Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

зі спеціальності 121 – Інженерія програмного забезпечення

На тему: Розробка коммент-боту для автоматичного розпізнавання та фільтрації спаму в коментарях Telegram

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-20-1
_____ / Р. О. Фарафонов /

Керівник кваліфікаційної роботи _____ / Д. В. Швець /

Завідувач кафедри _____ / А. М. Стрюк /

Кривий Ріг
2024

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: бакалавр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри

_____ А. М. Стрюк

«____» _____ 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу

студента групи ІПЗ-20-1 Фарафонову Ростиславу Олеговичу

1. Тема: «Розробка коммент-боту для автоматичного розпізнавання та фільтрації спаму в коментарях Telegram» затверджена наказом по КНУ №275с від «15» квітня 2024 р.

2. Термін подання студентом закінченої роботи: «10» червня 2024 р.

3. Вихідні дані по роботі: розроблювана система повинна забезпечити автоматичне розпізнавання та фільтрацію спам-повідомлень у коментарях Telegram, підтримуючи навчання на основі прикладів, мінімальна точність розпізнавання спаму – 90%.

4. Зміст пояснювальної записки (перелік питань, що їх треба розробити):
виконати аналіз існуючих програмних рішень для фільтрації спаму, вибрати метод розпізнавання спаму, спроектувати автоматизовану систему розпізнавання спаму, здійснити програмну реалізацію розробленої системи, провести тестування розробленої системи.

5. Перелік ілюстративного матеріалу: функціональна схема, блок-схема розробленого алгоритму, схема взаємодії модулів системи, схема баз даних, знімки екранних форм.

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Пошук літературних джерел та огляд інтернет-ресурсів з заданої тематики	14.01.24 – 21.01.24
2	Аналіз існуючих методів вирішення проблеми	22.01.24 – 04.02.24
3	Формулювання актуальності роботи і постановка завдань	05.02.24 – 18.02.24
4	Оформлення матеріалів першого розділу роботи	19.02.24 – 03.03.24
5	Розробка функціональної системи та алгоритму програми	04.03.24 – 17.03.24
6	Оформлення матеріалів другого розділу роботи	18.03.24 – 31.03.24
7	Розробка баз даних, інтерфейсу програмного забезпечення, програмних модулів	01.04.24 – 21.04.24
8	Оформлення додатків	22.04.24 – 28.04.24
9	Тестування створення системи	29.04.24 – 12.05.24
10	Оформлення пояснювальної записки	13.05.24 – 08.06.24

Дата видачі завдання: «12» січня 2024 р.

Студент: _____ / Р. О. Фарафонов /

Керівник роботи: _____ / Д. В. Швець /

РЕФЕРАТ

СПАМ, МЕТОДИ РОЗПІЗНАВАННЯ, ФІЛЬТРАЦІЯ, МАШИННЕ НАВЧАННЯ, ОБРОБКА ПРИРОДНОЇ МОВИ, ТЕЛЕГРАМ-БОТ

Пояснювальна записка: 51 с., 1 табл., 25 рис., 2 дод., 20 джерел.

Метою кваліфікаційної роботи є створення програмного забезпечення для автоматичного розпізнавання та фільтрації спаму в коментарях месенджера Telegram за допомогою коммент-боту.

У роботі проведено аналіз сучасного стану виявлення та фільтрації спаму в соціальних мережах, включаючи методи фільтрації на основі правил, байєсівські фільтри, методи машинного навчання та алгоритми глибокого навчання. Виконано аналіз проблеми спаму в коментарях Telegram, розглянуто наслідки для користувачів та платформи.

Для розв'язання задачі було вибрано методи машинного навчання та обробки природної мови, які забезпечують високу точність і ефективність у виявленні та блокуванні спаму. Виконано аналіз існуючих аналогів, зокрема сучасних рішень для фільтрації спаму в Telegram та використання ботів у месенджерах.

Розроблено архітектуру програмного забезпечення коммент-боту, описано компоненти та їх взаємодію, включаючи модулі для збору та попередньої обробки даних, навчання моделей, інтеграції з Telegram API, управління налаштуваннями та генерації звітів.

Виконано тестування розробленого коммент-боту, оцінено ефективність роботи за допомогою відповідних метрик. Сформульовано рекомендації для подальшого розвитку та вдосконалення системи.

ABSTRACT

SPAM, DETECTION METHODS, FILTERING, MACHINE LEARNING, NATURAL LANGUAGE PROCESSING, TELEGRAM BOT

Explanatory Note: 51 pages, 1 table, 25 figures, 2 appendices, 20 references.

The aim of the qualification work is to develop software for automatic detection and filtering of spam in Telegram comments using a comment bot.

The study analyzes the current state of spam detection and filtering in social networks, including rule-based filtering methods, Bayesian filters, machine learning methods, and deep learning algorithms. The problem of spam in Telegram comments is analyzed, and the consequences for users and the platform are considered.

To solve the problem, machine learning and natural language processing methods were chosen, ensuring high accuracy and efficiency in detecting and blocking spam. Existing solutions were analyzed, including modern spam filtering solutions in Telegram and the use of bots in messengers.

The software architecture of the comment bot was developed, and the components and their interactions were described, including modules for data collection and preprocessing, model training, integration with the Telegram API, settings management, and report generation.

The developed comment bot was tested, and its performance was evaluated using appropriate metrics. Recommendations for further development and improvement of the system were formulated.

ЗМІСТ

ВСТУП

Соціальні мережі є невід'ємною частиною сучасного життя, вони дозволяють спілкуватися, ділитися інформацією та обмінюватися ідеями. Однак зараз, разом з розвитком соціальних мереж, зростає і проблема спаму, що може призвести до втрати конфіденційності, фінансів та інших негативних наслідків. Спам в коментарях може бути шкідливими для користувачів, тому що можуть містити небезпечні посилання, фішингові повідомлення або інший контент.

Виявлення та фільтрація спаму в коментарях Telegram є актуальною задачею, що потребує ефективного та ефективного механізму. Сучасні методи виявлення спаму в соціальних мережах часто базуються на правилах та фільтрах, які можуть бути не ефективними проти нових типів спаму.

Мета цієї роботи полягає в розробці комент-боту для автоматичного виявлення та фільтрації спаму в коментарях Telegram. Комент-бот повинен бути здатним виявляти спам-коментарі з високою точністю та ефективністю, що дозволить покращити безпеку та комфорт користувачів месенджера. Для досягнення цієї мети, буде використовуватися машинне навчання та обробка природної мови, що дозволить боту навчитися виявляти спам-коментарі на основі їхнього змісту та поведінки користувачів.

Галузь застосування результатів цієї роботи включає соціальні мережі, онлайн-комунікацію та інформаційну безпеку. Результати цієї роботи можуть бути використані для покращення безпеки та ефективності соціальних мереж, а також для розробки нових систем виявлення та фільтрації спаму.

1. АНАЛІЗ ПРОБЛЕМИ ТА ОБҐРУНТУВАННЯ ВИБРАНОГО НАПРЯМКУ

1.1 Аналіз сучасного стану виявлення та фільтрації спаму в соціальних мережах

Для виявлення та фільтрація спаму в соціальних мережах потребується постійний моніторинг та вдосконалення методів. Традиційні підходи часто виявляються недостатньо ефективними у боротьбі з новими формами спаму. Висока динаміка соціальних мереж та винахідливість спамерів роблять боротьбу зі спамом постійним викликом.

Сучасний спам у соціальних мережах не обмежується лише небажаними повідомленнями. Він включає фішинг-атаки, небезпечні посилання, фейкові новини та соціальний інжиніринг, які спрямовані на обман користувачів та збір їхніх особистих даних. Спамери активно використовують проксі-сервери, анонімні мережі та фальшиві облікові записи, щоб приховати свою ідентичність і уникнути блокування. Завдяки цьому ускладняється виявлення спаму і розробники змушені постійно оновлювати механізми захисту.

Завдяки інноваціям у галузі машинного та глибокого навчання, ми отримали нові інструменти для ефективного виявлення спаму. За допомогою машинного навчання аналізуються величезні масиви даних, які виявляють шаблони, характерні для спаму. Алгоритми глибокого навчання можуть бути натреновані на реальних прикладах спаму, що забезпечує високоточне виявлення, навіть, найменш очевидних форм спаму. Наприклад, вони здатні розпізнавати патерни в поведінці користувачів або в текстових повідомленнях, це дозволяє знижувати кількість хибнопозитивних результатів.

Однак, постійна еволюція методів спаму створює нові виклики. Розробляються нові способи обману, включаючи використання соціальних методів маніпуляції та розподілених атак. Використовуються тимчасові облікові

записи, генеруються унікальні повідомлення для кожної атаки або комбінуються різні методи, щоб зробити свої дії менш очевидними. Завдяки цьому потрібно постійно адаптуватися та враховувати нові види загроз.

Незважаючи на значний прогрес у цій галузі, виявлення та фільтрація спаму в соціальних мережах потребують постійного розвитку. Сучасні дослідження фокусуються на створенні адаптивних систем, які можуть швидко реагувати на нові загрози. Розробляються нові алгоритми поведінкових аналізів, інтеграцій з іншими технологіями безпеки, для створення більш надійного захисту.

1.2 Визначення проблеми

1.2.1 Опис проблеми спаму в коментарях Telegram

Спам в коментарях Telegram - це поширена проблема, що впливає на якість спілкування та досвід користувачів на платформі.

Спам-коментарі можуть мати різні форми, включаючи:

- Реклама
- Образливі висловлювання
- Шкідливі посилання
- Фішинг
- Віруси

Спам-коментарі в Telegram мають серйозні наслідки такі як, порушення приватності та безпеки користувачів, зниження довіри до платформи та погіршення досвіду користувачів. Крім того, повідомлення використовуються для поширення шкідливих програм та вірусів, що може призвести до серйозних наслідків для користувачів.

Окрема проблема полягає в тому, що відкритість платформи та легкість створення нових акаунтів сприяють поширенню спаму в коментарях.

Статистика свідчить про те, що спам-коментарі становлять до 30% усіх коментарів в Telegram, а деякі канали та групи зазнають навіть вищого рівня спаму. Тому необхідно розробити ефективне рішення для вирішення проблеми спаму в коментарях Telegram, щоб забезпечити безпечне та зручне спілкування на платформі.

1.2.2 Наслідки для користувачів та платформи

Спам-коментарі в Telegram можуть мати серйозні наслідки для користувачів та платформи, включаючи:

- Порушення приватності
- Зниження довіри
- Погіршення досвіду
- Втрата часу
- Погіршення репутації

Ці наслідки є серйозними та довготривалими, тому необхідно розробити ефективні механізми боротьби з спамом, щоб забезпечити безпечне та зручне спілкування в Telegram.

Якщо не вжити заходів для боротьби з спамом, то це може призвести до втрати довіри користувачів до платформи та зниження їхньої активності. Крім того, спам-коментарі можуть бути використані для поширення шкідливих програм та вірусів, що може призвести до серйозних наслідків для користувачів.

Окрім того, спам-коментарі можуть бути використані для маніпуляції з громадською думкою, поширення фейкових новин та дезінформації. Це може мати серйозні наслідки для суспільства в цілому, тому боротьба з спамом є дуже важливою задачею.

