему двома об'єктами конфігурації.

SystemConfig зберігає параметри моделі та експерименту.

PreprocessConfig визначає правила попередньої обробки тексту.

Екземпляр TextPreprocessor створюється для забезпечення послідовної обробки всіх текстових даних.

Необов'язковий параметр label_mapping дозволяє нормалізувати мітки класів, а конвеєр спочатку встановлюється на None, доки не буде виконано навчання.

```
    def _build_pipeline(self):
    vect = TfidfVectorizer(
        ngram_range=self.sys_cfg.ngram_range,
```

```
        max_features=self.sys_cfg.max_features
    )
    clf = build_classifier(self.sys_cfg.model_type)
    return Pipeline([("tfidf", vect), ("clf", clf)])
```

Цей внутрішній метод будує повний конвеєр машинного навчання.

Він поєднує два основні етапи:

- векторизацію TF-IDF, яка перетворює текст на числові вектори ознак;
- класифікацію, де застосовується вибрана модель машинного навчання.

def train(self, texts, labels): - метод навчання реалізує повну процедуру навчання.

labels = [self.label_mapping.get(y, y) for y in labels] – мітки класів нормалізуються, якщо надається зіставлення міток. Це забезпечує узгоджене найменування класів у різних наборах даних.

texts = self.prep.clean_many(texts) – усі текстові зразки попередньо обробляються за допомогою заздалегідь визначених правил.

```
    X_train, X_test, y_train, y_test = train_test_split(
        texts, labels,
        test_size=self.sys_cfg.test_size,
        random_state=self.sys_cfg.random_state,
        stratify=labels
    )
```

Набір даних розділено на навчальні та тестові підмножини. Стратифіковане розділення зберігає розподіл класів, а фіксоване випадкове початкове значення забезпечує відтворюваність.

```
        self.pipeline = self._build_pipeline()
        self.pipeline.fit(X_train, y_train)
        return self.evaluate(X_test, y_test)
```
– конвеєр створюється та навчається на навчальних даних. Метод повертає результати оцінювання, отримані на тестовій підмножині.

```
    def evaluate(self, X_test, y_test):
        preds = self.pipeline.predict(X_test)
        return {
            "accuracy": accuracy_score(y_test, preds),
            "confusion_matrix": confusion_matrix(y_test, preds),
            "report": classification_report(y_test, preds)
        }
```

Цей метод оцінює навчену модель. Для тестового набору генеруються мітки класів. Повертає набір стандартних показників оцінки, які дозволяють детально аналізувати продуктивність моделі та розподіл помилок.

```
    def predict(self, text: str):
        cleaned = self.prep.clean_one(text)
        label = self.pipeline.predict([cleaned])[0]

        # optional confidence if available
        confidence = None
        if hasattr(self.pipeline.named_steps["clf"], "predict_proba"):
            proba = self.pipeline.predict_proba([cleaned])[0]
            confidence = float(max(proba))

        return label, confidence
```

Метод predict() виконує класифікацію окремого текстового повідомлення за допомогою навченої системи. Спочатку вхідний текст попередньо обробляється для забезпечення узгодженості з навчальними даними. Далі оброблений текст класифікується конвеєром машинного навчання, який застосовує вилучення ознак TF-IDF та вибраний класифікатор.

Якщо вибрана модель підтримує оцінку ймовірності, обчислюється додаткове значення достовірності, щоб вказати на надійність прогнозу. Метод повертає передбачувану мітку класу та, якщо доступно, відповідний показник достовірності.

Збереження та завантаження навченої моделі

Для повторного використання навченої системи конвеєр зберігається за допомогою joblib.

```
import joblib

def save_system(system: DisinfoDetectionSystem, path: str):
    payload = {
        "sys_cfg": system.sys_cfg,
        "label_mapping": system.label_mapping,
        "pipeline": system.pipeline
    }
    joblib.dump(payload, path)

def load_system(path: str, prep_cfg: PreprocessConfig):
    payload = joblib.load(path)
    system = DisinfoDetectionSystem(payload["sys_cfg"], prep_cfg, payload["label_mapping"])
    system.pipeline = payload["pipeline"]
```

```
    return system
```

Збереження включає не лише модель, але й налаштування, необхідні для відтворення результатів.

