НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Автоматизованих систем обробки інформації і управління
(повна назва кафедри)

«На правах рукопису»
УДК 004.91

До захисту допущено:
В.о. завідувача кафедри
_____ Олександр ПАВЛОВ
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютеризованих систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Інтелектуальна система дослідження та аналізу текстів на предмет наявності спойлерів у оглядах медіаконтенту»

Виконав:
студент VI курсу, групи ІП-92мп
Кущ Антон Віталійович _____

Науковий керівник:
старший викладач
Головченко Максим Миколайович _____

Рецензент:
доцент кафедри ОТ, к.т.н.
Болдак Андрій Олександрович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Автоматизованих систем обробки інформації і управління

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма - «Інженерія програмного забезпечення комп'ютеризованих систем»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Кущу Антону Віталійовичу

1. Тема дисертації «Інтелектуальна система дослідження та аналізу текстів на предмет наявності спойлерів у оглядах медіаконтенту», науковий керівник дисертації старший викладач Головченко Максим Миколайович, затверджені наказом по університету від «26» жовтня 2020 р. № 3132-с

2. Термін подання студентом дисертації _____

3. Об'єкт дослідження аналіз текстових даних на предмет наявності спойлерів

4. Вхідні дані

5. Перелік завдань, які потрібно розробити проаналізувати існуючі методи для визначення спойлерів у тексті; реалізувати три методи для визначення спойлерів у текст; порівняти результати ефективності реалізованих методів та обрати найефективніший; розробити програмний додаток, який буде використовувати обраний метод; розробити стартап-проект для створеної системи

6. Орієнтовний перелік графічного (ілюстративного) матеріалу _____

7. Орієнтовний перелік публікацій V Всеукраїнська науково-практична конференція молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2020) (Київ, 2020 р.)

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Графічний	доц. Ліщук К.І.		

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Дослідження літератури та документації	25.05.2020	
2	Аналіз існуючих рішень	05.07.2020	
3	Проектування системи	08.08.2020	
4	Розробка методів та алгоритмів	28.08.2020	
5	Розробка серверного та клієнтського програмного забезпечення	25.09.2020	
6	Тестування програмного забезпечення	30.10.2020	
7	Оформлення документації	21.11.2020	
8	Подання роботи на попередній захист	26.11.2020	
9	Подання роботи на основний захист		

Студент Кущ А.В.

Науковий керівник Головченко М.М.

РЕФЕРАТ

Актуальність теми:

– застосування методів семантичного аналізу даних для визначення спойлерів у оглядах медіаконтенту;

– відсутність автоматизованої системи для визначення спойлерів у оглядах медіаконтенту платформи Telegram.

Мета дослідження: створення інтелектуальної системи дослідження та аналізу текстів на предмет наявності спойлерів у оглядах медіаконтенту.

Для реалізації поставленої мети були сформульовані наступні **завдання:**

– проаналізувати існуючі методи для визначення спойлерів у тексті;

– реалізувати три методи для визначення спойлерів у тексті;

– порівняти результати ефективності реалізованих методів та обрати найефективніший;

– розробити програмний додаток, який буде використовувати обраний метод;

– розробити стартап-проект для створеної системи.

Об'єкт дослідження: аналіз текстових даних на предмет наявності спойлерів.

Предмет дослідження: алгоритми та методи для автоматизованого визначення спойлерів у оглядах медіаконтенту.

Методи дослідження: методи обробки текстових даних, засновані на семантичному аналізі з використанням глибинного навчання.

Наукова новизна: науковими результатами магістерської дисертації є:

– застосування методів семантичного аналізу текстових даних для визначення спойлерів у оглядах медіаконтенту;

– розробка комплексного рішення автоматизованої системи для визначення спойлерів у оглядах медіаконтенту платформи Telegram.

Практичне значення отриманих результатів визначається тим, що реалізована система може бути використана для автоматизованого визначення спойлерів у оглядах медіаконтенту у групах платформи Telegram, обговореннях тощо.

Зв'язок роботи з науковими програмами, планами, темами:робота виконувалась на кафедріавтоматизованих систем обробкиінформації та управлінняНаціональноготехнічногоуніверситетуУкраїни «Київськийполітехнічнийінститутім. ІгоряСікорського» в рамках теми «Інтелектуальна система дослідження та аналізутекстів на предмет наявності спойлерів у оглядах медіаконтенту».

Публікації: наукові положення дисертації опубліковані в тезах V Всеукраїнської науково-практичної конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2020) (Київ, 2020 р.).

Апробація: основні положення роботи доповідались і обговорювались на V Всеукраїнській науково-практичній конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2020) (Київ, 2020 р.).

Ключові слова: СЕМАНТИЧНИЙ АНАЛІЗ ТЕКСТОВИХ ДАНИХ, ОГЛЯДИ МЕДІАКОНТЕНТУ, СПОЙЛЕР, РЕКУРЕНТНІ НЕЙРОННІ МЕРЕЖІ

ABSTRACT

The relevance of the topic:

– application of semantic data analysis methods to identify spoilers in media content reviews;

– lack of anautomated system for determining spoilers in reviews of media content of the Telegram platform.

The purpose of the study: creation of an intelligent system of research and analysis of texts for the presence of spoilers in media content reviews.

To achieve this goal, the following **tasks** were set:

– analyze existing methods for determining spoilers in the text;

– implement three methods for determining spoilers in the text;

– compare the results of the effectiveness of the implemented methods and choose the most effective;

– develop a software application that will use the selected method;

– develop a startup project for the created system.

The object of research: analysis of text data for the presence of spoilers.

The subject of research: algorithms and methods for automated detection of spoilers in media content reviews.

Methods of research: methods of text data processing based on semantic analysis using deep learning concepts.

Scientific novelty: the scientific results of the research are:

– application of methods of semantic analysis of text data to determine spoilers in reviews of media content;

– development of a comprehensive solution for anautomated system for determining spoilers in reviews of media content of the Telegram platform.

The practical significance of the obtained results is determined by the fact that the implemented system can be used for automated detection of spoilers in media content reviews in groups of the Telegram platform, discussions, etc.

Relationship with working with scientific programs, plans, topics: the work was performed at the Department of Automated Information Processingand Management Systems of the National Technical University of Ukraine “Kyiv Polytechnic Institute Igor Sikorsky” within the topic “Intelligent system of research and analysis of texts for the presence of spoilers in media content reviews”.

Publications: scientific provisions of the research were published in the abstracts of the V All-Ukrainian scientific-practical conference of young scientists and students "Information systems and management technologies" (ISTU-2020) (Kyiv, 2020).

Testing: the main provisions of the work were reported and discussed at the V All-Ukrainian scientific-practical conference of young scientists and students "Information systems and management technologies" (ISTU-2020) (Kyiv, 2020).

Keywords: SEMANTIC ANALYSIS OF TEXT DATA, MEDIA CONTENT REVIEWS, SPOILER, RECURRENT NEURAL NETWORKS.

ЗМІСТ

ВСТУП

Перед початком перегляду медіаконтенту, у людей існує тенденція – коротко переглянути зміст кінофільму або почитати коментарі, які дали інші користувачі у каналах для обговорення месенджеру Telegram. Серед цих коментарів не рідко з’являються повідомлення, які повністю або частково розповідають про перебіг подій у фільмі, деякі ключові повороти – так звані спойлери. Дані коментарі часто відбирають бажання дивитися кінострічку через ненароком набуті знання про ключові повороти або кінцівку фільму. Одним із шляхів вирішення проблеми є наявність окремої людини (адміністратора), який буде відповідати за те, щоб фільтрувати повідомлення, які з’являються у обговоренні, на предмет спойлерів.Наявність інтелектуальної системи змогла б автоматизувати даний процес та зменшити вплив людської похибки.

Огляд літератури попередників показав, що тема визначення спойлерів є актуальною та розглядається до сьогоднішніх днів. Проте не існує комплексної архітектури для вирішення даної проблеми у рамках месенджеру Telegram. Також попередники в основному використовують методи, які виконують лінгвістичний аналіз текстів, що показує меншу точність у визначенні, якщо порівнювати з методами семантичного аналізу. Отже, існує потреба у реалізації інтелектуальної системи, яка б змогла автоматизувати процес визначення спойлерів у тексті, зменшити людський фактор.

Дана дисертаційна робота бере за мету розробку інтелектуальної системи, яка автоматизує процес визначення спойлерів у оглядах медіаконтенту, яка унеможливить виникнення вищесказаних проблем у користувачів та зможе відрізнити спойлери серед коментарів до медіаконтенту.При цьому необхідно реалізувати емпіричні методи для виділення найефективнішої моделі для визначення спойлерів у тексті, методи системного аналізу для вибору структури

інтелектуальної системи, методи об’єктно-орієнтованого програмування для розробки системи для автоматизації визначення спойлерів.

Для реалізації поставленої мети були сформульовані наступні завдання:

– проаналізувати існуючі методи для визначення спойлерів у тексті;

– реалізувати три методи для визначення спойлерів у тексті;

– порівняти результати ефективності реалізованих методів та обрати найефективніший;

– розробити програмний додаток, який буде використовувати обраний метод;

– розробити стартап-проект для створеної системи.

Об’єктом дослідження є аналіз текстових даних на предмет наявності спойлерів.

Предметом дослідження є алгоритми та методи для автоматизованого визначення спойлерів у оглядах медіаконтенту.

Науковою новизною даної дисертаційної роботи є використання методів семантичного аналізу текстів для вирішення задачі визначення спойлерів та розроблене комплексне рішення інтелектуальної системи, яка автоматизує процес визначення спойлерів у оглядах медіаконтенту у платформі Telegram.

Практичне значення одержаних результатів полягає у готовності до застосування даної інтелектуальної системи у різного роду проектах, які містять медіаконтенту та відгуки на них.

Усі результати, наведені у дисертації, отримані самостійно. У статтях та роботах здобувачеві належить ідея створення автоматизованої системи для визначення спойлерів у оглядах медіаконтенту та використання методів семантичного аналізу текстів.

Результати досліджень викладені у дисертації були оприлюдненіу V Всеукраїнській науково-практичній конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2020) (Київ, 2020 р.).

За результатами дисертації опубліковано статтю з темою «Інтелектуальна система дослідження та аналізу текстів на предмет наявності спойлерів у оглядах медіаконтенту» у V Всеукраїнській науково-практичній конференції молодих вчених та студентів «Інформаційні системи та технології управління».

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметного середовища

Кіноіндустрія є однією із найбільших індустрій у сфері розважання у світі. Станом на початок 2020 року світова кіноіндустрія демонструвала неймовірні показники касових надходжень – понад 42 млрд доларів. Даний показник впевнено зростав до 2020 року. Незважаючи на пандемію коронавірусу та спадок прибутку підприємств, дана індустрія залишається на лідируючих позиціях та приковує увагу все більшої кількості людей.

До предметного середовища даної дисертаційної роботи входить опис кінострічок разом із коментарями користувачів, які зацікавлені розділяти свої думки з приводу кінострічки з іншими людьми.Серед людей існує тенденція перед переглядом кінострічки подивитися короткий опис фільму та коментарі, які залишили інші користувачі щодо їх відношення після перегляду фільму. Серед даних коментарів можуть бути повідомлення, які містять так звані спойлери – описані частини кінострічки, які можуть видавати ключові події або повороти сюжету. Огляд літератури показав, що спойлери в основному не мають позитивного впливу на людей перед переглядом кінострічки. Саме тому існує потреба у фільтруванні повідомлень користувачів у обговореннях. Зазвичай користувачі залишають свої коментарі щодо фільму на сайтах, форумах та, у тому числі, у каналах месенджеру Telegram.

Telegram – це багатоплатформний месенджер, який створений з метою надати користувачам можливість обмінюватися повідомленнями та медіафайлами багатьох форматів.У 2013 році було випущено перший клієнт Telegram. З тих пір з’явилися додатки Telegram для таких операційних систем, як Android, iOS, Windows Phone, Windows, macOS та GNU/Linux.

До недавнього часу користувачі Telegram могли обмінюватися думками щодо переглянутої кінострічки у приватних повідомленнях або приватних групових чатах. Проте 30 вересня 2020 року у черговому оновленні платформи, розробники повідомили про реліз нового функціоналу для чатів Telegram каналів – коментарі. Таким чином, кожен, хто є підписником каналу, може відправити свій коментар під публікацією.Цей доданий функціонал дозволив користувачам Telegram обмінюватися думками у публічних каналах. Слід сказати, що коментарі усередині каналів користуються неабиякою популярністю та чи не кожен адміністратор публічного каналу додав функціональність коментарів до свого каналу.

Зважаючи на вищеописане оновлення Telegram, любителі кінострічок тепер можуть обмінюватися думками щодо переглянутого медіаконтенту у коментарях каналів.

1.2 Аналіз текстових даних

Для досягнення мети даної роботи необхідно виконати аналіз коментарів користувачів, зважаючи на кінострічку до якої вони пишуть цей коментар.Отже, потрібно проаналізувати сюжет кінострічки та з’ясувати, чи наявні спойлери у коментарі, що відправив користувач.Сюжет медіаконтенту зазвичай описується у описі до фільму. При чому підприємства, які зацікавлені у тому, щоб люди переглядали їх контент, зацікавлені у тому, щоб даний опис не мав жодних спойлерів та моментів, які можуть видати ключові події у сюжеті. Можна вважати, що опис кінострічки підходить для даної задачі. Отже, при аналізі спойлерів потрібно взяти до уваги опис медіаконтенту та коментар користувача – задача зводиться до аналізу текстових даних.

Аналіз текстових даних – це галузь інформатики та машинного навчання, що намагається вирішити завдання подання людської мови у комп’ютері.

Дані поділяються на структуровані та неструктуровані. До структурованих даних належать статистики, бази даних, електронні таблиці. Для таких даних типовими задачами є побудова графіків залежностей, діаграм, виділення аномалій. Прикладом може послужити побудова графіку кількості продажів продуктової компанії протягом місяця роботи.

Більшість текстових даних належать до неструктурованих даних. Для виконання більш розвинутого аналізу з даних соціальних мереж, опитувань, звітів новин, оглядів тощо, необхідно спільно використовувати ряд методів аналізу тексту машинного навчання. Ручне впорядковування та читання великих обсягів текстових даних у наш час просто не має сенсу – це дорого, трудомістко і набагато менш точно, ніж правильно навчені моделі машинного навчання. Дані методи дозволяють класифікувати дані. Прикладом використання таких моделей може послужити визначення недостовірних новин у додатку Twitter.

Одним із типів аналізу тексту є NaturalLanguageProcessing.

NaturalLanguageProcessing (NLP) – галузь, яка вивчає синтез природної мови, або іншимисловами – розуміння мови людини. Для вирішення цих задач зазвичай використовуються штучний інтелект.

Результатами застосування NLP, наприклад, є машинний переклад, електронні словники, розпізнавання мови у різних форматах тощо. На даний час, NLP вирішує багато задач, які стосуються аналізу текстових даних – виділення багато корисних фактів з тексту. Наприклад, методи NLP зможуть успішно проаналізувати текст та виділити персонажів розповіді, які є жіночої статі, зважаючи на синоніми, текстові обороти, визначення іменованої сутності тощо.

Отже методи NLP аналізують текст на лінгвістичному рівні, що є достатнім у більшості задач. Проте для визначення спойлерів необхідно зважати на семантичну подібність. Для вирішення даної задачі існує NaturalLanguageInference (NLI).

ЗадачоюNaturalLanguageInferenceє визначення чи є гіпотеза правдивою, нейтральною або хибною при даній передумові. Основною концепцією NLI є силогістичне сполучення – вимагає розуміння тексту на семантичному, а не на лінгвістичному рівні. NLI отримує нові знання, беручи явне твердження, зроблене у вихідному тексті, та модифікуючи його, щоб сформувати нове твердження, яке, можливо, взагалі не з'являлося в тексті.

NLI засновується на двох концепціях:

- гіперо-гіпонімія;
- монотонність універсальної кількісної оцінки.

Гіперо-гіпонімія показує родо-видові відношення в лексико-семантичній системі. Гіперонім – це слово ширшого значення, яке означає загальне, родове поняття, наприклад, тварина. Гіпонім – це слово вужчого значення, яке означає об'єкт, як елемент класу, наприклад, вовк або кіт.

Універсальна кількісна оцінка у NLI позначає, що для кожного об'єкту множини виконується та чи інша умова. Наприклад, всі коти – тварини.

Монотонність універсальної кількісної оцінки у NLI означає, що при заміні гіпероніму на слово більш ширшого значення, або гіпоніму на слово більш вужчого значення, наше твердження залишається вірним. Термін монотонність використовується через природу гіперонімів та гіпонімів, вони монотонно змінюються на більш ширші або більш вужчі поняття відповідно.

Отже, зважаючи на все вищесказане можна зробити логічний ланцюжок: кіт є твариною, кіт Том є твариною. Таким чином, із двох окремих тверджень у двох не пов'язаних між собою частинах тексту отримано абсолютно нові знання. Отже, система, що базується на NLI, може витягти ці конкретні знання там, де лексичний підхід до NLP не зможе.

Вищеописані поняття можуть змінюватись у різних реалізаціях NLI, але вони є базовими у NLI.

1.3 Огляд літератури

Спойлери є невід'ємною частиною перегляду медіаконтенту через розвиток інформаційних технологій та соціальних мереж. Існують наукові роботи, які намагаються визначити той ефект, який вони приносять людям – користь чи шкоду. Згідно с науковою роботою [1], вплив спойлерів збільшує те, що називається «reactance» - негативні почуття, які люди відчувають, коли стикаються зі втратою свободи, наприклад, коли щось або хтось забирає вашу здатність приймати рішення. Також вони вважають, що людей навчили вірити, що спойлери змусять нас насолоджуватися чимось менше і що вони забирають у нас свободу насолоджуватися історією без сторонніх впливів. Навіть саме слово «спойлер» має негативний підтекст, що зумовлює виникнення негативних емоцій у людей. Тому з точки зору психології, дана задача є актуальною та важливою.

При пошуку комплексного програмного додатку для платформи Telegram, аналогів знайдено не було, але для повноти аналізу необхідно зазначити дані додатки, які є потенційними конкурентами.

Серед потенційних конкурентів можна виділити SpoilerProtection 2.0 та SpoilerShield. Вони мають дуже схожий функціонал – визначають спойлери у даних ресурсах:

- Facebook (дошка оголошень, чат тощо);
- Twitter;
- Youtube;
- Google;
- сайти новин.

Недоліком даних програмних продуктів є те, що для визначення спойлерів потрібно заздалегідь додавати назви стрічок, серіалів у додаток для того, щоб була коректно виконана фільтрація. При перегляді оглядів медіаконтенту даний крок є

не потрібним. Крім того, через великий набір контенту з-поміж якого виконується пошук спойлерів, алгоритми втрачають свою ефективність: про це свідчать негативні відгуки користувачів.