1.3 Аналіз існуючих аналогів

1.3.1 Сучасні рішення для фільтрації спаму в Telegram

Існують різні рішення для виявлення та блокування спаму в коментарях Telegram.

Наприклад:

- Ручна модерація
- Автоматичні боти
- Алгоритми машинного навчання
- Фільтри ключових слів
- Обробка природної мови
- Комбіновані методи

Для ефективної фільтрації спаму в Telegram використовуються різні підходи. Ручна модерація передбачає перевірку та видалення спаму адміністраторами вручну, це є ефективним, але дуже трудомістко. Автоматичні боти повинні миттєво реагувати на спам, але їх налаштування вимагає певних технічних знань. Алгоритми машинного навчання навчаються на великих наборах даних і можуть самостійно виявляти спам. Фільтри ключових слів блокують повідомлення на основі попередньо визначених слів і фраз. Обробка природної мови аналізує текст з урахуванням контексту і семантики для точнішого виявлення спаму. Комбіновані методи поєднують кілька підходів для досягнення максимальної ефективності.

1.3.2 Використання ботів у месенджерах

Перед розробкою програмного забезпечення важливо аналізувати існуючі аналоги, щоб виявити їх сильні та слабкі сторони, знайти можливості для покращення та уникнути поширених помилок. У контексті розробки комент-боту для фільтрації спаму в Telegram, розглянемо популярні інструменти, що вже використовуються для цієї мети, та їхні особливості.

Спочатку розглянемо SpamBot (Рис. 1.3.1) — популярний бот у Telegram, який бореться зі спамом. Він використовує базу даних спам-акаунтів і автоматично видаляє їхні повідомлення. Проте він діє лише проти вже відомих акаунтів і не дає можливості налаштування фільтрації під конкретні потреби.

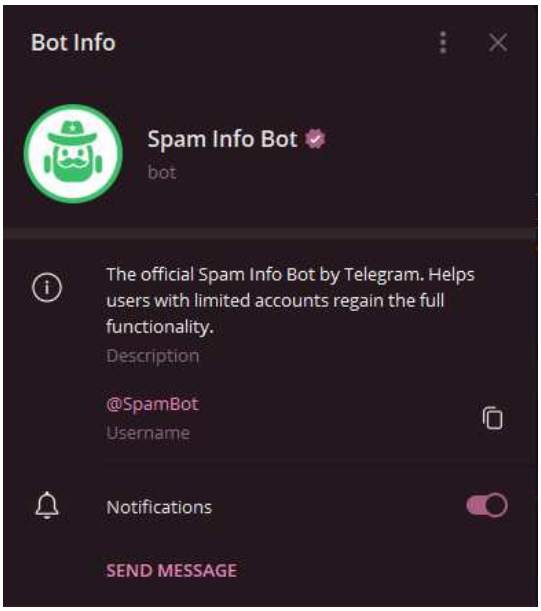


Рисунок 1.3.1 Бот SpamBot

Тепер розглянемо Combot (Рис. 1.3.2). Цей бот має кілька корисних функцій, таких як фільтрація спаму, аналіз активності групи та статистика. Проте його складне налаштування через багато функцій може бути складним для новачків, а деякі можливості доступні тільки за плату.

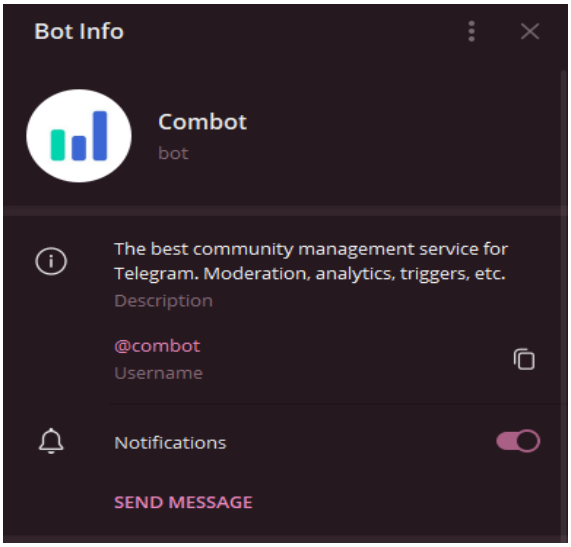


Рисунок 1.3.2 Бот Combot

І наостанок, розглянемо бота Shieldy (Рис. 1.3.3), який перевіряє нових учасників групи для захисту від спаму. Проте цей процес може бути незручним для новачків, а функціонал бота обмежений лише перевіркою нових учасників без додаткових можливостей для боротьби зі спамом.

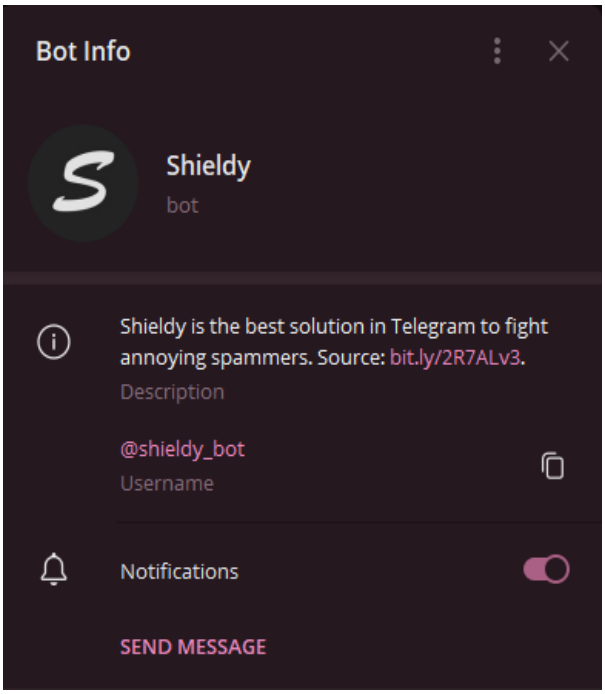


Рисунок 1.3.3 Бот Shieldy

Для створення комент-боту, який ефективно фільтруватиме спам у Telegram, важливо врахувати досвід існуючих рішень та інтегрувати сучасні технології, такі як машинне навчання. Це дозволить розробити адаптивний та універсальний інструмент з високою точністю виявлення спаму та зручністю для адміністраторів груп.

1.4 Формулювання актуальності і задач кваліфікаційної роботі

Telegram, як і інші месенджери, страждає від спамерських атак, що порушують безпеку та приватність користувачів. Однак, ефективні системи фільтрації спаму можуть бути рішенням цієї проблеми, забезпечуючи захист від шкідливого контенту та покращуючи досвід спілкування в месенджерах.

Які наслідки можуть мати спамерські атаки:

- Втрати даних та конфіденційності;
- Порушення безпеки облікових записів;
- Втрати часу та ресурсів на боротьбу з спамом;
- Погіршення репутації компаній та організацій, які використовують месенджери для спілкування з клієнтами.

Дослідивши проблему спамерських атак в месенджерах, зокрема в Telegram, визначено необхідність розробки ефективного інструменту фільтрації спаму, який допоможе захистити користувачів від шкідливого контенту та поліпшити загальний досвід спілкування в месенджері. Метою нашої роботи є створення такого інструменту, який би допоміг вирішити цю проблему.

Для досягнення цього необхідно вирішити наступні задачі:

- Розробити структурну архітектуру комент-боту, включаючи основні компоненти та модулі;
- Описати функціональну схему комент-боту;
- Оцінити ефективність розробленого комент-боту в боротьбі з спамом в Telegram;
- Визначити можливі способи покращення ефективності комент-боту;

2. МЕТОДОЛОГІЯ, ВИКОРИСТУВАНА У РОЗРОБЦІ КОМЕНТ-БОТУ

2.1 Вибір методів та алгоритмів розпізнавання спаму

Вибір ефективних методів та алгоритмів розпізнавання спаму є ключовим етапом у розробці комент-боту для Telegram. Мета полягає у створенні системи, яка точно і швидко виявляє та блокує спамові повідомлення. Аналізувалися різні методи машинного навчання та обробки природної мови для досягнення цієї мети.

1. Байєсівські фільтри: Використовуються ймовірнісний підхід для класифікації повідомлень на основі попередніх даних. Швидкі і точні, але ефективність може знижуватися при зміні тактик спамерів.

2. Підтримувальні векторні машини (SVM): Потужний метод для класифікації текстових даних, який ефективно розрізняє спам і легітимні повідомлення на основі багатовимірних характеристик тексту. Потребує ретельного налаштування, але забезпечує високу точність.

3. Нейронні мережі: Глибокі нейронні мережі (DNN) обробляють великі обсяги даних і виявляють складні патерни у тексті. Вони адаптивні і можуть покращувати ефективність з часом.

4. Обробка природної мови (NLP): Аналізує семантичний зміст повідомлень для точного виявлення спаму. Алгоритми, такі як BERT або GPT, враховують контекст і розпізнають спам навіть у складних випадках.

5. Гібридні методи: Поєднання різних методів (байєсівські фільтри, SVM, нейронні мережі і NLP) для створення надійних систем розпізнавання спаму.

2.2 Машинне навчання та обробки природної мови, використовуваних у боті

Для розробки ефективного комент-бота було використано методи машинного навчання та обробки природної мови:

Використовуємо нейронну мережу з двома лінійними шарами та сигмоїдною функцією активації для тренування моделі. Ця модель дозволяє класифікувати текстові повідомлення на спам та не спам з високою точністю.

Користуючись наступними методами обробки природної мови підготовляємо текст до тренування моделі.

Метод CountVectorizer дозволяє виділити ознаки з текстових повідомлень та представити їх у вигляді числових векторів. Модель машинного навчання ефективніше класифікує текстові повідомлення.

За допомогою методу Лемматизація слова нормалізуємо слова до їх основної форми, це зменшує розмірність простору ознак та покращує результати класифікації.

Вилучення стоп-слів видаляє стоп-слова з тексту, це покращує результати класифікації шляхом зменшення шуму в даних.

Використання цих методів машинного навчання та обробки природної мови дозволило розробити ефективний комент-бот, який може класифікувати коментарі з високою точністю.

2.3 Процедури збору та попередньої обробки даних

Для розробки ефективного комент-бота було проведено наступні процедури збору та попередньої обробки даних:

1. Збір даних
2. Попередня обробка даних
3. Навчання моделей
4. Інтеграція з Telegram API
5. Тестування

Зібрано великий обсяг даних, який включав як спам-повідомлення, так і легітимні повідомлення. Джерелами даних були відкриті бази даних спаму, історії чатів та спеціально створені набори даних.

Далі нормалізуємо та токенізуємо дані від зайвих символів. Проведена робота з видалення HTML-тегів, конвертацію тексту до нижнього регістру, видалення стоп-слів та лематизацію.

Навчаємо алгоритми машинного навчання на підготовлених даних. Для цього використаємо різні методи валідації, такі як крос-валідація, для оцінки точності моделей та налаштування їх параметрів.

За допомогою BotFather створюємо бота у Telegram, після чого інтегруємо алгоритми розпізнавання спаму з API Telegram. Завдяки цьому бот автоматично обробляє повідомлення в реальному часі.

Проводимо тестування бота на різних наборах даних для оцінки його ефективності. Це включало як автоматизовані тести, так і ручне тестування з боку користувачів, щоб забезпечити високу точність і швидкість роботи бота.