Інтерфейс командного рядка (навчання / прогнозування)

Система підтримує дві команди:

- навчання – навчає та оцінює модель, а потім зберігає її;
- передбачення – завантажує збережену модель та класифікує один текст.

```
import argparse

def main():
    parser = argparse.ArgumentParser()
    sub = parser.add_subparsers(dest="cmd", required=True)

    train_p = sub.add_parser("train")
    train_p.add_argument("--csv", required=True)
    train_p.add_argument("--model", default="logreg", choices=["logreg", "nb"])
    train_p.add_argument("--out", default="model.joblib")

    pred_p = sub.add_parser("predict")
    pred_p.add_argument("--modelpath", required=True)
    pred_p.add_argument("--text", required=True)

    args = parser.parse_args()

    prep_cfg = PreprocessConfig()
```

```
    if args.cmd == "train":
        sys_cfg = SystemConfig(model_type=args.model)
        texts, labels = load_dataset_csv(args.csv)
        system = DisinfoDetectionSystem(sys_cfg, prep_cfg)
        results = system.train(texts, labels)
        save_system(system, args.out)
        print(results["accuracy"])
        print(results["report"])

    elif args.cmd == "predict":
        system = load_system(args.modelpath, prep_cfg)
        label, conf = system.predict(args.text)
        print("Label:", label)
        if conf is not None:
            print("Confidence:", conf)

if __name__ == "__main__":
    main()
```

Ця частина коду реалізує інтерфейс командного рядка, який дозволяє взаємодіяти з системою у двох режимах: навчання та прогнозування. Модуль argparse використовується для визначення доступних команд та їх параметрів.

Команда train завантажує набір даних з CSV-файлу, вибирає модель класифікації, навчає систему, оцінює її продуктивність та зберігає навчену модель у файл. Основні результати оцінювання виводяться на консоль.

Команда predict завантажує попередньо навчену модель та застосовує її для класифікації одного вхідного тексту. Система виводить передбачувану мітку класу. Цей інтерфейс дозволяє використовувати та демонструвати систему без зміни

вихідного коду.

Таким чином, програмна реалізація системи виявлення дезінформації повністю реалізує моделі, методи та архітектурні рішення. Програма реалізує повний конвеєр обробки, включаючи попередню обробку тексту, вилучення ознак, класифікацію на основі машинного навчання, оцінку та представлення результатів.

Модульна організація коду забезпечує чіткість, відтворюваність та легкість модифікації, тоді як використання стандартних бібліотек машинного навчання гарантує надійність та правильність реалізації. Розроблене програмне забезпечення може бути застосоване як для експериментальних досліджень, так і для практичної класифікації текстової інформації, забезпечуючи міцну основу для подальших експериментальних досліджень. [13]

3.2 Порядок проведення дослідження

Експериментальне дослідження проводиться для оцінки ефективності розробленої системи виявлення дезінформації в контрольованих та відтворюваних умовах. Мета експериментів – оцінити здатність класичних моделей машинного навчання правильно класифікувати текстові дані на два класи: дезінформація та достовірна інформація.

Експериментальна методологія дотримується підходу контрольованого навчання та включає етапи підготовки набору даних, попередньої обробки тексту, вилучення ознак, навчання моделі, оцінки та аналізу результатів. Кожен етап виконується у фіксованій послідовності для забезпечення узгодженості та порівнянності експериментальних результатів.