Іншим потенційним конкурентом можна виділити програмний додаток HideItBot. Даний продукт реалізований у вигляді Telegram боту з яким користувач може комунікувати за допомогою введення команди у чат – @hideItBot. Задачою даного програмного продукту є приховування спойлерів, які містяться у повідомленнях користувачів Telegram. Можливі варіанти використання даного продукту:

– надсилання спойлера книги/серії/фільму в груповому чаті, щоб його потенційно побачили не всі користувачі, а лише ті які натиснули на кнопку для отримання спойлеру;

– надсилання повідомлення іншому користувачу у приватному чаті, у якому текст у сповіщенні буде прихований.

Недоліком даного альтернативного рішення є те, що інфраструктура додатку не містить серверу для автоматизованого аналізу текстових даних, а тому користувачі мають самостійно вирішувати, коли приховувати повідомлення, а коли відправляти без змін.

З точки зору методів, якими вирішується проблема визначення спойлерів у тексті існують наукові роботи, які випускаються і до сьогоднішнього дня.

Найбільш давні наукові роботи в своїй більшості засновуються на простому тематичному моделюванню [2],застосуванні частих дієслів та іменованих сутностей [3], застосуванні зовнішніх метаданих суб’єктів аналізу (наприклад, жанрів, видів) [4].

Очевидно, що методи відповідності ключових слів вимагають присутності людського фактору і, тому, не використовуються широко в різних сценаріях

застосувань. Крім того, дані методи зазвичай мають низьку точність, оскільки вони сприймають багато позитивних коментарів як спойлери.

Наступні наукові роботи успішно використовували підходи нейронних мереж для задачі класифікації речень та документів.Зокрема, підхід на основі згорткової нейронної мережіпоказав покращення у 4-х серед 7-ми задач сентимент аналізу речень та класифікації питань[5]. Також окремого вказання вартує наукова робота [6]. Вона використовує підхід, який базується на hierarchicalattentionnetworks для класифікації документів. Експериментальні результати показали, що дана модель краще, аніж моделі попередників та є ефективною у визначенні важливих слів та речень.

З деяких причин методи глибокого навчання у визначенні спойлерів були застосовані лише починаючи з 2018 року, зокрема у роботі [7].У даній роботі використовуються два кодери для навчання нейронної мережі. У якості набору властивостей застосовується властивості жанру, до яких належить кінострічка, та властивості речення. Також додатково застосовується реалізація алгоритму, який враховує жанр до якого належить кінострічка для визначення спойлерів.

Ще однією цікавою науковою роботою є робота [8]. Задачею даної роботи є визначення спойлерів у так званих time-synccomments – коментарях, які з’являються у реальному часі від користувачів при перегляді кінофільму. Концепція коментування фільмів у реальному часі є дуже популярною у Китаї. У роботі була запропонована SBN-IVA для класифікації коментарів на спойлери та ні. У цій структурі спочатку витягуються текстові особливості коментаря за допомогою кодеру. Далі розробляється мережа, заснована на схожості (SBN), щоб отримати схожість сусідів та ключових кадрів відповідно до семантичної подібності та часових межкоментаря. Потім впроваджуєтьсяInteractiveVarianceAttention(IVA), щоб усунути вплив шумових коментарів. Нарешті, отримується ймовірність спойлеру на основі різниці між

схожістю сусідів та ключових кадрів. Експерименти показують, що показник SBN-IVA в середньому на 11,2% перевищує найсучасніший метод оцінки F1.

Як видно з огляду літератури, тема класифікації тексту на спойлери є актуальною та доцільною для дисертаційного дослідження. Вже існують відомі моделі для вирішення проблеми та розроблюються нові. Проте, в існуючих роботах попередників в основному застосовуються лексичні властивості речень, що дає меншу точність у визначенні спойлерів, адже для даної задачі важливим є максимальне отримання контекстної інформації та виділення нових тверджень базуючись на контекстній інформації.

Проблемою даної предметної області є реалізація визначення спойлерів на основі семантичної подібності текстів,виділення методу, який є найефективнішим у задачі визначення спойлерів у оглядах медіаконтенту та побудови комплексного рішення автоматизованої системи для визначення спойлерів.

Аналіз, проведений вище, показав, що існують методи для класифікації текстових даних, але вони мають багато недоліків, зокрема: в основному вони виконують аналіз текстових даних на основі лексичних властивостей та інших метаданих, а також вони не були застосовані до пошуку спойлерів. Крім того, проблема побудови комплексної автоматизованої системи, яка буде використовувати натреновану модель, також залишається відкритою. Отже, необхідно вибрати декілька моделей для семантичного аналізу, дослідити їх ефективність на задачі визначення спойлеру, обрати найефективнішу та, у результаті, розробити автоматизовану систему для визначення спойлерів у оглядах медіаконтенту.

Новизна дисертаційного дослідження полягає у використанні методів семантичного аналізу текстів для вирішення задачі визначення спойлерів та розробці комплексної автоматизованої системи для визначення спойлерів у тексті,

яка має вирішити проблему багатьох людей, у яких відпадає бажання переглядати медіаконтент після отримання спойлеру.

1.4 Постановка задач дослідження

Метою дослідження є створення інтелектуальної системи дослідження та аналізу текстів на предмет наявності спойлерів у оглядах медіаконтенту. Для досягнення мети необхідно розробити програмне забезпечення для аналізу текстових даних на предмет наявності спойлерів, автоматизовану систему, яка буде використовувати програмне забезпечення аналізу тексту. На виході повинне бути комплексне рішення для автоматизованого аналізу текстів для визначення спойлерів. Більш детальний опис задач, які необхідно виконати для досягнення мети наведений нижче:

- обрати методи, які можуть бути застосовані для класифікації тексту;
- виконати аналіз обраних алгоритмів/моделей для визначення найефективнішого на основі метрик матриці помилок;
- розробити веб-сервер, який містить треновану нейронну мережу, яка приймає на вхід опис медіаконтенту та його огляд, та виводить класифікацію щодо того, відноситься він до спойлеру, чи ні;
- розробити Telegram бот, який буде виконувати фільтрування спойлерів у чаті обговорення кінострічки;
- розробити стартап-проект створеної системи для того, щоб проаналізувати ринок та зрозуміти наскільки проект є технологічно здійсненним та актуальним.

При розробці Telegram боту необхідно врахувати те, що для відображення прихованого коментаря, необхідно заздалегідь зберігати коментарі, які виявилися спойлерами у окрему базу даних. Для унікальної ідентифікації запису у базі даних необхідно взяти ідентифікатор чату, ідентифікатор коментаря, номер частини

коментаря у якості комплексного первинного ключа.При розробці комплексної системи необхідно досягти атомарності та масштабованості кожної частини системи.

Висновок до розділу

У даному розділі було проведено аналіз актуальності теми дисертаційної роботи, аналіз конкурентів, огляд літератури попередників. З огляду на все наведене вище можна зробити висновок, що дана тема є актуальною, як з соціальної точки зору, так і з точки зору вибору методів та алгоритмів для вирішення проблеми. До сих пір з'являються наукові статті, які намагаються вирішити проблему визначення спойлерів. Це свідчить про актуальність та доцільність теми. Також у процесі пошуку комплексного рішення для Telegram, аналогів не було знайдено.

2 МЕТОДИ АНАЛІЗУ ТЕКСТОВИХ ДАНИХ

2.1 Попередня обробка текстових даних

Для тренування моделей було обрано набір даних ресурсу IMDB, який складається з двох файлів формату JSON – IMDB_movie_details.json та IMDB_reviews.json. Кількість даних датасету наведена у таблиці 2.1.

Таблиця 2.1 – Кількість даних датасету

Тип даних	Кількість записів
Описи кінострічок	1572
Відгуки користувачів	573913
Відгуки, які містять спойлери	150924

Типи даних, які зберігаються в цих файлах наведені у таблицях 2.2 та 2.3.

Таблиця 2.2 – Опис даних файлу IMDB_movie_details.json

Назва атрибуту	Тип даних	Короткий опис
movie_id	string	Унікальний ідентифікатор кінострічки
plot_summary	string	Короткий зміст сюжету кінострічки, який не містить спойлерів
duration	string	Тривалість кінострічки
genre	string[]	Жанри кінострічки
rating	string	Рейтинг кінострічки
release_date	string	Дата виходу кінострічки
plot_synopsis	string	Зміст сюжету, який містить спойлери

Таблиця 2.3 – Опис даних файлу IMDB_reviews.json

Назва атрибуту	Тип даних	Короткий опис
review_date	string	Дата, коли було написано відгук
movie_id	string	Унікальний ідентифікатор кінострічки
user_id	string	Унікальний ідентифікатор користувача, який написав відгук
is_spoiler	bool	Прапорець спойлеру
rating	string	Рейтинг, який користувач дав кінострічці
review_text	string	Текст відгуку користувача
review_summary	string	Короткий зміст відгуку

Насправді, даний набір даних містить набагато більше атрибутів, проте їх можна опустити, через те, що вони є непотрібними для задачі виявлення спойлерів. Отже, для формування вихідних даних, необхідно об’єднати два файли за атрибутом movie_id. Після цього отримуємо набір даних, який містить наступні атрибути:

- movie_id;
- plot_synopsis;
- review_summary + review_text;
- is_spoiler.

Нейронні мережі потребують на вхід кодування слів у реченні. Тому, наступним етапом є кодування слів оглядів та відгуків кінострічок у так звані wordembeddings.

Word embeddings – це представлення слів у реченні у вигляді векторів, які кодують семантичне значення кожного слова. Існує безліч моделей для кодування слів у тексті. Серед найпопулярніших:

- bag-of-words;
- term-document;
- TF-IDF;
- word2vec.

Підхід Bag-of-words бере вектор, довжина якого дорівнює довжині словнику слів, та будує one-hotencoding вектор для кожного слова. Далі виконується складання всіх цих векторів слів у реченні. На виході отримуємо підрахунок кількості слів у тексті у одному векторі. Назва підходу підказує, що положення слів у тексті втрачається.

Недоліками даного підходу є те, що він не кодує семантичне значення слів у тексті та використовує дуже багато пам’яті для збереження векторів – їх довжина дорівнює довжині словника слів.

Term-document – це підхід, який кодує слова у реченні у вигляді матриці «слово-документ». Дана матриця рахується за допомогою добутку двох матриць – «слово-тема» та «тема-документ».

Незважаючи на те, що затрати пам’яті менші, аніж у Bag-of-words, даний підхід не підходить через те, що слова кодуються у вигляд матриці.

TF-IDF – це модифікація Term-document та показує відношення числа текстівв яких зустрілося шукане слово, до загальної кількості текстів в корпусі.

Усі вищеописані підходи мають один великий недолік – вони не справляються із словниками великих об’ємів. Через це розмір матриці представлення слів у тексті міг досягати мільйонів рядків до мільйонів стовпців. Через дану проблему у 2013 році ТомашМиколов запропонував підхід до векторизації слів у тексті, який називається word2vec.

Word2vec засновується на гіпотезі, що слова, які зустрічаються в однакових оточеннях, мають близькі значення. Прогнозується ймовірність слова з його оточення (контексту). Тобто, навчаються такі вектори слів, щоб вірогідність, присвоєна моделлю слову, була близькою до ймовірності знайти це слово в цьому середовищі в реальному тексті. Хоча модель не містить явно ніякої семантики, а лише статистичні властивості корпусу тексту, виявляється, що навчена модель word2vec може фіксувати деякі семантичні властивості слів. Експерименти ТомашаМиколова показали, що його модель кодує семантичне значення слів, а також деякі синтаксичні зв'язки, такі, як співвідношення одиночного і множинного числа.

Тому для векторизації слів у тексті було використано модель word2vec, яка кодує семантичне значення слів огляду медіаконтенту та відгуку до нього. Таким чином на виході ми отримуємо два wordembeddings для опису та відгуку, які подаємо на вхід нейронної мережі.

2.2 Постановка задачі класифікації текстових даних

Задача класифікації текстових даних на «спойлер» та «не спойлер» полягає у знаходженні значення невідомої функціїФ(р, с), якщо дано опис медіаконтенту та огляд цього медіаконтенту,рта с відповідно. Дана функція повинна виконати семантичний аналіз двох текстів та визначити оцінку їх семантичної подібності. Дана оцінка може варіюватися від 0 до 1. При цьому будь-які вхідні дані повинні бути прокласифіковані незалежно від значення функції.

При цьому мають виконуватись наступні твердження при оцінці результатів виконання функції:

- огляд є спойлером, якщо $\Phi(p, c)>0.5$;
- огляд не є спойлером, якщо $\Phi(p, c)< 0.5$.

Для кожного набору вхідних даних може бути визначена лише одна оцінка функції.Вигляд та опис різновидів функцій наведено у наступному розділі.

2.3 Методи вирішення задачі

Після виконання огляду літератури у попередньому розділі стало очевидно, що дослідження методів та алгоритмів повинно спиратися на методи глибинного навчання. Результати досліджень викладених у наукових роботах попередників показують, що дані методи є більш ефективними, аніж більш прості методи, які базуються на відповідності ключових слів. Тому у даному розділі буде описано методи, якими можливо вирішити проблему дисертаційної роботи та їх порівняльні оцінки.

МодельManhattan LSTM [9] бере за основу принцип того, що, якщо умовно два речення (фрази) виражають одну і ту ж ідею – вони мають бути семантично схожі. А точніше семантична схожість має допомогти у визначенні схожих речень або фраз. До того ж хороша модель не повинна залежати від кількості слів або синтаксису написання. Дана проблема до сих пір є актуальною через те, що такі моделі, як bag-of-words/tf-IDF (які є надзвичайно популярними у NLP), якраз дуже сильно спираються на специфічність термінів.

Як зрозуміло із назви цієї моделі, її основою є LSTM. LSTM – це різновид рекурентних нейронних мереж (RNN), які добре підходять через те, що приймають змінну довжину вхідних даних (речень або фраз у нашому випадку). Як відомо, LSTM використовує послідовні дані (x_1,...,x_T), де при кожній ітерації T виконуються оновлення вектора прихованого стану. Отримання вектора прихованого стану наведено у (2.1).

Manhattan LSTM – це покращення звичайного LSTM. У даній моделі береться дві мережі LSTM для кожного з порівнюваних текстів. Модель

використовує LSTM для зчитування у векторів слів, що представляють кожне вхідне речення, і використовує його останній прихований стан як векторне представлення для кожного речення. Згодом подібність між цими поданнями використовується як предиктор семантичної подібності. Вектори порівнюються для того, щоб отримати оцінку схожості. Оцінка схожості рахується на основі Манхетенської відстані між векторами речень.Підрахунок семантичної подібності наведений у (2.2).

Модель LSTM на основі attention механізму [10]є надстройкою над попередньою через додатковий шар attention механізму. Тобто модель складається з двох частин: двонаправлена LSTM та attention механізм. По суті, attention механізм – це сума всіх попередніх векторів прихованого стану. Результуючі зважені вектори прихованого стану LSTM вважаються вбудованими реченнями. Attention механізм дозволяє вбудованому реченню мати доступ до всіх попередніх прихованих станів LSTM, а сам LSTM має доступ лише до попереднього прихованого стану.

Нижче наведено рисунок 2.1, який відображає отримання матриці H, яка відображає речення у вигляді матриці векторів слів.

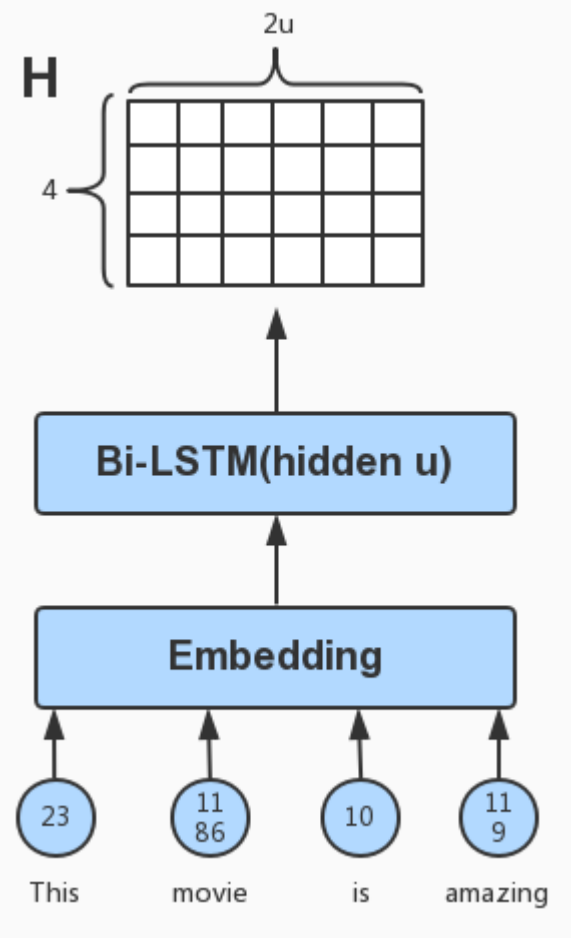

Рисунок 2.1 – Матриця H векторів слів

Нижче наведено рисунок 2.2, який відображає отримання attention матриці A.

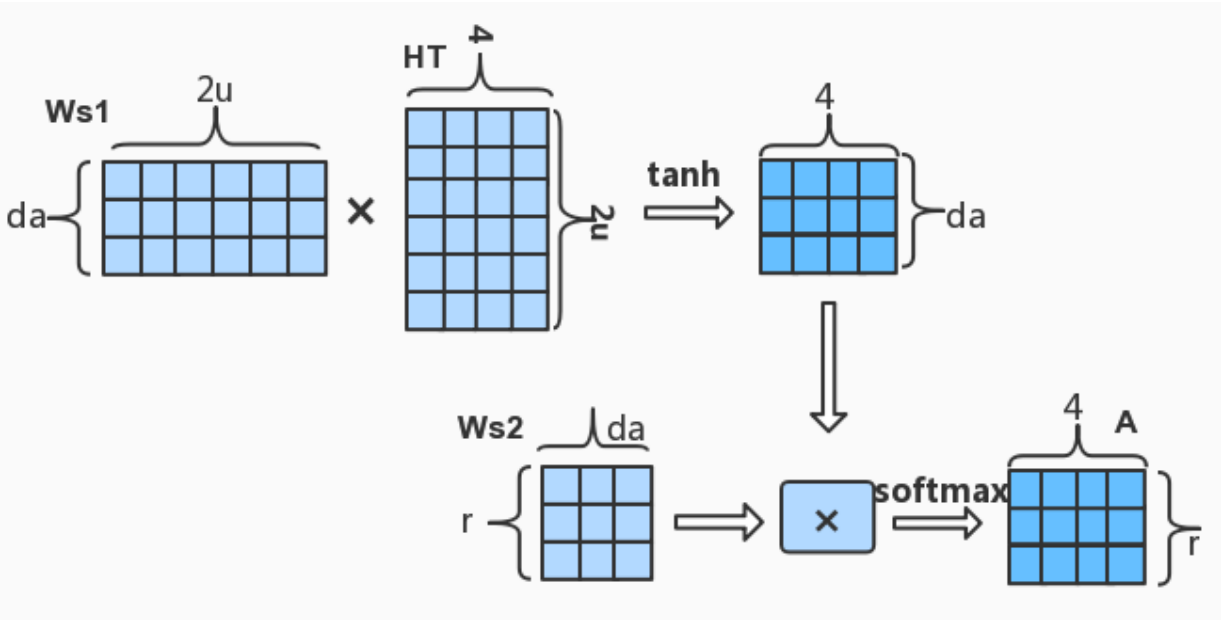

Рисунок 2.2 – Attention матриця A

Так як великим реченням властиво мати декілька логічних частин, які розділені сполучниками, є необхідність у приміненніattention механізму на декількох частинах речення. Таким чином, attention вектор перетворюється у матрицю, яка містить усі вектори у кількості, яка дорівнює кількості логічних частин речення. Рівняння дляattention матриці наведено у (2.3).

