МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри Комп'ютеризованих
систем захисту інформації

_____ Михайло СТЕПАНОВ
«____»_____ 2023 р.

На правах рукопису
УДК 004.056.5:510.22(043.3)

КВАЛІФІКАЦІЙНА РОБОТА
ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

Тема: Метод класифікації деструктивних текстових даних

Виконавець: Олександр БІЛИЙ
Керівник: д.т.н., доцент Людмила ТЕРЕЙКОВСЬКА
Консультант розділу «Охорона навколишнього середовища»: к.т.н., доцент Тетяна ДМИТРУХА
Нормоконтролер: д.т.н., доцент Людмила ТЕРЕЙКОВСЬКА

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: Магістр

Спеціальність: 125 «Кібербезпека»

Освітньо-професійна програма: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ
«__» _____ 2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

здобувача вищої освіти Білого Олександра Олеговича

1. Тема: *Метод класифікації деструктивних текстових даних*
затверджена наказом ректора від «15» вересня 2023 р. № 1814/ст.
2. Термін виконання: з 16.10.2023 р. по 31.12.2023 р.
3. Вихідні дані: розглянути підхід щодо сучасних методів та реалізацій класифікації деструктивних текстових даних; проаналізувати наявні рішення; програмно реалізувати метод класифікації деструктивних текстових даних.
4. Зміст пояснювальної записки: аналіз існуючих рішень в області класифікації деструктивних текстових даних; розробка методу класифікації деструктивних текстових даних; програмна реалізація методу класифікації деструктивних текстових даних.

5.КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

№ з/п	Етапи виконання кваліфікаційної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	16.10.2023	*Виконано*
2.	Аналіз літературних джерел	22.10.2023	*Виконано*
3.	Обґрунтування вибору рішення	25.10.2023	*Виконано*
4.	Збір інформації	31.10.2023	*Виконано*
5.	Аналіз існуючих рішень в області класифікації деструктивних текстових даних	14.11.2023	*Виконано*
6.	Розробка методу класифікації деструктивних текстових даних	25.11.2023	*Виконано*
7.	Програмна реалізація методу класифікації деструктивних текстових даних	07.12.2023	*Виконано*
8.	Огляд завдання сучасного природоохоронного законодавства	12.12.2023	*Виконано*
9.	Перевірка на антиплагіат	13.12.2023	*Виконано*
10.	Оформлення і друк пояснювальної записки	16.12.2023	*Виконано*
11.	Оформлення презентації	18.12.2023	*Виконано*
12.	Отримання рецензій від рецензента	22.12.2023	*Виконано*

6. Консультанти з окремих розділів

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв
Охорона навколишнього середовища	Дмитруха Т.І.		

7. Дата видачі завдання: «16» жовтня 2023 р.

Здобувач вищої освіти Олександр БІЛИЙ

(підпис, дата)

Керівник кваліфікаційної роботи Людмила ТЕРЕЙКОВСЬКА

(підпис, дата)

РЕФЕРАТ

Кваліфікаційна робота на тему: «Метод класифікації деструктивних текстових даних» складається зі вступу, основної частини, що містить 4 розділи, 3 висновки до кожного розділу, загального висновку, 2 додатків та списку використаної літератури. Загальний обсяг роботи – 89 сторінок. Робота містить 13 рисунків, 1 графік, 24 формули та 7 таблиць. Список використаних джерел включає 48 джерел.

Метою кваліфікаційної роботи є програмна реалізація методу класифікації деструктивних текстових даних.

У кваліфікаційній роботі розглянуті питання щодо сучасних методів та реалізацій класифікації деструктивних текстових даних.

Проведені дослідження та програмна реалізація методу базується на сучасних методах машинного навчання, обробки природної мови та штучного інтелекту, наївного баєсівого класифікатору, методу максимальної ентропії, TF-IDF векторизації, Word Embeddings та F-1 міри.

Розроблена метод відносяться до галузі інформаційної безпеки і може бути використаний як для персонального використання так і для використання малим, середнім чи великим бізнесом.

Запропонований метод дозволяє забезпечити швидку, надійну та просту у використанні класифікацію деструктивних текстових даних.

Ключові слова: КЛАСИФІКАЦІЯ, ТЕКСТОВІ ДАНІ, ВРАЗЛИВА ІНФОРМАЦІЯ, БАЄСІВ КЛАСИФІКАТОР, МАШИНЕ НАВЧАННЯ,

ЗМІСТ

ВСТУП 7

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ В ОБЛАСТІ КЛАСИФІКАЦІЇ ДЕСТРУКТИВНИХ ТЕКСТОВИХ ДАНИХ 10

1.1. Стандартний підхід до класифікації деструктивних тестових даних 10

1.2. Проблематика наявних рішень класифікації деструктивних текстових даних 12

1.3. Висновки до розділу 32

РОЗДІЛ 2. РОЗРОБКА МЕТОДУ КЛАСИФІКАЦІЇ ДЕСТРУКТИВНИХ ТЕКСТОВИХ ДАНИХ 34

2.1. Методи та алгоритми сортування текстового матеріалу з загрозливою інформацією 34

2.2. Метод обробки вхідних даних 59

2.2.1. TF-IDF векторизація 59

2.2.2. Word Embeddings 63

2.3. Математичне забезпечення методу класифікації 67

2.4. Висновки до розділу 69

РОЗДІЛ 3. СИСТЕМА КЛАСИФІКАЦІЇ ДЕСТРУКТИВНИХ ТЕКСТОВИХ ДАНИХ 71

3.1. Вибір мови програмування 71

3.2. Архітектура системи 73

3.3. Програмна складова методу 80

3.4. Експериментальні дослідження 83

3.5. Висновки до розділу 86

РОЗДІЛ 4. ЗАВДАННЯ СУЧАСНОГО ПРИРОДООХОРОННОГО ЗАКОНОДАВСТВА 88

ВИСНОВКИ 93

Список використаної літератури 94

ДОДАТОК А 99

ДОДАТОК Б 102

ДОДАТОК В 104

ВСТУП

Актуальність. Зростання впливу інтернету та соціальних мереж призвело до збільшення кількості деструктивних текстових матеріалів, які можуть завдати шкоди індивідам, організаціям та громадським інтересам. Ця шкідлива інформація може вплинути на репутацію, фінансовий стан, а також соціальну, психологічну та інформаційну безпеку. Соціальні медіа стали основним майданчиком обміну інформацією та взаємодії між користувачами. Проте це також стало основним середовищем для поширення деструктивного контенту. Методи класифікації деструктивних текстових даних можуть допомогти соціальним мережам виявляти та блокувати шкідливий контент, забезпечуючи кращу інформаційну безпеку для користувачів. Для багатьох інтернет-платформ інформаційна безпека та захист від деструктивного контенту стали пріоритетними завданнями. Зростання потужності штучного інтелекту (AI) та машинного навчання відкриває нові можливості для розробки більш точних та автоматизованих методів класифікації деструктивних текстових даних. Використання AI може значно полегшити інформаційну безпеку та забезпечити ефективну реакцію на загрози [1].

Усі ці фактори підкреслюють актуальність роботи, спрямованої на розробку та вдосконалення методів класифікації деструктивних текстових даних. Ця робота важлива для забезпечення інформаційної безпеки, збереження довіри та зручності користувачів у цифровому світі і має потенціал позитивно вплинути на суспільство в цілому.

Мета роботи: розробка та дослідження методу класифікації деструктивних текстових даних з метою підвищення інформаційної безпеки в онлайн-середовищі. Робота спрямована на досягнення наступних завдань:

- Аналіз існуючих методів та підходів до класифікації деструктивних текстових даних з метою вибору найбільш підходящого методу для досягнення найкращих результатів.
- Розробка ефективного та точного алгоритму класифікації деструктивних текстових даних, який здатний автоматично визначати та класифікувати шкідливий контент в текстах.
- Програмна реалізація методу класифікації деструктивних текстових даних за використання інноваційних методів машинного навчання, обробки природної мови та штучного інтелекту та експериментальна верифікація отриманих результатів.

Галузь застосування: розроблений метод класифікації деструктивних текстових даних може знайти своє застосування в соціальних мережах, допомагаючи виявляти та блокувати образливі коментарі і забезпечувати безпеку користувачів, а також для медіа-ресурсів та новинних сайтів для фільтрації шкідливого контенту.

Об'єкт дослідження: процеси розпізнавання деструктивних текстових даних в онлайн-середовищі.

Предмет дослідження: метод класифікації деструктивних текстових даних.

Методи дослідження: базуються на використанні мови Phyton для роботи з інноваційними методами машинного навчання, обробки природної мови та штучного інтелекту для розробки методу класифікації деструктивних текстових даних.

Новизна одержуваних результатів: отримав подальший розвиток метод класифікації деструктивних текстових даних, що за рахунок адаптації параметрів, роботи з інноваційними методами машинного навчання, обробки природної мови

та штучного інтелекту дозволяє підвищити ефективність розпізнавання деструктивних текстових даних.

Практична цінність: розроблено інструмент для класифікації деструктивних текстових даних; розроблена методика проведення експерименту з використанням розробленого програмного засобу класифікації деструктивних текстових даних.

Апробація. Основні положення роботи доповідалися та обговорювалися на конференції:

- Міжнародна науково-практична конференція «ЖИВУЧІСТЬ ТА РЕЗИЛЬЄНТНІСТЬ – 2023» (Київ: Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова)

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ В ОБЛАСТІ КЛАСИФІКАЦІЇ ДЕСТРУКТИВНИХ ТЕКСТОВИХ ДАНИХ

1.1. Стандартний підхід до класифікації деструктивних тестових даних

У сучасному інформаційному суспільстві обмін інформацією через текстові дані став невід'ємною частиною нашого повсякденного життя. Однак разом із безмежним потоком корисної інформації, текстові дані також несуть у собі потенційну загрозу у вигляді деструктивних текстів. Деструктивні текстові дані, такі як ненависть, мова ворожнечі, дезінформація та пропаганда, становлять значну загрозу для суспільства. Для ефективного запобігання та протидії поширенню таких даних необхідно мати ефективні методи їхньої класифікації. На сьогоднішній день існує широкий спектр методів класифікації деструктивних текстових даних. Однак ці методи мають ряд проблем, які обмежують їхню ефективність.

Рис. 1. Стандартний алгоритм навчання та роботи програмної класифікації деструктивних текстових даних [2].

Збір даних.

На цьому етапі збирається набор даних, який містить як деструктивні, так і недеструктивні текстові дані. Набір даних повинен бути репрезентативним для популяції деструктивних текстових даних, які можуть зустрічатися в реальному світі. Набір даних для класифікації деструктивних текстових даних може бути зібраний з різних джерел, таких як:

- Соціальні мережі. Деструктивні текстові дані часто зустрічаються в соціальних мережах, таких як Facebook, Twitter і Reddit.
- Новинні статті. Деструктивні текстові дані можуть зустрічатися в новинних статтях, які стосуються політичних конфліктів, соціальних проблем або інших тем, які можуть бути сприйняті як деструктивні.
- Веб-сайти. Деструктивні текстові дані можуть зустрічатися на веб-сайтах, які присвячені деструктивним темам, таким як ненависть, мова ворожнечі та дезінформація.

Маркування даних.

На цьому етапі модель машинного навчання навчається на наборі даних, який був зібраний і позначений на попередньому етапі. Модель машинного навчання вчиться виявляти закономірності, які відрізняють деструктивні текстові дані від недеструктивних. Маркування даних для класифікації деструктивних текстових даних може проводитися вручну або автоматично. При ручному маркуванні даних тексти у наборі даних класифікуються людьми, які є експертами в області деструктивного контенту. Ручне маркування є точним методом, але воно може бути трудомістким і дорогим. При автоматичному маркуванні даних використовуються алгоритми машинного навчання для класифікації текстів у наборі даних. Автоматичне маркування може бути менш точним, ніж ручне маркування, але воно може бути більш ефективним з точки зору витрат.

Навчання моделі.

Навчання моделі машинного навчання для класифікації деструктивних текстових даних може проводитися за допомогою різних алгоритмів, таких як:

- Навчання на прикладах. Цей алгоритм використовує набор даних, який містить як деструктивні, так і недеструктивні текстові дані. Модель машинного навчання вчиться виявляти закономірності, які відрізняють деструктивні текстові дані від недеструктивних.
- Навчання на ознаках. Цей алгоритм використовує набор даних, який містить текстові дані та інформацію про них, таку як ключові слова, граматичні конструкції та семантична структура. Модель машинного навчання вчиться використовувати цю інформацію для класифікації текстових даних.

Використання.

На цьому етапі модель машинного навчання використовується для класифікації нового тексту як деструктивного або недеструктивного.

На етапі експлуатації модель машинного навчання використовується для класифікації нового тексту як деструктивного або недеструктивного. Модель машинного навчання використовує інформацію, на якій вона навчалася, щоб визначити чи містить новий текст деструктивний контент.

1.2. Проблематика наявних рішень класифікації деструктивних текстових даних

Однією із ключових проблем, пов'язаних з класифікацією деструктивних текстових даних, є недостатня точність і надійність методів[3,4]. Точність - це

міра того, наскільки часто метод класифікації правильно визначає тип тексту[3]. Надійність - це міра того, наскільки часто метод класифікації дає однаковий результат при повторному використанні[4].

Ця проблема може мати серйозні наслідки для спроби виявлення та блокування деструктивного контенту, оскільки може спричинити помилки, як у визначенні справжнього негативного контенту, так і у визначенні законного або нешкідливого тексту. В даному контексті, низька точність і надійність методів може призвести до поширення деструктивних текстових даних або, навпаки, до надмірного блокування контенту, що може обмежити свободу слова та доступ до інформації.

Багато існуючих методів класифікації деструктивних текстових даних мають низьку точність і надійність. Це означає, що вони часто помиляються при виявленні та блокуванні деструктивних текстових даних.

Причини:

- Недостатня кількість та якість тренувальних даних для навчання моделей, особливо для рідкісних категорій.
- Неможливість врахувати складний контекст та нюанси мовлення (сарказм, іронію, мовні ігри).
- Залежність результату від обраного алгоритму, що може бути не найбільш ефективним.
- Швидке появу нових тем та форм деструктивної інформації, що ще не відображені в моделях.

Наслідки:

- Висока ймовірність пропуску частини токсичних даних або помилкових флагів.
- Необхідність постійного уточнення та оновлення моделей, що ускладнює процес.

- Втрата довіри користувачів та модераторів до результатів автоматичної фільтрації.
- Ризик поширення деструктивної інформації через помилки алгоритмів.

Наприклад, метод класифікації, який покладається на аналіз ключових слів і фраз, може помилково класифікувати як деструктивний текст, який є жартом або іронією. Або метод класифікації, який покладається на аналіз семантичної структури тексту, може помилково класифікувати як деструктивний текст, який є законним політичним дискурсом.

Необхідно розробляти нові методи класифікації деструктивних текстових даних, які мають більш високу точність і надійність [2,5]. Це допоможе забезпечити ефективне виявлення та блокування деструктивних текстових даних, що є важливою умовою для захисту суспільства від негативних наслідків деструктивного контенту [4].

Ось кілька конкретних рекомендацій щодо підвищення точності і надійності методів класифікації деструктивних текстових даних:

- Використання великих наборів даних для навчання методів класифікації. Це допоможе методам класифікації навчитися розпізнавати більш широкий спектр деструктивних текстових даних.
- Використання передових технологій машинного навчання та штучного інтелекту. Ці технології можуть допомогти методам класифікації виявляти більш тонкі закономірності, які відрізняють деструктивні текстові дані від законних.
- Розробка методів класифікації, які є більш адаптивними до змін. Це допоможе методам класифікації ефективно виявляти нові типи деструктивних текстових даних, які можуть з'являтися з часом.

Нездатність адаптуватися до змін - це ще одна важлива проблема, пов'язана з наявними методами класифікації деструктивних текстових даних.

Деструктивні дані постійно змінюються і розвиваються. Деструктивні користувачі постійно знаходять нові способи маскувати свої повідомлення, щоб уникнути виявлення. Наприклад, вони можуть використовувати кодування, шифрування або інші методи, щоб приховати свій зміст.

Існуючі методи класифікації деструктивних текстових даних, такі як аналіз на основі машинного навчання, можуть бути неефективними для виявлення нових типів деструктивних даних [5]. Це пов'язано з тим, що такі методи схильні до упереджень, пов'язаних із даними, на яких відбувалось навчання моделей [6].Це може призвести до того, що нові типи деструктивних даних будуть поширюватися безперешкодно.

Приклади деструктивних даних, які можуть бути не виявлені існуючими методами класифікації:

- Деструктивні дані, які містять прихований зміст. Деякі деструктивні текстові дані містять прихований зміст, який може бути важко розпізнати. Наприклад, деструктивні текстові дані можуть бути зашифровані або написані за допомогою коду.
- Деструктивні дані, які написані мовами, які не підтримуються існуючими методами класифікації. Багато існуючих методів класифікації деструктивних текстових даних підтримують лише кілька мов. Це може призвести до того, що деструктивні текстові дані, написані іншими мовами, будуть не виявлені.

Необхідно розробляти нові методи класифікації деструктивних текстових даних, які є більш адаптивними до змін. Ці методи повинні бути здатні виявляти нові типи деструктивних даних, які можуть з'являтися з часом.

Висока вартість і трудомісткість - це ще одна проблема, пов'язана з наявними методами класифікації деструктивних текстових даних. Багато існуючих методів класифікації деструктивних текстових даних вимагають

значних витрат і зусиль. Це може ускладнити їхнє впровадження в реальних умовах [5].

Приклади високої вартості і трудомісткості методів класифікації деструктивних текстових даних:

- Використання ручного аналізу. Деякі методи класифікації деструктивних текстових даних вимагають ручного аналізу експертів. Це може бути дуже трудомістким і дорогим процесом.
- Використання великих наборів даних. Багато методів класифікації деструктивних текстових даних вимагають великих наборів даних для навчання. Це може бути дорогим і трудомістким процесом.
- Використання складних алгоритмів. Деякі методи класифікації деструктивних текстових даних використовують складні алгоритми, які можуть бути дорогими для реалізації.

Необхідно розробляти нові методи класифікації деструктивних текстових даних, які є більш доступними. Ці методи повинні бути менш доступними та зрозумілими, щоб їх можна було легко впровадити в реальних умовах [3].

Ось кілька конкретних рекомендацій щодо зниження вартості і трудомісткості методів класифікації деструктивних текстових даних:

- Використання більш простих і швидких алгоритмів на кшталт логістичної регресії замість глибокого навчання для попередньої фільтрації.
- Автоматизований збір тренувальних даних за рахунок передплати на сторонні API або машинного тегування частини даних.
- Розгортання моделей у хмарному середовищі для масштабування та зниження ТО витрат на обслуговування серверів.
- Використання сервісів crowdsourcing та модерації спільнотою для дешевшого отримання зворотного зв'язку та оновлення моделей.

- Розробка штучного інтелекту, здатного самостійно контекстно тлумачити тексти і виявляти ненав'язливі категорії.
- Подальше стандартизування процесів та інтеграція зі загальними платформами моніторингу.

Розробка нових методів класифікації деструктивних текстових даних, які є більш доступними, є важливою науковою задачею, яка має потенціал для значного підвищення ефективності боротьби з деструктивним контентом.

Крім вищезазначених проблем, існують також інші, які слід враховувати при розробці нових методів класифікації деструктивних текстових даних. Наприклад, важливо, щоб методи класифікації були справедливими і недискримінаційними. Це означає, що вони не повинні класифікувати законний контент як деструктивний, а деструктивний контент - як законний [7]. Також важливо, щоб методи класифікації були прозорими. Це означає, що користувачі повинні розуміти, як працюють методи класифікації, і виявляти потенційні проблеми. Розробка нових методів класифікації деструктивних текстових даних є складною задачею. Однак ця задача є важливою, оскільки вона може допомогти захистити суспільство від негативних наслідків деструктивного контенту.

Плагіни - це програмні модулі, які розширюють функціональність основної програми [8]. Вони можуть додавати нові функції, можливості або налаштування. Плагіни використовуються в різних типах програм, включаючи веб-браузери, текстові редактори, графічні редактори та ігри. Як працюють плагіни? Плагіни зазвичай працюють, доповнюючи основну програму новим кодом. Цей код може бути написаний на тому ж мовою програмування, що і основна програма, або на іншій мові. Плагіни зазвичай встановлюються в основну програму через спеціальний інтерфейс. Цей інтерфейс може бути вбудований в основну програму або бути доступним через веб-сайт або інший ресурс.

Великою популярністю користуються доповнення для поштових програм, а саме фільтри спаму, доповнення для тестування електронної пошти за допомогою антивірусу.

Принцип роботи плагінів полягає у тому, що основна програма надає послуги, які може використовувати плагін. Сюди входять розширення, які мають можливість реєструватися в основній програмі, а також протоколи обміну даними з іншими розширеннями [8]. Плагіни залежать від послуг, які надає основний додаток, і в зазвичай не використовуються окремо. На відміну від цього, основний додаток запускає плагін незалежно, даючи можливість кінцевим користувачам динамічно додавати та оновлювати плагіни, не вносячи жодних змін до основної програми.

Тож плагіни чудово підходять під задачі пов’язані з моніторингом веб-сторінок та текстового контенту, задля запобігання накрапляння на загрозливу інформацію або, щонайменше, попередження про її наявність на конкретних ресурсах.

Adult Blocker - це плагін для браузера, який блокує доступ до дорослого контенту. Він використовує алгоритми машинного навчання для виявлення дорослого контенту.

Плагін Adult Blocker працює в два етапи:

1. Виявлення дорослого контенту: Плагін використовує алгоритми машинного навчання для виявлення дорослого контенту в текстах, які відображаються на веб-сторінках. Алгоритми машинного навчання навчаються на наборі даних, який містить приклади дорослого контенту.
2. Блокування дорослого контенту: Коли плагін виявляє дорослий контент, він блокує його. Це означає, що текст буде прихований від користувача.