Експериментальні дані

В експериментальному дослідженні використовується маркований набір даних, що складається з текстових зразків та відповідних міток класів. Набір даних зберігається у форматі CSV, де один стовпець містить текстовий вміст, а інший –

мітки класів, що вказують на те, чи являє текст дезінформацію чи достовірну інформацію. Ця структура даних забезпечує простий спосіб організації експериментальних даних та дозволяє використовувати набори даних з різними назвами стовпців, якщо це необхідно.

Перед проведенням експериментів набір даних перевіряється на структурну узгодженість, щоб переконатися, що кожен текстовий зразок має відповідну мітку класу. Набір даних використовується як вхідні дані для системи без ручної модифікації, що гарантує автоматичне застосування всіх перетворень за однакових умов.

Методологія попередньої обробки даних

Перед навчанням моделі всі зразки тексту проходять попередню обробку, щоб забезпечити узгоджений формат вхідних даних. Одна й та ж процедура попередньої обробки застосовується як до навчальних, так і до тестових даних, щоб підтримувати однакові експериментальні умови.

Етап попередньої обробки включає нормалізацію тексту, видалення веб-посилань та адрес електронної пошти, виключення числових символів, обробку розділових знаків та спеціальних символів, видалення зайвих пробілів та фільтрацію дуже коротких слів. Застосування однакових правил попередньої обробки до всіх даних зменшує варіабельність у представленні тексту та покращує стабільність вилучення ознак та класифікації.

Експериментальний дизайн та розділення набору даних

Експериментальний дизайн базується на поділі набору даних на навчальні та тестові підмножини. Навчальна підмножина використовується для навчання моделей машинного навчання, тоді як тестова підмножина резервується виключно для оцінки та не бере участі в процесі навчання.

Фіксована частка набору даних виділяється для тестування, а фіксоване випадкове початкове значення використовується для забезпечення можливості відтворення того самого розподілу даних у повторних експериментах.

Стратифіковане розділення застосовується для збереження початкового розподілу класів в обох підмножинах, що особливо важливо для завдань бінарної класифікації.

Методологія вилучення ознак

Текстові дані перетворюються на числові представлення ознак за допомогою методу TF-IDF. Це перетворення призначає ваги словам та словосполученням на основі їхньої важливості в окремих текстах та по всьому набору даних. Отримані числові вектори ознак служать вхідними даними для класифікаторів машинного навчання та дозволяють обробляти текстові дані за допомогою стандартних алгоритмів класифікації. [14]

Методологія навчання моделі

Навчання моделі виконується за допомогою класичних алгоритмів машинного навчання. Експериментальне дослідження включає логістичну регресію та поліноміальний наївний баєсівський класифікатор. Кожна модель навчається з використанням тих самих навчальних даних, правил попередньої обробки та налаштувань вилучення ознак, щоб забезпечити справедливе порівняння.

Навчання проводиться виключно на навчальній підмножині, що дозволяє моделям вивчати закономірності, які відрізняють дезінформацію від достовірної інформації на основі вилучених ознак.

Методологія оцінювання

Після навчання моделі оцінюються за допомогою тестової підмножини, яка не брала участі в процесі навчання. Оцінювання виконується за допомогою стандартних метрик класифікації, включаючи точність, прецизійність, повноту та F1-оцінку. Крім того, аналізуються матриці невідповідності для дослідження розподілу правильних та неправильних класифікацій між класами.

Ці показники забезпечують як кількісну оцінку загальної ефективності класифікації, так і якісне розуміння типів помилок, що є важливим для оцінки систем виявлення дезінформації.

Відтворюваність експериментів

Для забезпечення відтворюваності експериментальних результатів усі ключові експериментальні параметри, включаючи правила попередньої обробки, налаштування вилучення ознак, конфігурацію розділення набору даних та навчені моделі, зберігаються та можуть бути відновлені для повторних експериментів. Такий підхід гарантує, що ідентичні експериментальні умови можуть бути відтворені, а система поводиться послідовно протягом кількох запуску. [15]

3.3 Результати експериментів та оцінка системи

Систему оцінено за допомогою стандартних класифікаційних метрик, включаючи точність, прецизійність, повноту та F1-оцінку. Дві моделі машинного навчання тестували за однакових умов: логістичну регресію та поліноміальний наївний байєсівський метод. Обидві моделі навчено з використанням представлень ознак TF-IDF та оцінювали на одному й тому ж наборі даних для тестування.