2.4 Метрика оцінки ефективності бота

Для оцінки ефективності бота в розпізнаванні спаму в коментарях Telegram було використано ряд метрик, які дозволяють зрозуміти, наскільки добре він працює і якість його роботи.

Однією з ключових метрик є точність (accuracy). Вона визначає відсоток правильно класифікованих спамових і неспамових коментарів відносно загальної кількості коментарів у тестовому наборі даних. Висока точність свідчить про те, що бот правильно класифікує більшість коментарів, що надходять до нього.

Додатково, для оцінки ефективності використали такі метрики, як чутливість (recall) та специфічність (specificity). Чутливість визначає відсоток спамових коментарів, які були правильно визначені ботом, а специфічність — відсоток неспамових коментарів. Ці метрики допомагають зрозуміти, як добре бот впорався з розпізнаванням спаму та неспаму, зокрема, його здатність правильно визначати спам без заборони неспамових коментарів і навпаки.

Крім того, використали матриця плутанини (confusion matrix), яка показує кількість правильно та неправильно класифікованих коментарів для кожного класу (спам і неспам). Ця матриця дає змогу детальніше оцінити роботу бота та виявити можливі області для поліпшення.

Використання всіх цих метрик дозволяє об'єктивно оцінити ефективність роботи бота в реальних умовах та зробити необхідні корективи для покращення його роботи.

3. АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Загальна архітектура комент-боту

Комент-бот є складним програмним продуктом, призначеним для забезпечення високої ефективності виявлення спаму та поліпшення якості спілкування в Telegram. Основна мета архітектури бота (Рис. 3.1.1) полягає у забезпеченні надійної фільтрації спаму з мінімальним впливом на звичайні коментарі користувачів, а також у забезпеченні масштабованості та гнучкості системи.

Ключовими принципами архітектури є:

- Модульність: Розділення системи на окремі модулі, кожен з яких виконує специфічні функції.
- Інкапсуляція: Зменшення взаємозалежностей між модулями для спрощення тестування та підтримки.
- Масштабованість: Забезпечення можливості збільшення потужності обробки без значних змін у структурі системи.
- Гнучкість: Легке налаштування та адаптація до нових вимог або мовних середовищ.

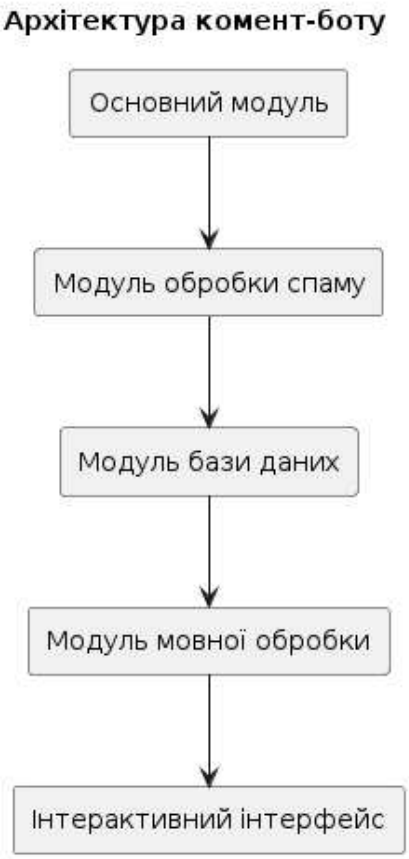


Рисунок 3.1.1 Архітектура комент-боту

Архітектура комент-боту включає наступні ключові компоненти:

1. Основний модуль: Відповідає за інтеграцію з API Telegram, обробку вхідних повідомлень та виклик відповідних модулів для фільтрації та обробки даних.
2. Модуль обробки спаму: Використовує моделі машинного навчання для аналізу текстів коментарів та визначення, чи є вони спамом.
3. Модуль бази даних: Зберігає правила фільтрації, історію оброблених повідомлень, налаштування користувачів та журнали подій.
4. Модуль мовної обробки: Виявляє мову коментаря та обирає відповідну модель для аналізу.
5. Інтерактивний інтерфейс: Забезпечує користувачам можливість налаштування фільтрації та отримання допомоги через інтерактивні клавіатури.

3.2 Компоненти та їх взаємодія

Комент-бот для Telegram складається з кількох ключових компонентів (Рис. 3.2.1 та Рис. 3.2.2), кожен з яких виконує окрему роль, що забезпечує ефективну роботу всього програмного забезпечення.

Нижче наведено опис основних компонентів:

- Кореневі файли:
    - `bot.py`;
    - `config.py`;
    - `settings.py`;
    - `requirements.txt`;
    - `README.md`;
- PostgreSQL зберігає дані про користувачів, повідомлення, правила фільтрації, налаштування та журнали дій. Використовується ORM для простого доступу.
- Модулі класифікації спаму з алгоритмами машинного навчання. `spam_classifier.py` перевіряє кожне повідомлення.
- Модулі багатомовності дозволяють взаємодію з користувачами різними мовами. `language_detection.py` визначає мову користувача і завантажує переклади.
- `user_commands.py` та `base_commands.py` обробляють команди користувачів, перенаправляючи їх до відповідних обробників. Обробники зворотних викликів обробляють події інлайн-клавіатур.
- Модулі інлайн-клавіатур створюють інтерфейси для взаємодії з користувачем та мають обробники зворотних викликів.

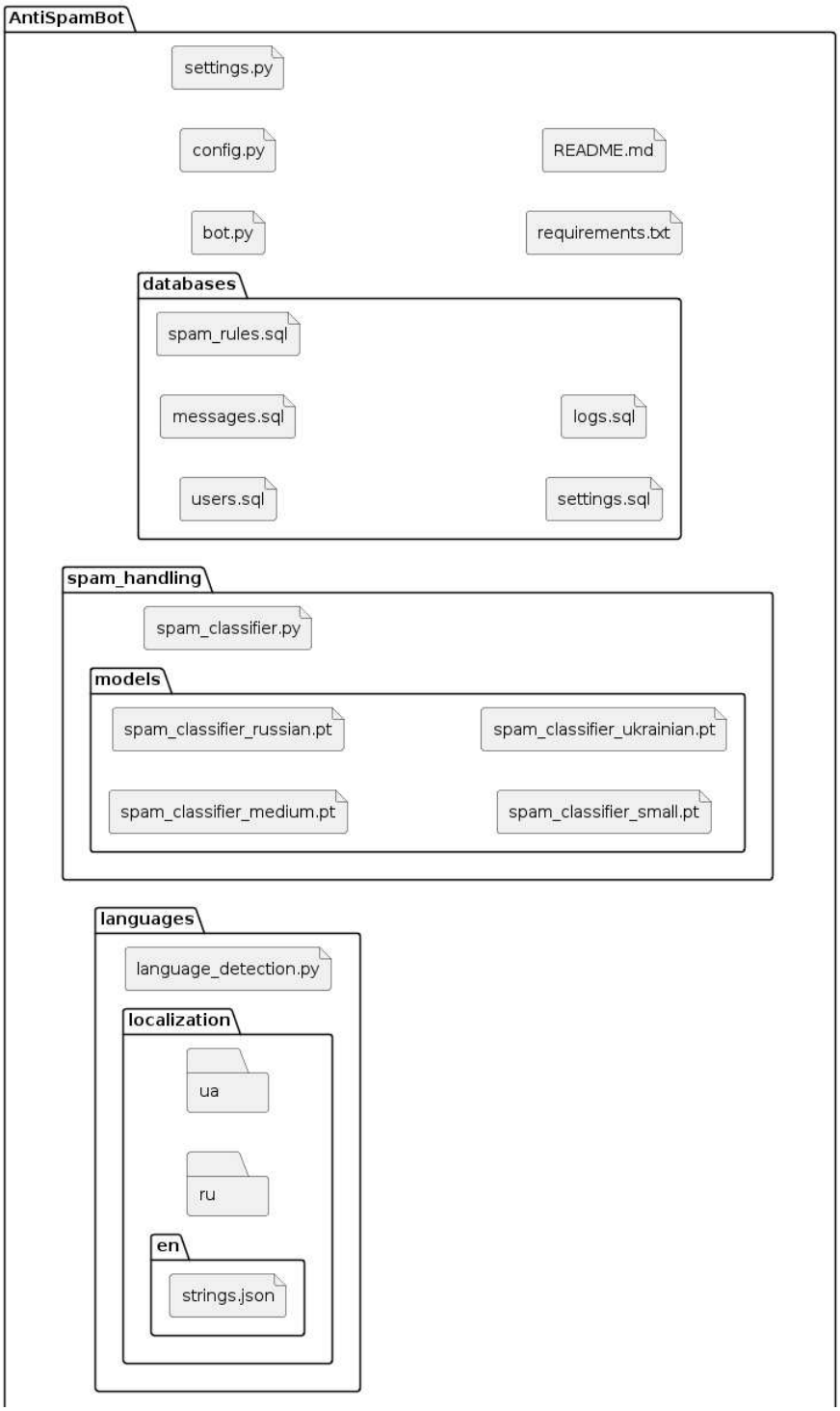


Рисунок 3.2.1 Компоненти та модулі бота

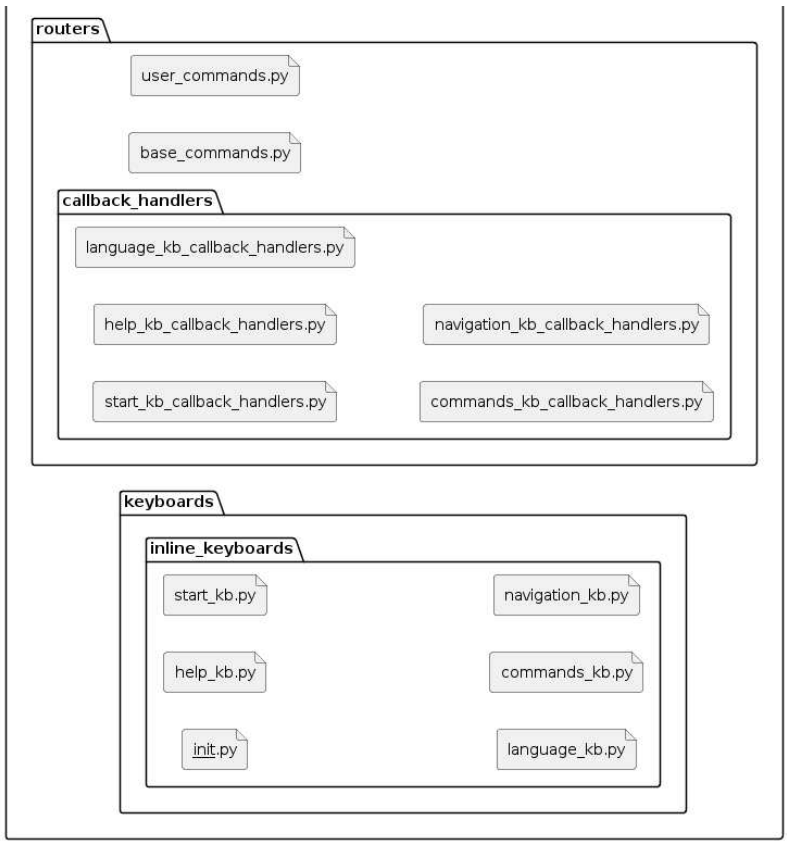


Рисунок 3.2.2 Продовження Компонентів та модулів бота

Взаємодія компонентів:

- При запуску `bot.py` ініціалізує конфігурації з `config.py` та `settings.py`, підключається до бази даних, завантажує модулі для обробки спаму та мовних налаштувань.
- Коли користувач взаємодіє з ботом, команди обробляються маршрутізаторами та відповідними обробниками, які викликають модулі для перевірки повідомлень на спам.
- Результати обробки спаму, налаштувань та інших дій зберігаються у базі даних.
- Взаємодія з користувачем здійснюється через інлайн-клавіатури, що дозволяють швидко та зручно керувати функціями бота.