Таким чином, для отримання вбудованого речення використовується матриця A та вектори прихованого стану H. Рівняння вбудованого речення наведено у (2.4).

Дана модель також може перетворити будь-яку послідовність слів у сталу довжину. Як показали досліди, дана модель відповідно має перевагу при великому розмірі речення, що є великим плюсом зважаючи на розміри резюме до фільмів.

Модель Enhanced LSTM (ESIM) [11] заснована на одній із задач Naturallanguageinference. Naturallanguageinference – це завдання визначити, чи є гіпотеза істинною, хибною чи невизначеною з урахуванням передумови.

Наприклад, нехай передумовою буде «На вулиці дощить». Згідно з цією передумовою, гіпотеза типу «Дощ заливає вулиці міста» буде істинною, а гіпотеза типу «Вулиця знаходиться на пагорбі» буде хибною. Для задачі дисертаційної роботи невизначена гіпотеза не потрібна, так як потрібно у будь-якому випадку визначити чи частина тексту спойлер, чи ні.

Підзадача, яку повинна вирішити дана модель – це так називаємаentailment задача – співвідношення вірне, коли істинність одного фрагмента тексту випливає з іншого тексту.

З назви моделі стає очевидним, що для того, щоб знайти прихований вектор також використовується LSTM, але його двонаправлений різновид (BiLSTM). BiLSTM обчислює результат, запускаючи два незалежних LSTM, один у прямому напрямку, а другий у зворотному напрямку. Потім отримані вектори об'єднуються.

Для отримання кінцевого результату текстові дані проходять три етапи.

Перший етап – кодування речень у вектори. Використовуючи BiLSTM, відображається зв’язок між словом у реченні та його контекстом.

Другий етап – процес підрахування attention. Використовуючи приховані вектори, які були отримані на попередньому етапі, підраховується матриця схожості.

Ваги матриці використовуються для обчислення контекстних векторів, що фіксують вигляд передумови для кожного слова в гіпотезі, і навпаки.

Потім модель використовує модуль генерації підкомпонентів.

Третій етап – агрегування результатів. Нарешті, для узагальнення результатів, другий цикл BiLSTM запускається для векторів підкомпонентів. Потім приховані стани поєднуються двома різними способами, обчислюючи максимальний та середній показники за часом (AvgPooling, MaxPooling).

2.4 Порівняння моделей

Так як дана задача зводиться до класифікації тексту на той, що містить спойлери та той, що не містить спойлери, було обрано метрики, які засновуються на матриці помилок (confusionmatrix).

Матриця помилок – це таблиця, яка показує ефективність алгоритму класифікації шляхом порівняння прогнозованого значення цільової змінної з її фактичним значенням. Основними термінами даної матриці є число вірно прогнозованих позитивних цілей TP, число фактично негативних цілей, які були спрогнозовані як позитивні FP, число фактично позитивних цілей, які були спрогнозовані як негативні FN та число вірно прогнозованих негативних цілей TN.

Отже, було обрано дані метрики для порівняння ефективності алгоритмів:

– точність (accuracy) – доля правильних передбачених спойлерів до загальної кількості спойлерів;

– втрата (loss) – доля неправильно передбачених спойлерів до загальної кількості спойлерів;

– точність (precision) – доля спойлерів, які були визначені як позитивні;

– повнота (recall) – це відношення правильно передбачених спойлерів до всіх спойлерів;

– F1 – це середньозважене значення точності (precision) та повноти (recall).

Precision та recall не залежить, на відміну від accuracy, від співвідношення класів і тому застосовні в умовах незбалансованих вибірок.

Для візуалізації значень метрик було побудовано графіки залежності значень accuracy та loss від номеру епохи. Також для порівняння значень метрики F1 було побудовано таблиці, які відображають ці значення для обох типів класифікації – «спойлер» та «не спойлер».

Результати для методу Manhattan LSTM наведені на рис. 2.3, рис. 2.4 та табл. 2.4.

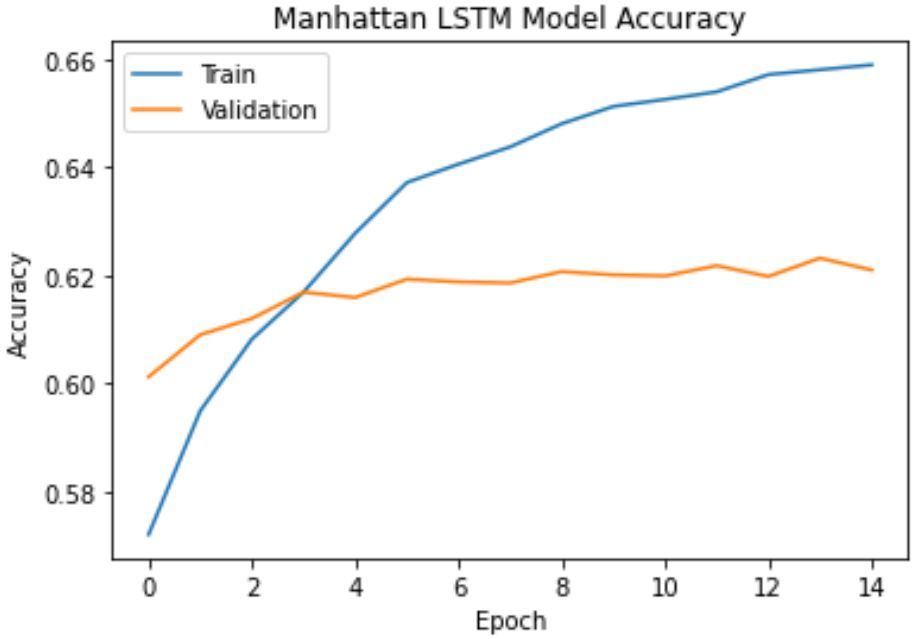

Рисунок 2.3 – Accuracy для моделі Manhattan LSTM

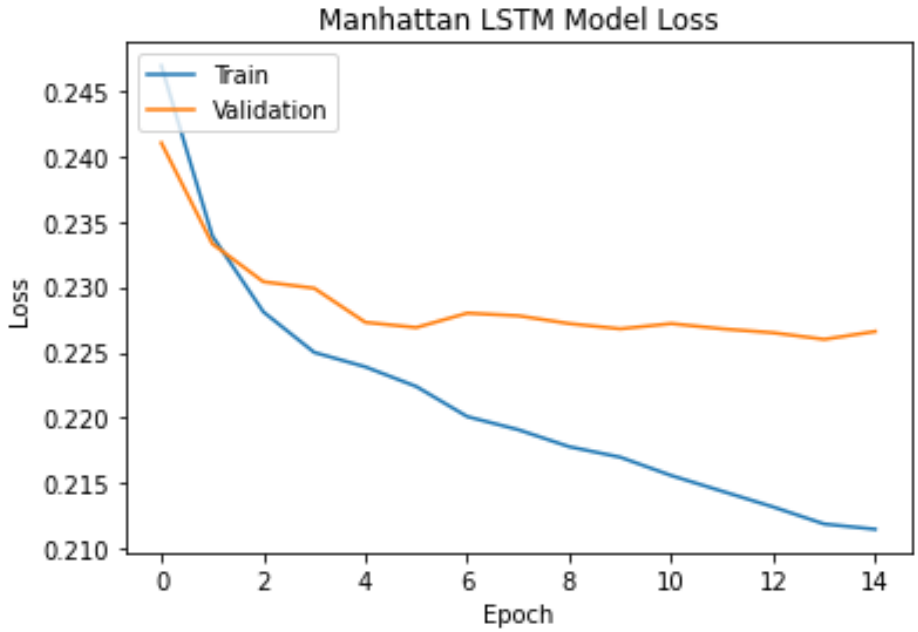

Рисунок 2.4 – Loss для моделі Manhattan LSTM

Таблиця 2.4 – Результати метрик для моделі Manhattan LSTM

	Precision	Recall	F1
Спойлер	0.71	0.14	0.23
Не спойлер	0.74	0.96	0.84

Результати для методу LSTM на основі attention механізму наведені на рис. 2.5, рис. 2.6 та табл. 2.5.

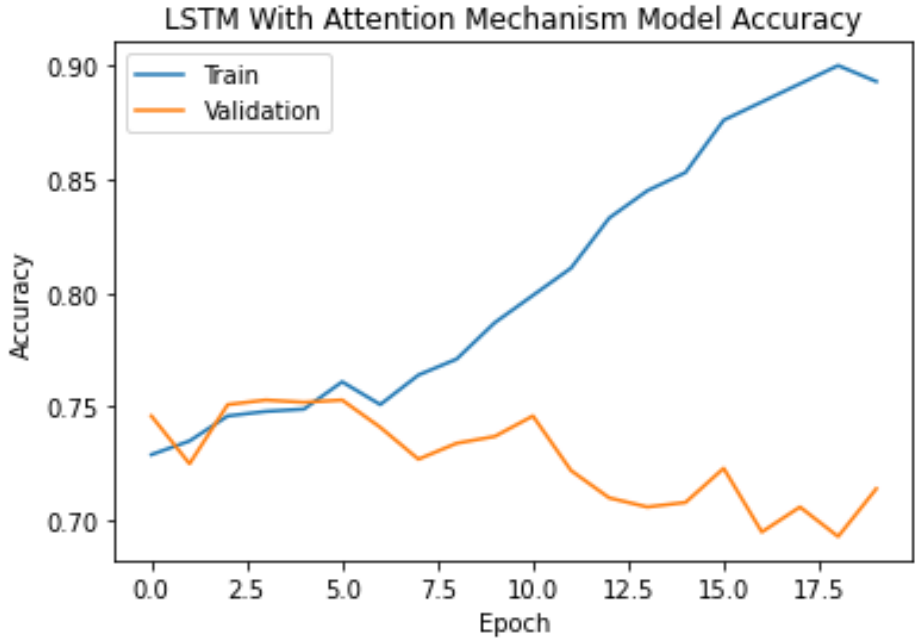

Рисунок2.5 – Accuracy для моделі LSTM на основі attention механізму

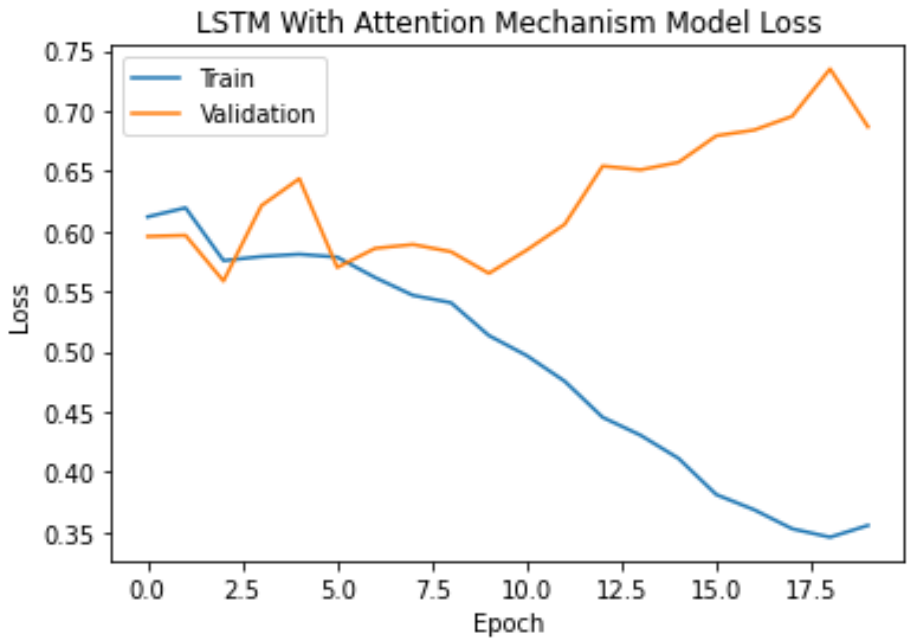

Рисунок2.6 –Loss для моделі LSTM на основі attention механізму

Таблиця2.5 – Результати метрик для моделі LSTM на основі attention механізму

	Precision	Recall	F1
Спойлер	0.36	0.56	0.44
Не спойлер	0.38	0.55	0.45

Результати для методу Enhanced LSTM (ESIM) наведені на рис. 2.7, рис. 2.8 та табл. 2.6.

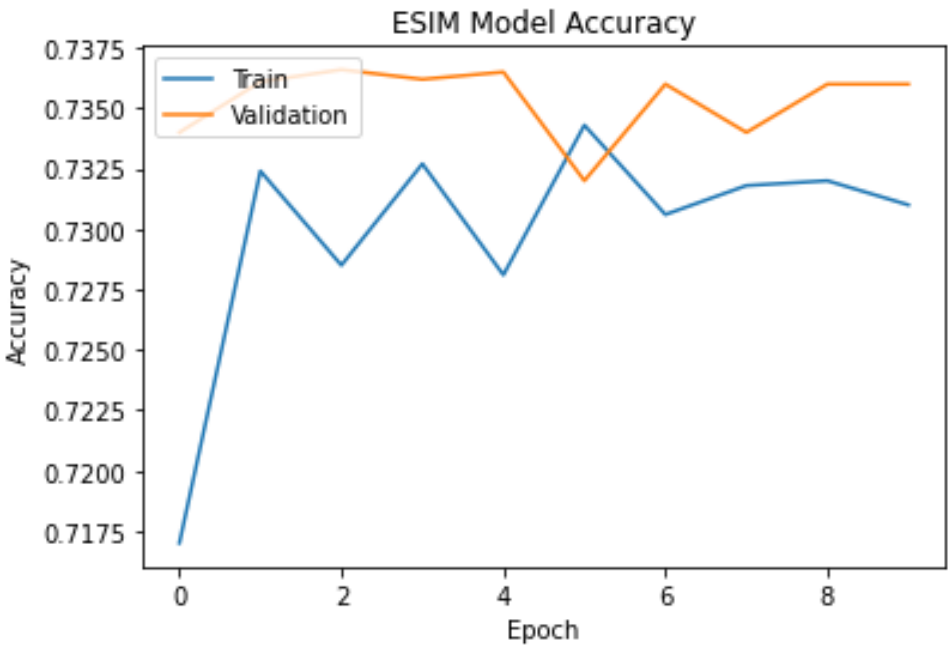

Рисунок2.7 –Accuracy для моделі Enhanced LSTM (ESIM)

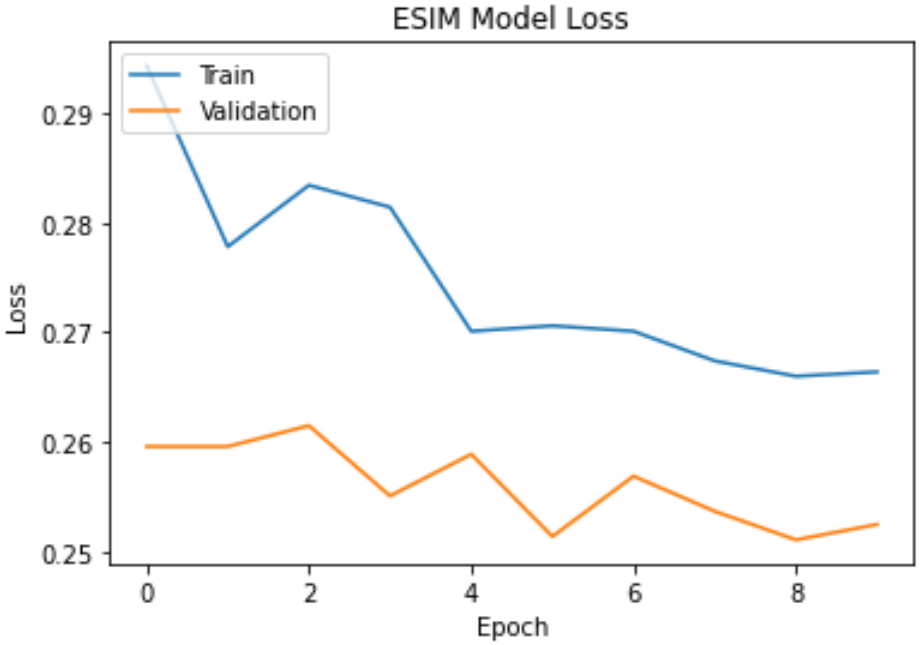

Рисунок2.8 –Loss для моделі Enhanced LSTM (ESIM)

Таблиця2.6 – Результати метрик для моделі Enhanced LSTM (ESIM)

	Precision	Recall	F1
Спойлер	0.64	0.69	0.66
Не спойлер	0.66	0.62	0.64

Як видно із результатів для методу Manhattan LSTM, recall для «спойлерів» є 0.14, що може казати про те, що дана модель буде рідше позначати справжні спойлериспойлерами. Натомість F1 для «не спойлерів» є дуже високим. Це каже про те, що дана модель позначає більшість текстів такими, що не є спойлерами, що не є ідеальним.

Як видно із результатів для методу LSTM на основі attention механізму середня accuracy є досить високою – на рівні 0.72, але показник loss більший, аніж у попередній моделі – ~0.63. Також незважаючи на те, що показник F1 є однаковим для обох класифікацій, проте він не є більшим аніж у попередній моделі.

Як видно із результатів для методу Enhanced LSTM (ESIM) показник середньої втрати є маленьким та майже рівний тому для методу Manhattan LSTM. Натомість F1 для обох класифікацій є майже однаковим та вищим аніж у Manhattan LSTM та LSTM на основі attention механізму. Це говорить про те, що дана модель має меншу залежність від відношення кількості «спойлерів» до «не спойлерів» у вхідних даних. Також показник точності даної моделі, що дорівнює ~0.73, є найвищим серед усіх моделей.

Отже, можна зробити висновок, що модель Enhanced LSTM (ESIM) загалом є покращенням, на відміну від Manhattan LSTM та LSTM на основі attention механізму, адже має найвищі показники accuracy, F1 для обох видів класифікації та показник loss, що дорівнює ~0.25.

2.5 Порівняння із методом лінгвістичного аналізу

Для наведення поліпшень у дослідженнях, описаних у попередньому розділі, необхідно порівняти методи для семантичного аналізу із методами для лінгвістичного аналізу тексту на предмет наявності спойлерів.

Для реалізації порівняння текстів на лексичні властивості було обрано класифікатор SVM (supportvectormachine), який належить до алгоритмів навчання із вчителем.

Навчання із вчителем це одна із задач машинного навчання, яка полягає у навчанні функції на парах вхідних-вихідних значень. Для даної задачі необхідно мати набір позначених навчальних даних, які містять вхідний вектор та вихідне значення. Задача визначення спойлерів отримує на вхід набір прокласифікованих даних на основі наявності спойлерів у оглядах медіаконтенту.