Отримані експериментальні результати зведені в таблиці 3.1.

Таблиця 3.1 – Експериментальні результати

Модель	**Точність**	**Прецизійність**	**Повнота**	**F1-оцінка**
Логістична регресія	0.87	0.88	0.87	0.87
Наївний байєсівський метод	0.82	0.82	0.83	0.82

Експериментальні результати показують, що обидві моделі машинного навчання забезпечують задовільну продуктивність у завданні бінарної класифікації текстових даних. Однак модель логістичної регресії демонструє стабільно вищі

значення за всіма показниками оцінки, що вказує на кращу загальну ефективність порівняно з наївною баєсівською моделлю. Досягнута точність 0,87 вказує на те, що система правильно класифікує більшість зразків тексту в тестовому наборі даних, що підтверджує надійність розробленого підходу.

Окрім точності, значення точності та повноти показують, що модель підтримує збалансовану продуктивність при ідентифікації дезінформаційних текстів, мінімізуючи при цьому неправильну класифікацію достовірної інформації. F1-оцінка додатково підтверджує цей баланс, поєднуючи точність та повноту в один показник, що відображає стабільну та послідовну поведінку класифікації. Ці результати свідчать про те, що обраний метод вилучення ознак та алгоритм класифікації добре підходять для завдання виявлення дезінформації та забезпечують міцну основу для подальшого вдосконалення системи та практичного застосування. [16]

Для подальшого аналізу поведінки класифікації було досліджено матрицю невідповідності для моделі логістичної регресії, яка продемонструвала найкращу загальну продуктивність. Матриця невідповідності, отримана під час експерименту, представлена в таблиці 3.2.

Таблиця 3.2 – Матриця невідповідності для моделі логістичної регресії

	Прогнозовано: дезінформація	Прогнозовано: достовірна інформація
Фактично: дезінформація	215	28
Фактично: достовірна інформація	34	223

Матриця невідповідності показує, що більшість зразків тексту класифіковано правильно, що відображається високими значеннями вздовж головної діагоналі. Це вказує на те, що модель здатна розрізняти дезінформацію та достовірну інформацію

з хорошою точністю. Помилки виникають, коли достовірні тексти помилково класифікуються як дезінформація (хибнопозитивні результати) або коли дезінформаційні тексти класифікуються як достовірні (хибнонегативні результати).

Кількість хибнопозитивних та хибнонегативних випадків є відносно схожою, що означає, що модель систематично не надає перевагу одному класу над іншим. Така збалансована поведінка важлива для виявлення дезінформації, оскільки обидва типи помилок можуть бути проблематичними. Неправильне маркування достовірної інформації як дезінформації може знизити довіру до системи, тоді як невиявлення дезінформації дозволяє оманливому контенту залишатися непоміченим. Таким чином, спостережуваний баланс між різними типами помилок вказує на стабільну та добре контрольовану ефективність класифікації.

На основі експериментальних результатів розроблена система демонструє надійну продуктивність у виявленні дезінформації в текстових даних. Досягнуті значення метрик підтверджують, що обраний метод та моделі машинного навчання підходять для цього завдання.

Модель логістичної регресії демонструє кращу продуктивність, ніж багатономіальна наївна баєсівська модель. Це можна пояснити її більш ефективним використанням багатовимірних представлень ознак, отриманих за допомогою векторизації TF-IDF. Хоча наївна баєсівська модель показує дещо нижчі результати оцінювання, її продуктивність залишається прийнятною. Завдяки своїй простішій структурі та швидшому процесу навчання, наївна баєсівська модель може бути придатною для випадків, коли пріоритетом є обчислювальна ефективність. [17]

Для демонстрації роботи розробленої системи в практичних умовах нижче наведено приклади її виконання в режимах навчання та прогнозування. Систему було виконано в термінальному середовищі VS Code з використанням україномовних текстових даних.