Ця взаємодія забезпечує цілісність роботи комент-боту, дозволяючи йому ефективно виконувати свої функції з фільтрації спаму та надавати якісний користувацький досвід.

3.3 Опис потоку даних та обробки

Потік даних та обробки в комент-боті для Telegram відбувається наступним чином, як показано на діаграмі архітектури комент-боту:

Потік даних та обробки в комент-боті:

1. Отримання повідомлення
2. Передача повідомлення до модуля обробки
3. Попередня обробка тексту
4. Передача повідомлення до модуля класифікації
5. Класифікація повідомлення
6. Передача повідомлення до модуля фільтрації
7. Блокування спам-повідомлення

Основний модуль отримує нове повідомлення від користувача в Telegram, забезпечуючи початковий зв'язок між користувачем та ботом. Після отримання повідомлення передається до модуля обробки спаму для подальшої перевірки та аналізу. Модуль мовної обробки виконує нормалізацію та токенізацію, готуючи текст для подальшої класифікації. Після попередньої обробки повідомлення передається до модуля обробки спаму, де відбувається класифікація на основі алгоритмів машинного навчання.

Модуль обробки спаму використовує алгоритм машинного навчання, який був навчений на великому обсязі даних, для визначення повідомлення спам/не спам. Залежно від результату класифікації повідомлення передається до модуля фільтрації: спам-повідомлення блокуються або видаляються, не спам зберігаються в базі даних. Модуль фільтрації відповідає за блокування або видалення спам-повідомлень, забезпечуючи чистоту та якість комунікації в чаті.

3.4 Структура даних або бази даних (БД)

3.4.1 Опис структури даних

Для забезпечення ефективного функціонування коммент-боту для Telegram необхідно створити структуровану базу даних (Рис. 3.4.1, 3.4.2, 3.4.3, 3.4.4, 3.4.5, 3.4.6), яка зберігатиме всі необхідні дані про повідомлення, користувачів та налаштування.

Для цього було обрано реляційну базу даних PostgreSQL, яка дозволяє створювати складні структури даних та забезпечує високу продуктивність і надійність.

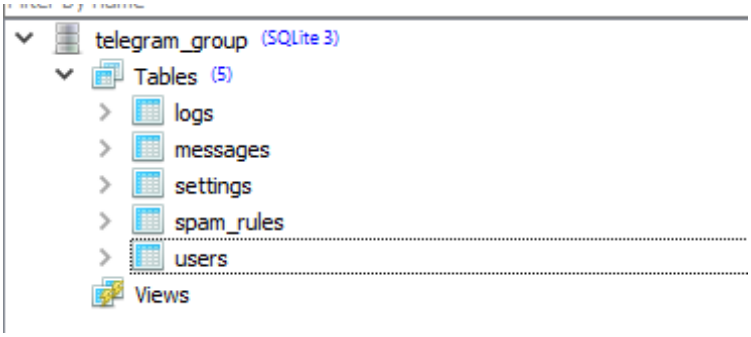


Рисунок 3.4.1 Структура БД

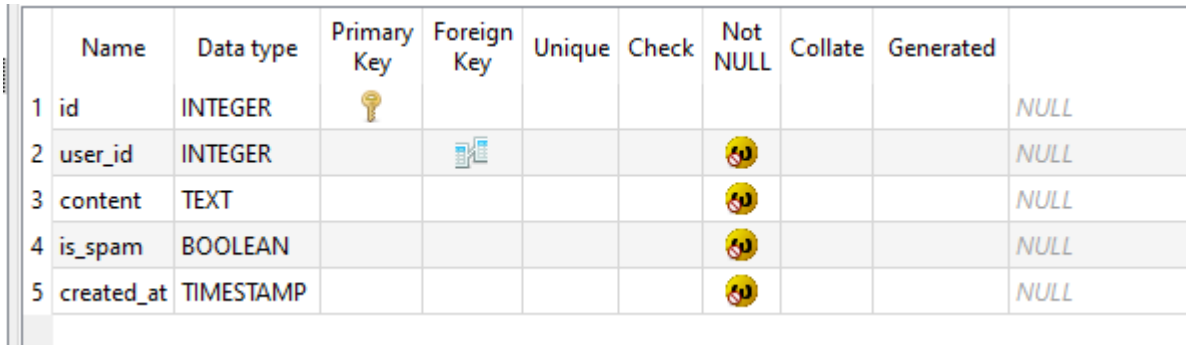
Основні таблиці бази даних (Рис. наведені нижче:

1. users
   - id (PRIMARY KEY);
   - username;
   - is_admin;
   - created_at;

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	id	INTEGER								NULL
2	username	TEXT								NULL
3	is_admin	BOOLEAN								NULL
4	created_at	TIMESTAMP								NULL

Рисунок 3.4.2 Вигляд таблиці users

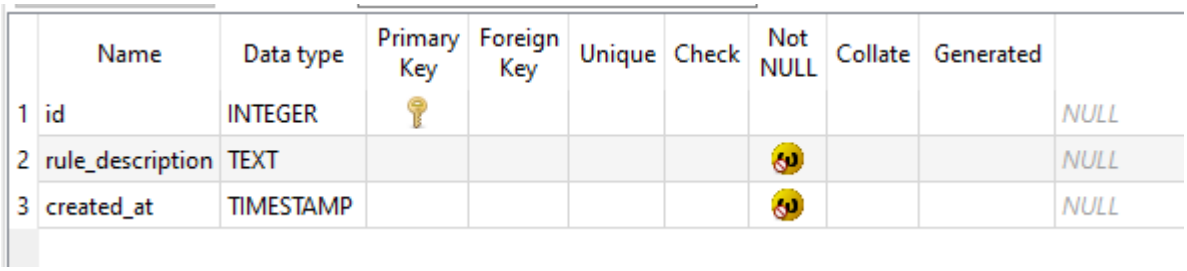
2. messages
    - id (PRIMARY KEY);
    - user_id (FOREIGN KEY);
    - content;
    - is_spam;
    - created_at;



	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	id	INTEGER								NULL
2	user_id	INTEGER								NULL
3	content	TEXT								NULL
4	is_spam	BOOLEAN								NULL
5	created_at	TIMESTAMP								NULL

Рисунок 3.4.3 Вигляд таблиці messages

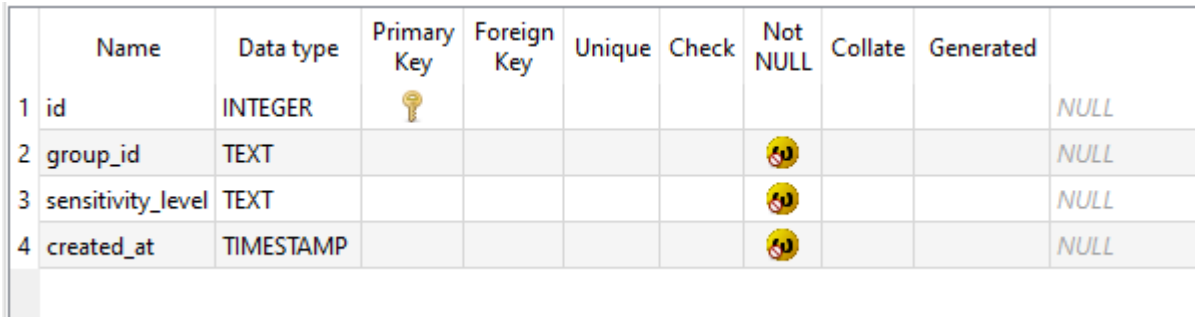
3. spam_rules
    - id (PRIMARY KEY);
    - rule_description;
    - created_at;



	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	id	INTEGER								NULL
2	rule_description	TEXT								NULL
3	created_at	TIMESTAMP								NULL

Рисунок 3.4.4 Вигляд таблиці spam_rules

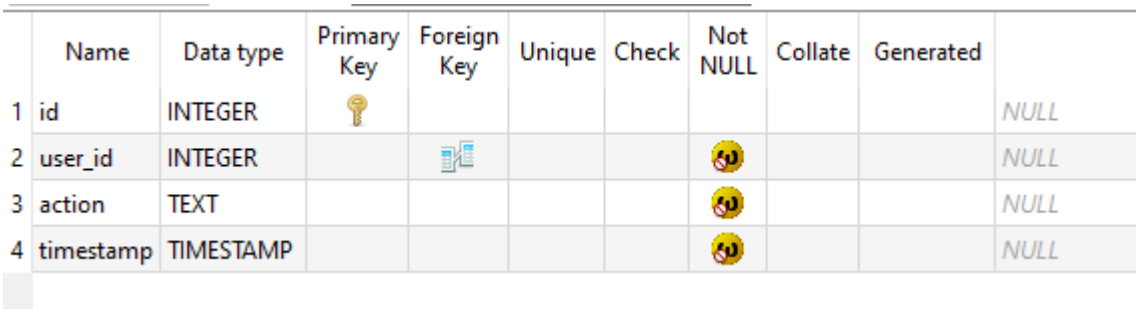
4. settings
    - id (PRIMARY KEY);
    - group_id;
    - sensitivity_level;
    - created_at;



	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	id	INTEGER								NULL
2	group_id	TEXT								NULL
3	sensitivity_level	TEXT								NULL
4	created_at	TIMESTAMP								NULL

Рисунок 3.4.5 Вигляд таблиці settings

5. logs
    - id (PRIMARY KEY);
    - user_id (FOREIGN KEY);
    - action;
    - timestamp;



	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	id	INTEGER								NULL
2	user_id	INTEGER								NULL
3	action	TEXT								NULL
4	timestamp	TIMESTAMP								NULL

Рисунок 3.4.6 Вигляд таблиці logs

3.4.2 Модель бази даних

Модель бази даних (Рис. 3.4.7) включає в себе основні таблиці, їх поля та зв'язки між ними.