Для задачі класифікації було обрано метод опорних векторів SVM.Даний класифікатор приймає на вхід набір даних поділених на категорії та виконує категоризацію нових текстових даних. SVM виконує класифікацію шляхом побудови гіперплощини, яка розділяє вектори у н-мірному просторі на дві категорії. Суттю SVM є знаходження найкращої гіперплощини, тобто такої, що відстань до найближчих векторів є максимальною. На рисунку 2.9 зображений приклад найкращої гіперплощини у тримірному просторі.

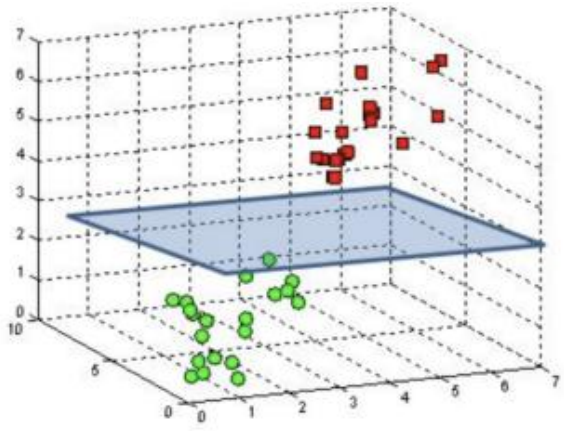

Рисунок 2.9 – Найкраща гіперплощина у задачі з трьома наборами властивостей

Класифікація тексту на предмет наявності спойлерів пов’язана із такими базовими проблемами обробки природньої мови, як визначення наскільки дія описана в реченні є завершеною [12] та визначення які речення належать до подій, які описані пізніше.

Перша проблема заснована на тому факті, що спойлери зазвичай містять слова, які описують обдумані, дієві, завершені дії. Наприклад, така дія, як «втекти» або «виконати побіг» є завершеною, впливає на об’єкт до якого застосовується.

Друга проблема заснована на тому факті, що спойлери зазвичай описують події, які знаходяться хронологічно пізніше, аніж ті знання, які на даний момент має читач.

Речення оглядів медіаконтенту були подані у вигляді векторів лексичних властивостей слів, н-грам та н-грам символів. У якості лексичної властивості текстів було обрано відноснуважливістьтерміна в документі та в цілому тексті або ж скорочено – TF-IDF.TF-IDF це показник, який визначає важливість терміну в документі засновуючись на двох оцінках:

– відношення кількості разів входжень токена у тексті до загального числа токенів у тексті;

– інверсія частоти, кількості входжень слова у текстах.

Вектори TF-IDF були згенеровані на трьох типах токенів текстів:

– словах – матриця, яка представляє TF-IDF оцінку кожного слова у текстах;

– н-грамах – матриця, яка представляє TF-IDF оцінку кожної н-грами у текстах;

– н-грамах символів – матриця , яка представляє TF-IDF оцінку кожної н-грами на рівні символів у текстах.

Перед тим, як формувати вектори властивостей, тексти оглядів медіаконтенту було оброблено наступним чином. Спочатку були приведено увесь текст до нижнього регістру. Потім огляди було розбито на токени, використовуючи метод word_tokenize бібліотеки NLTK.Потім було видалено так звані stopwords, які не несуть важливості у аналізі тексту, наприклад, та, або, його тощо. Потім було видалено усі символи, які не є літерами або числами. Потім було проведено операції лематизації та стемінгу, що означає приведення слів до їх базової форми шляхом відкидання таких частин слова, як суфікс, закінчення.

Далі огляди медіаконтенту були розбиті на тренувальну та тестувальну вибірки у співвідношенні 70% до 30%.

Для оцінення класифікатора його було натреновано на тренувальній вибірці та перевірено на тестовій вибірці. Для визначення показнику точності (accuracy) було взято частину правильно визначених речень як «спойлери» та «не спойлери». У результаті були отримані наступні результати точності:

- для векторів TF-IDF на рівні слів – 61%;
- для векторів TF-IDF на рівні н-грам – 57%;
- для векторів TF-IDF на рівні н-грам символів – 62%.

Можна зробити висновок, що дані показники точності є меншими, аніж ті для методу семантичного аналізу тексту, що дорівнює 73%. При використанні методу семантичного аналізу досягається покращення у точності на 11%.

Отже, можна зробити висновок, що застосування сучасних моделей нейронних мереж для вирішення задачі визначення спойлерів, шляхом семантичного аналізу тексту, дало значене покращення у порівнянні із методом для лексичного аналізу тексту.

Висновок до розділу

Аналіз текстових даних є комплексною задачою, яка може бути вирішена безліччю способами.У даному розділі було описано вхідні дані, приведено процес попередньої обробки вхідного датасету.

Також у даномурозділі було зроблено огляд методів для знаходження спойлерів, заснованих на рекурентних нейронних мережах із застосуванням модифікацій, таких як attention механізм, LSTM та двонаправлена LSTM. Моделі викладені у даномурозділі були порівняні використовуючи метрики, такі як accuracy, loss, precision, recall та F1. Для показників accuracy та loss було побудовано графіки їх значень на кожній епосі тренування моделі. Для показників precision, recall та F1 було побудовано таблиці їх порівняння під час класифікації текстів на «спойлери» та «не спойлери». На основі побудованих графічних та числових значень метрик, було обрано модель, яка є найбільш ефективною при аналізі даних оглядів медіаконтенту та відгуків на них.

Також у даному розділі було проведено порівняння із методом, який виконує лексичний аналіз текстів на основі класифікатора SVM та трьох наборах лексичнихвласивостей. Результати порівняння показали, що застосування методів семантичного аналізу на основі рекурентних нейронних мереж та семантичного аналізу дало значне покращення точності визначення спойлерів, а саме на 11%.

3 АВТОМАТИЗОВАНА СИСТЕМА АНАЛІЗУ ТЕКСТОВИХ ДАНИХ

3.1 Архітектура автоматизованої системи

Метою автоматизованої системи є забезпечення користувачів Telegram каналів надійним інструментом, який буде автоматично розпізнавати спойлери у коментарях до публікацій.

24 червня 2015 року команда Telegram випустила чергове оновлення платформи, у якому було повідомлено про реліз API для ботів та платформи для незалежних розробників для створення ботів.

Сам по собі Telegram бот – це акаунт Telegram, якими керує програмне забезпечення, а не люди, і вони часто мають функції штучного інтелекту. Вони можуть робити такі речі, як навчати, грати, шукати, транслювати, запам'ятовувати, підключатись, інтегруватися з іншими службами. Основні завдання, які можуть вирішувати боти:

- інтеграція з сервісами;
- виконання електронної оплати рахунків;
- створення ігор;
- отримання індивідуальних сповіщень та новин тощо.

Акаунти Telegram ботів не потребують окремого прив’язаного номеру телефона. Користувачі можуть спілкуватися із ботами двома шляхами:

- надсилати повідомлення та команди ботам, спілкуючись з ними в чаті або додаючи їх до груп;
- надсилати запити безпосередньо з поля введення, ввівши @username та запит бота. Це дозволяє надсилати результат виконання задачі боту безпосередньо в будь-який чат, групу чи канал.

Що стосується розробників ботів, для того, щоб створити Telegram бот, можна використати будь-який фреймворк будь-якої мови програмування, яка має

можливість виконувати запити на HTTPS-інтерфейс.Повідомлення, команди та запити, надіслані користувачами, передаються програмному забезпеченню, що працює на серверах. Брокерський сервер виконує усі функції шифрування та зв’язку з API Telegram. Спілкування із сервером відбувається через HTTPS-інтерфейс, який пропонує спрощену версію API Telegram.

Перед розробкою бота, було отримано унікальний токен, який потрібен для того, щоб створити так званий TelegramBotClient. TelegramBotClient – це клієнт, який дає змогу виконувати асинхронні запити по відношенню до бота.

Для задачі автоматизованого визначення спойлерів Telegram бот підходить, адже даний підхід є повністю автоматизованим та не потребує частого адміністрування програмного забезпечення боту. Після написання серверної частини боту, даний програмний продукт може бути задеплоєний на один із хмарних провайдерів, таких як, Azure, Aws, GoogleCloud, Heroku тощо.

Для отримання максимальної ефективності та мінімальних витрати по адмініструванню, Telegram бот повинен бути розроблений таким чином, щоб відслідковувати кожне повідомлення, яке надходить у коментарі кожної публікації та визначати чи наявні спойлери у коментарі. Далі, якщо результату розпізнавання спойлеру є негативним, бот повинен ігнорувати повідомлення, Інакше, якщо результату розпізнавання спойлеру є позитивним, бот повинен зберегти вихідний коментар користувача у базу даних та замінити даний коментар на такий, що містить повідомлення із прихованим контентом та кнопкою, яка при натисканні, покаже прихований вміст. При отриманні команди натискання на дану кнопку, Telegram бот повинен витягти вихідне повідомлення із спойлером та повернути його Telegram клієнту.

Загальна схема оперування програмного забезпечення боту зображена на рис. 3.1.

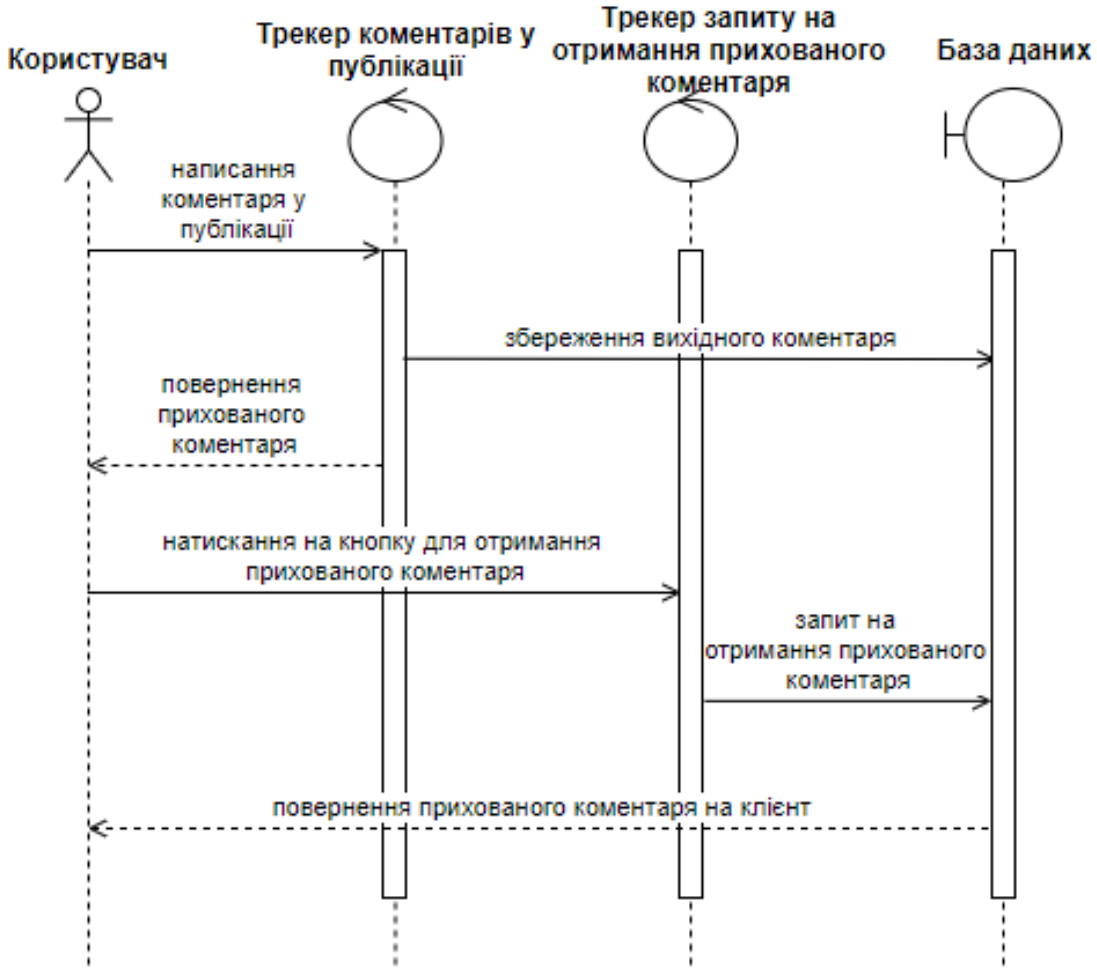

Рисунок 3.1 – Діаграма послідовності Telegram боту

Як видно із діаграми послідовності, яка зображена на рис. 3.1, трекер коментарів у публікаціях та трекерзапитів на отримання прихованого коментаря є асинхронними задачами, які виконуються потоком виділеним у ThreadPool.

Для написання серверної частини Telegram боту, була обрана платформа .NetCore 3.1. Дана платформа має можливість виконання асинхронних задач. Дана властивість платформи чудово працює для завдань Telegram боту, що полягають у слідкуванні за командами введення нового коментаря у публікації та запиту на отримання прихованого коментаря.Усі команди, які приходять від Telegram клієнта оброблюються у так званому ThreadPool, тим самим не блокується потік, який відповідає за виконання програми на .NetCore 3.1.

ThreadPool – це вбудована система у платформі .NetCore. Вона надає пул потоків, керований класом ThreadPool, яким керує система. Як розробнику, не потрібно мати справу із витратами на управління потоками.ThreadPool складається із робочих потоків, які вже були створені. Коли потік завершує виконання власної таски, він повертається до пулу потоків. Таким чином значно скорочуються витрати ресурсів та часу на створення нових потоків. Потоки з пулу потоків підходять для фонових процесів.

База даних повинна зберігати вихідні повідомлення, які були розпізнані системою визначення спойлерів, як такі, що містять спойлери. База даних повинна мати таблицю із даною структурою:

– атрибут текстового типу, який зберігає унікальний ідентифікатор чату, в якому було введено повідомлення;

– атрибут цілочисельного типу, який зберігає унікальний ідентифікатор повідомлення;

– атрибут цілочисельного типу, який зберігає частину коментаря;

– атрибут текстового типу, який зберігає вихідне повідомлення, яке містить спойлер.

У якості бази даних було обраноPostgresDB. Ситуація із збереженням коментарів із спойлерами потребуєсталої структури бази даних ташвидкості зчитування даних та запису даних у таблиці.Необхідною умовою у виборі СУБД також був варіант безкоштовної СУБД з відкритим вихідним кодом. Саме тому PostgresDBє найкращим у даній ситуації.

Користувач Telegram каналу має змогу вводити коментарі під публікаціями та отримати приховані повідомлення, які містять спойлери. У свою чергу, адміністратор Telegram каналу має змогу додавати бота до свого каналу. Для коректної роботи боту, адміністратор повинен надати боту роль адміністратора у

каналі для обговорень. Діаграма сценаріїв використання автоматизованої системи користувачем зображена на рис. 3.2.

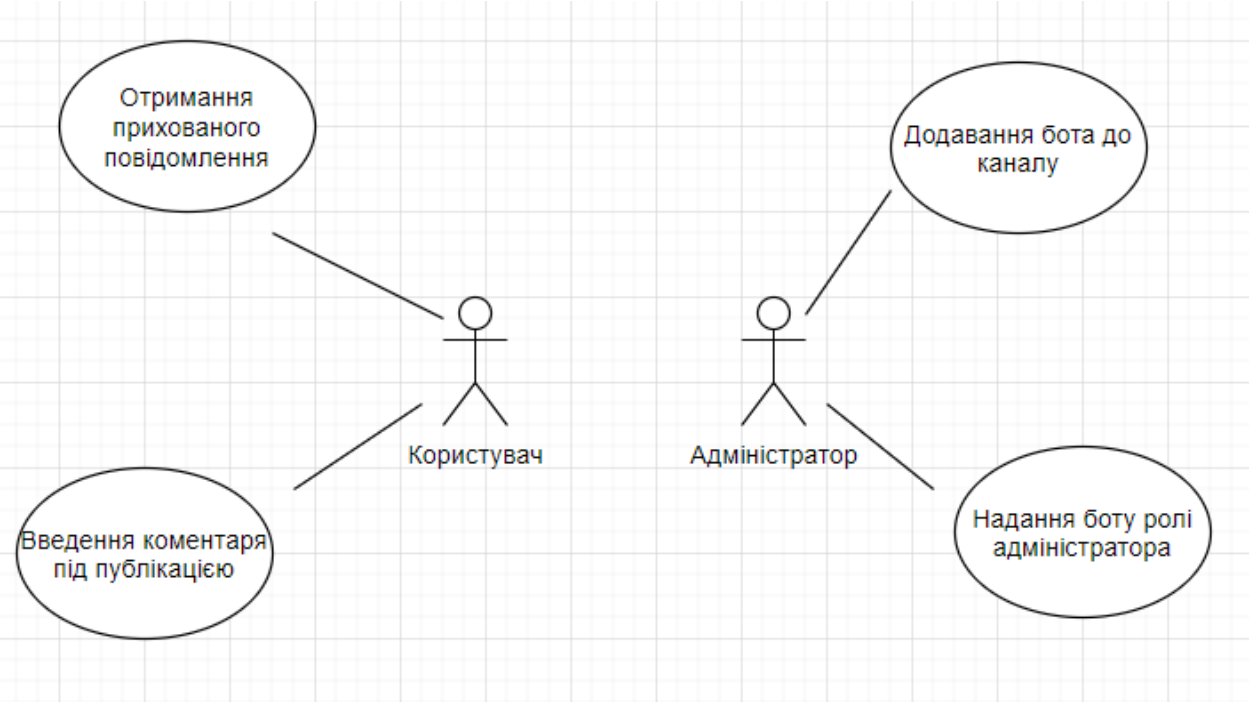

Рисунок 3.2 – Діаграма сценаріїв використання системи

3.2 Архітектура системи для визначення спойлерів

Система для визначення спойлерів має за мету зберігати натреновану модель, яка була описана у попередньому розділі, та мати HTTPS-інтерфейс для отримання результату розпізнавання спойлеру у коментарі до публікації кінострічки. Також важливою вимогою є асинхронне виконання визначення спойлеру, щоб не блокувати потік, який запросив дану дію.

Для написання моделей визначення спойлерів було обрано сервіс на Python, який виконує аналіз текстових даних за допомогою методів машинного навчання.

Для підрахування метрик, таких як точність, F1-score тощо, обуло застосовано бібліотеку sklearn.Sklearn – це безкоштовна бібліотека машинного навчання із відкритим вихідним кодом, яка написана на мові Python. Всередині бібліотека використовує іншу бібліотеку під назвою numpy для того, щоб

виконувати операції з матрицями та векторами з великою продуктивністю. Також sklearn повністю інтегрується з бібліотекою для відображення графіків, яка описана нижче.

Для відображення графіків було використано бібліотеку matplotlib.Matplotlib – це велика бібліотека на мові програмування Python, яка містить інструменти для будування графіків та інших візуальних елементів. Було використано колекцію функцій, що належить до пакету pyplot, які дають можливість створювати графіки залежностей.