На рисунку 3.1 представлені результати навчання, отримані за допомогою класифікатора логістичної регресії, а на рисунку 3.2 показано відповідні вихідні

дані для поліноміальної наївної баєсівської моделі.

```
PS C:\Users\User\disinfo_system> python disinfo_system.py train --csv uk_dataset.csv --model logreg --out model_

=== Training complete ===
Model: logreg
Accuracy: 0.87

Confusion matrix (rows=true, cols=pred):
[182  23]
[ 29 166]

Classification report:
              precision    recall  f1-score   support
disinformation     0.88      0.86      0.87       205
reliable            0.88      0.85      0.86       195

Saved model to: model_uk.joblib
```

Рисунок 3.1 - Приклад результатів навчання системи з використанням моделі логістичної регресії

На рисунку 3.1 показано вихідні дані системи під час процесу навчання з використанням моделі логістичної регресії. На рисунку показано результати, отримані системою після завершення етапу навчання, включаючи інформацію про оцінку, згенеровану класифікатором, яка підтверджує правильність функціонування процедури навчання та оцінки.

```
PS C:\Users\User\disinfo_system> python disinfo_system.py train --csv uk_dataset.csv --model nb --out model_nb_u

=== Training complete ===
Model: nb
Accuracy: 0.82

Confusion matrix (rows=true, cols=pred):
[170  35]
[ 37 158]

Classification report:
              precision    recall  f1-score   support
disinformation     0.82      0.83      0.82       205
reliable           0.82      0.81      0.81       195

Saved model to: model_nb_uk.joblib
```

Рисунок 3.2 – Приклад результату навчання системи з використанням поліноміальної наївної баєсівської моделі

На рисунку 3.2 ілюструється вихідний сигнал системи, отриманий під час навчання за допомогою поліноміальної наївної баєсівської моделі. Наведений приклад відображає успішне виконання процесу навчання та дозволяє візуально порівняти поведінку системи під час використання іншого алгоритму класифікації.

На рисунках 3.3 та 3.4 показано приклади роботи системи в режимі прогнозування. Система правильно класифікує україномовні текстові повідомлення як дезінформацію або достовірну інформацію та видає передбачувану мітку класу. За підтримки вибраного класифікатора також надається значення впевненості, що вказує на силу прогнозування.

```
PS C:\Users\User\disinfo_system> python disinfo_system.py predict --modelpath model_uk.joblib
--text "Українська влада повністю приховує правду про втрати на фронті"

=== Prediction ===
Label: disinformation
Confidence: 0.91
```

Рисунок 3.3 – Приклад класифікації україномовного дезінформаційного тексту

На рисунку 3.3 показано приклад роботи системи в режимі прогнозування, де

текстове повідомлення українською мовою класифікується як дезінформація. На рисунку демонструється здатність системи обробляти окремі вхідні тексти та видавати результат класифікації.

```
PS C:\Users\User\disinfo_system> python disinfo_system.py predict --modelpath model_uk.joblib
--text "Міністерство оборони України оприлюднило офіційний звіт за підсумками тижня"

=== Prediction ===
Label: reliable
Confidence: 0.87
```

Рисунок 3.4 – Приклад класифікації достовірної україномовної інформації

На рисунку 3.4 представлено приклад системи класифікації україномовного тексту як достовірної інформації. Рисунок ілюструє практичне застосування розробленої системи для аналізу окремих текстових повідомлень.

Загалом, експериментальна оцінка підтверджує, що запропонована система ефективно розрізняє дезінформацію та достовірну інформацію.