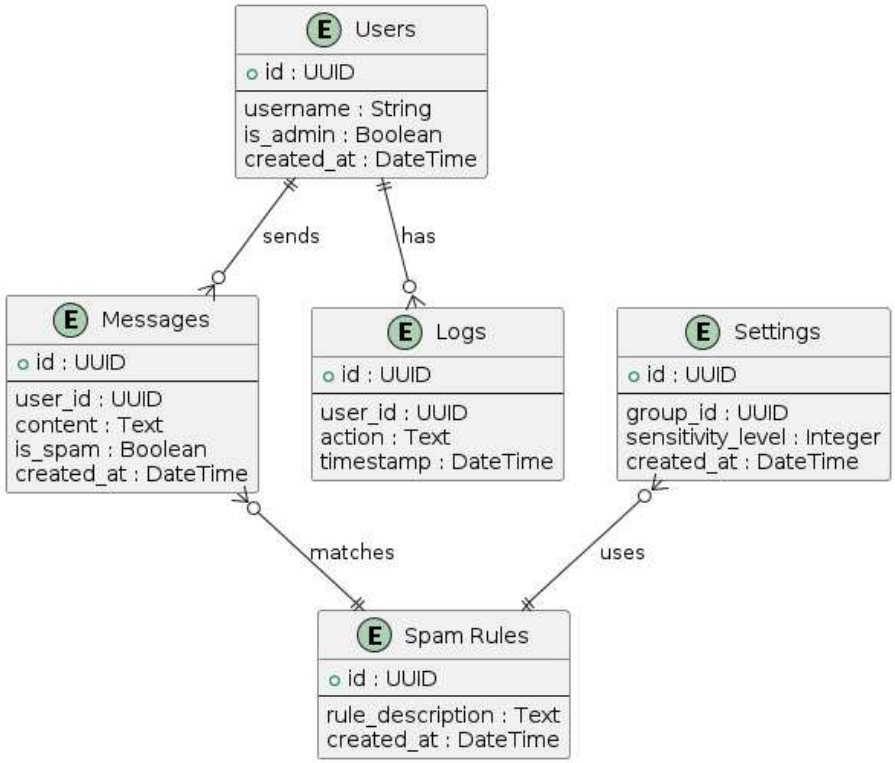


Рисунок 3.4.7 Модель бази даних

Основні зв'язки між таблицями:

- users → messages: Один користувач може надсилати багато повідомлень (один до багатьох).
- users → logs: Один користувач може мати багато записів у логах (один до багатьох).
- settings: Налаштування є унікальними для кожної групи або каналу в Telegram.
- spam_rules: Зберігає правила, які використовуються для фільтрації спаму

Такий підхід до структури даних забезпечує зручне зберігання та доступ до необхідної інформації для ефективної роботи коммент-боту, дозволяючи легко масштабувати систему та адаптувати її до нових вимог.

4. РОЗРОБКА ТА РЕАЛІЗАЦІЯ КОММЕНТ-БОТУ

4.1 Деталі реалізації програмного забезпечення

Для розробки коммент-боту для автоматичного розпізнавання та фільтрації спаму в коментарях Telegram було обрано сучасні програмні засоби, що забезпечують ефективність, гнучкість і високу продуктивність системи. Основною мовою програмування був обраний Python завдяки його простоті, читабельності коду, широкому вибору бібліотек для різних задач і великій спільноті розробників, що сприяє швидкій підтримці та доступу до ресурсів для навчання і вирішення проблем.

Серед використовуваних бібліотек та фреймворків:

- Aiogram – асинхронний фреймворк для розробки Telegram-ботів, який забезпечує високу продуктивність та зручність у використанні. Aiogram підтримує асинхронне програмування, що дозволяє обробляти багато запитів одночасно, підвищуючи швидкість реагування бота.
- PyTorch – потужна бібліотека для машинного та глибокого навчання, що дозволяє легко створювати і налаштовувати моделі з підтримкою GPU для прискорення навчання. PyTorch забезпечує гнучкість у розробці моделей і зручність у налаштуванні нейронних мереж.
- Scikit-learn – популярна бібліотека для машинного навчання, що підтримує широкий вибір алгоритмів. Вона забезпечує легкість у використанні та високу продуктивність, що дозволяє швидко та ефективно тренувати моделі для розпізнавання спаму.
- NLTK (Natural Language Toolkit) – бібліотека для роботи з природною мовою, яка надає широкий набір інструментів для попередньої обробки тексту та аналізу повідомлень. Вона використовується для

токенізації, стемінгу, лематизації та інших завдань, що є важливими для якісного аналізу тексту.

- PostgreSQL – система керування базами даних з відкритим вихідним кодом, яка забезпечує надійність, стабільність, гнучкість та підтримку розширень для зберігання даних про повідомлення, налаштування бота та статистику. PostgreSQL використовується для збереження результатів обробки та забезпечення швидкого доступу до даних.

Для реалізації проекту також необхідні такі додаткові бібліотеки:

- csv – стандартна бібліотека Python для роботи з файлами у форматі CSV, що дозволяє легко зберігати та читати дані у табличному форматі. Використовується для обробки даних, збереження результатів та їх подальшого аналізу.
- pandas – потужна бібліотека для маніпуляції даними та аналізу. Вона забезпечує високу продуктивність та гнучкість у роботі з великими наборами даних, включаючи їх зчитування, обробку та аналіз.

За допомогою застосування цих бібліотек і фреймворків можна створити високопродуктивний та гнучкий коммент-бот для Telegram, що здатен ефективно розпізнавати та фільтрувати спам. Кожна з вибраних технологій забезпечує конкретні переваги.

Усі ці програмні засоби разом забезпечують високу точність розпізнавання спаму, ефективність обробки повідомлень і зручність у використанні для адміністраторів груп та каналів Telegram. Використання цих інструментів дозволяє створити надійну систему, яка відповідає сучасним вимогам до якості і продуктивності, а також може легко адаптуватися до нових викликів і типів спаму.

4.2 Особливості застосування (копії екрану діючого ПЗ)

4.2.1 Інтерфейс користувача

Інтерфейс користувача комент-боту для Telegram розроблений з урахуванням простоти та зручності взаємодії з програмою. Основні елементи інтерфейсу включають в себе наступні компоненти:

- Головне вікно програми

Це початкове вікно, в яке користувач попадає. Головне вікно надає доступ до чату з ботом, дозволяючи користувачам швидко орієнтуватися та виконувати необхідні дії.

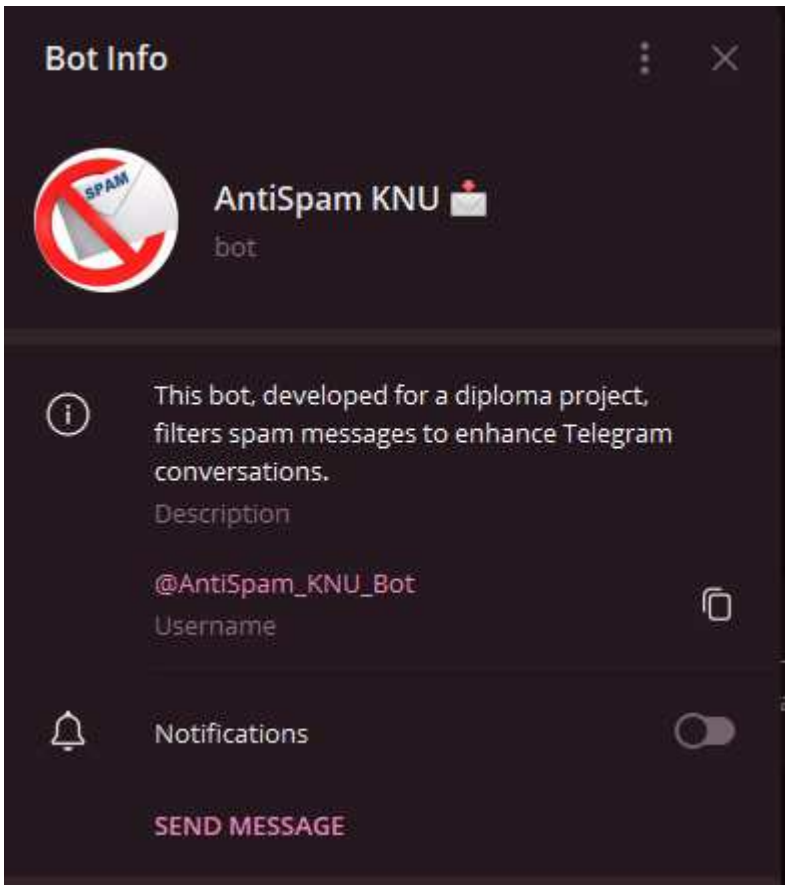


Рисунок 4.2.1 Головний екран боту

- Стартове привітання

Після додавання бота до групи або чату, користувачі отримують привітальне повідомлення (Рис. 4.2.2), яке включає інформацію про основні функції бота та інструкції щодо початку роботи. Це допомагає новим користувачам швидко зрозуміти, як користуватися ботом і які команди доступні.

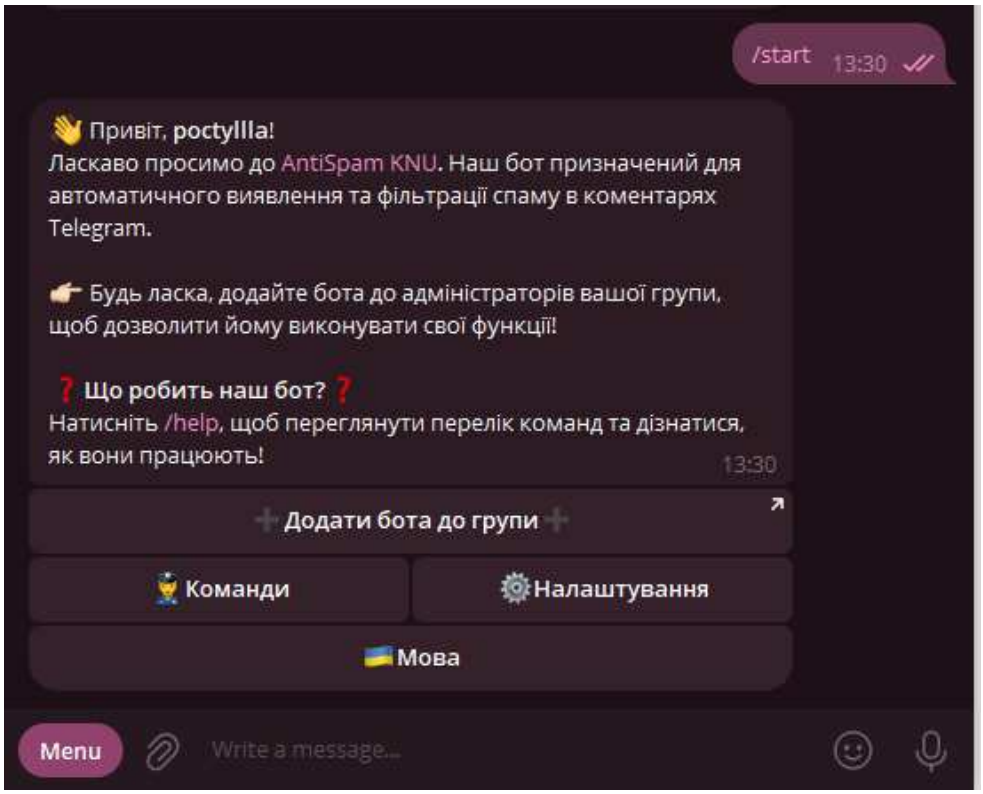


Рисунок 4.2.2 Стартове привітання

- Командний інтерфейс

Командний інтерфейс дозволяє користувачам взаємодіяти з ботом за допомогою текстових команд (Рис. 4.2.3). Це включає команди для налаштування фільтрації спаму, управління ключовими словами, отримання допомоги та інші команди для адміністративних задач.