Плагін Adult Blocker можна налаштувати відповідно до ваших потреб. Наприклад, ви можете вибрати, які типи дорослого контенту будуть блокуватися. Ви також можете вибрати, чи буде відображатися попередження, коли плагін блокує контент.

Принцип роботи плагіна Adult Blocker:

Плагін Adult Blocker використовує два основних типи алгоритмів машинного навчання для виявлення дорослого контенту:

- Алгоритми на основі наборів правил: ці алгоритми використовують набори правил для визначення, чи є текст дорослим. Набори правил складаються з виразів, які шукають у текстах. Наприклад, один із наборів правил може шукати в текстах слова або фрази, які є еротичними або сексуальними.
- Алгоритми машинного навчання: ці алгоритми навчаються на наборі даних, який містить приклади дорослого контенту. Після навчання алгоритми машинного навчання можуть виявляти дорослий контент, навіть якщо він не відповідає наборам правил.

Плагін Adult Blocker використовує комбінацію цих двох типів алгоритмів для підвищення точності виявлення дорослого контенту.

Переваги плагіна Adult Blocker:

- Може допомогти вам захистити ваших дітей від дорослого контенту. Дорослий контент може бути шкідливим для дітей.
- Є простим у використанні. Плагін Adult Blocker не вимагає від вас жодних технічних знань.
- Є доступним для безкоштовного використання.

Недоліки плагіна Adult Blocker:

- Він може іноді блокувати нешкідливий контент.

- Не може захистити вас від всього дорослого контенту. Дорослий контент може бути прихований у формах, які не можуть бути виявлені плагіном Adult Blocker.

Підсумок. Плагін Adult Blocker - це потужний інструмент, який можна використовувати для захисту від дорослого контенту. Він не є ідеальним, але він може допомогти вам захистити ваших дітей від більшості дорослого контенту.

ProCon Latte - це безкоштовний браузерний плагін для виявлення токсичних і деструктивних текстових даних в онлайн-комунікаціях. Він працює на основі моделей машинного навчання, які аналізують контент сторінок та виявляють ознаки насильства, екстремізму, порнографії тощо. Процес класифікації відбувається в режимі реального часу при завантаженні будь-якої веб-сторінки. Якщо детектор виявляє підозрілі фрагменти, користувач отримує попередження у вигляді невеликого pop-up. Також у плагіні передбачені налаштування рівнів чутливості та можливість додавати власні критерії фільтрації.

Принцип роботи плагіна ProCon Latte полягає в такому:

По-перше, він має вбудовані моделі машинного навчання, підготовлені на великому кількості перевірених даних про токсичний і нейтральний контент.

По-друге, при кожному відвідуванні сторінки плагін у режимі реального часу виділяє текстові фрагменти, зображення та інші елементи для аналізу.

По-третє, ці дані подаються на вхід тренованих моделей для класифікації і виділення ознак деструктивного змісту.

По-четверте, якщо ймовірність належності до небезпечної категорії перевищує поріг, користувач попереджається.

П'яте, система постійно оновлює моделі за рахунок залучення нових користувацьких даних та відгуків.

Таким чином досягається точна ідентифікація різних типів деструктивних даних безпосередньо в браузері, що полегшує процес модерації й користувачам, й платформам.

Переваги:

- Відкритий код, що дозволяє спільноті розвивати його.
- Безкоштовний та простий в установці. Не потребує створення облікового запису.
- Швидка перевірка контенту безпосередньо в браузері користувача.
- Постійне оновлення моделей за рахунок користувацьких даних.
- Різні налаштування рівня чутливості детекції.

Недоліки:

- Обмежена мовна підтримка (наразі тільки англійська).
- Точність й ефективність детекції може поступатися комерційним аналогам.
- Відсутні додаткові функції, такі як блокування доступу.
- Немає офіційної техпідтримки, все залежить від спільноти.
- Менш досконала обробка складних контекстів, нюансів мовлення.

Отже, ProCon Latte є корисним, але досить простим браузерним плагіном, спрямованим на виявлення токсичних текстових даних у веб-контенті. Він працює на основі моделей машинного навчання, які дозволяють у реальному часі сканувати зміст відвідуваних сторінок та виділяти підозрілі фрагменти.

Hate Speech Blocker - це браузерне розширення, яке дозволяє автоматично виявляти та блокувати мову ворожнечі в онлайн-контенті.

Воно працює на основі моделей машинного навчання, які проаналізували велику кількість даних, щоб навчитися розпізнавати ознаки ненависті, расизму, сексизму та інших форм токсичного мовлення.

При відвідуванні будь-якої веб-сторінки, Hate Speech Blocker миттєво сканує її текст та коментарі користувачів у пошуках відповідних фрагментів.

Якщо вони виявляються, то автоматично приховуються від читача. Користувач отримує лише інформацію про блокування.

Він працює на таких принципах:

- Машинне навчання - використовуються глибокі нейронні мережі, навчені на великих масивах позначених даних з мовою ненависті. Це дає змогу з високою точністю виявляти приховані прояви агресивної чи деструктивної мови.
- Статистичний аналіз - аналізує частоти слів, їх поєднання та розподіли для виявлення відхилень, характерних для мови ненависті. Доповнює нейронні моделі для підвищення точності.
- Морфологія - розбирає слова на складові та бере до уваги похідні, щоб ідентифікувати спотворені форми.
- Контекст - бере до уваги не лише окремі слова, а й їх зміст у реченнях та діалогах для кращого розуміння інтенцій.
- Багатомовність - працює з декількома мовами, зокрема англійською, іспанською, французькою та іншими.
- Постійне оновлення - моделі перенавчаються онлайн, щоб вчасно реагувати на зміни мови та нові форми мови ненависті.

Переваги застосування Hate Speech Blocker:

- Висока точність виявлення проявів мови ненависті, в тому числі прихованих і спотворених, завдяки сучасним методам машинного навчання.
- Швидка реакція на зміни мовленнєвого середовища та появу нових шкідливих виразів за рахунок постійного оновлення моделей.
- Автоматизація процесу перевірки контенту, що дозволяє модерувати великі потоки даних з мінімальними зусиллями людини.

- Служить профілактиці поширення деструктивних ідей, захищаючи користувачів від негативного впливу.

Однак існують і такі недоліки:

- Потенційна можливість помилкового блокування непроблематичних повідомлень через обмежену семантичну здатність системи.
- Залежність від якості навчальних даних - наявність в них усіх різновидів мови ненависті для узагальнення моделі.
- Неможливість блокування повністю нових типів шкідливого контенту до його додавання до тренувальної вибірки.
- Питання конфіденційності даних, які використовуються для навчання глибоких нейронних мереж системи.

У підсумку, Hate Speech Blocker є корисним інструментом для автоматичного виявлення та блокування проявів мови ненависті в соціальних мережах. Він ґрунтується на сучасних підходах обробки природної мови, зокрема глибокому навчанні та аналізі на статистичному, морфологічному і семантичному рівнях.

Завдяки комплексному застосуванню цих методів система досягає високої точності виявлення різноманітних форм деструктивного контенту. Постійне оновлення моделей дозволяє швидко реагувати на зміни. Це ефективно захищає користувачів і створює більш безпечне середовище онлайн-спільнот.

Висновки відповідно до порівняльного аналізу плагінів для блокування небажаного контенту на основі таблиці 1.1:

- Кожен з проаналізованих плагінів спеціалізується на певному виді небажаної інформації, маючи відповідні методи її виявлення. Це дозволяє ефективно протидіяти різноманітним загрозам в онлайн-середовищі.

- Основними методами детекції є машинне навчання, аналіз текстів/зображень, чорні списки. Їх застосування разом з оновленнями моделей забезпечує високу якість роботи.
- ProCon Latte є найменш прямолінійним, оскільки вимагає тонкого розуміння контексту для прийняття рішень. Інші ж працюють на більш чітких критеріях.
- Різні можливості настроювання дозволяють адаптувати кожен плагін під конкретні сайти та потреби модерації.
- Постійне вдосконалення технологій обробки даних допомагатиме підвищувати ефективність таких інструментів з часом.

Отже, плагіни успішно доповнюють один одного для створення безпечного онлайн-середовища.

Таблиця 1.1

Порівняльний аналіз плагінів для блокування деструктивних текстових матеріалів

Характеристика	**Hate Speech Blocker**	**ProCon Latte**	**Adult Blocker**
Призначення	Виявлення та блокування проявів мови ненависті, образ, залякування	Фільтрація занадто упереджених або радикальних думок	Виявлення та приховування порнографічних зображень, сексуальних текстів
Цільова платформа	Соцмережі, коментарі	Форуми, сайти новин	Будь-які вебсайти, пошукові запити

Метод детекції	Машинне навчання, аналіз тексту	Аналіз сентиментів та аргументів	Розпізнавання образів, чорні списки слів
Оновлення	Регулярні онлайн оновлення	Ручні оновлення розробником	Регулярні оновлення великого списку
Точність	Дуже точний, є ризик помилок	Може пропустити деякі випадки	Висока, але може блокувати не всі випадки
Налаштування	Фільтри і слова налаштовні користувачем	Сила фільтрації налаштовна керівником сайту	Заборонені слова та домени налаштовні користувачем
Мови	Багатомовний, в т.ч. українська	Англійська та європейські	Незалежний від мови, використовує образи/слова

Perspective API є нейронною мережею, розробленою компанією Google, яка призначена для аналізу та оцінки токсичності, насильства та інших небажаних аспектів в інтернет-коментарях.

В основі API лежить модель глибокого навчання, яка навчалася на величезній кількості людськи оцінених коментарів. Під час роботи вона аналізує мовні ознаки тексу на різних рівнях - від окремих слів до смислових конструкцій. На виході API повертає результат у вигляді кількох оцінок: токсичність, насильство, образливість, сексуальність та ін. Вони виражаються у відсотках та дають уявлення про ймовірність належності коментаря до певної категорії.

Сервіс інтегровано в багато веб-сайтів, форумів, соцмереж для фільтрації коментарів та блокування небажаних. Він постійно вдосконалюється завдяки збору даних від користувачів.

Інтерфейс API простий в користуванні - досить надіслати текст і отримати його оцінку. Це робить інструмент ефективним для модерації контенту у реальному часі.

Короткий опис принципу роботи Perspective API:

На вхід подається текст коментаря користувача. Спочатку API його токенізує - розбиває на окремі слова та семантичні одиниці. Потім токени проходять через нійронну мережу, яка була навчена на великій кількості анотованих прикладів. Мережа складається із шарів, кожен з яких екстрагує все більш складні лінгвістичні ознаки. Найбільш глибокі шари комбінують отримані риси у вищий рівень репрезентації, що дозволяє їм "зрозуміти" загальний сенс коментаря. На останньому шарі API обчислює ймовірність його належності до різних токсичних категорій. Це і є підсумковими оцінками, які повертаються користувачу. Така послідовність дає змогу моделі застосовувати глибокі мовні знання для якісного розпізнавання проблематичних аспектів в текстах. Результати постійно вдосконалюються завдяки оновленню мережі новими даними.

Переваги Perspective API:

- Висока точність результатів (понад 90%) завдяки застосуванню глибокого навчання.
- Зручний простий інтерфейс - лише необхідно надіслати текст та отримати відповідь.
- Можливість інтеграції в будь-які веб-сервіси для модерації контенту.
- Постійне оновлення моделі на базі нових даних, що підвищує якість.

Недоліки:

- Залежність від якості існуючого навчального набору. Може не враховувати локальні особливості.
- Обмеження щодо контексту, не враховується мова поза текстом.
- Вузька спеціалізація лише на виявленні деструктивних аспектів, без загального аналізу.
- Залежність від постійного доступу до серверів API, потреба в інтернет-з'єднанні.
- Комерційна модель - є обмеження на обсяги запитів для безкоштовного користування.

У цілому, Perspective API є потужним інструментом для класифікації деструктивних текстів, особливо коментарів та повідомлень в Інтернеті. Застосування нейронних мереж дозволяє досягати високої точності розпізнавання токсичності, насильства та інших аспектів. Інтеграція в веб-сайти, соціальні мережі дає можливість ефективно модерувати контент у реальному часі. Постійна підтримка та оновлення API удосконалює його функціонал. Утім, існують і певні обмеження, зокрема залежність від якісних даних для навчання. Подальший розвиток сфери класифікації деструктивних текстів пов'язаний з удосконаленням підходів машинного навчання та розширенням контекстного аналізу. Perspective API є корисним знаряддям дослідження для теми моєї роботи та аналізу його переваг і недоліків у порівнянні з іншими рішеннями.

Репозиторій PersLab/ToxicCommentClassifier на сервісі GitHub містить відкритий код навчаної моделі для виявлення токсичних коментарів та надає інтерфейс API до неї. Модель побудована на базі пре-тренованої перцептронної мережі BERT і навчалася на наборі даних отоксичнімих твітів. Для цього використовували позначки 6 токсичних категорій, включаючи образу, насильство, непристойність та ін. Після тренування модель зберігається у вигляді

ваг та архітектури нейронної мережі. Для її використання на сайтах та додатках розроблено JSON API.

Він дозволяє надсилати запити на перевірку текстів і отримувати відповідь у вигляді ймовірностей віднесення до кожної токсичної категорії. Реалізовано авторизацію та обмеження за обсягом запитів.

Такий підхід робить можливим використання навчаної моделі без необхідності її локальної інтеграції та компіляції. Це спрощує тестування API для модерації контенту в сервісах.

Принцип роботи моделі та API PersLab/ToxicCommentClassifier полягає в такому:

- Для виявлення токсичності використовується попередньо навчена мовна модель BERT. Це дозволяє ефективно розпізнавати смислові конструкції та контекст в текстах.
- Навчання моделі здійснювалося на великому аннотованому датасеті коментарів з мітками їх віднесення до токсичних категорій. Це дало змогу виділити корисні ознаки.
- Після тренування отримана модель зберігається у вигляді вагових коефіцієнтів всіх шарів нейронної мережі.
- Для роботи з моделлю розроблено JSON API. При надходженні коментаря, API токенізує його і подає послідовність токенів на вхід моделі.
- Модель обчислює винесення до кожної з токсичних категорій, що повертається користувачеві як результат.

Така схема дозволяє використовувати навчений класифікатор для перевірки текстів на токсичність в режимі реального часу шляхом запитів до API. Переваги моделі PersLab/ToxicCommentClassifier:

- Використання потужної мовної моделі BERT забезпечує високу якість розпізнавання.

- Відкритий доступ до коду та моделі дає змогу інтегрувати її в будь-які проекти.
- Наявність простого JSON API спрощує використання моделі без потреби в її локальній підтримці.

Недоліки:

- Залежність від початкового набору даних, може не враховувати особливості конкретних застосувань.
- Робота тільки з текстовим вмістом, поза контекстом.
- Лише англомовна модель, для інших мов потрібна адаптація.
- Не враховує еволюцію мови та слів, які раніше не були токсичними.
- Залежність від стабільності та підтримки API сервісу розробниками.

У підсумку, можна зробити такий висновок. Існує декілька цікавих програмних рішень, які використовують для виявлення та класифікації деструктивних текстових даних. Це дозволяє ефективно модерувати контент у соціальних мережах, форумах та інших онлайн-сервісах.

Перспектив API та модель PersLab/ToxicCommentClassifier є добре навченими інструментами з відкритим доступом, які можуть бути цікавими для аналізу в рамках моєї роботи. Водночас, існують і певні обмеження цих підходів, зокрема щодо контексту та міжнародної переносимості.

Подальший розвиток задачі класифікації деструктивних текстів пов'язаний з удосконаленням методів штучного інтелекту та збором якісних тренувальних даних для різних мов та сфер. Це допоможе створити більш універсальні та надійні рішення.

Було розглянуто одні з найвідоміших та найпопулярніших плагіни для блокування деструктивних текстових матеріалів. Ці плагіни використовують різні алгоритми для виявлення таких матеріалів. Однак вони можуть бути не завжди ефективними, оскільки вони не завжди можуть правильно визначити, чи

є текст деструктивним. Але також варто розглянули інших методів виявлення деструктивних текстових матеріалів. Ці методи використовують різні підходи, для досягнення поставлених цілей.

Бібліотека SpaCy для мовного аналізу в Python є потужним інструментом, що містить цікаві моделі для виявлення деструктивних аспектів в текстах.

Однією з її складових є классифікатори, що навчалися розпізнавати образи, насильство, токсичність та інші небажані аспекти. Для цього використовувались великі масиви анотованих даних.

Після навчання моделі зберігаються у вигляді модулів, які можна імпортувати в код та застосовувати. Для цього текст проходить попередню обробку та токенізацію за допомогою ядра бібліотеки.

Отримана послідовність токенів подається на вхід класифікатора, який повертає ймовірності категорій. Це дає змогу аналізувати будь-які вхідні текстові дані прямо в коді.

SpaCy є цінним інструментарієм для машинного аналізу мови та виявлення ймовірно небезпечного контенту в соціальних мережах, блогах тощо. Подальше вдосконалення моделей збільшить точність розпізнавання.

Принцип роботи класифікаторів деструктивного контенту в бібліотеці SpaCy полягає в такому:

- Для побудови моделей використовуються алгоритми глибокого навчання, такі як CNN або BiLSTM.
- Навчання проводиться на великих аннотованих датасетах, де тексти мітяться присутністю певних категорій.
- Мова класифікатора - це послідовність векторів від SpaCy word vectors, які репрезентують значення слів.
- Вектори проходять крізь шари нейронної мережі, яка виділяє ознаки на різних рівнях абстракції.

- Останній шар обчислює ймовірності віднесення до кожної класу.
- Отримані моделі зберігаються і доступні як Python-модулі та компоненти ядра SpaCy.
- Користувач подає текст на обробку сервісу, який використовує процесинг SpaCy та отримані вектори.
- Вони подаються на класифікатор, який швидко повертає результат аналізу.

Така схема дозволяє ефективно аналізувати тексти безпосередньо в кодовій базі.

Переваги класифікаторів деструктивного контенту з бібліотеки SpaCy:

- Висока якість моделей, навчаних на великих даних.
- Інтеграція в потужне ядро SpaCy, що спрощує підготовку даних.
- Відкритий вихідний код та можливість вдосконалення моделей.
- Наявність Python-інтерфейсу для простого застосування в проектах.

Недоліки:

- Обмеженість до окремих токсичних категорій та англомовних даних.
- Залежність від початкового набору тренувальних даних.
- Потреба в локальному встановленні Python та бібліотеки.
- Не враховує динаміку мови та контекст поза текстом.
- Можливі обмеження обчислювальних ресурсів для великих текстів.

Загалом це корисний інструментарій, але вимагає розвитку підходів для підвищення ефективності.

У підсумку, можна зробити такий висновок. Існує декілька цікавих відкритих рішень, які дають змогу аналізувати тексти на предмет виявлення деструктивних чи шкідливих фрагментів. Це допомагає покращити модерацію контенту в Інтернеті.

Перспектив API, модель PersLab та класифікатори SpaCy є потужними інструментами для цієї мети. Однак, потребують подальшого удосконалення, особливо щодо міжнародної переносимості.

Майбутній розвиток пов'язаний із штучним інтелектом, зокрема застосуванням новітніх методів навчання мовних моделей. Це зробить їх більш семантичними і чутливими до контексту.

1.3. Висновки до розділу

У даному розділі було проведено обширний аналіз деструктивних текстових матеріалів, стандартного підходу до їх класифікації та принципу роботи цього підходу. Відзначено, що деструктивні тексти мають негативну спрямованість та можуть завдати шкоди як індивідуумам, так і суспільству в цілому. Стандартний підхід до класифікації таких текстових даних включає в себе використання алгоритмів машинного навчання, що базуються на навчанні на позитивних і негативних прикладах.

Під час огляду інструментів для блокування деструктивних текстових матеріалів, було виявлено, що існують різні програмні рішення та бібліотеки, призначені для виявлення та фільтрації такого контенту. Деякі з цих інструментів, такі як Hate Speech Blocker, ProCon Latte, Adult Blocker, використовують правила та ключові слова для визначення деструктивного контенту, тоді як інші, наприклад, Perspective API та Репозиторій PersLab/ToxicCommentClassifier, базуються на аналізі тональності та інших лінгвістичних ознак тексту. Бібліотека SpaCy, з іншого боку, надає інструменти для обробки текстової інформації та аналізу.

Завдяки цьому огляду, стало очевидним, що проблема класифікації деструктивних текстових даних є актуальною та важливою. Однак, низька точність і надійність методів їх класифікації може створювати серйозні виклики та наслідки. Необхідно подальше дослідження та розробка більш ефективних методів класифікації, які б враховували мовні та культурні варіації, а також еволюцію мови та способів виразу.

Через постійний розвиток технологій та розширенню інтернет-середовища, робота над забезпеченням точності та надійності методів класифікації деструктивних текстових матеріалів залишається актуальною та перспективною галуззю досліджень. Вдосконалення методів виявлення та блокування деструктивних текстових даних в сучасному інформаційному суспільстві є важливим завданням для забезпечення безпеки та етичного користування інтернетом.

РОЗДІЛ 2. РОЗРОБКА МЕТОДУ КЛАСИФІКАЦІЇ ДЕСТРУКТИВНИХ ТЕКСТОВИХ ДАНИХ

2.1. Методи та алгоритми сортування текстового матеріалу з загрозливою інформацією

У сучасному цифровому світі деструктивні текстові дані становлять серйозну загрозу безпеці окремих осіб та суспільства в цілому. Ефективна класифікація та модерація такого контенту є критично важливою задачею для забезпечення безпечного й стабільного функціонування онлайн-простору [9].