Для створення та будування моделі було використано бібліотеку mxnet, а саме набір функцій пакету gluon.Mxnet – це бібліотека глибинного навчання із відкритим вихідним кодом. Вона підтримує безліч мов програмування, наприклад, C++, Python, Java, JavaScript, Go, R, Scala тощо, а також підтримується такими хмарними провайдерами, як AmazonAws, Microsoft Azure. Тому дана бібліотека є підходящим варіантом у разі деплою програмного забезпечення у один із хмарних провайдерів. Дана інфраструктура дозволяє розподіляти навантаження на безліч інстансів центральних або графічних процесорів.Пакет функцій gluon містить функції для тренування рекурентних нейронних мереж та їх різновидів, зокрема LSTM (longshort-termmemory) та GRU (gatedrecurrentunit).

Для тренування моделей було використано бібліотеку mxnet та пакет функцій gluon, який дає можливість налаштовувати оптимізатор.

Для створення та зчитування датасетів для тренування та тестування було використано бібліотеку pandas. Pandas – бібліотека написана на мові програмування Python, яка дає можливість оброблювати та маніпулювати структурами даних, у тому числі таблицями. За допомогою неї було реалізовано створення файлів з даними для тренування.

Python сервіс було реалізовано за допомогою веб-фреймворку Flask – бібліотеки, яка надає інструменти для написання масштабованих веб-проектів.

Даний фреймворк був написаний на мові програмування Python та вважається мікро-фреймворком, адже основа проекту є максимально простою. На відміну від інших фреймворків, таких як Django, TurboGears, web2py тощо, yFlaskвідсутні вбудовані шари абстракцій доступу до бази даних, валідації тощо. Натомість Flask підтримує ряд доповнень, які можуть додати той, чи інший функціонал до веб-проекту. Серед найпопулярніших можна виділити pytest для додавання тестування до веб-проекту, celery для виконання фонових процесів тощо.

Для комунікації міжPython сервісом та Telegram ботомбуло реалізовано API за яким бот може отримати результати аналізу текстових даних. Реалізація доступу до Python сервісу за API дає деякі переваги, зокрема те, що сервіси комунікують за відомим протоколом та з використанням задокументованих сутностей. Крім того, якщо сервіси будуть якимось чином модифікуватися, API не буде змінюватися. Таким чином досягається масштабованість сервісів.

Telegram бот, який був описаний у попередньому підрозділі, надсилає запити на API Python сервісу та отримає результат аналізу вхідного тексту. Для отримання результату аналізу огляду медіаконтенту на предмет наявності спойлерів, необхідно виконати GETзапит на адресу /checkspoilerізнаступними параметрами тілазапиту у форматіJSON:

- plot – опис медіаконтенту;
- comment – огляд до даного медіаконтенту.

Python сервіс повертає результат у вигляді змінної, яка означає, що огляд виявився спойлером чи ні, truetafalseвідповідно.

У таблиці 3.1 наведені класи сервісу для аналізу текстових даних та їх призначення.

Таблиця 3.1 – Опис класів сервісу для аналізу текстових даних

Назва класу/файлу	Методи класу	Призначення
Trainer	train,get_data, build_model, iterate_epoch,train_and_validate	Виконуєпопереднюобробкуданих, будування та тренуваннямоделі
Dataloader	get_dataloader	Розбиття датасетів на бакети для досягнення більш швидкої ітерації
Tokenizer	clip_words, get_length, tokenize_dataset	Розбиттяописів та оглядів на токенислів
Indexer	token_to_index, indexate_dataset	Перетворення токенів у список індексів словника
Attention	forward	Отримання шару attention механізму
ApplyModelLayers	forward	Застосуванняшарів до моделі: attentionмеханізм, LSTM, embedding

Усі частини інтелектуальної системи зображені на рис. 3.3.

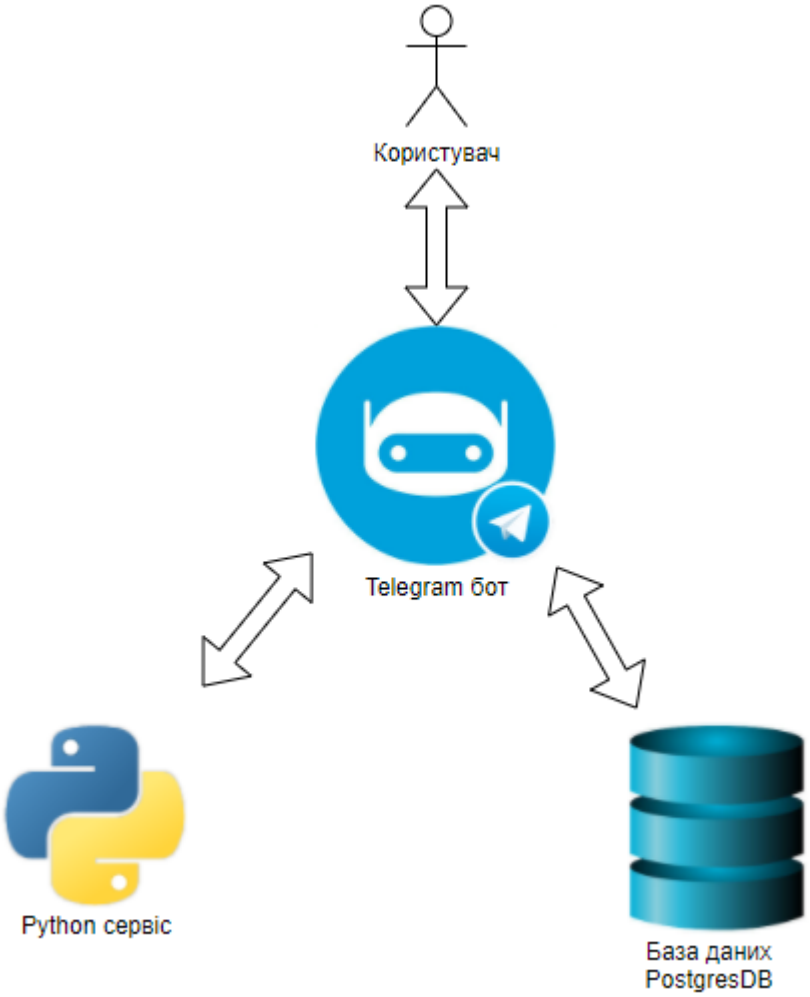

Рисунок 3.3 – Основні частини інтелектуальної системи

Висновок до розділу

У даному розділі було проведено аналіз архітектурних рішень для автоматизованої системи для визначення спойлерів у оглядах медіаконтенту. У результаті дослідження було побудовано схему взаємодії складових системи, обрано фреймворки для написання кожної частини системи. Аналіз та вибір технологій відбувався на основі вимог кожної складової інтелектуальної системи та на основі поточних тенденцій розвитку інформаційних технологій. Переваги обраних технологій були обґрунтовані.

Також було проведено аналіз сценаріїв використання системи та, в результаті, побудовано діаграму сценаріїв використання.

Для більш наочного представлення взаємодії автоматизованої системи із користувачем та залежними сервісами, було побудовано діаграму послідовності.

4 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ АНАЛІЗУ ТЕКСТУ

4.1 Опис програмного забезпечення

Автоматизована інтелектуальна система для визначення спойлерів у оглядах медіаконтенту складається з наступних модулів:

- Telegram бот;
- сервіс аналізу текстових даних;
- база даних.

Серверна частина Telegram боту була написана на об’єктно-орієнтованій мові програмування C# фреймворку .NetCore 3.1.Для виконання асинхронної обробки коментарів, які з’являються під публікаціями каналу, було реалізовано концепцію push технології.

Push технологія – це концепція поширення інформації, при якій сервер ініціює запит про оновлення деякої інформації, а клієнт у свою чергу підписується на оновлення серверу та отримує оновлення щойно вони сформовані на сервері.

Перед тим, як було реалізовано концепцію push технології, сервер та клієнт спілкувалися виключно запитами HTTP протоколу. При даному підході сервер не мав можливості проактивно інформувати клієнтів про оновлення інформації. Дана проблема виявилася найгострішою у таких додатках, як чати, невеликі ігри тощо.

Рішенням даної проблеми стала концепція почергових запитів на сервер для того, щоб отримати відповідь на питання «Чи є якість оновлення інформації?». Даний механізм був названийpolling. Загальноприйнятим було побачити клієнтські додатки, які неодноразово опитували сервери, щоб перевірити наявність нових даних. Даний підхід виявився надзвичайно неефективним через великі затрати на запити. При кожному новому запиті необхідно встановлювати нове вхідне з’єднання, HTTP заголовки мають бути розпарсені. Додатково

виділяються ресурси на генерування та доставлення відповіді серверу, яка, у більшості випадків, не буде мати нових даних або оновлень.

Модифікацією стандартного підходу polling є так званий longpolling. При даному підході сервер не закриває з'єднання із клієнтом на період поки не буде нових даних або оновлень на сервері, або ж до моменту досягнення терміну таймауту.Зі сторони клієнту, потрібно виконувати лише один запит на сервер. Коли приходить відповідь, клієнт може зробити ще один запит, при цьому зберігаючи з'єднання до отримання відповіді від серверу.Ще однією технікою для підвищення ефективності даного способу є збереження невеликого проміжку часу між запитами для зменшення навантаження на сервер. Для того, щоб використовувати longpolling сервер має бути налаштований на його використання. Схема роботи підходу longpolling зображена на рис. 4.1.

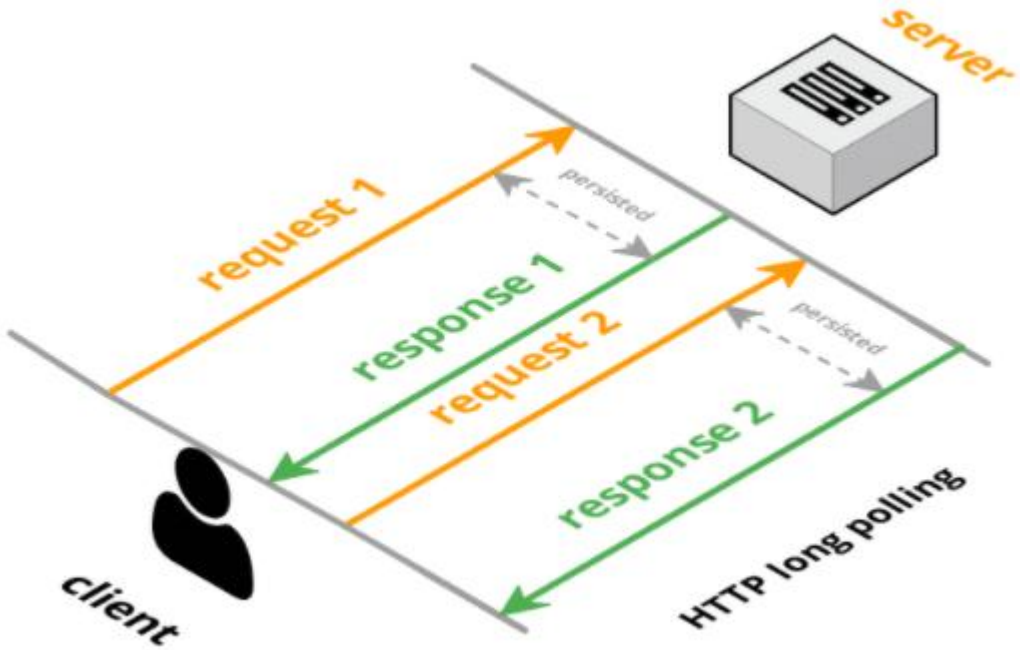

Рисунок 4.1 – Схема роботи підходу longpolling

Сервіс аналізу текстових даних було реалізовано на мові програмування Python. Спілкування між сервісом Python та Telegram ботом відбувається за допомогою REST API.

У якості бази даних для зберігання вхідних коментарів користувачів, які містять спойлери, було обрано базу даних PostgresDB.

4.2 Функціонал інтелектуальної системи

Для виконання автоматизованого визначення спойлерів у оглядах медаконтенту, було реалізовано Telegram бот. При першому вході у чат із ботом, з’являється невеликий опис функціоналу бота. Опис боту наведений на рисунку 4.2.

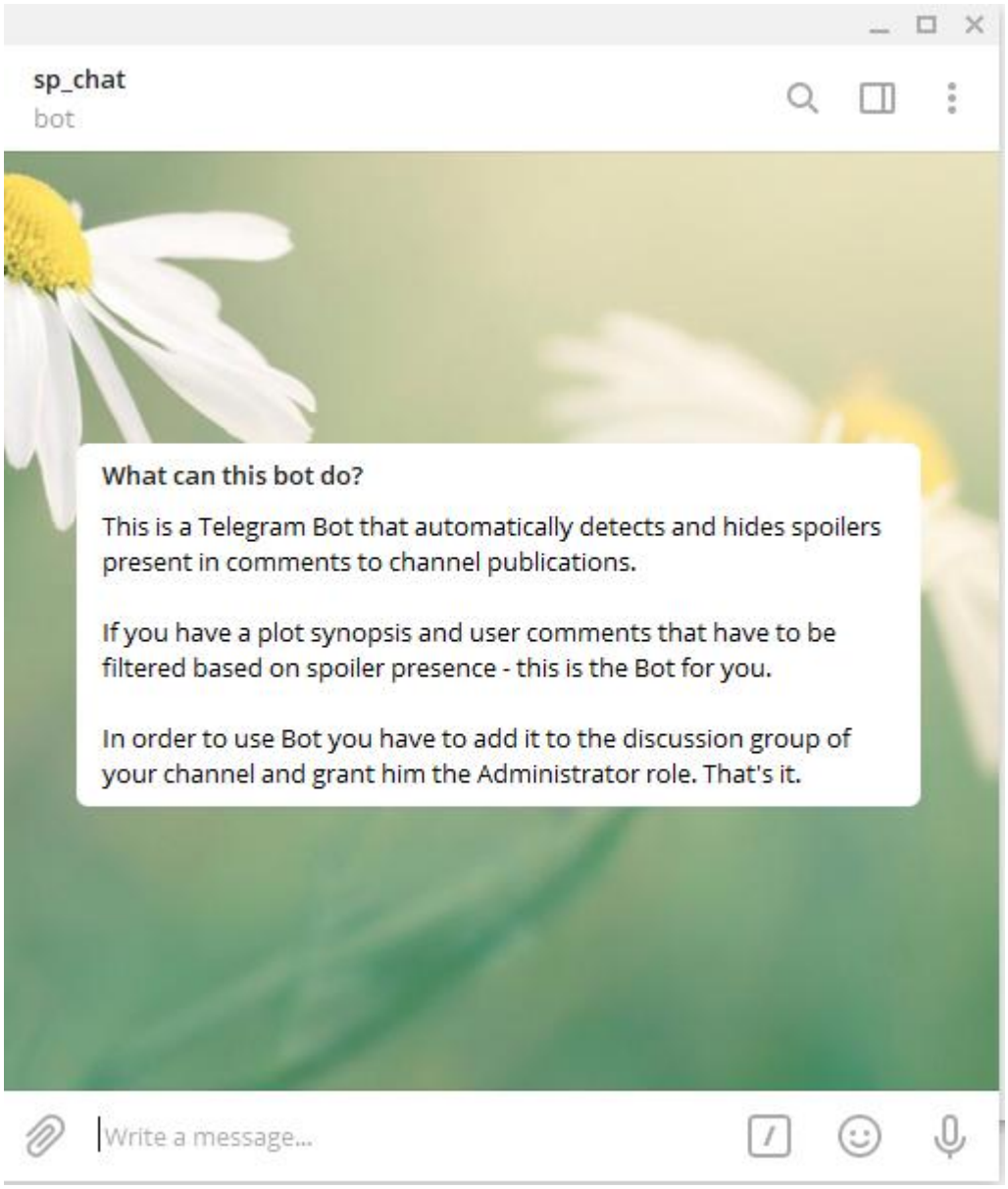

Рисунок 4.2 – Чат із ботом

Для того, щоб використовувати бот, Telegram канал має бути оснащений функцією коментарів та публікацій. Публікації мають містити короткий опис кінострічки, а всі користувачі можуть залишати коментарі під публікаціями.Для того, щоб бот коректно працював, необхідно додати його до чату каналу та надати йому роль «адміністратора». Дана вимога необхідна для того, щоб бот мав можливість додавати, змінювати та видаляти коментарі під публікаціями.

Типова публікація у Telegram каналі зображена на рисунку 4.3.

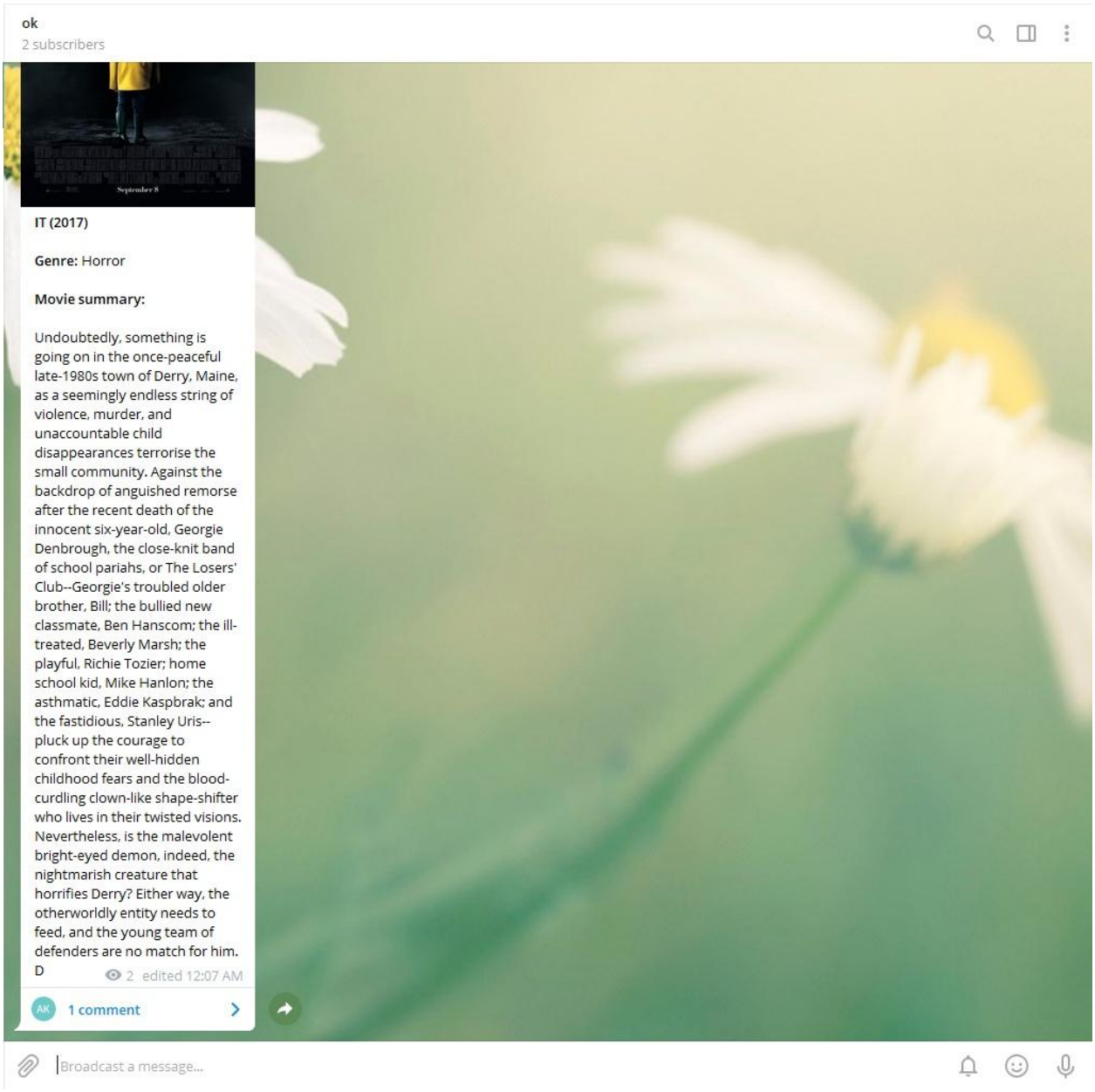

Рисунок 4.3 – Вигляд публікації у каналі

Якщо функція коментування увімкнена у налаштуваннях каналу, користувачі можуть залишати свої коментарі під публікаціями. Вигляд коментарів під публікаціями зображено на рисунку 4.4.