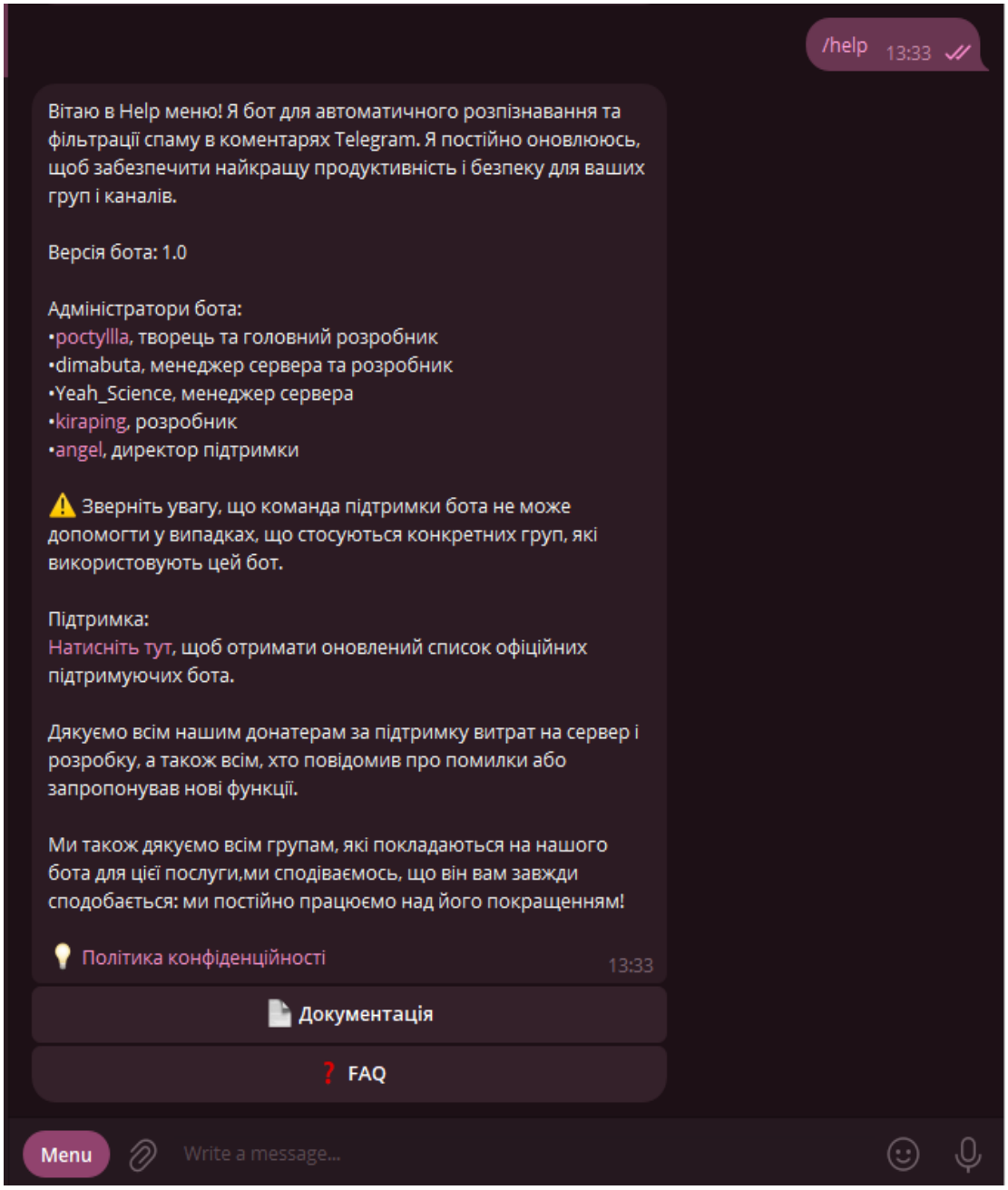


Рисунок 4.2.3 Вигляд команди help

Інтерфейс комент-бота побудований таким чином, щоб користувачі могли легко орієнтуватися у функціональності програми. Завдяки інтерактивним меню, чітким командним інструкціям та зрозумілому головному екрану, користувачі можуть ефективно налаштовувати бот відповідно до потреб своїх Telegram-груп. Логи та звіти додатково забезпечують прозорість роботи бота та дозволяють здійснювати моніторинг його ефективності.

4.2.2 Приклади роботи бота

Для демонстрації роботи комент-бота для автоматичного розпізнавання та фільтрації спаму в коментарях Telegram було створено кілька прикладів його функціональності.

Приклад 1: Запуск бота.

Щоб розпочати роботу з ботом, запустіть його та натисніть кнопку «Старт». Після потрапляння в стартове меню знайдіть і натисніть кнопку «Додати бота до групи». Додайте бота до вашої групи, призначте йому роль адміністратора та надайте необхідні права. Якщо ви создаєте бота на основі мого, то активуйте доступ до повідомлень через BotFather. Після виконання цих дій бот почне свою роботу.

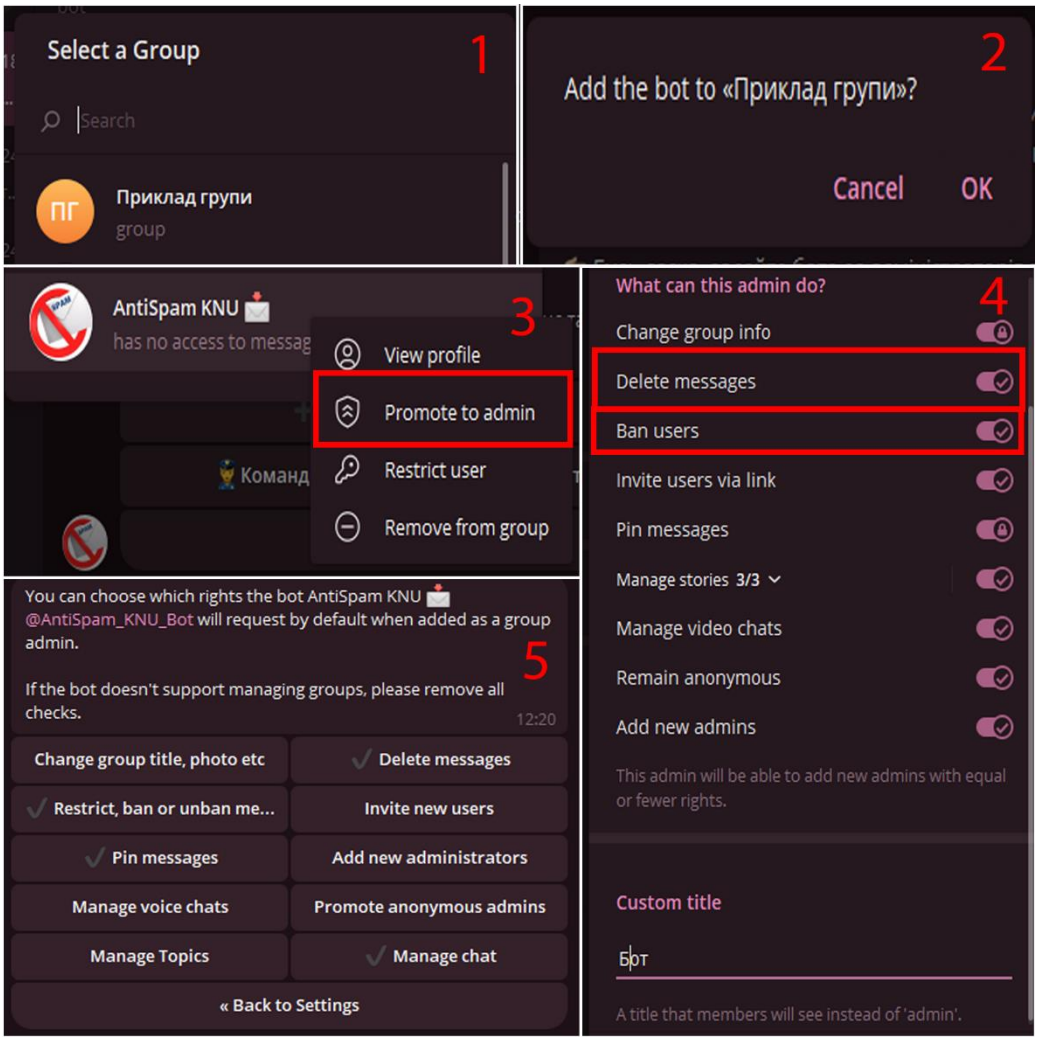


Рисунок 4.2.4 Запуск бота

Приклад 2: Логування роботи бота.

На скріншоті (Рис. 4.2.5) показано, як бот обробляє події, що надходять до нього. Кожна подія ідентифікується унікальним ідентифікатором і містить інформацію про тривалість обробки.

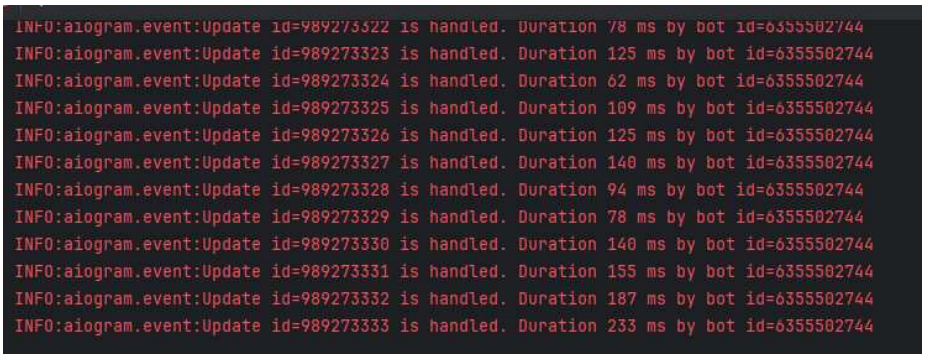


Рисунок 4.2.5 Логування роботи бота

Логи демонструють стабільну роботу бота з часом обробки, який коливається від 62 до 233 мілісекунд.

Приклад 3: Налаштування фільтрації спаму.

Скріншот (Рис. 4.2.6) демонструє використання команди /settings для налаштування фільтрації спаму. Користувач отримує повідомлення з інструкціями щодо можливих налаштувань:

- `/filter_level` - встановлення рівня фільтрації
- `/keywords` - керування ключовими словами для блокування.
- `/blacklist` - керування чорним списком користувачів.

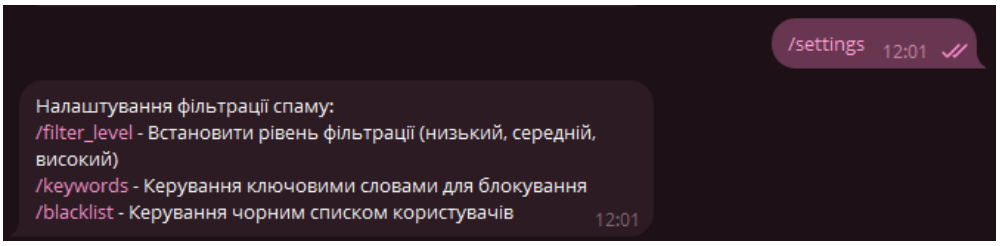


Рисунок 4.2.6 Налаштування фільтрації спаму

Таким чином, коммент-бот забезпечує ефективне та зручне управління фільтрацією спаму в Telegram-групах, що підтверджується як логами його роботи, так і можливостями налаштування, які він пропонує користувачам.

ВИСНОВОК

В процесі виконання дипломної роботи було досягнуто всіх поставлених цілей та виконано основні завдання, пов'язані з розробкою коммент-боту для автоматичного розпізнавання та фільтрації спаму в коментарях Telegram.