Статистичні методи

Статистичні методи використовують аналіз частоти та розподілу слів у текстах як основу для класифікації. Вони є фундаментальною технікою в обробці природної мови та аналізі тексту, яка спирається на витяг із вмісту так званих ознак типу «торба слів» [10]. Статистичні підходи довели свою ефективність для різноманітних завдань класифікації, включаючи розрізнення стилів письма та настроїв, завдяки їхній здатності виявляти зразки на поверхневому рівні в мові.

При застосуванні до деструктивних текстових даних, таких як мова ненависті, кіберзалякування, пропаганда радикалізації або плани насильства/самоушкодження, статистичні методи шукають явні статистичні аномалії, які можуть вказувати на наявність такого шкідливого чи незаконного вмісту. Незважаючи на відсутність глибшого семантичного розуміння мови, статистичні методи можуть служити першою лінією захисту та раннім індикатором для подальшого аналізу з використанням складніших методів НЛП.

Найпростішим, але найпоширенішим статистичним підходом є модель «мішка слів» (BOW). Тут текст представлений як невпорядкована сукупність його

слів, нехтуючи граматикою та навіть порядком слів, але зберігаючи множинність. Для заданого словника V розміром N кожен документ d характеризується N-вимірним вектором, що містить частоту термінів tf(t,d) кожного слова t у всіх документах у навчальному корпусі [10].

Загальним удосконаленням у порівнянні з використанням необроблених частот термінів є частотно-інверсне частотне зважування термінів (TF-IDF). TF вимірює, як часто термін з'являється в документі, але його покращує IDF, яка враховує, чи є термін поширеним чи рідкісним у всіх документах – рідкісні терміни мають більшу вагу. TF-IDF — це статистичний показник, який використовується для оцінки важливості слова для документа в колекції чи корпусі [11]. Отримується множенням значень TF і IDF.

Замість окремих слів, безперервні послідовності з n слів, відомі як n-грами, також можна використовувати як ознаки. Це фіксує короткі фрази та місцевість термінів, які надають додатковий контекст порівняно з сумкою слів. Загальні значення n коливаються в межах 1-3 грамів. N-грами довели свою ефективність для класифікації тексту на основі стилю та ранніх класифікаторів ненависті.

Завдяки вилученим статистичним характеристикам методи керованого машинного навчання можна використовувати для побудови класифікаційних моделей на основі позначених навчальних даних. Алгоритми, які зазвичай використовуються, включають логістичну регресію, наївну байєсовську систему та машину опорних векторів, які вивчають статистичні шаблони, що відрізняють різні класи документів [12].

Стандартні показники оцінки використовуються для кількісної оцінки ефективності моделей статистичної класифікації, а саме точності, запам'ятовування та оцінки F1. Точність вимірює частку прогнозованих позитивних результатів, які є фактичними позитивними результатами, пригадування розглядає частку фактичних позитивних результатів, правильно

ідентифікованих, а F1 є їх середнім гармонійним значенням, що врівноважує обидва значення. Сама по собі точність може ввести в оману для незбалансованих наборів даних.

Хоча статистичні підходи прості та швидкі, їм бракує семантичного розуміння мови. Моделі можуть відчувати труднощі з новими сленговими термінами або виявленням кваліфікованої обфускації шкідливої мови. Щоб вирішити цю проблему, статистичні ознаки можна комбінувати або збагачувати морфологічними, синтаксичними або семантичними методами. Додаткові функції метаданих, що представляють поведінку, властивості мережі або контекстні підказки, можуть додатково підвищити продуктивність певних програм.

Таким чином, статистичні методи забезпечують просту основу для автоматичної класифікації великих обсягів текстових даних. Їхній аналіз розподілу слів дозволяє виявити поверхневі візерунки, що вказують на різні стилі письма, почуття чи теми. Незважаючи на обмежену семантику, у поєднанні з іншими техніками NLP або додатковими метаданими статистичні моделі надають цінну інформацію, яка покращує виявлення деструктивного, шкідливого або заплутаного вмісту в Інтернеті.

Морфологічний аналіз

Морфологічний аналіз вивчає внутрішню лінгвістичну структуру слів, включаючи префікси, суфікси, основи слів і теги частини мови (POS). У застосуванні до текстів це виходить за межі простого представлення мішків слів і включає додаткову семантику з морфологічного складу слова. Для мов зі складною морфологією, таких як фінська чи турецька, це надає багатший набір функцій порівняно зі статистичними даними. Морфологічні методи знайшли особливе застосування в аналізі змішаного коду або багатомовного контенту, з яким можуть боротися законодавчі підходи [13].

Основною морфологічною процедурою є тегування частини мови, яке присвоює кожному слову граматичні позначки, такі як іменник, дієслово, прикметник. Це контекстуалізує слова за межі простих термінів. Функції на основі POS можуть вказувати на стилі написання, семантику та визначати стилі переслідування чи радикалізації. Підходи на основі правил і машинного навчання, як-от приховані моделі Маркова або умовні випадкові поля, зазвичай застосовуються для створення POS-тегерів.

Створення кореня намагається скоротити відмінювані або похідні слова до їх основи слова, основи чи кореня — частини слова, яка є спільною для всіх його відмінюваних варіантів. Це добре підходить для класифікації, але втрачає деяке значення. Лематизація продовжує морфологічний аналіз, намагаючись звести кожне слово до його леми або канонічної форми, яка є його базовим словом у словнику. Обидва допомагають агрегувати пов’язані варіанти слів у статистичних моделях або моделях машинного навчання.

Такі прояви, як префікси чи суфікси, вбудовані в слова, надають морфологічні ознаки. Для виявлення домагань орієнтовними ознаками були лайливі префікси. Виявлення регулярних комбінацій афіксів із відомих руйнівних списків слів може допомогти виявити неправильно написані або творчі варіанти [14]. Однак лише афіксам не вистачає семантичного контексту повнозначних слів.

Досконаліші схеми, як-от OntoNotes, класифікують слова за синтаксичними та семантичними типами – іменовані об’єкти, події, кількості тощо. Моделі, що використовують це, можуть краще вловлювати нюанси семантики. Але розробка комплексних морфологічних семантичних ресурсів потребує значних людських зусиль, що обмежує широке застосування.

Функції POS, похідні/лематизації та афікси часто поєднуються з набором слів у векторах ознак. Розширені морфологічні моделі виграють від додаткової

контекстуалізації порівняно з підходами, заснованими на чистих термінах. Розмірність залишається майже незмінною [15].

Подібні контрольовані алгоритми навчання як статистичні методи можуть бути застосовані до представлень морфологічних ознак, включаючи SVM, Наївний Байєс, глибоке навчання. Недавня робота застосувала згорточні нейронні мережі до вбудовування символів, що фіксують морфологічні ознаки для класифікації тексту.

Стандартні показники точності/відкликання оцінюють морфологічні підходи. Порівняння з сумкою слів допомагає оцінити внесок морфології. Комбіновані моделі, що використовують як статистичні, так і морфологічні характеристики, часто перевершують окремі методи, демонструючи їх взаємодоповнюючі переваги.

Включення морфологічного аналізу надає системам НЛП більш тонке розуміння словотвору та контекстів за межами чистого розподілу термінів. Ця додаткова семантика сприяє класифікації складної, багатомовної або заплутаної шкідливої мови, де статистичні підходи можуть дати збій. У поєднанні з іншими методами морфологічні методи сприяють надійнішому автоматичному виявленню деструктивного текстового вмісту.

Семантичні методи

У той час як статистичні методи аналізують зовнішні моделі термінів, а морфологія розглядає внутрішню структуру слів, семантичні підходи досліджують контекстне значення та зв’язки між мовними елементами.

Семантичні методи включають концептуальні представлення для виявлення шкідливого чи незаконного вмісту на основі його основної семантики, а не лише відкритих шаблонів слів. Це теоретично дозволяє виявляти перефразовані версії ефективніше, ніж поверхневі форми [16].

Вбудовування слів генерує щільні векторні представлення слів шляхом аналізу їхніх властивостей розподілу у великих корпусах без міток за допомогою мовних моделей нейронної мережі. Слова зі схожими контекстуальними значеннями мають ближчі вектори. Популярні методи включають вбудовування Word2Vec, GloVe та ELMo [17]. Вбудовування забезпечують рівень семантичної спорідненості, якого бракує статистичним і морфологічним характеристикам.

Складні мережі та онтології кодують концептуальні визначення та відносини, такі як синонімія, гіпернімія та меронімія між сутностями. Добре підібрані графіки знань забезпечують формальні семантичні представлення для моделювання. Наприклад, виявлення порушень онтології дозволило виявити антисоціальну поведінку, пов'язавши образливу мову з захищеними групами. Однак для розробки високоякісних семантичних ресурсів потрібні значні людські зусилля.

Обчислення полярності настроїв і оцінок суб'єктивності для термінів і речень використовує додаткову неявну семантику порівняно з Bag-of-Words. Загальнодоступна лексика настроїв виявила переслідування на основі виявлених рівнів агресії. Однак само почуття не гарантує, що повідомлення також буде деструктивним або образливим, якщо його вирвати з контексту.

Нещодавні моделі глибокого навчання, такі як BERT, ELMo та USE, створюють контекстуалізовані вбудовування, що враховують зв'язки між словам. Це фіксує багатозначні контексти на відміну від статичних вбудовувань [18]. Контекстні вбудовування досягли високої ефективності в таких завданнях, як виявлення шкідливих коментарів, де семантика є вирішальною порівняно з Bag-of-Words.

Більшість моделей, заснованих на семантиці, об'єднують додаткові функції зі статистичних, морфологічних джерел і джерел метаданих для покращення представлень. Фактично це означає, що семантика допомагає усунути

неоднозначність контекстів, одночасно доповнюючи функції, зберігаючи покриття над викидами. Продуктивність загалом покращується порівняно з ізоляцією будь-якого окремого методу.

Семантичні підходи зазвичай оцінюються за допомогою стандартних завдань класифікації тексту з використанням показників точності/запам’ятовування на позначених наборах даних. Вони також оцінюються за допомогою якісних прикладів, щоб перевірити, наскільки добре моделюються концептуальні нюанси для складних ситуацій, таких як перевірка фактів.

У той час як семантика покращує розуміння, підходи, які покладаються виключно на значення, не охоплюють незрозумілу мову. Розробка комплексних семантичних ресурсів вимагає величезних зусиль. Можливість інтерпретації знижена порівняно зі статистичними тенденціями. Поєднання семантики з іншими підказками пом’якшує ці недоліки для застосування в реальному світі [19].

Багатомовність залишається ключовою проблемою, оскільки семантичні ресурси повинні бути розроблені для кожної мови або перекладені. Вивчення міжмовних представлень і використання ресурсів різними мовами може допомогти усунути прогалини. Нові методи, що поєднують семантичне моделювання з контекстним обґрунтуванням, можуть краще відображати розуміння на рівні дискурсу, необхідне для виявлення деяких форм шкідливого мовлення.

Включення семантичного аналізу забезпечує концептуальну основу для класифікації мови поза поверхневими шаблонами. Це допомагає виявити перефразований образливий вміст. У поєднанні з іншими функціями семантика посилює надійність системи з часом, оскільки розуміння мови розвивається.

Загалом, семантичні методи роблять значний внесок в автоматичну ідентифікацію деструктивних текстових повідомлень.

Комбінований метод

У той час як статистичні, морфологічні та семантичні методи аналізують мову з різних лінгвістичних точок зору, комбінаторні підходи використовують сильні сторони багатьох методів. Замість того, щоб покладатися на окремі ознаки, комбінаторні моделі об’єднують різноманітні набори даних для представлення текстів через об’єднання лінгвістичних формалізмів. Ця багатогранна характеристика сприяє більш надійній і семантично нюансованій класифікації шкідливих або незаконних повідомлень.

Функції витягуються за допомогою статистичних, морфологічних і семантичних інструментів, а потім об’єднуються в супервектори. Наприклад, поєднання сумки слів із n-грамами, тегами частини мови, вставленням слів, оцінками настроїв і метаданими. Це надійно охоплює різноманітні контекстуальні аспекти, не пропускаючи унікальних ознак, які можна ідентифікувати лише за допомогою певних методів. Розмірність зростає, але й репрезентативна сила [20].

Отримані високовимірні простори ознак піддаються складним нелінійним моделям. Нещодавні дослідження ефективно застосовують глибокі нейронні мережі для вивчення високодискримінаційних спільних комбінацій, особливо згорткових і рекурентних архітектур, які вправно вловлюють довгострокові залежності між різнорідними модальностями.

Послідовні або паралельні ансамблі спочатку застосовують окремі моделі, а потім об’єднують шляхом голосування більшістю або укладання. Статистичні класифікатори використовують поверхневі закономірності, тоді як лінгвістичні особливості усувають неоднозначність контекстів [21]. Об’єднання думок

пом’якшує слабкі сторони будь-якої окремої техніки, щоб забезпечити додаткові переваги через підтвердження.

Спільна оптимізація пов’язаних допоміжних завдань із основною метою класифікації через багатозадачні мережі дозволяє вивчати корельовані представлення, які можна передавати. Наприклад, поєднання ідентифікації з атрибуцією допомагає краще сформулювати характеристики автора, які вказують на потенційну шкоду.

Оцінка за допомогою стандартних показників класифікації тексту порівняно з окремими базовими лініями, щоб кількісно оцінити покращення від інтеграції модальностей. Якісний аналіз визначає, як конкретні комбінації вирішують режими відмови ізольованих систем через усунення неоднозначності або введення глобальних закономірностей у просторі функцій.

Комбінаторні методи виграють від використання кількох внутрішніх властивостей, а не від надмірного використання будь-якої окремої точки зору. Це підвищує стійкість проти зміни концепції з часом, оскільки мова розвивається, зберігаючи при цьому широку семантичну застосовність у різних сферах через обширні багатогранні точки зору [22].

Конвеєри комбінованих доказів виявилися ефективними в промисловості для поміркування в масштабі. Гнучкі архітектури дозволяють доповнювати існуючі рішення шляхом поступового впровадження функцій, коли стають доступними додаткові дані, без повторного навчання з нуля.

Досягнення найбільш семантично інформованого розуміння вимагає ретельного вивчення з різних точок зору. Комбінаторні методи представляють поточний стан техніки, використовуючи лінгвістично обґрунтовані перетини функцій для комплексного моделювання тексту в напрямку надійної автоматизованої класифікації потенційно шкідливого вмісту [23].

Таблиця 2.1

Порівняння методів роботи з текстовими матеріалами

Метод	**Особливості**	**Переваги**	**Недоліки**
Статистичний	Базується на аналізі частоти слів та моделі "мішка слів"	Простий та швидкий; дозволяє виявити поверхневі шаблони	Бракує семантичного розуміння; чутливий до сленгу та обфускації
Морфологічний	Вивчає внутрішню морфологічну структуру слів, включаючи тегування ЧМ	Надає більш детальний контекст; добре для мов зі складною морфологією	Вимагає розробки лінгвістичних ресурсів; обмежена семантика
Семантичний	Досліджує контекстне значення і зв'язки між елементами за допомогою вбудовувань та онтологій	Покращує розуміння завдяки семантиці	Вимагає розробки ресурсів; нижча інтерпретаційна здатність ніж статистика
Комбінований	Поєднує різні лінгвістичні підходи для створення супервекторів ознак	Використовує сильні сторони окремих підходів; досягає найкращих результатів	Складніший за окремі методи; залежить від якості складових

Аналізуючи таблицю 2.1 можна зробити висновки, що комбінований метод забезпечує найкращі результати, оскільки використовує багатогранний та комплементарний аналіз, який дає найповніше розуміння природної мови в порівнянні з будь-яким окремим підходом.

Наївний баєсів класифікатор.

Наївний баєсів класифікатор (Naive Bayes classifier) - це статистичний алгоритм машинного навчання, що використовуєся для класифікації даних на основі теореми Баєса з наївним припущенням про незалежність між ознаками [24]. Цей класифікатор широко використовується в області машинного навчання та аналізу тексту для прогнозування класу або категорії, до якої належить певний об'єкт на основі його ознак.

Принцип роботи наївного баєсівого класифікатора базується на теоремі Баєса, яка визначає ймовірність настання події на основі попередніх умов. Для класифікації вхідного об'єкта за його ознаками використовуються умовні ймовірності [25]. Кожен елемент даних представляється як вектор ознак, і класифікатор обчислює ймовірність того, що цей вектор належить до певного класу.

Одним із ключових припущень наївного баєсівого класифікатора є наївність (наївність) - припущення про те, що всі ознаки (параметри) вважаються незалежними одна від одної у відношенні до класу, до якого належить об'єкт [26]. Це означає, що хоча це припущення не завжди відповідає реальному світу, але воно спрощує обчислення ймовірностей та робить алгоритм швидким і ефективним.

Щоб провести класифікацію за допомогою наївного баєсівого класифікатора, спочатку потрібно побудувати модель, обчислити умовні ймовірності для кожної класової мітки на основі навчального набору даних. Після цього для нових об'єктів класифікатор обчислює ймовірності належності до кожного класу на основі їхніх ознак, а потім визначає клас з найвищою ймовірністю.

Припустимо, документ поділено на декілька класів c_1,...,c^k, C – загальна безліч класів. Сенс моделі полягає є у тому, що імовірність того що документ d

потрапить в клас c, записується як $P(c|d)$:

$$P(c|d) = \frac{P(d|c)\,P(c)}{P(d)} \quad (2.1)$$

$P(d|c)$ – вірогідність зустріти документ d серед усіх документів класу c;

P(c) – безумовна вірогідність зустріти документ класу c в складовій документів;

P(d) – безумовна вірогідність документа наявності d в складовій документів;

Щоб оцінити умовну ймовірність $P(d|c) = P(t_1, t_2, ..., t_n |c)$, де t_k – терм з документа d, n – загальна кількість термів у документі (включаючи повторення), необхідно зробити припущення про незалежність умовних термінів і незалежність термінових позицій.

Іншими словами, природною мовою ми ігноруємо той факт, що поява одного слова часто тісно пов'язана з появою іншого (найімовірніше, що слово множення зустрінеться в одному тексті зі словом рівняння або розв'язок, ніж зі словом здоров'я). Крім того, одне й те саме слово має різну ймовірність зустрітись в різних місцях тексту. Для цих спрощень розглянута модель природної мови називається наївною [27].

Тому ймовірнісна модель забезпечує зручний спосіб прогнозування настання різних подій.

Оскільки метою класифікації є пошук найкращого класу для цього документа, завданням наївної баєсової класифікації є пошук найбільш раціонального класу cm, розрахованого за допомогою рівняння [28]:

$$c_m = \underset{c \epsilon C}{argmax} P(c|d) \quad (2.2)$$

c – клас;

d – документ;

argmax – елемент, на якому досягається максимум.

Значення цієї ймовірності неможливо розрахувати безпосередньо. Це тому, що навчальний масив повинен містити всі (або майже всі) можливі комбінації класів і документів. Проте, при використанні формули Баєса, можна переписати вираз для $P(d|c)$ таким чином:

$$c_m = \underset{c\epsilon C}{\operatorname{argmax}} \frac{P(d|c)P(c)}{P(d)} = \underset{c\epsilon C}{\operatorname{argmax}} \mathrm{P}(d|c)P(c) \qquad (2.3)$$

$P(d|c) -$ імовірність зустріти документ d серед документів класу c

P(c) – імовірність того, що зустрінеться клас c, незалежно від розглянутого документа

Знаменник $P(d)$ випущено, тому що він не залежить від c і, як наслідок, не впливає на знаходження максимуму.

Використовуючи навчальну множину, імовірність $P(c)$ можна визначити за формулою:

$$P(c) = Nc/N \qquad (2.4)$$

Nc – кількість документів з навчальної множини у класі c;

N – загальна кількість документів у навчальній множині.

Використовуючи правило множення ймовірностей незалежних подій [45], можна записати наступну формулу:

$$P(d|c) = P(t_1, t_2, \dots, t_n|c) = P(t_1|c)P(t_2|c) \dots P(t_n|c) = \prod_{k=1}^{n} P(t_k|c) \quad (2.5)$$

Оцінка імовірності $P(t|c)$ за допомогою навчальної множини буде розраховуватись за допомогою формули:

$$P(t|c) = \frac{T_{ct}}{T_c} \quad (2.6)$$

T_c – загальна кількість темінів в документах класу с;

T_{tc} – кількість входження терміна t до всіх документів класу с на будь яких позиціях. При підрахуванні всі повторні входження враховуються.

Потім, коли класифікатор навчений, тобто величини $P(t|c)$ та P(c) знайдені, можна за допомогою співвідношення відшукати клас документу:

$$c_m = \underset{c\epsilon C}{\operatorname{argmax}} \mathrm{P}(d|c)P(c) = \underset{c\epsilon C}{\operatorname{argmax}} P(c) \prod_{k=1}^{n} P(t_k|c) \quad (2.7)$$

На практиці зазвичай замість добутків використовуються логарифми, щоб запобігти переповненню останнього виразу через безліч дрібних факторів. Логарифм є монотонною функцією збільшення, тому логарифм не впливає на визначення максимального значення. Тому більшість реалізацій використовують таку формулу:

$$c_m = \underset{c\epsilon C}{\operatorname{argmax}} [logP(c) + \sum_{k=1}^{n} logP(t_k|c)] \quad (2.8)$$

Ця формула має просте пояснення. Шанси класифікувати документ, часто зустрічається класом вище, і доданок $\log P(c)$ вносить в загальну суму відповідний внесок. Величини $\log P(t_k|c)$ тим більше, ніж важливіше терм t для ідентифікації класу c, і, відповідно, тим вагомішим їх внесок в загальну суму.