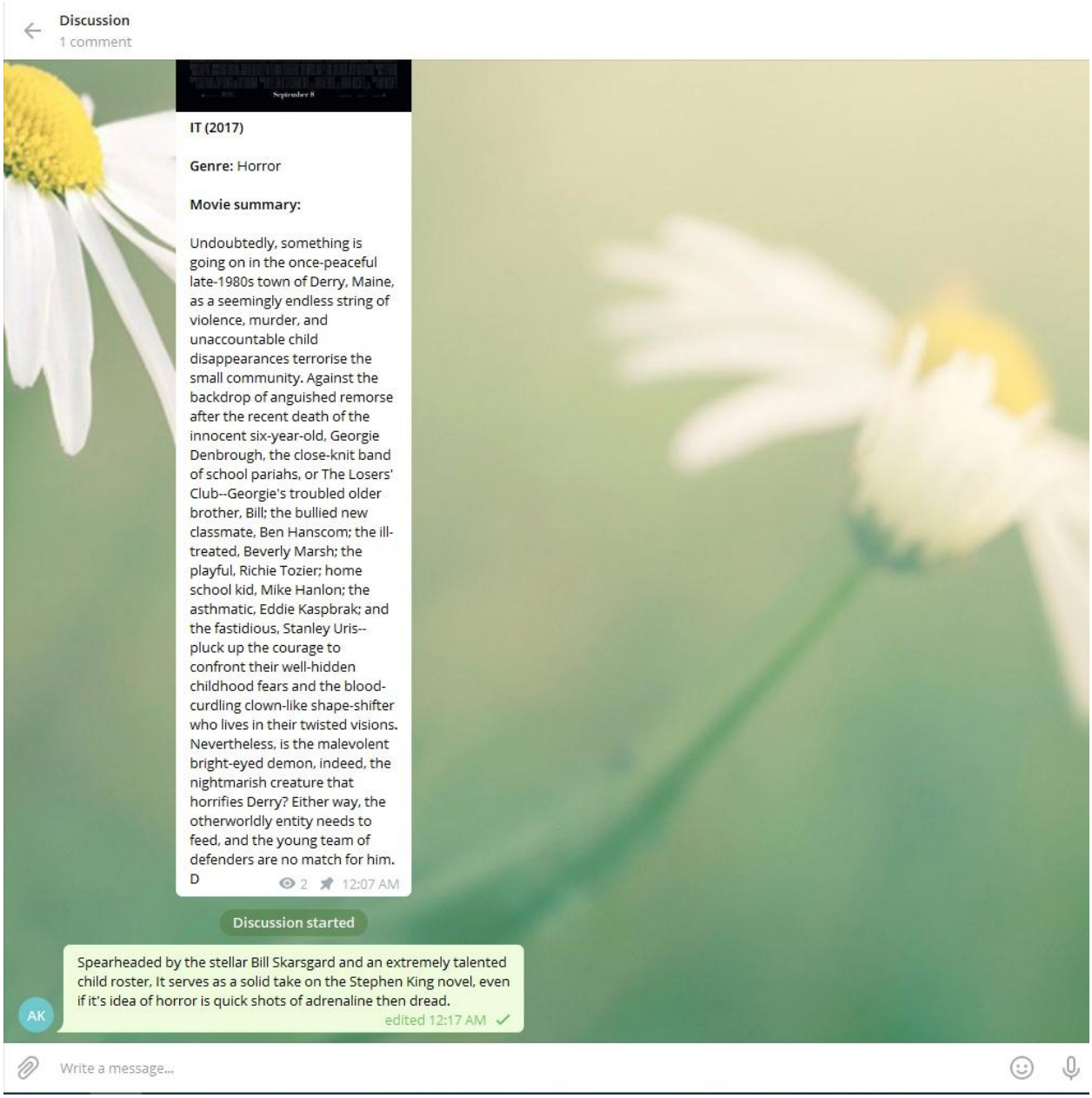

Рисунок 4.4 – Вигляд коментарів під публікаціями у каналі

При введенні коментаря під публікацією робиться аналізу текстового контенту коментаря на предмет наявності спойлеру. Результат залежить від публікації під якою даний коментар було залишено. Якщо результат аналізу тексту на спойлери є негативним, коментар з’являється під публікацією без змін.

Якщо результат аналізу тексту на спойлери є позитивним, під публікацією з’являється коментар, який містить інформацію про його автора, прихований зміст

коментаря та кнопку для отримання вмісту, який є прихованим. Даний результат зображений на рисунку 4.5.

Рисунок 4.5 – Коментар під публікацією із спойлером

При визначенні того, що коментар містить спойлер, на клієнт повертається прихований вміст без оригінального повідомлення. Це гарантує, що оригінальний вміст коментаря не відсилається ботом на клієнт.

Для того, щоб отримати прихований вміст коментаря, необхідно натиснутина кнопку під повідомленням. Даний результат зображений на рисунку 4.6.

Рисунок 4.6 – Отримання вмісту прихованого коментаря

При натисканні на кнопку отримання вмісту прихованого коментаря робиться запит у базу даних для отримання оригінального тексту за унікальним ідентифікатором повідомлення.

4.3 Розгортання програмного забезпечення

Інтелектуальна система складається із даних серверів:

- сервер Telegram боту;
- сервер аналізу текстових даних;
- сервер бази даних.

Сервер Telegram боту має бути встановлений разом із .NetCore 3.1. Сервер аналізу текстових даних має бути встановлений разом із Pyhonверсії 3. Сервер бази даних має бути встановлений разом із СУБД PostgresDB.

Для використання інтелектуальної системи локально, необхідно налаштувати усі сервери, які наведені у абзаці вище, а також створити телеграм бот та отримати токен, який необхідно використати як аргумент класу TelegramBotClient.Далі, необхідно додати бота до чату обговорень каналу та надати йому роль «адміністратора».

Для локального розгортання також можна використати docker-compose.yml файли дляTelegramботу та сервісу аналізу текстових даних.

Узагальнена діаграма розгортання інтелектуальної системи наведена на рисунку 4.7.

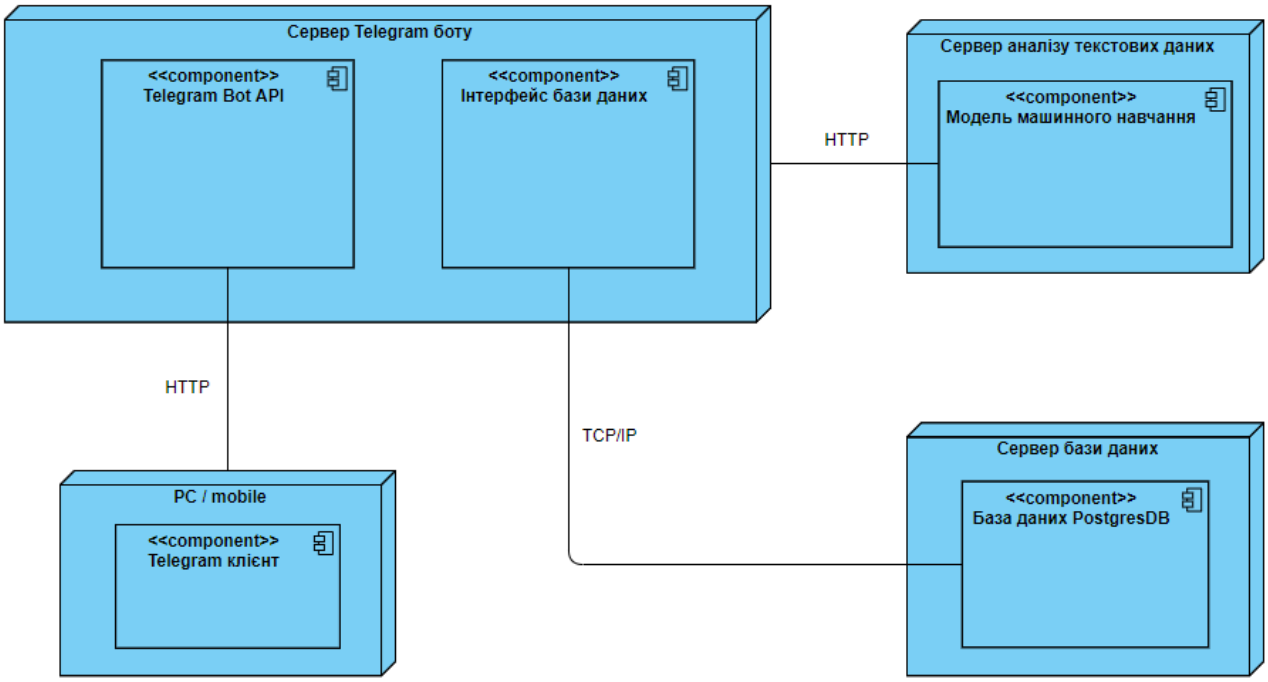

Рисунок 4.7 – Узагальнена діаграма розгортання інтелектуальної системи

Висновок до розділу

У даному розділі було проведено опис програмного забезпечення, яке було реалізовано у рамках інтелектуальної системи. Було описано сервери, які використовуються у системі та наведено способи комунікації між ними. На основі цього було побудовано діаграму розгортання.

Також у даному розділі було проведено опис функціональності системи та наведено основні способи використання разом із екранними формами використання системи.

5 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

5.1 Опис ідеї проекту

У даному розділі буде проведено детальний аналіз стартап проекту із глибоким дослідженням потенційних конкурентів, стану ринку та можливостей виходу на нього.

Стартап проект призначений для автоматизованого визначення спойлерів у оглядах медіаконтенту у платформі месенджеру Telegram.

Суть розробки стартап-проекту полягає у реалізації Telegram боту, який буде автоматично визначати спойлери у коментарях користувачів до публікацій у каналах. Звичайно після визначення спойлеру, система автоматично приховує зміст коментаря та надає можливість перегляду коментаря тим, хто цього захоче.

Цільовими споживачами даного програмного продукту є адміністратори каналів, які бажають мати функцію фільтрації спойлерів у власних чатах каналів, а також люди, які хочуть унеможливити отримання інформації про ключові повороти сюжету кінострічки.

Вигодою при використанні продукту для споживачів є семантичний аналіз текстів на предмет наявності спойлерів, автоматизований процес визначення спойлерів та легкість у конфігуруванні.

Ідея проекту описується у таблиці 5.1.

Таблиця 5.1 – Опис ідеї проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Інтелектуальна система дослідження та аналізу текстів на предмет наявності спойлерів у оглядах медіаконтенту	Семантичні дослідження, аналіз тексту.	За допомогою методів глибинного навчання виконується семантичний аналіз текстових даних на предмет наявності спойлерів. Використання даних методів дає більшу точність визначення.
	Розпізнавання спойлерів у оглядах медіаконтенту.	Люди уникають можливості отримання інформації про ключові повороти або моменти у сюжеті кінострічки, що зазвичай знижує бажання до перегляду.
	Автоматизований процес визначення спойлерів у оглядах.	Процес визначення спойлерів є автоматизованим, що означає відсутність людського фактору та підтримання з боку людини.

Враховуючи популярність месенджеру Telegram та коментарів під публікаціями, дана ідея є доволі актуальною.

Наступним кроком є визначення сильних, слабких та нейтральних сторін проекту у порівнянні із конкурентами.

Після виконання пошуку,прямих конкурентів знайдено не було. Проте існують потенційні конкуренти, які можуть у майбутньому увійти у даний сегмент ринку. Серед них, можна виділити SpoilerProtection 2.0 та HideItBot. У таблиці 5.2 наведено порівняння сильних, слабких та нейтральниххарактеристик ідеї проекту у порівнянні із потенційними конкурентами.

Таблиця 5.2 – Визначення сильних, слабких та нейтральниххарактеристик ідеї проекту.

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Spoiler Protection 2.0	HideIt Bot			
1	Семантичний аналізу тексту	+	-	-	-	-	+
2	Автоматизоване визначення у Telegram	+	-	-	-	-	+
3	Розпізнавання спойлерів у оглядах медіаконтенту	+	+	-	-	+	-

Згідно із таблицею наведеною вище, стартап проект має ряд переваг перед потенційними конкурентами, а саме: реалізація семантичного аналізу тексту на предмет виявлення спойлерів, автоматизоване визначення у платформі Telegram.

5.2 Технологічний аудит ідеї проекту

У даному розділі буде проведено технологічний аудит проекту та зроблено висновок щодо технологічної здійсненності проекту. Технології, які необхідно реалізувати при розробці проекту, наведені у таблиці 5.3.

Таблиця 5.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Семантичний аналіз тексту на предмет наявності спойлерів	Використання методів NLI для навчання нейронної мережі	Розробити	Розроблено автором проекту
2	Автоматизована система у Telegram	Telegram бот	Наявна	Доступна
3	Автоматизована система аналізу тексту	Веб сервер	Наявна	Доступна
Обрана технологія реалізації ідеї проекту: для реалізації Telegram боту було обрано мову програмування C# та інфраструктуру .NetCore 3.1. Для реалізації семантичного аналізу тексту та веб серверу було обрано мову програмування Python та веб фреймворк Flask.				

Як видно із вище наведеної таблиці, існують вже доступні технології для реалізації автоматизованої системи у Telegram та системи для аналізу тексту. Проте семантичний аналізу тексту на предмет наявності спойлерів потребує розроблення.

5.3 Аналіз ринкових можливостей запуску стартап-проекту

У даному розділі буде проведено аналіз ринку та його можливостей при вході проекту. Також буде наведено фактори загроз та можливостей після запуску проекту.Дане дослідження дозволить спланувати напрямок розвитку проекту у разі виникнення перешкод.

Характеристика потенційного ринку та потенційних клієнтів стартап-проекту наведені у таблицях 5.4 та 5.5 відповідно.

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проекту.

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	0
2	Загальний обсяг продаж, грн/ум.од	Немає точної публічної статистики
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Використання методів лінгвістичного аналізу та застарілих методів
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	Невідома

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Автоматизоване визначення спойлерів у оглядах медіаконтенту	Адміністратори каналів, які бажають фільтрувати спойлери у чатах. Люди, які бажають унеможливити отримання інформації про ключові повороти у кінострічках	Загальна ком'ютерна грамотність, ознайомленність із платформою Telegram	Максимальна точність у визначенні спойлерів. Автоматизоване визначення спойлерів. Простота у конфігуруванні.

Так як на ринку відсутні прямі конкуренти стартап-проекту, а ринок динамічно зростає можна зробити висновок, що вихід стартап-проекту на ринок є сприятливим.Характеристика потенційних клієнтів говорить про те, що важливою відмінністю клієнтів є їх ознайомленність із платформою Telegram.Офіційна статистика каже про те, що платформою Telegram в своїй більшості користуються люди віком від 18 до 34 років. Дана статистика фактично забезпечує, що

потенційні клієнти будуть володіти загальною комп’ютерною грамотністю та без проблем будуть використовувати розроблений продукт.Найбільшими вимогами користувачів до продукту є максимально точність у визначенні спойлерів, відсутність необхідності у втручанні та простота конфігурування.При реалізації продукту дані факти були взяті до уваги.

У таблиці 5.6 наведені фактори, які можуть потенційно перешкодити впровадженню проекту на ринку, у порядку зменшення значущості.

Таблиця 5.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	З’являється конкурент у вигляді великої компанії, який виконує аналогічні задачі	Відтік користувачів від власної системи до системи більш відомої компанії	Залучення інвестицій для просування та реклами системи. Пропонування користувачам активну роль у вдосконаленні системи, таким чином вони будуть мати можливість впливати на вектор розвитку системи
2	Зменшення популярності або потреби у платформі Telegram	Платформа Telegram втрачає велику кількість користувачів, які переходять на інші платформи	Через наявність системи для визначення спойлерів, її імплементація у рамках іншої платформи не має витратити багато ресурсів

Продовження таблиці 5.6

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
3	Збільшення ціни на хостинг хмарного провайдера	Збільшення ціни хостингу у хмарного провайдера	Використання власної апаратної інфраструктури або перехід до іншого хмарного провайдера

Як видно із таблиці 5.6, фактори загроз в основному походять від чинників, які не залежать від впровадження проекту на ринку. Однак, дані ситуації є цілком ймовірними. Можливими реаціями компанії на них можуть бути описання переваг проекту, залучення інвестицій або перехід на іншу платформу/власну апаратну інфраструктуру.

У таблиці 5.7 наведені фактори, які можуть потенційно сприяти впровадженню проекту на ринку, у порядку зменшення значущості.

Таблиця 5.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Збільшення популярності або потреби у платформі Telegram	Платформа Telegram отримує велику кількість користувачів, які переходять з інших платформ	Розповсюдження інформації про систему, її унікальність та переваги. Опис переваг системи на прикладі успішної інтеграції з існуючими каналами

Продовження таблиці 5.7

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
2	Збільшення попиту на автоматизовану обробку текстових даних	Збільшується попит на програму, яка автоматично визначає спойлери у коментарях під публікаціями	Опис переваг запропонованої системи автоматизації, наведення прикладу успішної інтеграції з існуючими каналами

Як видно із таблиці 5.7 фактори можливості в основному спираються на факт збільшення попиту на платформу в цілому, а також на збільшення попиту на автоматизоване визначення спойлерів – на даний момент існують системи, які виконують приховування спойлерів вручну користувачем. Основною та найдієвішою реакцією компанії є опис переваг реалізованого рішення та приведення прикладу успішної інтеграції з існуючими каналами.

У таблиці 5.8 наведено риси конкуренції на ринку.