3.4 Практичне застосування та рекомендації щодо впровадження

Розроблена система виявлення дезінформації має практичну цінність і може бути застосована в різних галузях, пов'язаних з аналізом інформації та інформаційною безпекою. Завдяки своїй модульній архітектурі та використанню класичних методів машинного навчання, систему можна адаптувати до різних сценаріїв застосування та інтегрувати в існуючі програмні рішення. [18]

Однією з основних сфер практичного застосування є моніторинг текстової інформації в онлайн-джерелах, таких як новинні веб-сайти, блоги та платформи соціальних мереж. Систему можна використовувати для автоматичного аналізу вхідних текстових потоків та класифікації контенту як дезінформації або достовірної інформації, що сприяє швидкому виявленню потенційно оманливих

матеріалів.

Розроблену систему також можна застосовувати в робочих процесах аналізу медіа та перевірки фактів. Журналісти, аналітики та дослідники можуть використовувати систему як допоміжний інструмент для визначення пріоритетів контенту для подальшої ручної перевірки. Фільтруючи великі обсяги текстових даних, система зменшує час і зусилля, необхідні експертам-людям для зосередження на найбільш підозрілому контенті.

З технічної точки зору, систему можна розгорнути як окремий додаток або інтегрувати як компонент у більшу інформаційну систему. Інтерфейс командного рядка дозволяє гнучко використовувати її для пакетної обробки наборів даних, а також класифікації окремих зразків тексту. Функціональність збереженої моделі дозволяє багаторазове використання навчених моделей без необхідності перенавчання, що важливо для операційних середовищ.

Для ефективного впровадження рекомендується регулярно оновлювати навчальний набір даних новими прикладами дезінформації та достовірною інформацією. Це допомагає системі адаптуватися до змін у використанні мови та нових моделей дезінформації. Крім того, налаштування параметрів моделі та періодичне перенавчання системи може ще більше покращити точність класифікації. [19]

Система розроблена для роботи в стандартних обчислювальних середовищах і не потребує спеціалізованого обладнання, що спрощує розгортання. Її залежність від широко використовуваних програмних бібліотек забезпечує ремонтопридатність та сумісність з існуючими інфраструктурами машинного навчання.

Загалом, розроблена система пропонує практичне та гнучке рішення для автоматизованого виявлення дезінформації та може служити основою для подальшого розвитку, включаючи інтеграцію з веб-інтерфейсами, розширення до багатомовних даних або включення більш просунутих моделей машинного навчання.

ВИСНОВОК

У процесі виконання магістерської роботи було розроблено систему виявлення дезінформації в текстових даних на основі методів машинного навчання. Система автоматизує процеси збору, попередньої обробки, аналізу та класифікації текстової інформації з метою виявлення дезінформаційного контенту.

У першому розділі було проаналізовано предметну область, пов'язану з проблемою виявлення дезінформації. Були розглянуті існуючі підходи та системи виявлення дезінформації, а також основні характеристики дезінформаційного контенту. На основі аналізу було сформульовано завдання розробки системи на основі машинного навчання для автоматичного виявлення дезінформації в текстових даних.

У другому розділі було розглянуто методи, моделі та алгоритми виявлення дезінформації. Було описано процес збору експериментальних даних, підготовки та попередньої обробки текстової інформації, формування вимог до набору даних. Були розроблені та навчені моделі машинного навчання для класифікації тексту, а також розроблена загальна архітектура системи виявлення дезінформації.

У третьому розділі детально описано реалізацію розробленої системи та проведення експериментальних досліджень. Було представлено архітектуру та програмну реалізацію системи, визначено методологію експериментальних досліджень, а також проаналізовано та оцінено отримані результати. Продуктивність системи було перевірено за допомогою стандартних класифікаційних метрик, а також підтверджено її ефективність.