Переваги наївного баєсівого класифікатора включають простоту реалізації, ефективність на великих об'ємах даних та досить хорошу роботу у випадках, коли наївне припущення про незалежність ознак відносно класу виправдане. Однак цей класифікатор може давати менш точні результати у випадку, коли ознаки сильно залежать одна від одної або коли у наборі даних присутня велика кількість шуму. Загалом, наївний баєсів класифікатор залишається важливим інструментом у сфері класифікації даних, зокрема в

аналізі тексту, фільтрації спаму, рекомендаційних системах та інших областях, де важлива точність і швидкодія.

Багаточленний наївний баєсів класифікатор.

Багаточленний наївний баєсів класифікатор (Multinomial Naive Bayes classifier) - це варіація наївного баєсівого класифікатора, який спеціалізується на обробці дискретних ознак та зазвичай використовується в задачах аналізу тексту, зокрема в класифікації тексту [29].

Цей класифікатор особливо корисний для задач, де ознаки описуються як кількість здійснених подій або кількість входжень певних слів у тексті. Наприклад, в аналізі тексту можна представити документ як вектор, де кожен елемент відповідає кількості входжень певного слова або терміну у тексті. Багаточленний наївний баєсів класифікатор враховує такі вектори ознак і використовує ймовірнісний підхід для класифікації текстів.

Головною відмінністю багаточленного наївного баєсівого класифікатора є використання розподілу Мультиноміального (Multinomial distribution) для моделювання умовних ймовірностей ознак в даних. Цей класифікатор припускає, що ознаки (слова, терміни) у тексті представлені як розподіл кількостей, а не як просто наявність або відсутність [29].

Одним з основних застосувань багаточленного наївного баєсівого класифікатора є аналіз тексту для класифікації документів на основі входження слов або термінів у них. Наприклад, в задачах фільтрації спаму або категоризації текстів за темами (тематична класифікація), цей класифікатор може бути дуже ефективним.

Хоча багаточленний наївний баєсів класифікатор вважається ефективним у багатьох випадках, він також має свої обмеження. Наприклад, він не враховує порядок слів у тексті та зв'язки між ними, що може призвести до втрати інформації про семантичні зв'язки. Крім того, якщо в наборі даних присутній шум

або деякі ознаки мають великий обсяг, це також може вплинути на точність класифікації [30].

Узагальнюючи, багаточленний наївний баєсів класифікатор є потужним інструментом для класифікації текстових даних, зокрема у випадках, коли розглядаються кількісні ознаки, такі як частота входження слів у текст. Його ефективність та використання залежать від конкретної задачі та властивостей даних, на яких він застосовується.

Гаусів наївний баєсів класифікатор.

Цей тип класифікатору застосовується тоді, коли ознаки належності є безперервними та не є дискретними. Тоді має місце припущення, що ці ознаки відбираються з нормального розподілу (розподіл Гауса рис. 2.1).

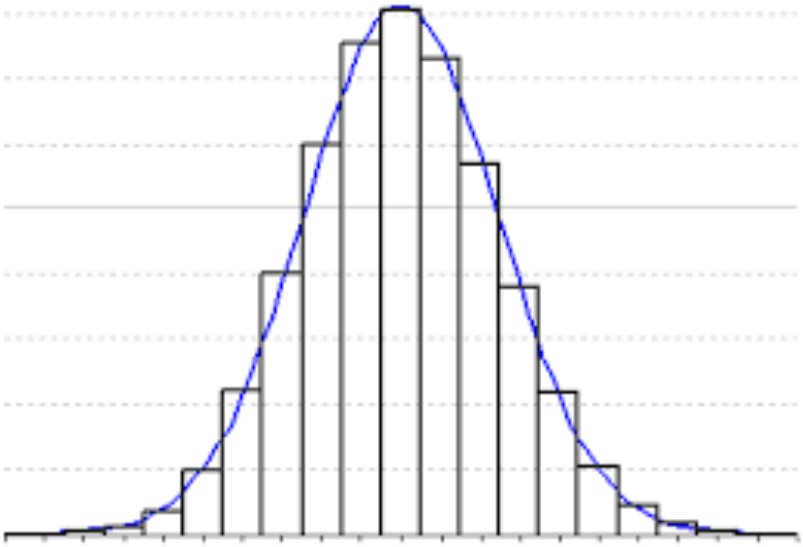


Рис 2.1. Нормальний розподіл.

Формула умовної імовірності має наступний вигляд:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i-\mu_y)^2}{2\pi\sigma_y^2}\right) \quad (2.9)$$

y – вектор міток;

x – вектор даних;

σ – середньоквадратичне відхилення;

μ – мода та медіана розподілу, математичне очікування.

Цей класифікатор вважає, що значення кожної ознаки для кожного класу формує гаусів розподіл і використовує цей розподіл для прогнозування класу нового зразка. Основна ідея полягає в тому, що для кожного класу використовується статистика, така як середнє значення і дисперсія (або коваріація у випадку багатовимірного вектора ознак), щоб описати гаусів розподіл цих ознак для даного класу [31].

Цей метод може бути ефективним для задач класифікації, особливо коли дані відповідають гаусівому розподілу, проте наївний баєсів класифікатор, включаючи гаусів наївний баєсів класифікатор, має свої обмеження. Наприклад, він передбачає незалежність між ознаками, що може бути нереалістичним для деяких даних. Також, якщо дані не відповідають гаусівому розподілу, цей класифікатор може працювати менш ефективно.

Як підсумок, можна зазначити, що наївний баєсів класифікатор найчастіше використовується саме для фільтрації текстовий матеріалів, з метою фільтрації спаму, розбиття тексту/документів на відповідні категорії, поліпшення системи рекомендацій тощо. Ці методи є ефективними, простими та швидкими у вирішенні поставлених задач, але основним недоліком є той факт, що для якісного функціонування ознаки повинні бути незалежними, але у більшості випадків досягнути такого результату неможливо, що має безпосередній вплив на роботу класифікатору.

Недоліком баєсівого класифікатору є те, що для його його коректної роботи необхідно мати вибірку, яка вміщає в себе змінні всіх можливих комбінацій, через що розмір вибірки експоненціально збільшується із збільшенням числа змінних.

Метод максимальної ентропії.

Метод максимальної ентропії (Maximum Entropy method або MaxEnt) - це статистичний метод, який використовується в машинному навчанні та

статистичному моделюванні для розробки ймовірнісних моделей, коли доступні обмежені дані та обмежена апріорна інформація про систему.

Головна ідея методу максимальної ентропії полягає в тому, щоб побудувати ймовірнісну модель на основі найбільш рівномірного розподілу, що можливий при заданих обмеженнях або обмеженій інформації. Зокрема, метод максимальної ентропії шукає ймовірнісну функцію, яка максимізує ентропію системи за обмеженнями, які враховуються у вигляді середніх значень відомих функцій (ознак) у відношенні до цієї функції [32].

У багатьох випадках, коли доступні обмежені або неповні дані про відношення між ознаками та результатом, метод максимальної ентропії може стати корисним для побудови ймовірнісних моделей без введення явних апріорних припущень. Це особливо важливо у випадках, коли інші методи стикаються з недостатністю даних або коли точні знання про структуру системи обмежені.

Метод максимальної ентропії застосовується у різних областях, включаючи обробку природної мови (наприклад, у задачах машинного перекладу, розпізнавання мови, аналізу текстів), біологію, екологію, фізику та інші галузі. Однією з особливостей методу максимальної ентропії є те, що він може враховувати різні види ознак та їх взаємодії, дозволяючи побудувати складні моделі, які враховують різноманітні аспекти вхідних даних. Відношення імовірності елементу даних x до мітки y визначається таким чином [32]:

$$P(y|x) = \frac{1}{Z(x)} \exp\left(\sum_{i=1}^{m} w_i f_i(x, y)\right) \qquad (2.10)$$

w_i – вектор ваг;

f_i – функція залежності, яка враховує взаємозв’язок між міткою і даними;

$\sum_{i=1}^{m} w_i f_i(x, y) -$ підсумування усіх результатів функцій залежностей

x – вектор даних;

y – вектор міток;

m – кількість елементів;

Z(x) – функція, яка допомагає нормалізувати імовірність, та має такий вигляд:

$$Z(x) = \sum_y \exp\left(\sum_{i=1}^{m} w_i f_i(x, y)\right) \quad (2.11)$$

Модель максимальної ентропії має такий ж самий підхід, як і логарифмічно – лінійна модель, проте вона не враховує послідовність вхідного вектору даних.

Незважаючи на свою потужність, метод максимальної ентропії також має свої обмеження, зокрема, вимагає достатньої кількості даних для навчання та може бути витратним з обчислювальної точки зору в разі великих обсягів даних.

У підсумку, метод максимальної ентропії є важливим інструментом для моделювання ймовірностей у випадках обмежених даних або неповної інформації про систему, дозволяючи побудувати ефективні ймовірнісні моделі без значних апріорних припущень.

Метод латентно-семантичного аналізу.

Метод латентно-семантичного аналізу (Latent Semantic Analysis, LSA) - це техніка обробки природної мови та метод аналізу текстів, спрямований на розуміння семантики слів та документів шляхом розкладання матриці термін-на-документ у просторі меншої розмірності [33].

Основна мета LSA полягає у виявленні і використанні прихованої (латентної) структури в текстових даних. Цей метод ґрунтується на матричній факторизації, де матриця термін-на-документ, що представляє вхідні дані (де

рядки - це слова або терміни, стовпці - документи), зменшується до меншої розмірності шляхом зниження розмірності простору інформації.

За допомогою методу сингулярного розкладання (Singular Value Decomposition, SVD), LSA витягує латентні (приховані) семантичні зв'язки між словами та документами [34]. В результаті цього процесу кожен документ та кожне слово представлені у новому просторі з меншою розмірністю, де семантично близькі слова чи документи розташовані близько одне до одного.

LSA може бути використаний для декількох завдань аналізу текстів: Пошук інформації та рекомендацій:

LSA може здійснювати пошук та рекомендації на основі семантичних зв'язків між словами та документами. Наприклад, він може рекомендувати подібні статті або документи на основі семантичної подібності.

Кластеризація документів: LSA може групувати семантично схожі документи у кластери або тематичні групи.

Пошук синонімів та антонімів: Він дозволяє визначати слова, які мають семантичну близькість до вихідного слова, що може бути корисним у розумінні семантики тексту. Модель представлення тексту, яка використовується у латентно-семантичному аналізу у своїй більшості схожа на те, як сприймає текст людина.

Приклад – метод LSA дає можливість визначити текст на відповідність тій чи іншій темі.

LSA використовує терм-документу матрицю в якості вихідної інформації.

Терм-документ матриця – це така математична матриця, яка дає опис частоті термів, що зустрічаються в колекції документів.

Рядки мають відповідність документам у колекції, а стовпці, у свою чергу, відповідають термінам. До даної матриці застосовується сингулярне розкладання.

Сингулярне розкладання – це така математична операція, яка займається розкладанням матриці на три складові. Сингулярне розкладання представляється у вигляді даної формули:

$$A = USV^T \tag{2.12}$$

A – вихідна матриця;

S – діагональна матриця, на діагоналі якої значення називають сингулярними коефіцієнтами матриці;

U і V^T – ортогональні матриці;

Сингулярне розкладання дозволяє виділити складові вихідної матриці, що є ключовими. Задум латентно-семантичного аналізу є у тому, що у якості матриці використовується терм-документна матриця, що містить у собі виключно перші лінійно незалежні компоненти, а також вона відображає основну структуру різних залежностей, які присутні у вихідній матриці. Вагові функції термів використовуються для визначення структури залежності .

Однак, LSA має свої обмеження, включаючи проблеми з урахуванням контексту та врахуванням синонімів, полісемії та амбігвітності. Вибір оптимальної розмірності простору LSA також може бути викликаний труднощами.

Незважаючи на це, метод латентно-семантичного аналізу є корисним інструментом для аналізу текстів та вирішення завдань обробки природної мови, допомагаючи у виявленні семантичних зв'язків у текстових даних.

Метод умовного випадкового поля.

Метод умовного випадкового поля (Conditional Random Fields, CRF) є популярним методом у машинному навчанні для моделювання залежностей та взаємозв'язків у послідовних даних, зокрема у задачах обробки природної мови (Natural Language Processing, NLP), розпізнавання об'єктів у зображеннях, сегментації послідовностей і багатьох інших [35].

CRF є моделлю з учителем, яка використовується для маркування послідовностей даних з урахуванням контексту. Основна ідея полягає в тому, щоб знайти найкращу послідовність міток або категорій для кожного елемента в послідовності, враховуючи контекст та зв'язки між сусідніми елементами.

Головна відмінність CRF від інших моделей, таких як наївні Баєсові класифікатори чи моделі Маркова, полягає у врахуванні не тільки поточного стану прийняття рішення, а й усього контексту [35]. Вона моделює умовній розподіл ймовірностей міток, враховуючи взаємозв'язки між всіма елементами послідовності та їхніми ознаками.

CRF зазвичай використовується у випадках, коли потрібно врахувати складні залежності між різними частинами даних. Наприклад, в задачах розпізнавання іменованих сутностей в тексті (NER), CRF може враховувати контекст та залежності між словами для точнішого визначення меж іменованих сутностей (наприклад, особи, місця, організації) у текстових даних [35].

На відміну від методу максимальної ентропії, метод умовного випадкового поля вирішує проблему з тим, що перевага надається елементам з найменшою кількістю переходів (label bias problem), це досягається шляхом глобального нормалізатору, також розглядається уся послідовність міток, а не тільки одна [36].

$$P(y|x) = \frac{1}{Z(x)} \exp\left(\sum_{j=1}^{n} \sum_{i=1}^{m} w_i f_i(y_{j-1}, y_j, x, j)\right) \qquad (2.13)$$

f_i – функція залежності, яка враховує взаємозв’язки між міткою і даними.

w_i – вектор ваг;

x – вектор даних;

y – вектор міток;

n – кількість елементів;

m – кількість міток;

$\sum_{i=1}^{m} w_i f_i(y_{j-1}, y_j, x, j)$ – підсумування усіх результатів функцій залежностей.

Z(x) – це глобальний нормалізатор, який враховує суму усіх можливих. послідовностей мітки y та знаходиться за формулою:

$$Z(x) = \sum_{y \in Y} \exp\left(\sum_{j=1}^{n} \sum_{i=1}^{m} w_i f_i(y_{j-1}, y_j, x, j)\right) \qquad (2.14)$$

f_i – функція залежності, яка враховує взаємозв’язки між міткою і даними;

$\sum_{i=1}^{m} w_i f_i(y_{j-1}, y_j, x, j)$ – підсумування усіх результатів функцій залежностей;

Переваги використання CRF включають можливість враховувати складні залежності між елементами послідовності, гнучкість у роботі з різноманітними типами ознак та відсутність припущень про незалежність ознак.

Однак, недоліками CRF може бути складність налаштування параметрів моделі, потреба у великій кількості даних для ефективного навчання, а також обчислювальна складність при роботі з великими обсягами даних.

У підсумку, метод умовного випадкового поля (CRF) є потужним інструментом для моделювання залежностей у послідовних даних та знаходження оптимальних маркувань враховуючи контекст, що робить його

корисним у різних областях, включаючи обробку природної мови, комп'ютерне зорове сприйняття, біоінформатику та інші.

Метод опорних векторів.

Метод опорних векторів (Support Vector Machine, SVM) є потужним алгоритмом у машинному навчанні, використовуваним для задач класифікації та регресії. Цей метод зазвичай використовується для роботи з наборами даних, які мають можливість лінійної чи не лінійної роздільної поверхні між класами [37].

Основна ідея SVM полягає у пошуку оптимальної розділної границі, що максимізує відстань між класами. У випадку класифікації з двома класами, це означає знаходження гіперплощини, яка найкращим чином розділяє два класи та має максимальну відстань (маржу) до найближчих точок кожного класу, відомих як опорні вектори.

SVM може використовувати різні функції ядра (kernel functions), такі як лінійна, поліноміальна, радіальна базисна функція (RBF), для вирішення не лінійних проблем розділення даних [37]. Ці функції ядра дозволяють SVM працювати у вищих просторових вимірах, тобто використовувати не лінійні границі розділення між класами.

Для певного набору навчальних вибірок, кожна з яких позначена як належить до однієї з двох категорій, навчальний алгоритм ОВМ будує модель, яка призначає новий зразок до однієї з категорій. Модель ОВМ представляє зразок як точку в просторі та відображає окремі категорії вибірки, розділені найширшим чистим розривом. Нові вибірки потім відображаються в тому самому просторі, і вибірки, які належать до категорії, прогнозуються на основі того, в яку сторону розриву вони потрапляють [38].

Маємо певний набір з n точок, що виглядають наступним чином: $\overrightarrow{x_1}, y_1$ ... $\overrightarrow{x_n}, y_n$, де y набуває значення 1 або -1, і демонструє клас, належним до якого є точка $\overrightarrow{x_i}$, що є p-вимірним дійсним вектором. Задача полягає у тому аби

знайти максимально розділову гіперплощину, що відділяє групу точок $\overrightarrow{x_i}$ для яких $y_i = 1$ та $y_i = -1$, і визначається у такий спосіб, що відстань між тією гіперплощиною та найближчою точкою $\overrightarrow{x_i}$ з кожної з груп є максимальною.

Будь-яка гіперплощина може містити форму запису, як множина точок $\overrightarrow{x_i}$, що дають можливість $\vec{w} * x - b = 0$, де у свою чергу $\vec{w}$ є вектором нормалі до даної гіперплощини.

Якщо дані, що використовуються для тренування є лінійно роздільними, то можна обрати дві паралельні гіперплощини, що розділяють два класи даних у такий спосіб, за якого відстань між ними є максимально великою. Область, що є обмежена цими двома гіперплощинами, називається «розділенням», а максимально розділова гіперплощина є гіперплощиною, що знаходиться посередині між цими двома. Такі гіперплощини можуть бути описані рівняннями $\vec{w} * x - b = 1$ та $\vec{w} * x - b = -1$.

Це все можна сформувати у задачу оптимізації, у якій потрібно мінімізувати $||\vec{w}||$ за умови $y_i(\vec{w} * \vec{x}_i - b) \geq 1$ для i = 1, … n. Тобто, $\vec{w}$ та b, що задовольняють задачу, визначають класифікатор $x \rightarrow sgn(\vec{w} * x - b)$, а максимальна розділова гіперплощина цілком визначається $\overrightarrow{x_i}$, які знаходяться у максимальній близості до неї. Ці $\overrightarrow{x_i}$ називають опорними векторами [38].

Метод опорних векторів застосовується у багатьох областях, включаючи класифікацію тексту, визначення об'єктів у зображеннях, аналіз даних у біології, медицині, фінансах та інших сферах.

У підсумку, SVM є потужним методом у машинному навчанні, який зазвичай демонструє високу точність та ефективність у вирішенні різноманітних задач класифікації та регресії, зокрема там, де є чітко визначена роздільна границя між класами.

У контексті розробки програмного модулю для сортування загрозливої текстової інформації, комбінований метод, що використовується разом з наївним баєсівим класифікатором та методом максимальної ентропії, може виявитися найбільш ефективним. Це дозволить поєднати переваги обох підходів, зберігаючи швидкість та ефективність наївного баєсівого класифікатора та компенсуючи його обмеження завдяки більшому врахуванню семантичної інформації за допомогою методу максимальної ентропії. Комбінація цих методів із комбінованими підходами дозволить уникнути обмежень наївного баєсівого класифікатора, забезпечуючи більш точну та повну аналітику текстів у програмному модулі для класифікації загрозливої текстової інформації.

2.2. Метод обробки вхідних даних

Для подальшої ралізації методу класифікації деструктивних текстових даних розглянемо використання TF-IDF векторизації та Word Embeddings, як ключових методів обробки текстових даних у контексті класифікації. Дані методи, сполучені разом, створюють багатоаспектний підхід до аналізу текстових даних, який дозволяє збагатити метод класифікації за рахунок різних характеристик та особливостей текстів. Такий підхід допомагає покращити здатність методу враховувати інформацію з різних аспектів текстів, що в свою чергу призводить до підвищення її точності та надійності у класифікації.

2.2.1. TF-IDF векторизація

TF-IDF (Term Frequency-Inverse Document Frequency) є одним із ключових методів векторизації тексту, що використовується для представлення слів у

числовому вигляді. Цей метод враховує частоту використання слова у конкретному документі (TF) та зважує її на частоту використання слова у всьому корпусі документів (IDF). Таким чином, TF-IDF дозволяє виділяти важливі слова для кожного документу на основі їхньої унікальності та значущості у контексті всього корпусу текстів [11].

Формула TF для слова у документі [39]:

$$TF(t,d) = \frac{\text{кількість разів,коли слово } t \text{ зустрічається в документі } d}{\text{загальна кількість слів у документі } d} \quad (2.15)$$

Це значення відображає, яка частка тексту складається з конкретного слова.

IDF— відображає, наскільки важливе слово є в межах усієї колекції документів, враховуючи його наявність у всьому корпусі текстів.

Формула IDF для слова у колекції документів [39]:

$$IDF(t) = \log\left(\frac{\text{кількість всіх документів у корпусі}}{\text{кількість документів,які містять слово } t+1}\right) \quad (2.16)$$

Тут +1 додається для уникнення ділення на нуль, а log допомагає зменшити вагу слів, які зустрічаються дуже часто у всій колекції.

TF-IDF обчислення:

TF-IDF для слова у документі обчислюється як добуток значень TF та IDF для цього слова [39]:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t) \quad (2.17)$$

Це значення показує важливість слова для конкретного документу, враховуючи його частоту в цьому документі та його унікальність у всій колекції.

Після обчислення TF-IDF для всіх слів у всіх документах, ми отримуємо числові вектори, що представляють кожен документ у корпусі текстів. Ці вектори можуть бути використані для подальшої обробки алгоритмами машинного навчання, такими як класифікатори чи кластеризаційні моделі.