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції - монополія/олігополія/ монополістична/чиста	Конкурентний ринок	Потреба у аналізі напрямку глибинного навчання, дослідження текстових даних та використанні нових методів, які дають переваги у точності у порівнянні зі старими. Потреба у постійному вдосконаленні методів визначення спойлерів для задоволення потреб користувачів
2. За рівнем конкурентної боротьби - локальний/національний/…	Глобальний	
3. За галузевою ознакою - міжгалузева/ внутрішньогалузева	Міжгалузева	
4. Конкуренція за видами товарів: - товарно-родова - товарно-видова - між бажаннями	Товарно-видова	
5.За характером конкурентних переваг - цінова / нецінова	Нецінова	
6. За інтенсивністю - марочна/не марочна	Марочна	

У таблиці 5.9 наведено детальний аналіз умов конкуренції на ринку.

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	Відсутні	Незважаючи на те, що потенційні конкуренти не впливають на ринок на даний момент, вони мають можливість спостерігати за системою на ринку та увійти у ринок, маючи інформаційну базу того, що позитивно вплинуло на кількість користувачів системи. Серед таких конкурентів можна виділити: SpoilerProtection 2.0 Spoilershield HideItBot ispoilerbot	-	Клієнти напряму впливають на наповнення ринку, адже чим точнішим є визначення спойлерів, тим більше користувачів будуть використовувати дану систему	Товари-замінники не мають загроз на стан ринку розробленої системи

Продовження таблиці 5.9

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Висновки:	Немає	Потенційні конкуренти мають вдосконалити власну інфраструктуру для того, щоб увійти у ринок. Наприклад, побудувати автоматизовану систему для визначення спойлерів, або реалізувати методи семантичного аналізу для аналізу тексту	-	Клієнти напряму впливають на ринок, адже вони оцінюють роботу системи та формують загальне відношення до її роботи.	Немає обмежень роботи через товари-замінники

Як видно із таблиці 5.9, можливість успішної роботи стартап-проекту на ринку є майже сто відсотковою, адже відсутні прямі конкуренти. Для того, щоб залишатися на лідируючій позиції на ринку, проект має зважати на зворотній зв'язок клієнтів та вдосконалювати продукт для забезпечення максимальної точності визначення спойлерів у коментарях користувачів.

У таблиці 5.10 наводяться фактори конкурентоспроможності та їх обґрунтування.

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Семантичний аналіз текстових даних на предмет наявності іспойлерів	Більш точне визначення спойлерів у оглядах медіаконтенту, зважаючи на семантичну подібність двох частин тексту
2	Автоматизована система для визначення спойлерів	Потреба користувачів у автоматизованому визначенні спойлерів, що передбачає відсутність потреби у підтриманні
3	Відкритий вихідний код	Можливість кожного користувача удосконалювати розроблену систему та впливати на її розвиток
4	Зручність та легкість у конфігуруванні	Мінімальні кроки для початку роботи програмного забезпечення

Як видно із таблиці 5.10, фактори конкурентоспроможності є досяжними, якщо проект постійно вдосконалюється. Також важливим фактором є вплив користувачів на подальший розвиток стартап-проекту.

У таблиці 5.11 наводиться порівняння сильних та слабких сторін стартап-проекту у порівнянні з конкурентами.

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з розробленим продуктом						
			–3	–2	–1	0	+1	+2	+3
1	Семантичний аналіз текстових даних на предмет наявності спойлерів	20	*						
2	Автоматизована система для визначення спойлерів	20		*					
3	Відкритий сирцевий код	16				*			
4	Зручність та легкість у конфігуруванні	16				*			

Як видно із таблиці 5.11, найбільшими факторами конкурентоспроможності стартап-проекту є застосування методів семантичного аналізу текстових даних для задачі визначення спойлерів, а також автоматизована система. Для збереження лідируючої позиції на ринку необхідно в майбутньому вдосконалювати дані фактори.

У таблиці 5.12 наводиться SWOT- аналіз стартап-проекту.

Таблиця 5.12 – SWOT- аналіз стартап-проекту

Сильні сторони: семантичний аналіз текстових даних на предмет наявності спойлерів, автоматизована система для визначення спойлерів	Слабкі сторони:невелика популярність системи та компанії

Продовження таблиці 5.12

Можливості: збільшення популярності або потреби у платформі Telegram, збільшення попиту на автоматизовану обробку текстових даних	Загрози: збільшення ціни на хостинг хмарного провайдера, з'являється конкурент у вигляді великої компанії, яка виконує аналогічні задачі

У таблиці 5.13 наводиться перелік альтернатив вровадження стартап-проекту, зважаючи на можливі фактори загроз або проекти конкурентів.

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Імплементація системи у рамках іншої платформи	Середня	1-3 місяці
2	Рекламна кампанія та розповсюдження інформації	Достатня	1-4 місяця
3	Розширення функціоналу	Достатня	3-6 місяця

Як видно із таблиці 5.13, найбільш сприятливою альтернативою є імплементація системи у рамках іншої платформи. Проте існують ризики пов'язані із специфікою реалізації системи у невідомій платформі, тому строки реалізації можуть збільшитися. Тому можна вважати, що найкращими альтернативами є розширення функціоналу та розповсюдження інформації про

стартап-проект, а саме про його переваги, унікальність та успішний досвід інтеграції з існуючими каналами.

5.4 Розроблення ринкової стратегії проекту

Перед тим, як описувати ринкову стратегію необхідно визначити цільові групи потенційних споживачів. Вони наведені у таблиці 5.14.

Таблиця 5.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Люди віком 12-17 років	Середня	Низький	Низька	Висока
2	Люди віком 18-25 років	Висока	Високий	Низька	Середня
3	Люди віком 26-39 років	Висока	Високий	Низька	Середня
4	Люди віком старше 40 років	Середня	Середній	Низька	Низька
Які цільові групи обрано: 2, 3					

У якості потенційних споживачів було обрано групи, які свідомо цікавляться кінострічками та мають потребу у фільтрації спойлерів. Також важливим є те, щоб цільова група споживачів була ознайомлена з платформою Telegram.

Базова стратегія розвитку наведена у таблиці 5.15.

Таблиця 5.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Стратегія спеціалізації	Ознайомлення із потребами потенційної групи споживачів, детальний опис конфігурування системи, розповсюдження інформації про систему	Легкість у інтеграції та максимальна точність у визначенні спойлерів	Стратегія диференціації

У якості базової стратегії було обрано стратегію диференціації, яка передбачає підлаштовування під потреби споживачів. Як альтернативу, було обрано стратегію спеціалізації, тобто концентрацію на потребах одного цільового сегменту.

Стратегія конкурентної поведінки на ринку наведена у таблиці 5.16.

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Так	Шукати нових споживачів	Так як проект є «першопрохідцем» основні характеристики товару будуть змінюватися в залежності від потреби споживачів	Оборонна стратегія лідера

У якості стратегії конкурентної поведінки було обрано оборонну стратегію лідера. Вона передбачає захист власної долі на ринку за рахунок розробки та вдосконалення технологічної частини системи. Дана стратегія передбачає менше ресурсів, аніж атакувальна, а саме тому добре підходить для даної системи.

Таблиця 5.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Семантичний аналіз тексту на предмет наявності спойлерів Автоматизована система визначення спойлерів Легкість у конфігуруванні	Стратегія диференціації	Висока точність визначення спойлерів, прямий вплив споживачів на розвиток системи	Точність класифікації Легкість Автоматизація

5.5 Розроблення маркетингової програми стартап-проекту

У даному розділі буде проведено розроблення маркетингової програми проекту. Спочатку необхідно сформувати маркетингову концепцію товару. Вона наведена у таблиці 5.18.

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Семантичний аналіз тексту на предмет наявності спойлерів	Аналіз оглядів медіаконтенту на предмет наявності спойлерів з високою точністю визначення за рахунок використання методів глибинного навчання	Більша точність визначення, аніж з використанням методів лінгвістичного аналізу або застарілих методів
2	Автоматизований процес визначення спойлерів	Відсутність людського фактору та підтримання з боку людини	Автоматизація процесу визначення спойлерів у рамках платформи Telegram
3	Максимальна легкість у конфігуруванні	Конфігурування у декілька кліків	Конфігурування не потребує додаткових дій з боку споживачів

У таблиці 5.19 наведено опис трьох рівнів моделі товару.

Таблиця 5.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Автоматизоване визначення спойлерів у оглядах медіаконтенту.</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Семантичний аналіз тексту
2. Автоматизована система визначення спойлерів</td><td>Нм
Нм</td><td>Тл/Е
Тл/Е</td></tr>
<tr><td colspan="3">Якість: стандарти відсутні. Якість забезпечується тестувальником.</td></tr>
<tr><td colspan="3">Пакування: сервер для аналізу текстових даних та серверна частина Telegram боту.</td></tr>
<tr><td colspan="3">Марка: розумне визначення спойлерів у оглядах медіаконтенту.</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="3">До продажу: -</td></tr>
<tr><td colspan="3">Після продажу: -</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання:користувачі можуть впливати на розвиток та набір функціоналу проекту.</td></tr>
</table>

Далі необхідно описати рівень цін на товари-аналоги та товари-замінники та вирішити, якою буде ціна на розроблюваний продукт. Рівні цін зображені у таблиці 5.20.

Таблиця 5.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	Безкоштовний	Безкоштовний	Будь-який	Ціна на товар може варіюватися від безкоштовної до 20$ щомісячно

З таблиці 5.20 можна побачити, що товари потенційних конкурентів є безкоштовними. Саме тому, при встановленні ціни на розроблюваний продукт необхідно брати межі від безкоштовної до 20$ щомісячної підписки, адже необхідно компенсувати витрати на перебування серверів у хмарного провайдера.

Важливим кроком у розробленні маркетингової програми проекту є визначення оптимальної системи збуту. Вона наведена у таблиці 5.21.

Таблиця 5.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Цільові клієнти: адміністратори каналів,люди, які бажають уникнути спойлерів.Клієнти в основному виконують	Підтримання серверів системи. Коректнефункціонування системи. Збір маркетингової інформації	Канал нульового рівня	Організація збуту власними силами

	закупівлю через Інтернет			

Сформована система збуту показує, що групи цільових клієнтів в основному виконують закупівлю товарів через Інтернет, тому немає необхідності у налагодженні додаткових однорівневих або дворівневих систем збуту. Організація збуту власними силами має зменшити додаткові витрати пов’язані із збутом за допомогою третіх сторін.

У таблиці 5.22 наведено канали, за якими буде відбуватися комунікація з цільовими клієнтами. Так як у якості системи збуту було обрано канал нульового рівня, важливим питанням є вибір каналів за якими буде відбуватися рекламування товару, його продвигання та комунікація із клієнтами загалом.

Таблиця 5.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікаційцільових клієнтів	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Потреба в точній системі визначення спойлерів	Месенджер, електронна пошта, соціальні мережі, форуми	Точність класифікації Легкість у конфігуруванні Автоматизація процесу визначення спойлерів	Донесення до споживачів ідеї автоматизованого визначення спойлерів з високою точністю, відкритість для вдосконалення,	Розумне визначення спойлерів у оглядах медіаконтенту

				легкість у конфігуруванні	

Як видно з таблиці 5.22, комунікація з потенційними клієнтами має відбуватися у месенджерах, через електронну пошту, соціальні мережі тощо. Для більш ефективнішого продвиганняпродукту, рекламне звернення має наголошувати на те, що система виконує розумне визначення спойлерів у оглядах медіаконтенту.

Висновки до розділу

У даному розділі було проаналізовано ринок проектів дослідження текстів на предмет наявності спойлерів у оглядах медіаконтенту на базі платформи Telegram. У результаті дослідження не було виявлено прямих конкурентів, проте були наведені потенційні конкуренти. Дані аналоги мають схожі концепції, проте було визначено переваги розроблюваної системи перед вже існуючими.

Також було проведено технологічну експертизу втілення стартап-проекту. Вона показала, що реалізація системи є актуальною та рентабельною. Проте для того, щоб залишатися на лідируючих позиціях на ринку, необхідно постійно удосконалювати програмний продукт.

ВИСНОВКИ

У даній дисертаційній роботі було реалізовано інтелектуальну систему для автоматизованого визначення спойлерів у оглядах медіаконтенту. По ходу дослідження та реалізації були отримані наступні результати:

– було виконано пошук та проаналізовано існуючі альтернативні рішення системи та зроблено висновок про те, що на ринку відсутні прямі конкуренти. У свою чергу було проведено аналіз та виділено переваги та недоліки перед потенційними конкурентами;

– було проведено аналіз сучасних методів класифікації текстових даних за допомогою нейронних мереж, які були використані для виконання семантичного аналізу тексту. Було реалізовано три методи та вибрано найефективніший на основі метрик матриці помилок;

– було досліджено та реалізовано метод для лінгвістичного аналізу тексту для вирішення проблеми визначення спойлерів. Порівняння показало, що методи семантичного аналізу дають більшу точність у визначенні спойлерів у оглядах медіаконтенту–73% на тестовій вибірці, на противагу 62% для методу лінгвістичного аналізу;

– було запропоновано та реалізовано архітектуру автоматизованого програмного додатку, який виконує аналіз текстових даних;

– було реалізовано програмний додаток у вигляді Telegram боту, який автоматизує процес визначення спойлерів у коментарях публікацій платформи Telegram.

Наукова новизна отриманих результатів полягає в застосуванні методів семантичного аналізу тексту для задачі визначення спойлерів у оглядах медіаконтенту, а також реалізація автоматизованої системи у платформі Telegram.

Задачі, які були описані у постановці завдання були виконані у повному обсязі та було досягнуто мету дисертаційного дослідження. У результаті цього було реалізовано методи для семантичного аналізу тексту, натреновано модель та реалізовано інтелектуальну систему, яка автоматизує процес визначення спойлерів у оглядах медіаконтенту.

СПИСОК ЛІТЕРАТУРИ

1) Johnson&Rosenbaum (Don’t) TellMeHowItEnds: Spoilers, Enjoyment, andInvolvementinTelevisionandFilm// 21:4, 582-612, DOI – 2018.

2) ShengGuo, NarenRamakrishnanFindingthestoryteller: Automaticspoilertaggingusinglinguisticcues // 23rd International Conference on Computational Linguistics – 2010.

3) SunghoJeon, SungchulKim, andHwanjoYuDon’tbespoiledbyyourfriends: Spoilerdetectionin TV programtweets // 7th International AAAI Conference on Weblogs and Social Media – 2013.

4) Jordan L. Boyd-Graber, KimberlyGlasgow, JackieSauterZajacSpoileralert: Machinelearningapproachestodetectsocialmediapostswithrevelatoryinformation // Proceedings of the American Society for Information Science and Technology 50(1) – 2013.

5) YoonKimConvolutionalneuralnetworksforsentenceclassification// EMNLP – 2014.

6) ZichaoYang, DiyiYang, ChrisDyer, XiaodongHe, Alexander J. Smola, Eduard H. HovyHierarchicalattentionnetworksfordocumentclassification // NAACL – 2016.

7) BuruChang, HyunjaeKim, RaehyunKim, DeahanKim, JaewooKang A deepneuralspoilerdetectionmodelusing a genre-awareattentionmechanism // PAKDD – 2018.

8) W Yang, W Jia, W Gao, X Zhou, Y LuoInteractiveVarianceAttentionbasedOnlineSpoilerDetectionforTime-SyncComments // CIKM – 2019.

9) JonasMueller, AdityaThyagarajanSiameseRecurrentArchitecturesforLearningSentenceSimilarity // AAAI-16 – 2016.

10) ZhouhanLin, MinweiFeng, CiceroNogueiradosSantos, MoYu, Bing Xiang, BowenZhou, YoshuaBengio A StructuredSelf-attentiveSentenceEmbedding // ICLR – 2017.

11) QianChen, XiaodanZhu, ZhenhuaLing, SiWei, HuiJiang, DianaInkpenEnhanced LSTM forNaturalLanguageInference // ACL – 2017.

12) Hopper PJ, Thompson SA Transitivityingrammaranddiscourse. Language // Language, 56(2), 251-299 – 1980.