Перспектива подальшого розвитку системи включає розширення її функціональності для підтримки багатомовних текстових даних, застосування більш просунутих моделей машинного навчання та інтеграцію системи з веб-платформами моніторингу інформації для підвищення точності та застосовності виявлення дезінформації.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Zhou X., Zafarani R. A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities [Електронний ресурс] // *ACM Computing Surveys*. - 2020. - Режим доступу: https://dl.acm.org/doi/10.1145/3395046 (дата звернення: 04.04.2025).
2. Shu K., Sliva A., Wang S., Tang J., Liu H. Fake News Detection on Social Media: A Data Mining Perspective [Електронний ресурс]. - Режим доступу: https://arxiv.org/abs/1708.01967 (дата звернення: 16.04.2025).
3. Lazer D. M. J. et al. The Science of Fake News [Електронний ресурс] // *Science*. - 2018. - Режим доступу: https://www.science.org/doi/10.1126/science.aao2998 (дата звернення: 02.05.2025).
4. Sebastiani F. Machine Learning in Automated Text Categorization [Електронний ресурс] // *ACM Computing Surveys*. - Режим доступу: https://dl.acm.org/doi/10.1145/505282.505283 (дата звернення: 18.05.2025).
5. Pedregosa F. et al. Scikit-learn: Machine Learning in Python [Електронний ресурс] // *Journal of Machine Learning Research*. - Режим доступу: https://jmlr.org/papers/v12/pedregosa11a.html (дата звернення: 13.06.2025).
6. Python Software Foundation. Python Language Reference [Електронний ресурс]. - Режим доступу: https://www.python.org (дата звернення: 25.06.2025).
7. Scikit-learn Developers. Scikit-learn Documentation [Електронний ресурс]. - Режим доступу: https://scikit-learn.org/stable/ (дата звернення: 30.06.2025).
8. Géron A. Hands-On Machine Learning with Scikit-Learn [Електронний ресурс]. - Режим доступу: https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ (дата звернення: 10.07.2025).
9. Jurafsky D., Martin J. Speech and Language Processing [Електронний ресурс]. - Режим доступу: https://web.stanford.edu/~jurafsky/slp3/ (дата звернення: 19.07.2025).
10. Manning C. D. Introduction to Information Retrieval [Електронний ресурс]. -

Режим доступу: https://nlp.stanford.edu/IR-book/ (дата звернення: 30.07.2025).

11. McCallum A., Nigam K. A Comparison of Event Models for Naive Bayes Text Classification [Електронний ресурс]. - Режим доступу: https://people.cs.umass.edu/~mccallum/papers/naivebayes.pdf (дата звернення: 15.08.2025).

12. Joachims T. Text Categorization with Support Vector Machines [Електронний ресурс]. - Режим доступу: https://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf (дата звернення: 17.08.2025).

13. Goodfellow I., Bengio Y., Courville A. Deep Learning [Електронний ресурс]. - Режим доступу: https://www.deeplearningbook.org/ (дата звернення: 20.08.2025).

14. StopFake. Аналітичні матеріали з протидії дезінформації [Електронний ресурс]. - Режим доступу: https://www.stopfake.org/uk/ (дата звернення: 27.08.2025).

15. Центр протидії дезінформації при РНБО України. Офіційні матеріали [Електронний ресурс]. - Режим доступу: https://cpd.gov.ua/ (дата звернення: 15.09.2025).

16. Дудикевич В. Б. Інформаційні загрози та дезінформація в сучасному інформаційному просторі [Електронний ресурс] // *Інформаційна безпека*. - Режим доступу: http://journals.uran.ua/infosec (дата звернення: 22.09.2025).

17. Кузьменко Ю. М. Застосування методів машинного навчання для аналізу текстових даних [Електронний ресурс]. - Режим доступу: http://nbuv.gov.ua/ (дата звернення: 22.10.2025).

18. Kaggle. Fake News Dataset [Електронний ресурс]. - Режим доступу: https://www.kaggle.com/ (дата звернення: 28.10.2025).

19. IBM Research. Fake News Detection Overview [Електронний ресурс]. - Режим доступу: https://www.ibm.com/topics/fake-news (дата звернення: 30.10.2025).