Подальше об'єднання цих векторів для усіх документів у корпусі текстів створює матрицю TF-IDF. У цій матриці кожен рядок представляє один документ,

а кожний стовпчик відповідає значенням TF-IDF для певного слова у всьому корпусі.

Отримана матриця TF-IDF може бути використана для тренування класифікаційних моделей, кластеризації текстів або для пошуку інформації. Числові представлення документів у вигляді TF-IDF векторів надають можливість алгоритмам машинного навчання працювати з текстом, що є числовою формою, забезпечуючи аналіз та обробку.

TF-IDF дозволяє враховувати важливість та унікальність кожного слова у документі, що може покращити точність класифікаційних моделей. Використання TF-IDF векторизації допомагає враховувати особливості текстів та їхніх ключових слів для кращого розуміння змісту.

Практичне використання методу TF-IDF [40]:

Таблиця 2.2

Приклад визначення частоти слова

	Текст 1	Текст 2	Текст 3
Початковий варіант	Машинне навчання - це галузь штучного інтелекту, де комп'ютери навчаються вирішувати завдання, не будучи явно запрограмованими для цього.	Велика частина машинного навчання полягає у використанні алгоритмів для аналізу та інтерпретації даних.	Штучний інтелект, включаючи машинне навчання, стає все більш популярним у різних галузях, від медицини до фінансів.
Після стемінгу	машин навчан - це галуз штучн інтелект де комп'ютер навчається виріш завдан не будуч явн запрограмован для цього	велик частин машин навчан полягає у використан алгоритм для аналіз т інтерпретаці даних	штучн інтелект включа машин навчан стає все більш популярн у різних галуз від медицин до фінансів

К-ть слів в документі	18	15	18
Слова, що зустрічаються 3 рази	навчан; TF = 3/18 = 0,1667	навчан; TF = 3/15 = 0,2	навчан; TF = 3/18 = 0,1667
Слова, що зустрічаються 2 рази	галуз, машин; TF = 2/18 = 0,1111	машин; TF = 2/15 = 0,1333	галуз; TF = 2/18 = 0,1111
Слова, що зустрічаються 1 раз	TF = 1/18 = 0,0556	TF = 1/15 = 0,0667	TF = 1/18 = 0,0556

Таблиця 2.3

Розрахунок значення IDF для найпоширеніших слів у текстах

Слово	DF(w)	IDF(w)
навчан	3	$\text{Log}(\frac{3}{3}) = 0$
машин	2	$\text{Log}(\frac{3}{2}) \approx 0.176$
галуз	2	$\text{Log}(\frac{3}{2}) \approx 0.176$

У цій таблиці показано значення DF для слів "навчан", "машин" та "галуз", а також обчислені значення IDF для цих слів у трьох текстах. Чим більше значення IDF, тим менш загальне (більш унікальне) слово у всьому корпусі текстів.

Таблиця 2.4

Визначення значення TF-IDF для кількох поширених слів у розрізі трьох документів

Слово	TF-IDF в Тексті 1	TF-IDF в Тексті 2	TF-IDF в Тексті 3
навчан	0 * TF = 0	0 * TF = 0	0 * TF = 0
машин	0.176 * TF = 0.176 * TF	0.176 * TF = 0.176 * TF	0.176 * TF = 0.176 * TF
галуз	0.176 * TF = 0.176 * TF	0.176 * TF = 0.176 * TF	0.176 * TF = 0.176 * TF

У таблиці показані значення TF-IDF для слів "навчан", "машин" та "галуз" у трьох різних текстах. Значення TF-IDF визначається як добуток значення TF на значення IDF для кожного слова у кожному тексті. У цьому випадку, оскільки значення IDF для "навчан" дорівнює нулю, значення TF-IDF для цього слова буде також дорівнювати нулю у всіх текстах.

2.2.2. Word Embeddings

Word Embeddings, або вбудування слів, є технікою векторного представлення слів у вигляді числових векторів у просторі реальних чисел. Цей метод спрямований на представлення слів у вигляді числових векторів, де семантично схожі слова мають близькі значення у векторному просторі [41].

Використання Word Embeddings є досить ефективним у багатьох задачах обробки природної мови та машинного навчання. Ось деякі ключові аспекти цього методу [42]:

Семантична репрезентація слів: Word Embeddings дозволяє представити слова у вигляді векторів, де схожі слова розташовані близько одне до одного у векторному просторі. Це означає, що вектори слів зберігають семантичну схожість слів: слова з подібними значеннями або контекстом розташовані поруч.

Здатність узагальнення: Word Embeddings можуть узагальнювати знання з обмеженої кількості даних. Вони можуть робити припущення про семантику слів на основі контексту, в якому вони з'являються. Наприклад, якщо в моделі вже бачили слова "собака" і "кіт" у певному контексті, вона може узагальнити ці знання для нових слів у тому ж контексті.

Використання в навчанні моделей машинного навчання: Word Embeddings є корисним інструментом для навчання моделей машинного навчання в задачах обробки природної мови, таких як класифікація текстів, машинний переклад,

відносини між словами, генерація тексту тощо. Векторне представлення слів полегшує навчання моделей, оскільки вони працюють з числовими даними.

Заснованість на контексті: Одна з головних переваг Word Embeddings полягає у врахуванні контексту слів. Вони можуть враховувати семантичні зв'язки між словами на основі їхнього спільного вживання у тексті.

Моделі Word2Vec, GloVe, FastText: Існують різні архітектури та моделі для створення Word Embeddings, такі як Word2Vec, GloVe (Global Vectors for Word Representation), та FastText, які використовують різні підходи для генерації векторів слів.

Усі ці аспекти роблять Word Embeddings потужним інструментом для роботи з текстовими даними у широкому спектрі застосувань, дозволяючи моделям краще розуміти та узагальнювати семантичні зв'язки між словами у текстах.

Практичне використання методу Word Embeddings:

Класифікація текстів на основі їх тональності: позитивні або негативні.

Підготовка даних:

- Отримання набору даних з позитивними та негативними коментарями.
- Перетворення тексту коментарів у векторні представлення за допомогою Word Embeddings.

Створення моделі для класифікації:

- Використання навчених вкладень слів для представлення текстів у векторній формі.
- Побудова моделі класифікатора, наприклад, з використанням методів глибокого навчання (наприклад, з використанням LSTM, CNN, або інших архітектур).

Вхідні дані:

- Векторні представлення текстів, позначені як позитивні або негативні.

Тренування та оцінка моделі:

- Модель тренується на навчальних даних з векторними представленнями коментарів.
- Оцінка моделі на валідаційному або тестовому наборі даних для визначення її точності та ефективності.

Математичні формули в цьому випадку будуть включати методи глибокого навчання, такі як нейронні мережі із шарами LSTM або CNN, а також функції втрат, наприклад, категоріальна крос-ентропія для багатокласової класифікації.

Наприклад, для моделі LSTM вектор x із Word Embeddings може бути поданий на вхід і оброблений шарами LSTM [43]:

$$h_t = LSTM(x_t, h_{t-1}) \tag{2.18}$$

де h_{t-1} - прихований стан LSTM у момент часу t.

Функція втрат для багатокласової класифікації може бути визначена як категоріальна крос-ентропія:

$$\text{Loss} = -\sum_{i=1}^{N} y_i * \log(\hat{y}_i) \tag{2.19}$$

де y - фактичний вихідний вектор, а $\hat{y}$ - передбачений вихідний вектор.

Така модель може бути навчена за допомогою тренувального набору даних і використовується для класифікації нових текстових матеріалів на позитивні та негативні.

Поєднання методів TF-IDF та Word Embeddings

Поєднання методів TF-IDF та Word Embeddings буде результативне при класифікації деструктивних текстових матеріалів. TF-IDF використовується для створення векторів для кожного тексту, які враховують частоту вживання слова у тексті та важливість цього слова для всього корпусу текстів. Word Embeddings

надають векторні представлення словам з урахуванням їх семантичних зв'язків. Поєднуючи ці два методи, можна отримати кращі векторні представлення текстів для подальшої класифікації.

Кроки поєднання TF-IDF та Word Embeddings:

Перетворення тексту в вектори TF-IDF:

- Обчислення TF-IDF для кожного слова в тексті за допомогою формул TF і IDF.
- Побудова вектора для кожного тексту, де кожне слово представлене значенням TF-IDF.

Отримання векторних представлень слів за допомогою Word Embeddings:

- Застосування навчених вкладень слів на словах у тексті для отримання їх векторних представлень.

Поєднання векторів TF-IDF та Word Embeddings:

- Об'єднання векторів TF-IDF та Word Embeddings для кожного тексту. Це може бути зроблено, наприклад, за допомогою конкатенації цих векторів.

Модель класифікації:

- Використання отриманих векторних представлень текстів для навчання моделі класифікації, наприклад, моделі глибокого навчання (наприклад, нейронної мережі).
- Використання згенерованої моделі для класифікації нових текстів на позитивні та негативні.

2.3. Математичне забезпечення методу класифікації

Одним з ключових аспектів у визначенні та класифікації деструктивних текстів є важливість кожної окремої ознаки або слова в тексті, що впливає на їхню класифікацію. Подальший аналіз важливості окремих ознак у текстах, що відносяться до деструктивних та недеструктивних категорій, може сприяти удосконаленню моделей класифікації та розумінню того, які саме аспекти текстів визначають їхню приналежність до різних класів.

Information Gain (інформаційний приріст) - це метрика, яка використовується у машинному навчанні для визначення важливості конкретної ознаки у класифікації даних. Цей показник оцінює, наскільки добре певна ознака або атрибут розділяє дані за їх класами. Основна ідея полягає в порівнянні ентропії (ступеня неоднорідності) даних до та після розділення за певною ознакою. Чим більший різниця між цими ентропіями, тим більший інформаційний приріст, що вказує на велику інформативність цієї ознаки для класифікації [44]. Використання інформаційного приросту дозволяє визначити, які ознаки або атрибути є найбільш важливими для розділення даних на різні класи, що сприяє удосконаленню моделей машинного навчання та підвищенню їхньої ефективності в процесі класифікації.

Ентропія визначає ступінь неоднорідності у наборі даних. Для множини даних S з p класами, ентропія обчислюється як:

$$\text{Entropy (S)} = -\sum_{i=1}^{p} p_i log_2(p_i) \qquad (2.20)$$

де p_i - ймовірність належності до класу *i*.

Після обчислення ентропії для всього набору даних, розглянемо окрему ознаку (слово, ознаку) і поділимо дані за цією ознакою. Для кожної можливої категорії цієї ознаки, обчислимо її внутрішню ентропію Entropy ($S_{feature}$).

Інформаційний приріст показує, наскільки добре певна ознака розділяє дані за їх класами. Обчислюється як різниця між поточною ентропією Entropy (S) та середньою взваженою ентропією після розділення даних за цією ознакою. Середня взважена ентропія після розділення обчислюється як сума внутрішньої ентропії кожної категорії ознаки, взважена її часткою у загальній множині [45]:

$$\text{Average Entropy after Split} = \sum_{categories} \frac{|S_{feature,\ category}|}{|S|} \times Entropy(S_{feature,\ category}) \qquad (2.21)$$

де $|S_{feature,\ category}|$ – кількість елементів у кожній категорії ознак, а $|S|$ – загальна кількість елементів.

Остаточний інформаційний приріст для даної ознаки обчислюється як різниця між поточною ентропією і середньою виваженою ентропією після розділення даних:

$$\text{Information Gain} = \text{Entropy(S)} - \text{Average Entropy after Split}$$

Високий інформаційний приріст вказує на те, що розділення за цією ознакою вносить значний внесок у зменшення неоднорідності в даних, тобто ця ознака важлива для класифікації.

Похибка в методі класифікації деструктивних текстових даних визначається за допомогою різних метрик, які оцінюють правильність передбачень моделі.

Precision вказує на відсоток документів, визначених моделлю як позитивні, і вони дійсно є позитивними. Це значення вимірює точність у визначенні позитивних прикладів моделлю.

$$\text{Precision} = \frac{True\ Positives}{True\ Positives + False\ Positives} \qquad (2.22)$$

True Positives: Кількість правильно визначених позитивних прикладів.

False Positives: Кількість неправильно визначених позитивних прикладів.

Високе значення Precision вказує на те, що модель відзначає лише дійсно позитивні приклади, без зайвих помилок у визначенні негативних прикладів як позитивних [45].

Recall визначає, яку частку дійсних позитивних прикладів модель визначила правильно відносно загальної кількості дійсних позитивних прикладів.

$$\text{Recall} = \frac{True\ Positives}{True\ Positives + False\ Negative} \tag{2.23}$$

False Negatives: Кількість неправильно визначених негативних прикладів.

Високе значення Recall показує, що модель відзначає більшу частину дійсних позитивних прикладів, уникнувши помилкового класифікування негативних прикладів як позитивних.

F1-міра представляє собою гармонічне середнє між Precision та Recall. Ця метрика об'єднує як точність, так і повноту моделі [46].

$$\text{F1 score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.24}$$

F1-міра дає більш збалансовану оцінку моделі, яка враховує як точність, так і повноту. Вона корисна, коли потрібно об'єднати обидві метрики для оцінки ефективності класифікації моделі. Чим вище F1-міра, тим краща ефективність класифікації моделі.

2.4. Висновки до розділу

Наївний баєсів класифікатор, відомий своєю ефективністю та швидкістю обробки великих обсягів текстових даних, базується на простому врахуванні частоти входження слів у текст, що сприяє швидкій та ефективній класифікації

текстів. Однак, його обмеження полягає в урахуванні семантичних зв'язків та послідовностей слів, що може вплинути на точність та повноту аналізу.

З іншого боку, метод максимальної ентропії дозволяє більш комплексно враховувати складні взаємозв'язки та семантичні аспекти між словами у тексті. Його використання може компенсувати обмеження наївного баєсівого класифікатора, але це може бути більш обчислювально витратним процесом.

У роботі було обрано метод векторизації TF-IDF, що дозволяє представити документи у вигляді числових векторів, враховуючи частоту термінів всередині документа та їх рідкісність у великому корпусі. Це дозволяє виділити найбільш інформативні терміни.

Також був розглянутий підхід Word Embeddings, що також перетворює текст на числові вектори на основі нейронних мереж, враховуючи контекст слова та дозволяючи виявити приховані семантичні зв'язки. З метою підвищення якості класифікації, було обрано метрику Information Gain для відбору найбільш інформативних ознак, а також F1-міру для оцінки точності моделі.

У сукупності це дозволить створити надійний метод класифікації текстів з деструктивним вмістом, що буде чітко та якісно виконувати свої функції за рахунок поєднання різних методів та підходів до класифікації текстових даних.

РОЗДІЛ 3. СИСТЕМА КЛАСИФІКАЦІЇ ДЕСТРУКТИВНИХ ТЕКСТОВИХ ДАНИХ

3.1. Вибір мови програмування

Python визнаний своєю простотою, гнучкістю та широким спектром можливостей, що робить його однією з найпопулярніших мов програмування у світі. Розглянемо детальніше особливості Python, які зробили його привабливим для широкого кола програмістів та розробників.

Python славиться своєю простотою та легкістю вивчення. Синтаксис мови максимально наближений до англійської мови, що робить код зрозумілим навіть для початківців. У порівнянні з іншими мовами, Python відомий своєю лаконічністю та здатністю вирішувати складні завдання за допомогою зручного та зрозумілого коду.

Однією з найбільших переваг Python є велика кількість бібліотек та модулів, що дозволяють розробникам ефективно створювати різноманітні застосунки. Наприклад, для аналізу даних існує бібліотека Pandas, для наукових обчислень - NumPy та SciPy, а для машинного навчання - Scikit-learn. Це дає можливість швидко та ефективно розв'язувати різноманітні завдання з програмування [47].

Python є однією з найбільш популярних мов для аналізу даних та роботи з машинним навчанням. Бібліотеки, такі як Pandas, NumPy та Scikit-learn, надають потужні інструменти для обробки, аналізу та моделювання даних. Pandas використовується для роботи з даними у формі таблиць, NumPy для чисельних

обчислень, а Scikit-learn - для створення та навчання моделей машинного навчання.

Python є крос-платформовою мовою програмування, що означає, що код, написаний на Python, може запускатися на різних операційних системах, таких як Windows, macOS та Linux, без будь-яких істотних змін. Це робить Python універсальним інструментом для розробки, дозволяючи створювати програми, які працюють на будь-яких пристроях та платформах.

Python використовує динамічну типізацію, що означає, що розробники можуть змінювати типи змінних під час виконання програми [47]. Це робить розробку більш гнучкою та зменшує час на написання коду. Також Python має автоматичне управління пам'яттю, що звільняє програмістів від потреби вручному вирішенні завдань з управління пам'яттю.

Таблиця 3.1

Порівняння Python з іншими мовами програмування

Характеристика	**Python**	**JavaScript (JS)**	**Java**	**C++**
Простота вивчення	Висока	Середня	Середня	Складна
Широкий спектр бібліотек	Великий	Середній	Великий	Великий
Популярність та спільнота	Висока	Висока	Висока	Середня
Застосування	Універсальний	Веб-розробка, фронтенд	Веб-розробка, додатки	Системне програмування, ігри

Java - мова програмування, яка, подібно до Python, використовується у різних галузях. Вона відома своєю високою швидкодією та переносимістю через використання віртуальної машини Java . Проте для виконання коду Java потрібна JVM, що може впливати на швидкодію та потребує встановлення відповідного

середовища на пристрої. Python має меншу швидкодію порівняно з Java, але вирізняється простотою та легкістю вивчення.

JavaScript використовується головним чином для веб-розробки і взаємодії з браузерами. Ця мова відрізняється від Python як в синтаксисі, так і в призначенні. JavaScript використовується для створення динамічного вмісту на веб-сайтах та має потужні бібліотеки для роботи з DOM та асинхронним програмуванням. В той час як Python широко використовується в аналізі даних, штучному інтелекті та інших сферах, JavaScript спеціалізується на веб-технологіях.

C++ є потужною мовою програмування, яка відома своєю ефективністю та швидкодією. Вона використовується для розробки операційних систем, вбудованих систем, грифічних додатків та інших завдань, де швидкодія є критичною [48]. У порівнянні з Python, C++ зазвичай вимагає більше коду для досягнення того ж результату, що може затягнути процес розробки.

Python - це універсальний інструмент, який поєднує в собі простоту з потужними функціональними можливостями. Ця мова програмування підходить для різноманітних завдань, включаючи обробку тексту, аналіз даних, веб-розробку та машинне навчання. Її широкий спектр бібліотек та активна спільнота роблять Python ідеальним вибором для реалізації методу класифікації деструктивних текстових даних .

3.2. Архітектура системи

Підготовка до використання методу. Введення тексту та категорії у форматі [текст, {категорія}] - це перший крок до створення корисного

інструменту для класифікації тексту. Для підготовки роботи з розробленим методом, надзвичайно важливою є створення бази даних, яка служитиме фундаментом для прийняття рішень щодо класифікації тексту користувача.

Ця програма, яка знаходиться у стадії підготовки до використання (Додаток Б), дозволяє користувачеві вводити тексти та відповідні категорії, у форматі, який легко розпізнається програмою. Це можуть бути будь-які тексти разом з відповідною категорією, що вказує на сутність цього тексту, на основі яких користувач хоче навчити програму класифікації.

Ця взаємодія з інтерфейсом дозволяє легко та зрозуміло внести дані для подальшої обробки. Власне, коли тексти та категорії введені, програма приступає до обробки цієї інформації.

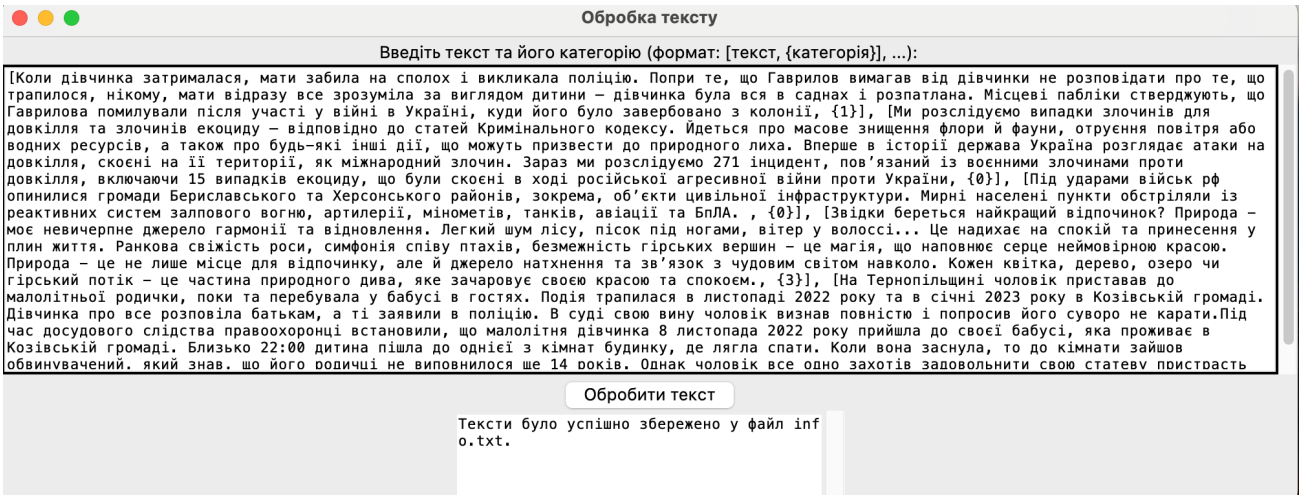


Рис. 3.1. Інтерфейс програми для створення бази даних векторизованих текстів

Програма розділяє введений текст та категорію, використовуючи певний формат запису, та виконує векторизацію цих текстів за допомогою TfidfVectorizer, який перетворює текстові дані у числові вектори. Це забезпечує можливість аналізувати та розуміти текстові дані на числовому рівні, що є ключовим аспектом багатьох алгоритмів машинного навчання.