Розроблений коммент-бот успішно виконує функцію розпізнавання та фільтрації спамових повідомлень у коментарях Telegram. Бот в реальному часі аналізує всі отримані повідомлення, використовуючи модель машинного навчання, що дозволяє ефективно відрізняти спам від легітимних повідомлень. Всі поставлені завдання були виконані відповідно до вимог та стандартів. В результаті досягнуто високу точність виявлення спаму, що значно підвищує якість та безпеку комунікації в чатах Telegram.

Розробка коммент-боту базувалася на таких основних компонентах: бібліотека Aiogram для інтеграції з Telegram API; модель машинного навчання для розпізнавання спаму, навчена на великому датасеті з прикладами спамових та легітимних повідомлень; SQLite база даних для зберігання повідомлень та журналу дій бота; Python як основна мова програмування, що забезпечує високу продуктивність та зручність у розробці.

Було розроблено повнофункціональний коммент-бот для Telegram, який включає такі основні компоненти: система збору та обробки повідомлень; механізм аналізу та розпізнавання спаму на основі моделі машинного навчання; інтерфейс користувача для взаємодії з ботом та налаштування його параметрів; система логування та звітності для відстеження дій бота та ефективності його роботи; модулі для автоматичного видалення спамових повідомлень та сповіщення користувачів про їх видалення.

Розроблений коммент-бот дозволяє значно покращити якість комунікації в чатах Telegram за рахунок автоматичної фільтрації спаму. Основні можливості, які надає розробка, включають: зменшення кількості спамових повідомлень, що

підвищує комфорт користувачів та знижує навантаження на адміністраторів чатів; можливість налаштування порогів та правил фільтрації спаму відповідно до потреб конкретного чату; забезпечення високої точності виявлення спаму завдяки використанню сучасних методів машинного навчання; ведення журналу дій для аналізу та вдосконалення роботи бота.

Для подальшого розвитку та вдосконалення розробленого коммент-боту рекомендується:

1. Покращення моделі розпізнавання спаму: Постійно оновлювати та тренувати модель машинного навчання на нових даних, що дозволить підвищити точність розпізнавання спаму.

2. Розширення функціональності: Додати нові функції, такі як автоматичне попередження користувачів про порушення правил, можливість блокування користувачів, які систематично надсилають спам.

3. Інтеграція з іншими сервісами: Розглянути можливість інтеграції бота з іншими платформами та сервісами для покращення його ефективності та розширення сфери застосування.

4. Оптимізація продуктивності: Проводити регулярні тести та оптимізувати код бота для забезпечення стабільної роботи при великому навантаженні.

5. Зворотній зв’язок від користувачів: Збирати та аналізувати зворотній зв’язок від користувачів для виявлення можливих проблем та потреб, що дозволить удосконалювати бот відповідно до їх очікувань.

Таким чином, розроблений коммент-бот є важливим інструментом для підтримки чистоти комунікацій у чатах Telegram, а запропоновані рекомендації сприятимуть його подальшому розвитку та вдосконаленню.

Перелік посилань

1. PEP 8 – Style Guide for Python Code. [Електронний ресурс]. – Режим доступу: https://peps.python.org/pep-0008/

2. "Telegram Bot API". Telegram. [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/api

3. Aiogram Documentation. [Електронний ресурс]. – Режим доступу: https://docs.aiogram.dev/en/latest/

4. Scikit-learn: Machine Learning in Python. Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. [Електронний ресурс]. – Режим доступу: https://scikit-learn.org/stable/

5. SQLite Documentation. [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/docs.html

6. Нурминский Д. Чистый Python: лучший учебник / Д. Нурминский. – СПб.: Питер, 2020. – 320 с.

7. Bishop, C. M. Pattern Recognition and Machine Learning / C. M. Bishop. – New York: Springer, 2006. – 738 p.

8. Гудфеллоу И., Бенджио Й., Курвил А. Глубокое обучение / И. Гудфеллоу, Й. Бенджио, А. Курвил. – М.: Вильямс, 2018. – 732 с.

9. Kleppmann, M. Designing Data-Intensive Applications / M. Kleppmann. – O'Reilly Media, 2017. – 614 p.

10. Hulten, G. Building Intelligent Systems: A Guide to Machine Learning Engineering / G. Hulten. – New York: Springer, 2018. – 346 p.

11. Ng, A. Machine Learning Yearning / A. Ng. [Електронний ресурс]. – Режим доступу: https://www.mlyearning.org/

12. Sweigart, A. Automate the Boring Stuff with Python / A. Sweigart. – San Francisco: No Starch Press, 2015. – 504 p.

13. McKinney, W. Python for Data Analysis / W. McKinney. – Sebastopol: O'Reilly Media, 2017. – 544 p.

14. Burkov, A. The Hundred-Page Machine Learning Book / A. Burkov. [Електронний ресурс]. – Режим доступу: http://themlbook.com/

15. Sutton, R. S., Barto, A. G. Reinforcement Learning: An Introduction / R. S. Sutton, A. G. Barto. – Cambridge: MIT Press, 2018. – 552 p.

16. Вакуленко И. Введение в машинное обучение / И. Вакуленко. – М.: ДМК Пресс, 2019. – 400 с.

17. "Telegram FAQ". Telegram. [Електронний ресурс]. – Режим доступу: https://telegram.org/faq

18. Lutz, M. Python: основи та використання / M. Lutz. – М.: Вильямс, 2018. – 1600 с.

19. Chollet, F. Deep Learning with Python / F. Chollet. – Manning Publications, 2017. – 384 p.

20. Bird, S., Klein, E., Loper, E. Natural Language Processing with Python / S. Bird, E. Klein, E. Loper. – Sebastopol: O'Reilly Media, 2009. – 504 p.

Додаток А

Результати експериментів та тестування

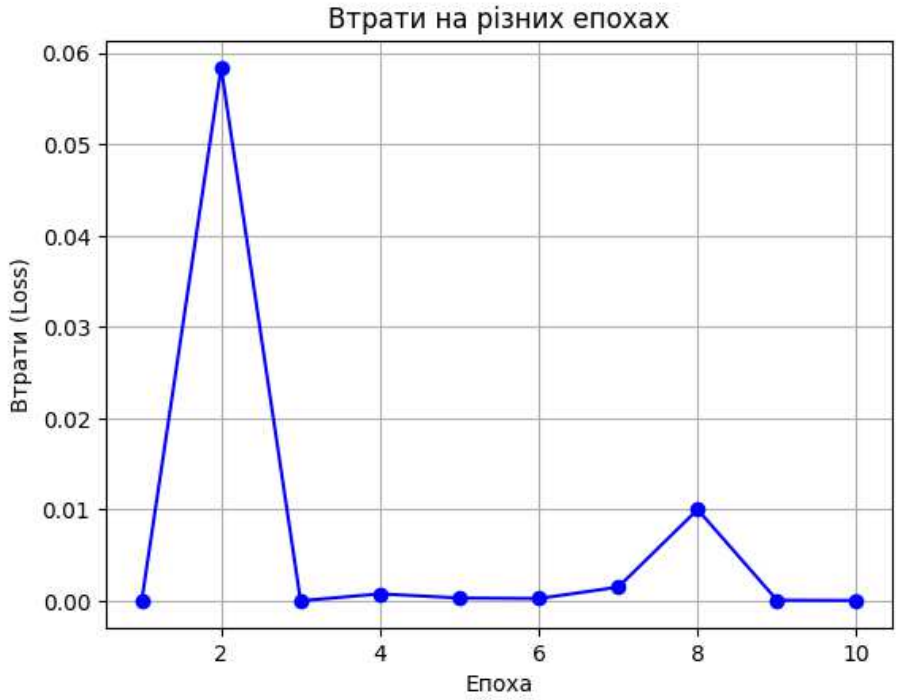


Рисунок А.1 Графік Втрат (Loss)

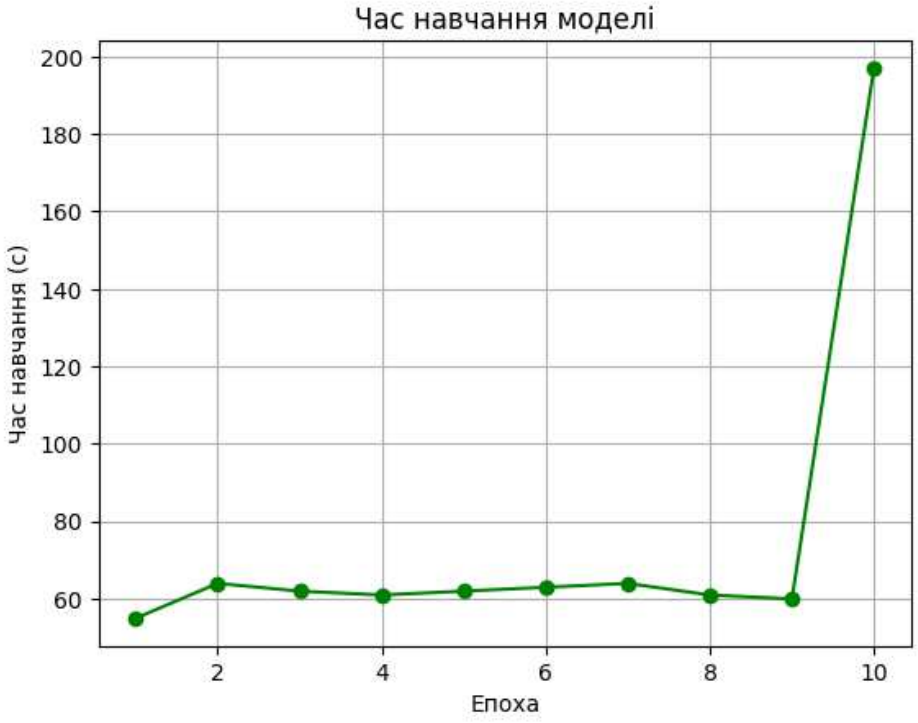


Рисунок А.2 Графік Часу Навчання

Рисунок А.3 Графік Коефіцієнта Впевненості (Confidence rate)

Окремо результати кожної моделі:

```
Using GPU
_____
INPUT_SIZE FOR spam_classifier_small: 8709
                                        Train Epoch: 99 [5440/5573 (97%)]       Loss: 0.000002
Training took approximately 55.52 seconds
```

Рисунок А.1 Результат тренування моделі spam_classifier_small

```
Using GPU
_____
INPUT_SIZE FOR spam_classifier_medium: 70237
                                        Train Epoch: 99 [5760/6047 (95%)]       Loss: 0.058406
Training took approximately 197.65 seconds
```

Рисунок А.2 Результат тренування моделі spam_classifier_medium

```
Using GPU
_____
INPUT_SIZE FOR spam_classifier_ukrainian: 15009
                                        Train Epoch: 99 [5440/5573 (97%)]       Loss: 0.000003
Training took approximately 63.61 seconds
```

Рисунок А.3 Результат тренування моделі spam_classifier_ukrainian

```
Using GPU
_____
INPUT_SIZE FOR spam_classifier_russian: 13341
                                        Train Epoch: 99 [5440/5573 (97%)]       Loss: 0.000761
Training took approximately 61.18 seconds
```