Однак, саме тут, після векторизації, відбувається важливий етап: програма зберігає векторизовані тексти у файл "info.txt" (рис.3.2). Кожен векторизований текст із своєю категорією записується у цей файл, створюючи своєрідну базу даних, яка стане основою для подальшої роботи основного методу. Цей файл, є своєрідним сховищем векторизованих текстів, які будуть використовуватися для порівняння та аналізу тексту користувача.

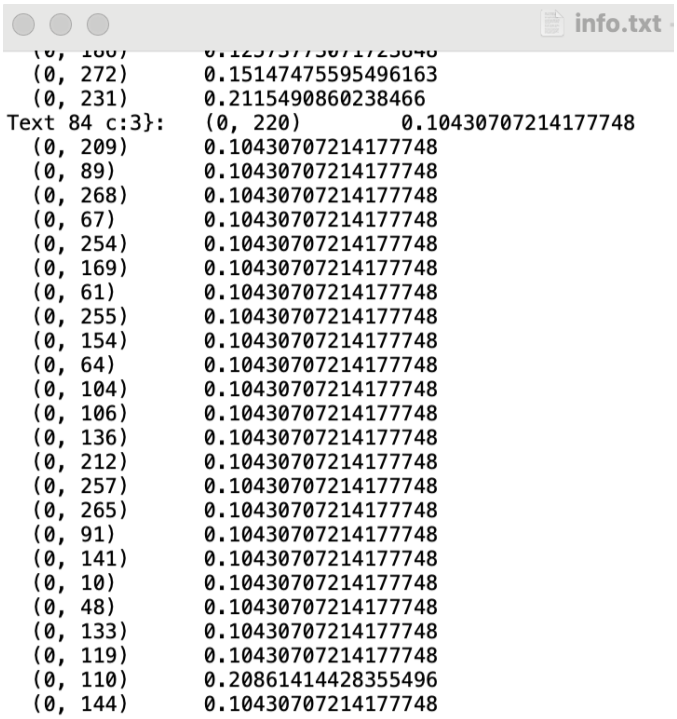


Рис. 3.2. Вміст файлу info.txt з векторизованими текстами

Таким чином, підготовка до роботи основної програми є важливим етапом, коли відбувається створення бази даних з векторизованими текстами, які послуговують основою для класифікації тексту користувача. Це дозволяє створювати ефективний та корисний інструмент для роботи з текстовими даними.

Для подальшої демонстрації роботи методу було створену не велику базу даних, що включає в себе по 20 текстів для кожної з п’яти категорії. Загалом 100 текстів, що слугують основою для подальшої класифікації текстів користувача.

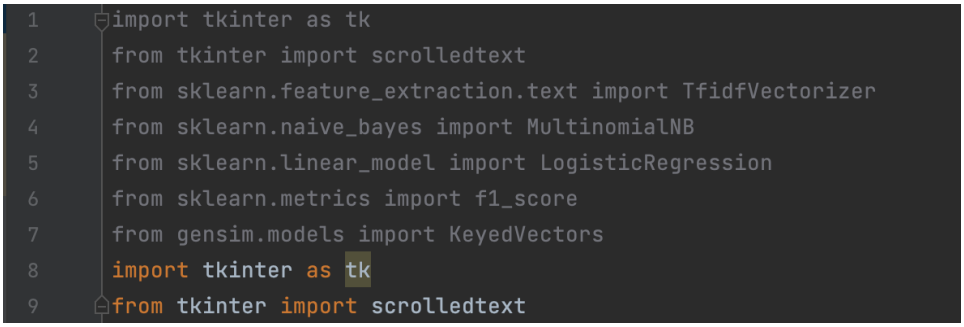
Принцип роботи методу класифікації деструктивних текстових даних.

Розроблений метод призначений для класифікації текстових матеріалів за їх характеристиками, такими як війна, насильство, порнографія та інше. Користувач може ввести текст у вікно програми, і метод застосовує навчені моделі машинного навчання, щоб визначити категорію цього тексту.

Після введення тексту, метод використовує два підходи до класифікації: наївний баєсівський класифікатор та метод максимальної ентропії (Logistic Regression), для передбачення категорії тексту. Результати класифікації відображаються у вікні результатів, де показується категорія, до якої належить текст, а також обчислена F1-міра для кожного методу класифікації.

Розроблений метод використовує файл info.txt як базу даних для класифікації текстів. У цьому файлі зберігаються векторизовані тексти та їх відповідні категорії. При класифікації, модуль використовує цей файл як основу для визначення категорії нового тексту, порівнюючи його з вмістом файлу info.txt.

Необхідні бібліотеки та доповнення:



```
1  import tkinter as tk
2  from tkinter import scrolledtext
3  from sklearn.feature_extraction.text import TfidfVectorizer
4  from sklearn.naive_bayes import MultinomialNB
5  from sklearn.linear_model import LogisticRegression
6  from sklearn.metrics import f1_score
7  from gensim.models import KeyedVectors
8  import tkinter as tk
9  from tkinter import scrolledtext
```

Рис. 3.3. Бібліотеки та доповнення

Фрагмент коду на рисунку 3.3 імпортує різні модулі та класи, необхідні для розробки програми, яка, ймовірно, включає в себе створення інтерфейсу користувача для класифікації тексту та використання моделей машинного навчання для цієї класифікації.

import tkinter as tk: Цей рядок коду імпортує модуль tkinter, який використовується для створення графічного інтерфейсу користувача (GUI) в Python. Його можна використовувати для створення вікон, кнопок, полів введення та інших елементів інтерфейсу.

from tkinter import scrolledtext: Цей рядок імпортує конкретний клас scrolledtext з модуля tkinter. Цей клас забезпечує можливість створення текстового поля з прокруткою для великого обсягу тексту в інтерфейсі.

from sklearn.feature_extraction.text import TfidfVectorizer: Цей рядок імпортує TfidfVectorizer з бібліотеки sklearn. TfidfVectorizer використовується для перетворення тексту у числові вектори за допомогою методу TF-IDF (Term Frequency-Inverse Document Frequency).

from sklearn.naive_bayes import MultinomialNB: Цей рядок імпортує клас MultinomialNB з бібліотеки sklearn. MultinomialNB - це реалізація наївного баєсівого класифікатора для роботи з категоріальними ознаками, яка часто використовується у класифікації тексту.

from sklearn.linear_model import LogisticRegression: Цей рядок імпортує клас LogisticRegression з бібліотеки sklearn. LogisticRegression - це модель, що використовує логістичну регресію для класифікації.

from sklearn.metrics import f1_score: Цей рядок імпортує функцію f1_score з бібліотеки sklearn. f1_score - це метрика, яка використовується для оцінки якості моделі класифікації на основі точності та відновлення. from gensim.models import KeyedVectors: Цей рядок імпортує клас KeyedVectors з бібліотеки gensim. KeyedVectors - це клас, що надає можливість роботи з векторними представленнями слів, такими як Word Embeddings.

Інтерфейс користувача (GUI):

Tkinter GUI: Tkinter - це стандартний пакет для створення графічного інтерфейсу користувача (GUI) в Python. Цей пакет дозволяє створювати

різноманітні вікна, кнопки, поля для введення тексту та інші елементи інтерфейсу. У даному випадку, Tkinter використовується для створення вікна, де користувач може вводити текст для класифікації та де результати відображаються.

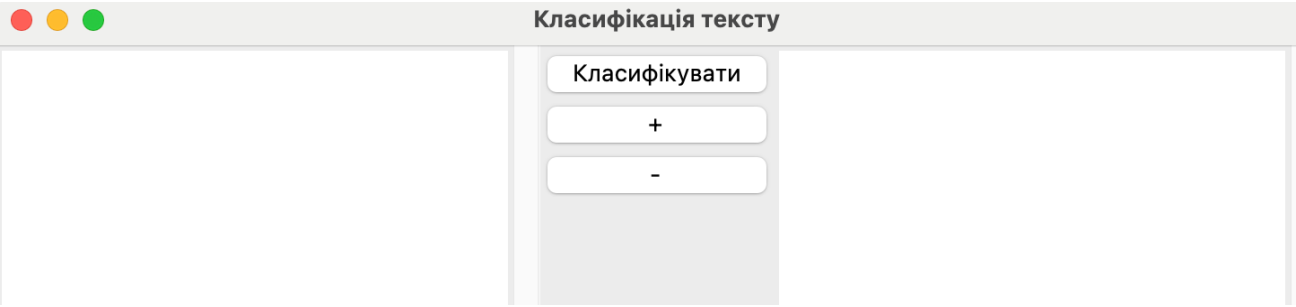


Рис. 3.4. Інтерфейс модулю

Кнопка "Класифікувати" має основну функцію, вона запускає процес класифікації тексту, що ввів користувач.

Кнопка "+" – можливість позитивно оцінити класифікацію, вона виконує дію збереження векторизованого тексту у файлі info.txt., тим самим розширюючи базу знань модулю.

Кнопка "-" використовується для позначення неправильно класифікованого тексту. Користувач може використовувати цю кнопку, щоб вказати, що результат класифікації був неправильним, і модуль спробує класифікувати текст знову.

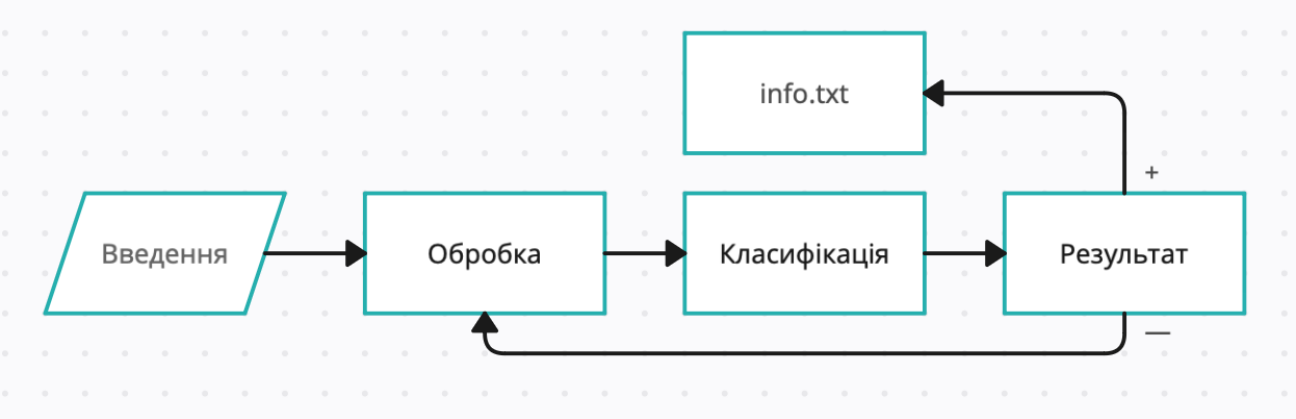


Рис. 3.5. Архітектура системи

На рисунку 3.5 зображена архітектура системи, що складається з чотирьох ключових етапів.

Етап 1. Введення. На цьому етапі користувач вводить текст, який бажає швидко класифікувати та віднести до певної категорії.

Етап 2. Обробка. На даному етапі відбувається векторицація тексту.

Word Embeddings: Це техніка, що використовується для перетворення слів у вектори чисел, які представляють семантичні значення цих слів у просторі векторів. У даному випадку, використовується модель Word Embeddings з бібліотеки Gensim для отримання векторного представлення тексту.

TF-IDF Vectorizer: TfidfVectorizer з бібліотеки scikit-learn використовується для створення числового представлення тексту на основі частотності та важливості слів.

Етап 3. Класифікація. Базуючись на даних з файлу info.txt модуль виконує класифікацію заданого тексту.

Naive Bayes (MultinomialNB) та Logistic Regression: Це дві моделі машинного навчання, які використовуються для класифікації тексту за заданими категоріями. Наївний баєсівський класифікатор (MultinomialNB) та метод максимальної ентропії (Logistic Regression) з scikit-learn використовуються для передбачення категорій текстів.

Етап 4. Метрики та результати. Виведення результатів класифікації користувачу.

Результат класифікації та метрики F1-міри виводяться на графічний інтерфейс користувача. F1-міра: Це метрика, яка використовується для оцінки точності моделей класифікації. Вона враховує як точність (precision), так і повноту (recall) моделі, роблячи оцінку її ефективності.

За бажанням користувач може доєднатись до тренування модулю шляхом оцінки результатів його роботи натиснувши кнопки “+” – позитивна оцінка та

доповнення бази даних модлю, “-” – негативна оцінка, що запустить повторний процес класифікації.

3.3. Програмна складова методу

Зчитування та обробка даних: Під час виконання програми відбувається зчитування даних з файлу info.txt та їх обробка. Файл info.txt використовується як база даних для навчання та вдосконалення моделі класифікації за допомогою додавання нових текстів та їх категорій. Це дозволяє програмі навчатися на нових даних та покращувати свою точність і ефективність. Фрагмент коду для цього виглядає наступним чином:

```
processed_texts = []
categories = []
with open("info.txt", "r", encoding="utf-8") as file:
    for line in file:
        parts = line.strip().split(": ")
        categories.append(int(parts[0].split()[1]))
        vector = list(map(float, parts[1][1:-1].split(', ')))
        processed_texts.append(vector)
```

Рис. 3.6. Зчитування даних з файлу info.txt

Отримані з файлу дані використовуються для підготовки даних для класифікації текстів користувача.

Машинне навчання: Програмна реалізація методу класифікації використовує ряд методів машинного навчання, а саме Наївний баєсівський класифікатор та метод максимальної ентропії, що входять до складу бібліотеки scikit-learn. Розглянемо фрагмент коду, що відповідає за створення та застосування цих моделей:

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

# Створення моделей класифікації
nb_classifier = MultinomialNB()
lr_classifier = LogisticRegression()

# Тренування моделей на вхідних даних
nb_classifier.fit(X_train, y_train)
lr_classifier.fit(X_train, y_train)

# Класифікація тексту за допомогою навчених моделей
prediction_nb = nb_classifier.predict(X_test)
prediction_lr = lr_classifier.predict(X_test)
```

Рис. 3.7. Реалізація методу класифікації в поєднанні з машинним навчання

Реалізація методів машинного навчання була забезпечена за допомогою бібліотеки scikit-learn, яка надає зручний інтерфейс для створення моделей класифікації.

Обробка природної мови: Для обробки текстових даних та їх підготовки до класифікації використовується метод векторизації TF-IDF та Word Embeddings. Реалізація цих методів є наступною:

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Векторизація тексту за допомогою TF-IDF
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Використання Word Embeddings
def get_word_embeddings(text):
    # Реалізація отримання векторів слів з тексту
```

Рис.3.8. Обробка природної мови.

Для отримання векторів слів з тексту використовувалась функція get_word_embeddings, яка залишалась нереалізованою в цьому коді. Ця функція є ключовою для отримання числових векторів, що представляють слова з тексту. TF-IDF дозволяє представити текст у вигляді числових векторів, враховуючи частоту термінів у документах та їх рідкісність у корпусі текстів. Word Embeddings використовує нейронні мережі для перетворення слів у числові вектори, що дозволяє виявляти семантичні зв'язки. Застосування цих методів у

поєднанні з моделями класифікації такими, як Наївний баєсів класифікатор та метод максимальної ентропії, дозволило створити модель, яка здатна ефективно впізнавати та фільтрувати деструктивний контент у текстах. Результати класифікації були оцінені за допомогою метрик, зокрема F1-міри, що підтверджує надійність та ефективність створеної моделі.

Взаємодія користувача з програмою: Розроблений програмний продукт має простий інтерфейс, який надає можливість введення тексту користувачем та отримання результатів класифікації. Нижче наведено приклад взаємодії користувача з програмою за допомогою графічного інтерфейсу Tkinter:

```
# Створення графічного інтерфейсу
root = tk.Tk()
text_input = scrolledtext.ScrolledText(root, width=40, height=5)
text_input.pack(fill=tk.BOTH, expand=True, side=tk.LEFT)

# ... (інші елементи графічного інтерфейсу)
root.mainloop()
```

Рис. 3.9. Створення графічного інетрфейсу.

Реалізація метрики якості класифікації F-1 міра:

Отримання прогнозів класифікатора: У коді, після побудови моделі та її навчання, ми отримуємо прогнози для тестових даних. prediction_nb та prediction_lr - це прогнози, зроблені двома різними класифікаторами (наприклад, наївний баєсівський класифікатор та метод максимальної ентропії) для тестових даних.

Обчислення F-1 міри: Далі в коді обчислюється F-1 міра для кожного з класифікаторів на підставі їхніх прогнозів та фактичних міток класів (які можуть бути заздалегідь відомі для тестових даних).

```
f1_nb = f1_score([3], prediction_nb, average='macro')
f1_lr = f1_score([3], prediction_lr, average='macro')
```

Рис. 3.10. Обчислення F-1 міра.

Функція f1_score() обчислює F-1 міру для кожного класифікатора.Параметр average='macro' вказує, що ми хочемо обчислити середнє значення F-1 міри для всіх класів. prediction_nb та prediction_lr - це прогнозовані класи відповідно від наївного баєсівського класифікатора та методу максимальної ентропії. [3] - це фактична мітка класу для тестових даних, яка використовується для порівняння з прогнозованими класами. Далі показники поєднуються і користувач бачить результат.

Крім цього, програмна складова методу забезпечує можливість розширення бази даних з використанням функцій додавання та видалення текстових даних. Ця можливість дозволяє вдосконалювати модель за рахунок навчання на нових даних, що підвищує точність та ефективність класифікації.

3.4. Експериментальні дослідження

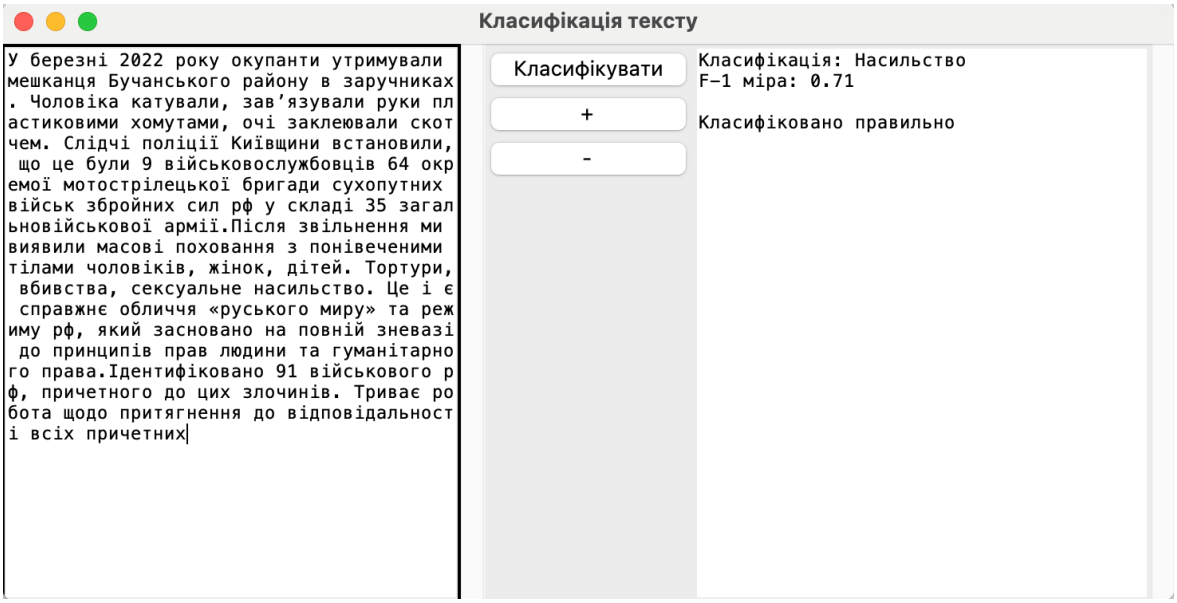
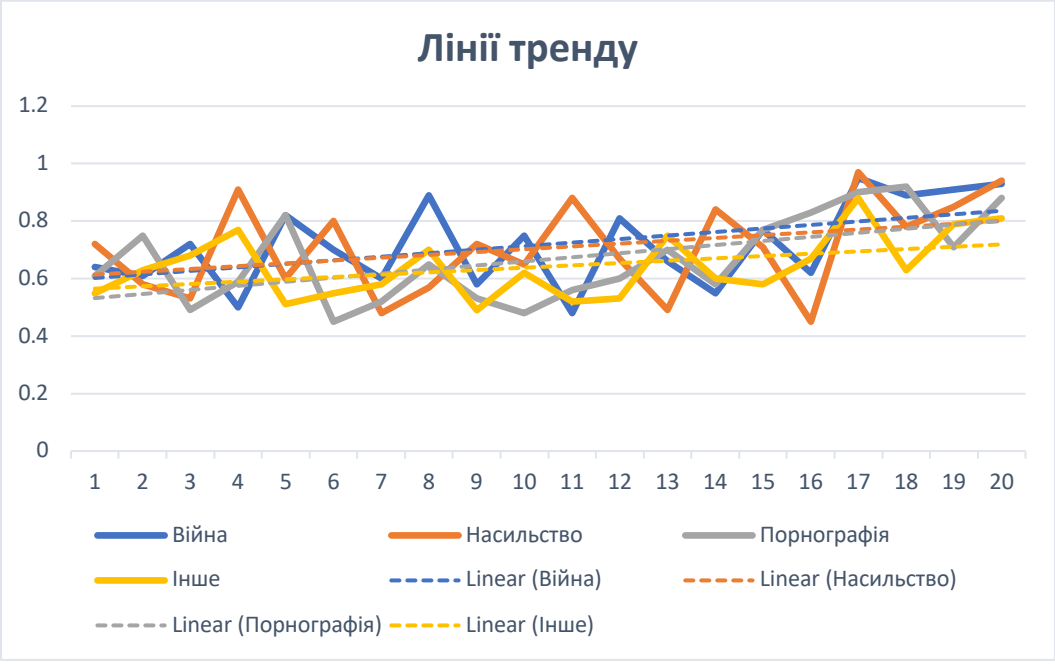


Рис.3.11. Класифікація 17 тексту з категорії «Насильство»

Після створення бази даних info.txt, що включає в себе 100 текстів з різних категорій було обрано 100 інших уривків текстів з книжок, новин тощо для експериментального дослідження. Така кількість зумовлена тим, що у разі успішної класифікації база даних даного модулю збільшиться вдвічі, а за мірою F-1 можна буде визначити чи відбувається покращення якості класифікації тексту.

На рисунку 3.11 зображено процес класифікації 17 текст з категорії «Насильство», що розповідає про катування москалями мешканця Бучанського району в березні 2022 року. За результатами класифікації модуль правильно визначив категорію до якої належить текст, а показник F-1дорівнює 0.71, що є досить хорошим результатом.

Після завершення тестування модулю було отримано сто показників, по двадцять в кожній категорії та побудовано таблицю 3.1 з усіма результатами, а також графік 3.1 на якому можна побачити зростаючі лінії тренду.



Графік 3.1. Графічне представлення результатів класифікації 100 текстів.

Таблиця 3.1

Табличне представлення показнику F-1 міра за 100 спроб класифікації.

Спроба	Війна	Насильство	Порнографія	Інше
1	0.64	0.72	0.61	0.55
2	0.61	0.58	0.75	0.63
3	0.72	0.53	0.49	0.68
4	0.5	0.91	0.59	0.77
5	0.82	0.6	0.82	0.51
6	0.7	0.8	0.45	0.55
7	0.6	0.48	0.52	0.58
8	0.89	0.57	0.65	0.7
9	0.58	0.72	0.53	0.49
10	0.75	0.65	0.48	0.62
11	0.48	0.88	0.56	0.52
12	0.81	0.67	0.6	0.53
13	0.66	0.49	0.7	0.75
14	0.55	0.84	0.58	0.6
15	0.77	0.71	0.77	0.58
16	0.62	0.45	0.83	0.67
17	0.95	0.97	0.9	0.88
18	0.89	0.78	0.92	0.63
19	0.91	0.85	0.71	0.79
20	0.93	0.94	0.88	0.81

Аналіз результатів за останніми спробами:

Категорія "Війна": Останні результати класифікації у категорії "Війна" варіювалися від 0.89 до 0.95 за мірою F-1. Це свідчить про високу точність класифікації для цієї категорії.

Категорія "Насильство": Останні результати класифікації у категорії "Насильство" показали значення F-1 від 0.78 до 0.97. Ці показники також вказують на високу точність у класифікації даної категорії.

Категорія "Порнографія": У цій категорії останні результати класифікації показали значення F-1 від 0.63 до 0.92. Це також свідчить про покращення точності класифікації порівняно з початковими даними.

Категорія "Інше": Останні результати класифікації у категорії "Інше" демонстрували значення F-1 від 0.63 до 0.81. Це показує стабільність та покращення методу класифікації у даній категорії.

Останні результати класифікації показали високу точність та покращення якості класифікації у всіх категоріях порівняно зі спочатку набутими результатами. Це свідчить про здатність методу навчатися на нових даних і підвищувати ефективність класифікації з плином часу. Такий підхід виявляється перспективним і підтримує думку про ефективність даного методу класифікації при поповненні бази даних новою інформацією.

3.5. Висновки до розділу

У третьому розділі даної роботи була проведена програмна реалізація методу класифікації деструктивних текстових даних з використанням інноваційних підходів машинного навчання та обробки природної мови.

В процесі розробки програмного рішення було здійснено імплементацію методів машинного навчання та використано відповідні бібліотеки, зокрема scikit-learn, що надає широкий спектр інструментів для побудови моделей класифікації.

Для створення моделі класифікації були обрані алгоритми Наївного баєсівського класифікатора та методу максимальної ентропії, які виявилися оптимальними для даної задачі. Крім того, в рамках програмної реалізації були використані методи векторизації тексту, зокрема TF-IDF та Word Embeddings, для числового представлення слів та текстів. Останній забезпечує виявлення семантичних зв'язків у тексті та дозволяє враховувати контекст слова.

Програма дозволяє користувачу вводити текст для його класифікації, а також забезпечує можливість розширення бази даних з використанням функцій "+"/"-". Це забезпечує навчання моделі на нових даних, що позитивно впливає на її точність та ефективність.

Загалом, розроблена програмна реалізація відповідає поставленим завданням роботи. Вона дозволяє ефективно впізнавати та класифікувати деструктивний контент у текстах, забезпечуючи користувачеві зручний інтерфейс для взаємодії з програмою та можливість покращення моделі за рахунок надання нових даних.

Ця програмна реалізація є значним кроком у напрямку створення ефективних інструментів для фільтрації деструктивного контенту в онлайн-середовищі, що підвищує інформаційну безпеку та створює передумови для подальших досліджень та вдосконалення методів класифікації текстових даних.

ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

РОЗДІЛ 4. ЗАВДАННЯ СУЧАСНОГО ПРИРОДООХОРОННОГО ЗАКОНОДАВСТВА

Природоохоронне законодавство є сукупністю правових норм, актів та механізмів, спрямованих на захист, збереження та раціональне використання природних ресурсів. Ця галузь права визначає основні принципи охорони природи, регулює взаємодію людини з природним середовищем та встановлює механізми контролю за використанням природних ресурсів для забезпечення сталого розвитку.

Природоохоронне законодавство включає в себе широкий спектр нормативно-правових актів, що стосуються охорони біорізноманіття, збереження водних, лісових, ґрунтових ресурсів, контролю за викидами шкідливих речовин, регулювання відходів та інших аспектів, спрямованих на підтримку екологічно стійкого розвитку суспільства.

У сучасному світі проблеми забруднення довкілля, знищення екосистем та виснаження природних ресурсів стають все більш актуальними. Суперечливий розвиток промисловості, зростання попиту на ресурси та погіршення стану довкілля підкреслюють необхідність розробки та впровадження ефективних природоохоронних законів. Створення та вдосконалення природоохоронного законодавства вимагає виваженості, науково обґрунтованих рішень та широкого суспільного консенсусу.

Ефективне природоохоронне законодавство — це ключовий інструмент для забезпечення сталого розвитку, збереження природних ресурсів та забезпечення здоров'я населення. Розгляд цієї теми дозволить не лише усвідомити важливість

охорони природи, але й розробити конкретні стратегії та механізми для досягнення цілей екологічної стабільності та гармонії між людиною та природою.

Збереження біорізноманіття та екосистем. Збереження біорізноманіття та екосистем є одним із найважливіших завдань сучасного природоохоронного законодавства. Це передбачає створення законодавчих актів, спрямованих на захист унікальних видів рослин і тварин, збереження та відновлення природних середовищ, підтримку екосистемних послуг. Природоохоронне законодавство має сприяти формуванню мережі охоронних територій, створенню резерватів та заповідників для збереження біорізноманіття та забезпечення його стійкості в умовах змін клімату та людської діяльності.

Боротьба зі забрудненням навколишнього середовища. Природоохоронне законодавство повинне активно боротися зі забрудненням повітря, води та ґрунтів. Це включає розробку строгих нормативів щодо викидів та видалень шкідливих речовин, встановлення вимог до очищення стічних вод та утилізації відходів. Природоохоронні закони мають стимулювати використання екологічно чистих технологій та інновацій, спрямованих на зменшення впливу людської діяльності на навколишнє середовище.

Раціональне використання природних ресурсів. Законодавчі норми повинні спрямовувати населення та підприємства на раціональне використання природних ресурсів, включаючи ліси, мінеральні ресурси, водні джерела та інші. Ефективне природоохоронне законодавство має регулювати видобуток ресурсів, сприяти відновленню природних балансів та запобігати їхньому виснаженню.

Захист від кліматичних змін та їх наслідків. Природоохоронне законодавство повинне бути спрямоване на захист від наслідків кліматичних змін. Це включає розробку стратегій адаптації до змін клімату, зменшення викидів парникових газів, підтримку використання відновлюваних джерел енергії та збереження природних ресурсів.

Недоліки і прогалини в законодавстві. Сучасне природоохоронне законодавство, незважаючи на свою важливість, має деякі недоліки та прогалини, які потребують уваги та виправлення. Деякі з них включають:

Неповнота або розбіжність нормативних актів: У законодавстві можуть бути прогалини або суперечливість між різними нормативними актами, що ускладнює їх реалізацію та контроль.

Недостатня ефективність контролю та виконання: Часто відсутність чіткого механізму контролю та санкцій за порушення природоохоронного законодавства призводить до його недієвості.

Відсутність міжнародної координації: Проблеми природоохорони не мають меж, тому відсутність координації між країнами у розробці та реалізації спільних заходів може ускладнити вирішення глобальних екологічних проблем.

Неадекватність до швидкісних змін: Природоохоронне законодавство часто не може швидко реагувати на нові екологічні виклики та технологічні зміни.

Виклики з позиції нових технологій та глобальних проблем. Сучасне природоохоронне законодавство стикається з новими технологіями та глобальними проблемами, які ставлять під сумнів ефективність і потребують актуалізації законодавства:

Технологічні інновації: Розвиток нових технологій, таких як штучний інтелект, біотехнології та генетично модифіковані організми, потребує адаптації законодавства для регулювання їх впливу на навколишнє середовище та біорізноманіття.

Глобальні екологічні проблеми: Зміна клімату, зникнення видів, забруднення морів та океанів — ці проблеми потребують міжнародної співпраці та нових підходів у законодавстві для їх ефективного вирішення.

Цифрова трансформація та екологія: Використання інформаційних технологій, зокрема розвиток цифрової економіки, також має вплив на

навколишнє середовище, і вимагає вдосконалення законодавства для регулювання цього впливу.

Ці виклики потребують постійного оновлення та удосконалення природоохоронного законодавства для того, щоб воно було адаптивним, ефективним та здатним відповідати на найбільш актуальні екологічні проблеми сучасності.

У контексті обговорення сучасного природоохоронного законодавства, важливо враховувати складні виклики, з якими стикається наше суспільство у збереженні та охороні природних ресурсів та навколишнього середовища. Аналізуючи основні завдання, проблеми та потенційні напрями вдосконалення природоохоронного законодавства, відкриваються ключові аспекти, які потребують уваги та дії.

Сучасне природоохоронне законодавство спрямоване на збереження біорізноманіття, боротьбу з забрудненням навколишнього середовища, раціональне використання природних ресурсів та захист від кліматичних змін. Ці завдання є критично важливими для створення екологічно стійкого майбутнього.

Однак, існують проблеми у сучасному природоохоронному законодавстві, які потребують уваги. Недоліки в законодавстві, які включають неповноту чи суперечливість нормативних актів, або недостатню ефективність контролю та виконання, ускладнюють його реалізацію. Також, відсутність міжнародної координації у вирішенні глобальних екологічних проблем може ускладнити захист навколишнього середовища на глобальному рівні.

Поруч з цим, сучасне природоохоронне законодавство повинне враховувати нові технології та глобальні виклики. Розвиток інноваційних технологій, які можуть мати як позитивний, так і негативний вплив на навколишнє середовище, потребує узгоджених та прогресивних підходів у законодавстві для регулювання їх впливу на екологію. Крім того, глобальні

екологічні проблеми, такі як зміна клімату та зникнення видів, вимагають міжнародної співпраці та спільних зусиль у розробці та виконанні стратегій для їх вирішення.

Загальний висновок полягає в тому, що удосконалення та адаптація природоохоронного законодавства до сучасних реалій є критично важливим для забезпечення сталого розвитку та збереження природи для майбутніх поколінь. Необхідність постійного оновлення та удосконалення законодавства, з огляду на швидко змінюючіся технологічні та екологічні умови, визначає стратегічний напрям для подальших дій та розвитку в даній сфері. Спільні зусилля всіх зацікавлених сторін, зокрема урядів, науково-екологічних організацій та громадськості, необхідні для досягнення ефективного та сталого природоохоронного законодавства.

ВИСНОВКИ

У ході написання даної кваліфікаційної роботи було досягнуто поставленої мети:

- У ході дослідження були оглянуті різні сервіси та плагіни, такі як Adult Blocker, ProCon Latte, Hate Speech Blocker, Perspective API та Репозиторій PersLab/ToxicCommentClassifier. Також були проаналізовані статичний, семантичний та морфологічний методи класифікації текстів. Після уважного аналізу вибір було зроблено на користь поєднання Наївного баєсівого класифікатора та Методу максимальної ентропії. Це дозволило підібрати найбільш підходящий метод для досягнення найкращих результатів.
- Для досягнення більшої ефективності було використано метод векторизації TF-IDF та Word Embeddings, що дозволило представити текст у вигляді числових векторів та виявляти семантичні зв'язки між словами. Ці методи були успішно впроваджені в програмній реалізації методу класифікації деструктивних текстових даних, що відобразилося на його точності та ефективності.
- Розроблений метод був протестований на реальних даних, де показав хорошу якість класифікації та надійність. Використання метрики F-1 міра дозволило оцінити точність моделі та підтвердити її ефективність.

Отже, на основі проведеного дослідження та розробки методу класифікації деструктивних текстових даних було досягнуто мету роботи. Застосування комбінації методів та уважний аналіз дозволили створити ефективний та точний метод для виявлення шкідливого контенту у текстах.

Список використаної літератури

1. Wulczyn, E., Thain, N., Dixon, L. (2017). Ex machina: Personal attacks seen at scale. WWW 2017.

2. Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y., Chang, Y. (2016). Abusive language detection in online user content. WWW 2016.

3. Dinakar, K., Reichart, R., Lieberman, H. (2011). Modeling the Detection of Textual Cyberbullying. ICWSM 2011.

4. Breitfeller, L., Trott, E., Zhang, C. (2019). Finding toxic language: A spaCy toxicity classifier.

5. Badjatiya, P., Gupta, S., Gupta, M., Varma, V. (2017). Deep learning for hate speech detection in tweets. WWW 2017.

6. Davidson T., Bhattacharya D., Weber I. Racial Bias in Hate Speech and Abusive Language Detection Datasets. ArXiv, 2020, abs/2005.12060.

7. Zhang, Qingquan, Jialin Liu, Zeqi Zhang, Junyi Wen, Bifei Mao, and Xin Yao. Mitigating Unfairness via Evolutionary Multi-objective Ensemble Learning. arXiv preprint arXiv:2210.16754 (2022).

8. Mozilla Developer Network. "Mozilla Plugin" WEB: https://developer.mozilla.org/en-US/docs/Mozilla/Plugins

9. Fung, Benjamin C.M., et al. "Text classification without negative examples revisit." IEEE Transactions on Knowledge and Data Engineering 28.1 (2015): c. 18-20.

10. Sivic, Josef (2009). "Efficient visual search of videos cast as text retrieval". IEEE Transactions on pattern analysis and machine intelligence, c. 591–605.

11. Rajaraman, A.; Ullman, J.D. (2011). "Data Mining". Mining of Massive Datasets. c. 1–17.

12. Breitinger, Corinna; Gipp, Bela; Langer, Stefan (2015). "Research-paper recommender systems: a literature survey". International Journal on Digital Libraries. с. 325–338.

13. Бучинський М.Я., Горик О.В., Чернявський А.М., Яхін С.В. ОСНОВИ ТВОРЕННЯ МАШИН / [За редакцією О.В. Горика, доктора технічних наук, професора, заслуженого працівника народної освіти України]. – Харків : Вид-во «НТМТ», 2017. — 448 с.

14. Brown, Dunstan (2012). "Morphological Typology". In Jae Jung Song (ed.). The Oxford Handbook of Linguistic Typology. с. 487–503.

15. Larasati, Sri. "The use of morphological analysis and part-of-speech tagging for improving Indonesian text categorization." Scientific Journal of Informatics 4, no. 2 (2017): с. 63-73.

16. Giannini, A. J. (2009); Semiotic and Semantic Implications of "Authenticity", Psychological Reports, 106(2): с. 611–612.

17. Winskel, Glynn (1993). The formal semantics of programming languages : an introduction. Cambridge, Mass.: MIT Press. p. xv. ISBN 978-0-262-23169-5.

18. Rogers, Anna; Kovaleva, Olga; Rumshisky, Anna (2020). "A Primer in BERTology: What We Know About How BERT Works" с. 842-866.

19. Yih, Scott Wen-tau, et al. "Semantic parsing via staged query graph generation: Question answering with knowledge base." arXiv preprint arXiv:2004.072138 (2020).

20. Kowsari, Kamran, et al. "HDLTex: Hierarchical deep learning for text classification." In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, с. 285-294. (2017).

21. Nguyen, Thien Hai, et al. "Ensemble of deep neural networks with noisy labeling for text classification (2018).

22. Subbian, Kamburugamuve, and Sam Prentice. "Combining word representations improves text classification." (2020).

23. Ganesh, Gayathri, Tanmoy Chakraborty, and Saptarshi Ghosh. "A survey: Natural language processing approach for analysing unstructured data." (2018): c.124-37.

24. McCallum, A. and Nigam K. «A Comparison of Event Models for Naive Bayes Text Classification», c. 41-48.

25. Caruana, R.; Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms.

26. Tibshirani R., Hastie T., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. — Verlag : Springer, (2013). — c. 746.

27. Michael Pazzani & Domingos, Pedro (1997) «On the optimality of the simple Bayesian classifier under zero-one loss». Machine Learning, c.103-137.

28. Kattan M , Demsar J, Mozina M, & Zupan B. (2004). «Nomograms for Visualization of Naive Bayesian Classifier», c.337—348.

29. Rish, Irina. "An empirical study of the naive Bayes classifier." IJCAI 2001 workshop on empirical methods in artificial intelligence 3 (2001): 41-46.

30. Rennie, Jason DM, et al. "Tackling the poor assumptions of naive bayes text classifiers."(2003) ICML. c. 23-28.

31. Xiang, Yuanyuan, et al. "Gaussian process classification for identifying migraine from non-cephalic photic stimuli induced pain: an exploratory study." Journal of neuroscience methods 260 (2016): c. 97-103.

32. Bajkova, A. T., (1992), The generalization of maximum entropy method for reconstruction of complex functions. c.313—320.

33. Bansal S. Comparison between the probabilistic and vector space model for spam filtering //International Journal of Computational Intelligence Techniques. – 2012. c. 82.

34. Deerwester, Scott, et al. "Indexing by latent semantic analysis." Journal of the American society for information science 41.6 (1990): c. 391-407.

35. McCallum, A.: Efficiently inducing features of conditional random fields (2003), c.52-57.

36. Lafferty, John, Andrew McCallum, and Fernando CN Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data." (2001).

37. Ivanciuc, Ovidiu; Applications of Support Vector Machines in Chemistry, in Reviews in Computational Chemistry, (2007), c. 291–400.

38. Ben-Hur, David, Siegelmann, Asa, Horn, Hava, and Vapnik, Vladimir; "Support vector clustering" (2001) Journal of Machine Learning Research, 2: c.125–137.

39. Jones K. S. A statistical interpretation of term specificity and its application in retrieval // Journal of Documentation : журнал. — MCB University : MCB University Press, 2004. — Т. 60, № 5. — с. 493-502.

40. Robertson, S. (2004). "Understanding inverse document frequency: On theoretical arguments for IDF". Journal of Documentation. c. 503–520.

41. Jurafsky, Daniel; H. James, Martin (2000). Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition.

42. Li, Yitan; Xu, Linli (2015). Word Embedding Revisited: A New Representation Learning and Explicit Matrix Factorization Perspective.

43. Gers, F. A.; Schmidhuber, J. (2001). "LSTM Recurrent Networks Learn Simple Context Free and Context Sensitive Languages.

44. Larose, Daniel T. (2014). Discovering Knowledge in Data: An Introduction to Data Mining. Hoboken, New Jersey: Wiley. c. 174–179.

45. Jothikumar, R. (2018). "Predicting Life time of Heart Attack Patient using Improved C4.5 Classification Algorithm". Research Journal of Pharmacy and Technology. c. 1951–1956.

46. Гущин, І. В.; Сич, Д. О. (2018). Аналіз впливу попередньої обробки тексту на результати текстової класифікації. с. 264–266.