Рисунок А.4 Результат тренування моделі spam_classifier_russian

Таблиця А.1 Результати впевненості

Повідомлення	Передбачення Спам / Не спам	Впевненість (що це спам)	Модель
Congrats! 1 year special cinema pass for 2 is yours. call 09061209465 now!	Спам	97.09%	small
По дорозі додому тобі доведеться купити собі гамбургер за 1 доллар. Я навіть не можу поворухнутися. Біль мене вбиває	Не спам	1.90%	ukrainian
Вы ВЫИГРАЛИ гарантированную 1000 фунтов стерлингов наличными или приз в размере 5000 фунтов стерлингов!	Спам	99.99%	russian
Спасибі за підписку на Ringtone, з вашого номеру буде оплачуватися плата розміром 500 гривень на місяць.	Спам	99.99%	ukrainian
I HAVE A DATE ON SUNDAY WITH WILL!!	Не спам	0.0028%	small
BUY AN IPHONE NOW!!!	Спам	74.79%	medium
Прекрасно, якщо ти так відчуваєш. Ось так воно і повинно бути.	Не спам	2.37%	ukrainian
I'm sorry I missed your call earlier. Can we talk later? If you're comfortable, of course. Best Regards, Jared.	Не спам	27.84%	medium

Додаток Б

Деталі реалізації

bot.py

```
import asyncio, logging

from aiogram import Bot, Dispatcher
from aiogram.enums import ParseMode
from aiogram.client.default import DefaultBotProperties
from config import settings

from routers import router as main_router


async def main ():
    dp = Dispatcher()
    dp.include_router(main_router)
    logging.basicConfig(level=logging.INFO)
    bot = Bot(
        token=settings.bot_token,
        default=DefaultBotProperties(
            parse_mode=ParseMode.HTML)
    )
    await dp.start_polling(bot)

if __name__ == "__main__":
    asyncio.run(main())
```

config.py

```
from pydantic_settings import BaseSettings, SettingsConfigDict

class Settings(BaseSettings):
    model_config = SettingsConfigDict(
        case_sensitive=False,
    )

    bot_token:str

settings = Settings()
```

start_kb.py

```
from enum import IntEnum, auto
from aiogram.filters.callback_data import CallbackData
from aiogram.types import InlineKeyboardMarkup
from aiogram.utils.keyboard import InlineKeyboardBuilder

class StartActions(IntEnum):
    start = auto()
    commands = auto()
    settings = auto()
    language = auto()
    back_to_start = auto()


class StartCbData(CallbackData, prefix="start"):
    action: StartActions


invite_url = "https://t.me/AntiSpam_KNU_Bot?startgroup=true"

def build_start_kb() -> InlineKeyboardMarkup:
    builder = InlineKeyboardBuilder()
    builder.button(text="➕Додати бота до групи➕",
                   url=invite_url,)
    builder.button(text="👮Команди",

callback_data=StartCbData(action=StartActions.commands).pack(),)
    builder.button(text="⚙️Налаштування",

callback_data=StartCbData(action=StartActions.settings).pack(),)
    builder.button(text="UAМова",

callback_data=StartCbData(action=StartActions.language).pack(),)
    return builder.adjust(1, 2, 1).as_markup()

def build_commands_kb() -> InlineKeyboardMarkup:
    builder = InlineKeyboardBuilder()
    builder.button(text="🔙Повернутись",

callback_data=StartCbData(action=StartActions.back_to_start).pack()
,)
    return builder.as_markup()


def build_settings_kb() -> InlineKeyboardMarkup:
    builder = InlineKeyboardBuilder()
    builder.button(text="🔙Повернутись",
```

```
callback_data=StartCbData(action=StartActions.back_to_start).pack()
)
    return builder.as_markup()


def build_language_kb() -> InlineKeyboardMarkup:
    builder = InlineKeyboardBuilder()
    builder.button(text="GBEnglish",

callback_data=StartCbData(action=StartActions.language).pack(),)
    builder.button(text="RUРусский",

callback_data=StartCbData(action=StartActions.language).pack(),)
    builder.button(text="🔙Повернутись",

callback_data=StartCbData(action=StartActions.back_to_start).pack()
,)
    return builder.adjust(2,1).as_markup()
```

start_kb_callback_handlers.py

```
from aiogram import F, Router
from aiogram.types import CallbackQuery
from aiogram.utils import markdown
from keyboards.inline_keyboards.start_kb import (
    StartCbData,
    StartActions,
    build_start_kb,
    build_commands_kb,
    build_language_kb,
    build_settings_kb,
)

router = Router()

async def start(call: CallbackQuery):
    await call.message.edit_text(
        text=f'👋 Привіт,
{markdown.hbold(call.from_user.first_name)}!\n'
             f'Ласкаво просимо до {markdown.hlink("AntiSpam KNU",
"https://t.me/AntiSpam_KNU_Bot")}. '
             'Наш бот призначений для автоматичного виявлення та
фільтрації спаму в коментарях Telegram.\n\n'
             '👉 Будь ласка, додайте бота до адміністраторів вашої
групи, щоб дозволити йому виконувати свої функції!\n\n'
             f'❓{markdown.hbold("Що робить наш бот?")}❓\n'
```

```
            'Натисніть /help, щоб переглянути перелік команд та
дізнатися, як вони працюють!',
        disable_web_page_preview=True,
        reply_markup=build_start_kb(),
    )

@router.callback_query(StartCbData.filter(F.action ==
StartActions.start))
async def start_handler(call: CallbackQuery):
    await start(call)

@router.callback_query(StartCbData.filter(F.action ==
StartActions.commands))
async def commands(call: CallbackQuery):
    await call.answer()
    await call.message.edit_text(
        text='Ось доступні команди:',
        reply_markup=build_commands_kb(),
    )

@router.callback_query(StartCbData.filter(F.action ==
StartActions.settings))
async def settings(call: CallbackQuery):
    await call.answer()
    await call.message.edit_text(
        text='Налаштування фільтрації спаму:\n'
             '/filter_level - Встановити рівень фільтрації
(низький, середній, високий)\n'
             '/keywords - Керування ключовими словами для
блокування\n'
             '/blacklist - Керування чорним списком користувачів',
        reply_markup=build_settings_kb(),
    )

@router.callback_query(StartCbData.filter(F.action ==
StartActions.language))
async def language(call: CallbackQuery):
    await call.answer()
    await call.message.edit_text(
        text='Оберіть мову:',
        reply_markup=build_language_kb(),
    )

@router.callback_query(StartCbData.filter(F.action ==
StartActions.back_to_start))
async def back_to_start(call: CallbackQuery):
    await start(call)
```

base_commands.py

```
from aiogram import Router, types
from aiogram.filters import CommandStart, Command
from aiogram.utils import markdown

from keyboards.inline_keyboards.start_kb import (
    build_start_kb,
)
from keyboards.inline_keyboards.help_kb import (
    build_help_kb,
)
router = Router()


@router.message(CommandStart())
async def handle_start(message: types.Message):
    markup = build_start_kb()
    await message.answer(text=
                                f'👋 Привіт,
{markdown.hbold(message.from_user.first_name)}!\n'
                                f'Ласкаво просимо до
{markdown.hlink("AntiSpam KNU", "https://t.me/AntiSpam_KNU_Bot")}.
'
                                'Наш бот призначений для
автоматичного виявлення та фільтрації спаму в коментарях
Telegram.\n\n'
                                '👉 Будь ласка, додайте бота до
адміністраторів вашої групи, щоб дозволити йому виконувати свої
функції!\n\n'
                                f'❓{markdown.hbold("Що робить наш
бот?")}❓\n'
                                'Натисніть /help, щоб переглянути
перелік команд та дізнатися, як вони працюють!',
                           disable_web_page_preview=True,
                           reply_markup=markup,
                           )


@router.message(Command("help"))
async def handle_help(message: types.Message):
    markup = build_help_kb()
    await message.answer(
        text='Вітаю в Help меню! Я бот для автоматичного
розпізнавання та фільтрації спаму в коментарях Telegram. '
             'Я постійно оновлююсь, щоб забезпечити найкращу
продуктивність і безпеку для ваших груп і каналів.\n\n'
             'Версія бота: 1.0\n\n'
             'Адміністратори бота:\n'
```

```
            f'•{markdown.hlink("poctyllla","t.me/poctyllla")},
творець та головний розробник\n'
            '•dimabuta, менеджер сервера та розробник\n'
            '•Yeah_Science, менеджер сервера\n'

f'•{markdown.hlink("kiraping","https://t.me/darknessio_pasione")},
розробник\n'
            f'•{markdown.hlink("angel",
"https://t.me/aryan_angel")}, директор підтримки\n\n'
            '⚠️ Зверніть увагу, що команда підтримки бота не може
допомогти у випадках, що стосуються конкретних груп, '
            'які використовують цей бот.\n\n'
            'Підтримка:\n'
            f'{markdown.hlink("Натисніть
тут","https://t.me/+p_tkvmduMtZiMTky")}, щоб отримати оновлений '
            f'список офіційних підтримуючих бота.\n\n'
            'Дякуємо всім нашим донатерам за підтримку витрат на
сервер і розробку, а також всім, хто повідомив '
            'про помилки або запропонував нові функції.\n\n'
            'Ми також дякуємо всім групам, які покладаються на
нашого бота для цієї послуги,ми сподіваємось, що він '
            'вам завжди сподобається: ми постійно працюємо над
його покращенням!\n\n'
            f'{markdown.hlink("💡 Політика
конфіденційності","https://www.yourbotprivacy.link")}',
        disable_web_page_preview=True,
        reply_markup=build_help_kb()
    )



@router.message(Command("settings"))
async def handle_settings(message: types.Message):
    await message.answer(text='Налаштування фільтрації спаму:\n'
                              '/filter_level - Встановити рівень
фільтрації (низький, середній, високий)\n'
                              '/keywords - Керування ключовими
словами для блокування\n'
                              '/blacklist - Керування чорним
списком користувачів'
                         )
```

routers/commands/init.py

```
__all__ = ("router", )

from aiogram import Router
from .base_commands import router as base_commands_router
from .user_commands import router as user_commands_router
from .spam_commands import router as spam_commands_router

router = Router()

router.include_routers(
    base_commands_router,
    user_commands_router,
    spam_commands_router,
    )
```

routers/init.py

```
__all__ = ("router", )

from aiogram import Router

from .admin_handlers import router as admin_router
from .callback_handlers import router as callback_router
from .commands import router as commands_router

router = Router()

router.include_routers(
    callback_router,
    commands_router,
    admin_router,
)
```

routers/commands/spam_commands.py

```
from aiogram import Router, types
from spam_handling.spam_classifier import load_model_and_vectorizer
import torch
from langdetect import detect

router = Router()

def detect_language(text):
    try:
        return detect(text)
    except:
        return "unknown"

@router.message()
async def handle_spam(message: types.Message):
    try:
        language = detect_language(message.text)
        if language == "ru":
            model_type = "russian"
        elif language == "uk":
            model_type = "ukrainian"
        else:
            model_type = "medium"

        model, vectorizer = load_model_and_vectorizer(model_type)

        message_vector =
vectorizer.transform([message.text]).toarray()
        message_tensor = torch.tensor(message_vector,
dtype=torch.float32)

        with torch.no_grad():
            output = model(message_tensor)
        prediction = output.item()

        if prediction > 0.9:
            await message.delete()
    except Exception as e:
        print(f"Error processing message: {str(e)}")
```