47. Borderies, Olivier (24 January 2019). "Pythran: Python at C++ speed !".

48. Stroustrup, Bjarne (12 June 2020). "Thriving in a crowded and changing world: C++ 2006–2020".

ДОДАТОК А

Програмна реалізація методу класифікації деструктивних текстових матеріалів

```
import tkinter as tk
from tkinter import scrolledtext
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score
import numpy as np
def get_word_embeddings(text):
    pass
def classify_text():
    input_text = text_input.get("1.0", tk.END)
    processed_texts = []
    categories = []
    with open("info.txt", "r", encoding="utf-8") as file:
        for line in file:
            parts = line.strip().split(": ")
            categories.append(int(parts[0].split()[1]))
            vector = list(map(float, parts[1][1:-1].split(', ')))
            processed_texts.append(vector)
    tfidf_vectorizer = TfidfVectorizer()
    X_test_text = tfidf_vectorizer.fit_transform([input_text])
    text_embeddings = np.array(processed_texts)
    if text_embeddings.size != 0:
        X_test = np.hstack((X_test_text.toarray(), text_embeddings))
```

Продовження додатку А

```
 else:
        X_test = X_test_text
    prediction_nb = nb_classifier.predict(X_test)
    prediction_lr = lr_classifier.predict(X_test)
    f1_nb = f1_score([3], prediction_nb, average='macro')
    f1_lr = f1_score([3], prediction_lr, average='macro')
    result_text.delete("1.0", tk.END)
    result_text.insert(tk.END, "Прогноз за допомогою наївного баєсівського класифікатора:\n")
    result_text.insert(tk.END, f"Категорія: {categories[prediction_nb[0]]}\n")
    result_text.insert(tk.END, f"F1-міра: {f1_nb}\n\n")
    result_text.insert(tk.END, "Прогноз за допомогою методу максимальної ентропії:\n")
    result_text.insert(tk.END, f"Категорія: {categories[prediction_lr[0]]}\n")
    result_text.insert(tk.END, f"F1-міра: {f1_lr}\n\n")
def on_positive_click():
    input_text = text_input.get("1.0", tk.END)
    with open("info.txt", "a", encoding="utf-8") as file:
        file.write(f"Text {len(open('info.txt').readlines()) + 1} c:{selected_category.get()}: {input_text}\n")
    result_text.insert(tk.END, "Текст було успішно додано у файл info.txt.\n")
def on_negative_click():
    classify_text()
def paste_text(event):
    text = root.clipboard_get()
    text_input.insert(tk.END, text)
```

Продовження додатку А

```
root = tk.Tk()
root.title("Класифікація тексту")
text_input = scrolledtext.ScrolledText(root, width=40, height=5)
text_input.pack(fill=tk.BOTH, expand=True, side=tk.LEFT)
button_frame = tk.Frame(root)
button_frame.pack(side=tk.LEFT, fill=tk.Y)
classify_button = tk.Button(button_frame, text="Класифікувати", command=classify_text, bg="lightblue", relief=tk.RAISED)
classify_button.pack(fill=tk.X)
positive_button = tk.Button(button_frame, text="+", command=on_positive_click, bg="lightgreen", relief=tk.RAISED, width=2)
positive_button.pack(fill=tk.X)
negative_button = tk.Button(button_frame, text="-", command=on_negative_click, bg="pink", relief=tk.RAISED, width=2)
negative_button.pack(fill=tk.X)
result_text = scrolledtext.ScrolledText(root, width=40, height=5)
result_text.pack(fill=tk.BOTH, expand=True)
selected_category = tk.StringVar()
category_label = tk.Label(button_frame, text="Категорія:")
category_label.pack()
category_entry = tk.Entry(button_frame, textvariable=selected_category)
category_entry.pack()
root.mainloop()
```

ДОДАТОК Б

Програмна реалізація наповнення бази даних info.txt

```
import tkinter as tk
from tkinter import scrolledtext
from sklearn.feature_extraction.text import TfidfVectorizer
def process_text():
    input_text = text_input.get("1.0", tk.END)
    if input_text.strip() == "":
        result_text.delete("1.0", tk.END)
        result_text.insert(tk.END, "Помилка: Ви не ввели жодного тексту для обробки.\n")
        return
    # Розділення введеного тексту за комами
    text_category_pairs = [pair.strip() for pair in input_text.split("], [")]
    processed_texts = []
    categories = []
    for pair in text_category_pairs:
        # Розділення тексту на текст та категорію
        pair = pair.strip("[]")
        text_part, category_part = pair.rsplit(", {", 1)
        text_part = text_part.strip()
        category_part = category_part.replace("]", "[").strip()
        processed_texts.append(text_part)
        categories.append(category_part)
    # Векторизація текстів за допомогою TfidfVectorizer
    tfidf_vectorizer = TfidfVectorizer()
    X = tfidf_vectorizer.fit_transform(processed_texts)
```

Продовження додатку Б

```
    # Запис векторизованих текстів у файл
    with open("info.txt", "w", encoding="utf-8") as file:
        for i, (text_vector, category) in enumerate(zip(X, categories)):
            file.write(f"Text {i + 1} c:{category}: {text_vector}\n")
    result_text.delete("1.0", tk.END)
    result_text.insert(tk.END, "Тексти було успішно збережено у файл info.txt.\n")
def paste_text(event):
    text = root.clipboard_get()
    text_input.insert(tk.END, text)
# Створення графічного інтерфейсу
root = tk.Tk()
root.title("Обробка тексту")
text_label = tk.Label(root, text="Введіть текст та його категорію (формат: [текст, {категорія}], ...):")
text_label.pack()
text_input = scrolledtext.ScrolledText(root, width=40, height=5, wrap=tk.WORD)
text_input.pack(fill=tk.BOTH, expand=True)
text_input.bind("<Command-v>", paste_text)  # Обробник події для вставки тексту
process_button = tk.Button(root, text="Обробити текст", command=process_text)
process_button.pack()
result_text = scrolledtext.ScrolledText(root, width=40, height=5)
result_text.pack()
root.mainloop()
```

ДОДАТОК В
Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

КВАЛІФІКАЦІЙНА РОБОТА ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР» НА ТЕМУ:

МЕТОД КЛАСИФІКАЦІЇ ДЕСТРУКТИВНИХ ТЕКСТОВИХ ДАНИХ

Виконавець:
студент групи БІ-241М **Олександр БІЛИЙ**
Керівник:
д.т.н., доцент, професор Людмила ТЕРЕЙКОВСЬКА

1

Мета роботи

Розробка та дослідження методу класифікації деструктивних текстових даних з метою підвищення інформаційної безпеки в онлайн-середовищі. Робота спрямована на досягнення наступних завдань:

- Аналіз існуючих методів та підходів до класифікації деструктивних текстових даних з метою вибору найбільш підходящого методу для досягнення найкращих результатів.
- Розробка ефективного та точного алгоритму класифікації деструктивних текстових даних, який здатний автоматично визначати та класифікувати шкідливий контент в текстах.
- Програмна реалізація методу класифікації деструктивних текстових даних за використання інноваційних методів машинного навчання, обробки природної мови та штучного інтелекту та експериментальна верифікація отриманих результатів.

2

Продовження додатку В

Актуальність

Зростання впливу інтернету та соціальних мереж призвело до збільшення кількості деструктивних текстових матеріалів, які можуть завдати шкоди індивідам, організаціям та громадським інтересам.

Методи класифікації деструктивних текстових даних можуть допомогти соціальним мережам виявляти та блокувати шкідливий контент, забезпечуючи кращу інформаційну безпеку для користувачів. Для багатьох інтернет-платформ інформаційна безпека та захист від деструктивного контенту стали пріоритетними завданнями. Зростання потужності штучного інтелекту та машинного навчання відкриває нові можливості для розробки більш точних та автоматизованих методів класифікації деструктивних текстових даних.

Ця робота важлива для забезпечення інформаційної безпеки, збереження довіри та зручності користувачів у цифровому світі і має потенціал позитивно вплинути на суспільство в цілому.

3

Об'єкт дослідження: процеси розпізнавання деструктивних текстових даних в онлайн-середовищі.

Предмет дослідження: метод класифікації деструктивних текстових даних.

Новизна одержаних результатів: отримав подальший розвиток метод класифікації деструктивних текстових даних, що за рахунок адаптації параметрів, роботи з інноваційними методами машинного навчання, обробки природної мови та штучного інтелекту дозволяє підвищити ефективність розпізнавання деструктивних текстових даних.

4

Продовження додатку В

Практична цінність:

- розроблено інструмент для класифікації деструктивних текстових даних;
- розроблена методика проведення експерименту з використанням розробленого програмного засобу класифікації деструктивних текстових даних.

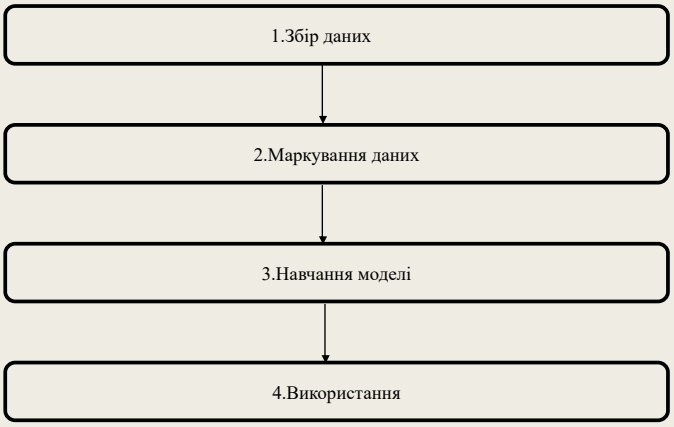
Галузь застосування: розроблений метод класифікації деструктивних текстових даних може знайти своє застосування в соціальних мережах, допомагаючи виявляти та блокувати образливі коментарі і забезпечувати безпеку користувачів, а також для медіа-ресурсів та новинних сайтів для фільтрації шкідливого контенту.

Методи дослідження: базуються на використанні мови Phyton для роботи з інноваційними методами машинного навчання, обробки природної мови та штучного інтелекту для розробки методу класифікації деструктивних текстових даних.

Апробація Білий О.О. Проблематики класифікації деструктивних текстових даних// Живучість та резильєнтність – 2023: міжнародна науково-практична конференція 19 жовтня 2023 р.: тези доповіді. – К., 2023. – С.142-143.

5

ТИПОВА СХЕМА КЛАСИФІКАЦІЇ ДЕСТРУКТИВНИХ ТЕКСТОВИХ ДАНИХ



6

Продовження додатку В

Проблематика наявних рішень класифікації деструктивних текстових даних

1. **Точність і Надійність.** Деякі моделі можуть мати тенденцію до неправильної класифікації, особливо коли мова або контекст використання текстів змінюються або стають складнішими для розуміння. Це може призвести до помилкових визначень, коли деякі текстові дані неправильно класифікуються як деструктивні або навпаки.
2. **Нездатність адаптуватися до змін.** Проблема полягає в тому, що існуючі методи класифікації деструктивних текстових даних, таких як спам, образливі коментарі або фейкові новини, часто ґрунтуються на певних статичних правилах та алгоритмах. Однак зміни в мові, контексті та способах вираження деструктивності текстів стають все більш складними й варіативними.
3. **Висока вартість і трудомісткість**. Багато існуючих методів класифікації деструктивних текстових даних вимагають значних витрат і зусиль. Це може ускладнити їхнє впровадження в реальних умовах.

7

Порівняльний аналіз плагінів для блокування деструктивних текстових матеріалів

Характеристика	Hate Speech Blocker	ProCon Latte	Adult Blocker
Призначення	Виявлення та блокування проявів мови ненависті, образ, залякування	Фільтрація занадто упереджених або радикальних думок	Виявлення та приховування порнографічних зображень, сексуальних текстів
Цільова платформа	Соцмережі, коментарі	Форуми, сайти новин	Будь-які вебсайти, пошукові запити
Метод детекції	Машинне навчання, аналіз тексту	Аналіз сентиментів та аргументів	Розпізнавання образів, чорні списки слів
Оновлення	Регулярні онлайн оновлення	Ручні оновлення розробником	Регулярні оновлення великого списку
Точність	Дуже точний, є ризик помилок	Може пропустити деякі випадки	Висока, але може блокувати не всі випадки
Налаштування	Фільтри і слова налаштовні користувачем	Сила фільтрації налаштовна керівником сайту	Заборонені слова та домени налаштовні користувачем
Мови	Багатомовний, в т.ч. українська	Англійська та європейські	Незалежний від мови, використовує образи/слова

8

Продовження додатку В

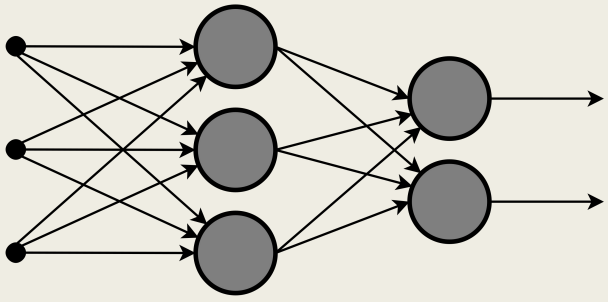
Порівняння методів роботи з текстовими матеріалами

Метод	Особливості	Переваги	Недоліки
Статистичний	Базується на аналізі частоти слів та моделі "мішка слів"	Простий та швидкий; дозволяє виявити поверхневі шаблони	Бракує семантичного розуміння; чутливий до сленгу та обфускації
Морфологічний	Вивчає внутрішню морфологічну структуру слів, включаючи тегування ЧМ	Надає більш детальний контекст; добре для мов зі складною морфологією	Вимагає розробки лінгвістичних ресурсів; обмежена семантика
Семантичний	Досліджує контекстне значення і зв'язки між елементами за допомогою вбудовувань та онтологій	Покращує розуміння завдяки семантиці	Вимагає розробки ресурсів; нижча інтерпретаційна здатність ніж статистика
Комбінований	Поєднує різні лінгвістичні підходи для створення супервекторів ознак	Використовує сильні сторони окремих підходів; досягає найкращих результатів	Складніший за окремі методи; залежить від якості складових

9

Машинне навчання та обробка природної мови

Застосуванням алгоритмів машинного навчання до обробки природної мови можна значно полегшити та прискорити роботу з великими обсягами текстової інформації, забезпечивши точні та швидкі результати в класифікації текстів у різних галузях, що робить цей підхід дуже важливим у сучасному світі інформації та даних.



10

Продовження додатку В

МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ МЕТОДУ КЛАСИФІКАЦІЇ ДЕСТРУКТИВНИХ ТЕКСТОВИХ ДАНИХ (1)

$$P(y|x) = \frac{1}{Z(x)} \exp\left(\sum_{i=1}^{m} w_i f_i(x, y)\right)$$

w_i – вектор ваг;

f_i – функція залежності, яка враховує взаємозв'язок між міткою і даними;

$\sum_{i=1}^{m} w_i f_i(x, y)$ – підсумування усіх результатів функцій залежностей

x – вектор даних;

y – вектор міток;

m – кількість елементів;

$$c_m = \underset{c \epsilon C}{\operatorname{argmax}}\left[logP(c) + \sum_{k=1}^{n} logP(t_k|c)\right]$$

c_m - категорія, яка максимізує вираз.

$\underset{c \epsilon C}{\operatorname{argmax}}$ - оператор, що визначає значення c, яке максимізує вираз.

$logP(c)$ - логарифм ймовірності вибору категорії c.

$logP(t_k|c)$ - логарифм умовної ймовірності слова (або ознаки) t_k у вибраній категорії c.

11

МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ МЕТОДУ КЛАСИФІКАЦІЇ ДЕСТРУКТИВНИХ ТЕКСТОВИХ ДАНИХ (2)

$$TF(t, d) = \frac{\text{кількість разів, коли слово } t \text{ зустрічається в документі } d}{\text{загальна кількість слів у документі } d}$$

$$DF(t) = \log\left(\frac{\text{кількість всіх документів у корпусі}}{\text{кількість документів,які містять слово } t} + 1\right)$$

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

TF-IDF векторизація

$$h_t = LSTM(x_t, h_{t-1})$$

$$\text{Loss} = -\sum_{i=1}^{N} y_i * \log(\hat{y}_i)$$

Word Embeddings

12

Продовження додатку В

ВИЗНАЧЕННЯ ЕФЕКТИВНОСТІ ЗАПРОПОНОВАНИХ РІШЕНЬ

$$\text{Entropy (S)} = -\sum_{i=1}^{p} p_i log_2(p_i)$$

$$\text{Average Entropy after Split} = \sum_{categories} \frac{|S_{feature,\,category}|}{|S|} \times Entropy(S_{feature,\,category})$$

$$\text{Information Gain} = \text{Entropy(S)} - \text{Average Entropy after Split}$$

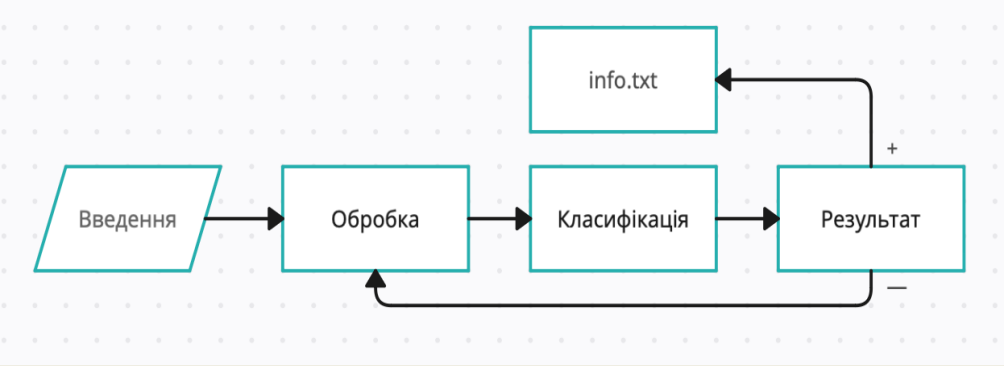
$$\text{Precision} = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$\text{Recall} = \frac{True\ Positives}{True\ Positives + False\ Negative}$$

$$\text{F1 score} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

13

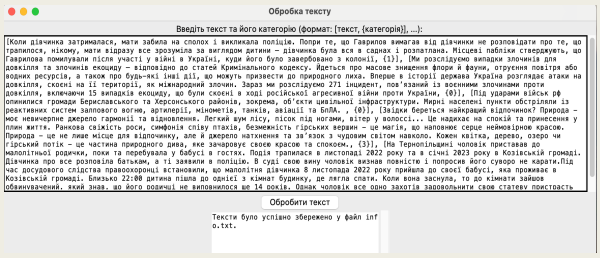
АРХІТЕКТУРА СИСТЕМИ КЛАСИФІКАЦІЇ ДЕСТРУКТИВНИХ ТЕКСТОВИХ ДАНИХ



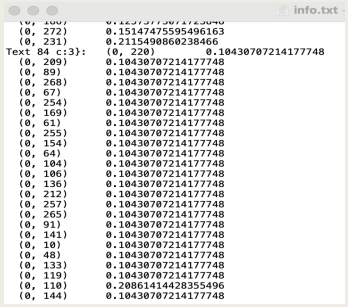
14

Продовження додатку В

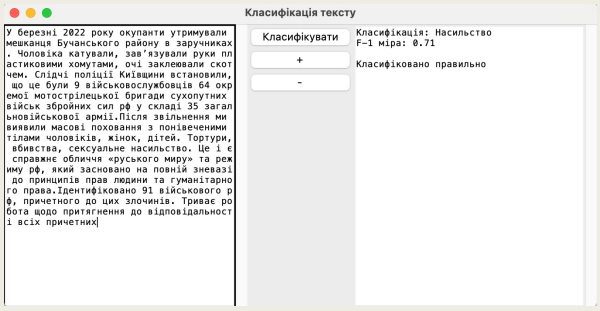
СТВОРЕННЯ БАЗИ ДАНИХ INFO.TXT ТА КЛАСИФІКАЦІЯ



Інтерфейс заповнення заповнення info.txt



Вміст info.txt



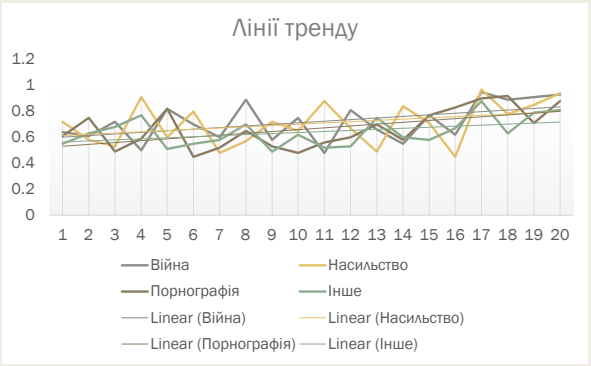
Процес класифікації

15

ЕКСПЕРЕМЕНАЛЬНІ ДОСЛІДЖЕННЯ

Спроба	Війна	Насильство	Порнографія	Інше
1	0.64	0.72	0.61	0.55
2	0.61	0.58	0.75	0.63
3	0.72	0.53	0.49	0.68
4	0.5	0.91	0.59	0.77
5	0.82	0.6	0.82	0.51
6	0.7	0.8	0.45	0.55
7	0.6	0.48	0.52	0.58
8	0.89	0.57	0.65	0.7
9	0.58	0.72	0.53	0.49
10	0.75	0.65	0.48	0.62
11	0.48	0.88	0.56	0.52
12	0.81	0.67	0.6	0.53
13	0.66	0.49	0.7	0.75
14	0.55	0.84	0.58	0.6
15	0.77	0.71	0.77	0.58
16	0.62	0.45	0.83	0.67
17	0.95	0.97	0.9	0.88
18	0.89	0.78	0.92	0.63
19	0.91	0.85	0.71	0.79
20	0.93	0.94	0.88	0.81

Табличне представлення показнику F-1 міра за 100 спроб класифікації



Графічне представлення результатів класифікації 100 текстів

16

Продовження додатку В

ВИСНОВКИ

У ході написання даної кваліфікаційної роботи було досягнуто поставленої мети та завдань:

- У ході дослідження були оглянуті різні сервіси та плагіни, такі як Adult Blocker, ProCon Latte, Hate Speech Blocker, Perspective API та Репозиторій PersLab/ToxicCommentClassifier. Також були проаналізовані статичний, семантичний та морфологічний методи класифікації текстів. Після уважного аналізу вибір було зроблено на користь поєднання Наївного баєсівого класифікатора та Методу максимальної ентропії. Це дозволило підібрати найбільш підходящий метод для досягнення найкращих результатів.
- Для досягнення більшої ефективності було використано метод векторизації TF-IDF та Word Embeddings, що дозволило представити текст у вигляді числових векторів та виявляти семантичні зв'язки між словами. Ці методи були успішно впроваджені в програмній реалізації методу класифікації деструктивних текстових даних, що відобразилося на його точності та ефективності.
- Розроблений метод був протестований на реальних даних, де показав хорошу якість класифікації та надійність. Використання метрики F-1 міра дозволило оцінити точність моделі та підтвердити її ефективність.

17