Додаток А

Лістинг коду

```
from flask import Flask, request, jsonify
from md import Md
from mxnet import nd

mdobj = Md()
model = mdobj.train()

app = Flask(__name__)

@app.route('/checkspoiler')
def checksp():
    input = request.get_json()
    tokenized_comment = mdobj.indexer.vocab[
mdobj.tokenizer.tokenizer(input['comment'])]
    tokenized_plot = mdobj.indexer.vocab[mdo
bj.tokenizer.tokenizer(input['plot'])]

    comment_input = nd.array(tokenized_comme
nt, ctx=mdobj.ctx).reshape(1,-1)
    plot_input = nd.array(tokenized_plot, ct
x=mdobj.ctx).reshape(1,-1)
    pred, att = model(comment_input, plot_in
put)

    spoiler_float = nd.argmax(pred, axis=1).
asscalar()

    return "false" if spoiler_float < 0.5 el
se "true"
```

```
import os
import json
import zipfile
import time
import itertools
import numpy as np
import mxnet as mx
import multiprocessing as mp
import gluonnlp as nlp
from mxnet import gluon, nd, init
from mxnet.gluon import nn, rnn
from mxnet import autograd, gluon, nd
import pandas as pd
from sklearn.metrics import accuracy_score,
f1_score
np.random.seed(500)
mx.random.seed(500)
from preprocess import mp_tokenizer, mp_inde
xer
from dataloader import Dataloader
from applylayers import ApplyModelLayers

class Md():
    ctx = None
    indexer = None
    tokenizer = None

    def train(self):
        data = self.get_data()
        data.dropna(inplace=True)

        label2num = {'contradiction':0, 'neu
tral':1, 'entailment':2}
        dataset = [[left, right, label] for
left, right, label in \
                zip(data['sentence1'], data[
'sentence2'], data['label']) if label!='-' ]
        train_dataset, valid_dataset = nlp.d
ata.train_valid_split(dataset, valid_ratio=.
1)
        self.tokenizer = nlp.data.SpacyToken
izer('en')
        length_review = 300
        length_plot = 600

        self.tokenizer = mp_tokenizer(self.t
okenizer, length_review, length_plot)

        train_dataset_token, train_data_leng
ths = self.tokenizer.tokenize_dataset(train_
dataset)
        valid_dataset_token, valid_data_leng
ths = self.tokenizer.tokenize_dataset(valid_
dataset)

        embedding = 'glove.42B.300d'
```

```
        self.indexer = mp_indexer(train_data
set_token, embedding)
        train_dataset = self.indexer.indexat
e_dataset(train_dataset_token)
        valid_dataset = self.indexer.indexat
e_dataset(valid_dataset_token)

        dataloader = Dataloader()
        train_dataloader, valid_dataloader =
 dataloader.get_dataloader(train_dataset, va
lid_dataset, train_data_lengths)

        model = self.build_model(self.indexe
r)

        class_weight = None
        loss_name = 'sce'
        optim_name = 'adam'
        lr = 0.001
        clip = 2.5
        nepochs = 10

        trainer = gluon.Trainer(model.collec
t_params(), optim_name, {'learning_rate': lr
, 'clip_gradient': clip})

        loss = gluon.loss.SoftmaxCrossEntrop
yLoss()

        self.train_and_validate(train_datalo
ader, valid_dataloader, model, loss, \
            trainer, self.ctx, nepochs, clas
s_weight=class_weight, loss_name=loss_name)

        return model

    def get_data(self):
        data_folder = './'
        file_name = 'train.csv'
        file_path = data_folder + file_name
        data = pd.read_csv(file_path)

        return data

    def build_model(self, indexer):
        vocab_len = len(indexer.vocab)
        emsize = 300
        nhidden = 300
        nlayers = 2
        nfc = 256
        nclass = 2
        drop_prob = 0.5

        self.ctx = mx.cpu()

        model = ApplyModelLayers(vocab_len,
emsize, nhidden, nlayers, nfc, nclass, drop_
prob)
        model.initialize(init=init.Xavier(),
 ctx=self.ctx)
        model.embedding_layer.weight.set_dat
a(indexer.vocab.embedding.idx_to_vec)
        model.embedding_layer.collect_params
().setattr('grad_req', 'null')

        return model

    def calculate_loss(self, x_left, x_right
, y, model, loss, class_weight):
        pred, att = model(x_left, x_right)
        y = nd.array(y.astype('int32', copy=
False), ctx=self.ctx)
        l = loss(pred, y)
        return pred, l

    def iterate_epoch(self, data_iter, model
, loss, trainer, ctx, is_train, epoch, class
_weight=None, loss_name='sce'):
        loss_val = 0.
        total_pred = []
        total_true = []
        n_batch = 0

        for batch_x_left, batch_x_right, bat
ch_y in data_iter:
            batch_x_left = batch_x_left.as_i
n_context(ctx)
            batch_x_right = batch_x_right.as
_in_context(ctx)
```

```
            batch_y = batch_y.as_in_context(
ctx)

            if is_train:
                with autograd.record():
                    batch_pred, l = self.cal
culate_loss(batch_x_left, batch_x_right, \

    batch_y, model, loss, class_weight)

                l.backward()
                trainer.step(batch_x_left.sh
ape[0])

            else:
                batch_pred, l = self.calcula
te_loss(batch_x_left, batch_x_right, \

batch_y, model, loss, class_weight)

            batch_pred = nd.argmax(nd.softma
x(batch_pred, axis=1), axis=1).asnumpy()
            batch_true = np.reshape(batch_y.
asnumpy(), (-1, ))
            total_pred.extend(batch_pred.tol
ist())
            total_true.extend(batch_true.tol
ist())

            batch_loss = l.mean().asscalar()

            n_batch += 1
            loss_val += batch_loss

            if is_train and n_batch % 400 ==
 0:
                print('epoch %d, batch %d, b
atch_train_loss %.4f, batch_train_acc %.3f'
%
                    (epoch, n_batch, batch_l
oss, accuracy_score(batch_true, batch_pred))
)

        F1 = f1_score(np.array(total_true),
np.array(total_pred), average='binary')
        acc = accuracy_score(np.array(total_
true), np.array(total_pred))
        loss_val /= n_batch

        if is_train:
            print('epoch %d, learning_rate %
.5f \n\t train_loss %.4f, acc_train %.3f, F1
_train %.3f, ' %
                (epoch, trainer.learning_rat
e, loss_val, acc, F1))
            if epoch % 2 == 0:
                trainer.set_learning_rate(tr
ainer.learning_rate * 0.9)
        else:
            print('\t valid_loss %.4f, acc_v
alid %.3f, F1_valid %.3f, ' % (loss_val, acc
, F1))

    def train_and_validate(self, data_iter_t
rain, data_iter_valid, model, loss, trainer,
 \
                ctx, nepochs, class_weight=N
one, loss_name='sce'):
        for epoch in range(1, nepochs+1):
            start = time.time()
            is_train = True
            self.iterate_epoch(data_iter_tra
in, model, loss, trainer, ctx, is_train, epo
ch, class_weight, loss_name)

            is_train = False
            self.iterate_epoch(data_iter_val
id, model, loss, trainer, ctx, is_train, epo
ch, class_weight, loss_name)
            end = time.time()
            print('time %.2f sec' % (end-
start))
            print("*"*100)


from mxnet import gluon
import gluonnlp as nlp
import time
import multiprocessing as mp
import itertools
```

```
class mp_tokenizer:
    def __init__(self, tokenizer, clip_lengt
h1, clip_length2):
        self.tokenizer = tokenizer
        self.clipper1 = nlp.data.ClipSequenc
e(clip_length1)
        self.clipper2 = nlp.data.ClipSequenc
e(clip_length2)

    def clip_words(self, data):
        left, right, label = data[0], data[1
], int(data[2])
        left, right = self.clipper1(self.tok
enizer(left.lower())), self.clipper2(self.to
kenizer(right.lower()))
        return left, right, label

    def get_length(self, data):
        return float(len(data[0]))

    def tokenize_dataset(self, data):
        start = time.time()
        with mp.Pool() as pool:
            data = gluon.data.ArrayDataset(p
ool.map(self.clip_words, data))
            lengths = gluon.data.ArrayDatase
t(pool.map(self.get_length, data))
        end = time.time()
        return data, lengths

    def __call__(self, data):
        return self.tokenizer(data)

class mp_indexer:
    def __init__(self, dataset_token, embedd
ing):
        self.dataset_token = dataset_token
        self.seqs = [ sample[0] + sample[1]
for sample in dataset_token]
        self.counter = nlp.data.count_tokens
(list(itertools.chain.from_iterable(self.seq
s)))
        self.vocab = nlp.Vocab(self.counter,
 max_size=40000)
        self.vocab.set_embedding(nlp.embeddi
ng.GloVe(source=embedding))

    def token_to_index(self, data):
        return self.vocab[data[0]], self.voc
ab[data[1]], data[2]

    def indexate_dataset(self, data):
        with mp.Pool() as pool:
            data_new = pool.map(self.token_t
o_index, data)
        return data_new

import gluonnlp as nlp
from mxnet import gluon, nd, init

class Dataloader():
    def get_dataloader(self, train_dataset,
valid_dataset, train_data_lengths):
        batch_size = 32
        bucket_num = 10
        bucket_ratio = 0.5
        batchify_fn = nlp.data.batchify.Tupl
e(nlp.data.batchify.Pad(axis=0), \

nlp.data.batchify.Pad(axis=0),

nlp.data.batchify.Stack())

        batch_sampler = nlp.data.sampler.Fix
edBucketSampler(
            train_data_lengths,
            batch_size=batch_size,
            num_buckets=bucket_num,
            ratio=bucket_ratio,
            shuffle=True)
        print(batch_sampler.stats())

        train_dataloader = gluon.data.DataLo
ader(
            dataset=train_dataset,
            batch_sampler=batch_sampler,
            batchify_fn=batchify_fn)
```

```
        valid_dataloader = gluon.data.DataLo
ader(
            dataset=valid_dataset,
            batch_size=batch_size,
            shuffle=False,
            batchify_fn=batchify_fn)
        return train_dataloader, valid_datal
oader

from mxnet import autograd, gluon, nd
from mxnet.gluon import nn

class Attention(nn.Block):
    def __init__(self, **kwargs):
        super(Attention, self).__init__(**kw
args)

    def forward(self, inp_left, inp_right):
        att = nd.batch_dot(inp_left, nd.tran
spose(inp_right, axes = (0, 2, 1)))
        inp_left_dot = nd.batch_dot(nd.softm
ax(att, axis=-1), inp_right)
        inp_right_dot = nd.batch_dot(nd.soft
max(nd.transpose(att, axes=(0, 2, 1)), axis=
-1), inp_left)
        aug_left = nd.concat(inp_left, inp_l
eft_dot, inp_left-
inp_left_dot, inp_left*inp_left_dot, dim=-1)
        aug_right = nd.concat(inp_right, inp
_right_dot, inp_right-
inp_right_dot, inp_right*inp_right_dot, dim=
-1)
        return aug_left, aug_right, att

from mxnet import autograd, gluon, nd
from mxnet.gluon import nn, rnn
from attention import Attention

class ApplyModelLayers(nn.Block):
    def __init__(self, vocab_len, embsize, n
hidden, nlayers, nfc, nclass, drop_prob, **k
wargs):
        super(ApplyModelLayers, self).__init
__(**kwargs)
        with self.name_scope():
            self.drop_prob = drop_prob
            self.embedding_layer = nn.Embedd
ing(vocab_len, embsize)
            self.bilstm1 = rnn.LSTM(nhidden,
 num_layers=nlayers, dropout=drop_prob, bidi
rectional=True)
            self.bilstm2 = rnn.LSTM(nhidden,
 num_layers=1, dropout=drop_prob, bidirectio
nal=True)
            self.align_att = Attention()
            self.output_layer = nn.HybridSeq
uential()
            self.output_layer.add(nn.Dense(n
fc, activation='tanh'), nn.Dropout(rate=drop
_prob), \
                                  nn.Dense(n
fc, activation='tanh'), nn.Dropout(rate=drop
_prob), \
                                  nn.Dense(n
class))

    def forward(self, inp_left, inp_right):
        inp_embed_left = self.embedding_laye
r(inp_left)
        inp_embed_right = self.embedding_lay
er(inp_right)
        h_output_left = self.bilstm1(nd.tran
spose(inp_embed_left, axes=(1, 0, 2)))
        h_output_right = self.bilstm1(nd.tra
nspose(inp_embed_right, axes=(1, 0, 2)))
        m_left, m_right, att = self.align_at
t(nd.transpose(h_output_left, axes=(1, 0, 2)
), \

          nd.transpose(h_output_right, axes=
(1, 0, 2)))
        v_left = self.bilstm2(nd.transpose(m
_left, axes=(1, 0, 2)))
        v_right = self.bilstm2(nd.transpose(
m_right, axes=(1, 0, 2)))
        v_left = nd.transpose(v_left, axes=(
1, 0, 2))
        v_right = nd.transpose(v_right, axes
=(1, 0, 2))
```

```
        v_left_avg = nd.sum(v_left, axis=1)
/ v_left.shape[1]
        v_right_avg = nd.sum(v_right, axis=1
) / v_right.shape[1]
        v_left_max = nd.max(v_left, axis=1)
        v_right_max = nd.max(v_right, axis=1
)
        dense_input = nd.concat(v_left_avg,
v_left_max, v_right_avg, v_left_max, dim=-1)
        output = self.output_layer(dense_inp
ut)
        return output, att

import pandas as pd
import numpy as np
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import
LabelEncoder
from collections import defaultdict
from nltk.corpus import wordnet as wn
from sklearn.feature_extraction.text import
TfidfVectorizer
from sklearn import model_selectionsvm
from sklearn.metrics import accuracy_score

import nltk
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')

np.random.seed(500)
Corpus_json =
pd.read_json('./data/reviews.json',
lines=True).\
drop_duplicates('review_text').sample(frac=1
)
Corpus_json.rename(columns =
{'review_text':'text',
'is_spoiler':'label'}, inplace = True)
Corpus_json =
Corpus_json.drop(['review_date', 'movie_id',
'user_id', 'rating', 'review_summary'], axis
= 1)
Corpus_json.to_csv('./data/corpus_own2.csv',
index=False)
Corpus =
pd.read_csv("./data/corpus_own2.csv",encodin
g='latin-1')
print(Corpus)

Corpus['text'].dropna(inplace=True)
Corpus['text'] = [entry.lower() for entry in
Corpus['text']]
Corpus['text']= [word_tokenize(entry) for
entry in Corpus['text']]
tag_map = defaultdict(lambda : wn.NOUN)
tag_map['J'] = wn.ADJ
tag_map['V'] = wn.VERB
tag_map['R'] = wn.ADV
for index,entry in
enumerate(Corpus['text']):
Final_words = []
word_Lemmatized = WordNetLemmatizer()
    for word, tag in pos_tag(entry):
        if word not in
stopwords.words('english') and
word.isalpha():
word_Final =
word_Lemmatized.lemmatize(word,tag_map[tag[0
]])
Final_words.append(word_Final)
Corpus.loc[index,'text_final'] =
str(Final_words)
```

```
Train_X, Test_X, Train_Y, Test_Y =
model_selection.train_test_split(Corpus['tex
t_final'],Corpus['label'],test_size=0.3)

Encoder = LabelEncoder()
Train_Y = Encoder.fit_transform(Train_Y)
Test_Y = Encoder.fit_transform(Test_Y)

tfidf_vect =
TfidfVectorizer(analyzer='word',
token_pattern=r'\w{1,}', max_features=5000)
tfidf_vect.fit(Corpus['text_final'])
xtrain_tfidf = tfidf_vect.transform(Train_X)
xtest_tfidf = tfidf_vect.transform(Test_X)

tfidf_vect_ngram =
TfidfVectorizer(analyzer='word',
token_pattern=r'\w{1,}', ngram_range=(2,3),
max_features=5000)
tfidf_vect_ngram.fit(Corpus['text_final'])
xtrain_tfidf_ngram =
tfidf_vect_ngram.transform(Train_X)
xtest_tfidf_ngram =
tfidf_vect_ngram.transform(Test_X)

tfidf_vect_ngram_chars =
TfidfVectorizer(analyzer='char',
token_pattern=r'\w{1,}', ngram_range=(2,3),
max_features=5000)
tfidf_vect_ngram_chars.fit(Corpus['text_fina
l'])
xtrain_tfidf_ngram_chars =
tfidf_vect_ngram_chars.transform(Train_X)
xtest_tfidf_ngram_chars =
tfidf_vect_ngram_chars.transform(Test_X)

SVM = svm.SVC(C=1.0, kernel='linear',
degree=3, gamma='auto')
SVM.fit(xtrain_tfidf,Train_Y)
predictions_SVM = SVM.predict(xtest_tfidf)
print("SVM Accuracy Score words level ->
",accuracy_score(predictions_SVM,
Test_Y)*100)

SVM = svm.SVC(C=1.0, kernel='linear',
degree=3, gamma='auto')
SVM.fit(xtrain_tfidf_ngram,Train_Y)
predictions_SVM =
SVM.predict(xtest_tfidf_ngram)
print("SVM Accuracy Score ngrams level ->
",accuracy_score(predictions_SVM,
Test_Y)*100)

SVM = svm.SVC(C=1.0, kernel='linear',
degree=3, gamma='auto')
SVM.fit(xtrain_tfidf_ngram_chars,Train_Y)
predictions_SVM =
SVM.predict(xtest_tfidf_ngram_chars)
print("SVM Accuracy Score char ngrams level
-> ",accuracy_score(predictions_SVM,
Test_Y)*100)


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using System.Threading;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using Newtonsoft.Json;
using SpoilerTelegramBot.Models;
using Telegram.Bot;
using Telegram.Bot.Exceptions;
using Telegram.Bot.Extensions.Polling;
using Telegram.Bot.Types;
using Telegram.Bot.Types.Enums;
using Telegram.Bot.Types.ReplyMarkups;

namespace SpoilerTelegramBot
{
       static class Program
       {
              private static
SpoilerCheckServiceSpoilerCheckService;
```

```
        private static
TelegramBotClientBot;
        private static Regex regex =
new Regex(@"[^\s\\]");
        private const int
MaxAlertCharactersCount = 200;

        private static
readonlyAutoResetEvent _waitHandle = new
AutoResetEvent(false);

        static async Task
Main(string[] args)
        {
                using (var dbContext =
new SpoilerContext())
                {
                        await
dbContext.Database.MigrateAsync();
                }

                SpoilerCheckService =
new SpoilerCheckService();

                Bot = new
TelegramBotClient("token");

                var me = await
Bot.GetMeAsync();

                var cts = new
CancellationTokenSource(TimeSpan.FromDays(2)
);

                Bot.StartReceiving(
                        new
DefaultUpdateHandler(HandleUpdateAsync,
HandleErrorAsync),
                        cts.Token
                );


        Console.WriteLine($"Started listening
for {me.Username}");
                Console.ReadLine();

                Console.CancelKeyPress
+= (o, e) =>
                {

        Console.WriteLine("Exit");

        _waitHandle.Set();
                };
                _waitHandle.WaitOne();

                cts.Cancel();
        }

        public static async Task
HandleUpdateAsync(ITelegramBotClientbotClien
t, Update update,

        CancellationTokencancellationToken)
        {
                var handler =
update.Type switch
                {

        UpdateType.Message
=>BotOnMessageReceived(update.Message),

        UpdateType.CallbackQuery
=>BotOnCallbackQueryReceived(update.Callback
Query),

        UpdateType.EditedMessage
=>BotOnMessageReceived(update.EditedMessage)
,
                        _
=>Task.CompletedTask
                };

                try
                {
                        await handler;
                }
                catch (Exception ex)
                {
                        throw ex;
                }
        }

        private static async Task
BotOnMessageReceived(Message message)
        {

        if(message.ReplyToMessage == null ||
message.ReplyToMessage.Text == @"\start")
                        return;

                var plotCaption =
message.ReplyToMessage.CaptionEntities.Last(
);

                var isSpoiler = await
SpoilerCheckService.SpoilerCheckClient.Check
Spoiler(new SpoilerCheck
                {
                        comment =
message.Text,
                        plot =
message.ReplyToMessage.Caption.Substring(plo
tCaption.Offset + plotCaption.Length)
                });
```

```
                    if (!isSpoiler)
                        return;

                    try
                    {
                        await
Bot.DeleteMessageAsync(
        message.Chat.Id,
        message.MessageId
                        );
                    }
                    catch
(ApiRequestException e)
                    {
                        return;
                    }


        Console.WriteLine($"Receive message
type: {message.Type}");

                    var partsCount =
(int)Math.Ceiling((decimal)message.Text.Leng
th / 200);

                    var inlineKeyboard =
new InlineKeyboardMarkup(new[]
                    {

        GetButtons(GetSpoilerParts(message,
partsCount))
                    });

                    using (var dbContext =
new SpoilerContext())
                    {
                        var
spoilerComments =
GetSpoilerComments(message, partsCount);
                        await
dbContext.SpoilerComments.AddRangeAsync(spoi
lerComments);
                        await
dbContext.SaveChangesAsync();
                    }

                    var hiddenMessage =
regex.Replace(message.Text, "\u2588");

                    await
Bot.SendTextMessageAsync(
                        message.Chat.Id,
                        $@"Comment from
*[{message.From.FirstName}
{message.From.LastName}](tg://user?id={messa
ge.From.Id})*\!\!\! {hiddenMessage}",

        replyToMessageId:
message.ReplyToMessage.MessageId,
                        replyMarkup:
inlineKeyboard,
                        parseMode:
ParseMode.MarkdownV2
                    );
            }

            private static
List<SpoilerCallback>GetSpoilerParts(Message
message, int partsCount)
            {

        List<SpoilerCallback>spoilerCallbacks
= new List<SpoilerCallback>(partsCount);
                    for (int i = 0;
i<partsCount; i++)
                    {

        spoilerCallbacks.Add(new
SpoilerCallback
                        {
                            ChatId =
message.Chat.Id.ToString(),
                            MessageId
= message.MessageId,

        MessagePart = i + 1
                        });
                    }

                    return
spoilerCallbacks;
            }

            private static
List<SpoilerComment>GetSpoilerComments(Messa
ge message, int partsCount)
            {

        List<SpoilerComment>spoilerComments =
new List<SpoilerComment>(partsCount);
                    for (int i = 1; i<=
partsCount; i++)
                    {
                        var startIndex =
i * MaxAlertCharactersCount -
MaxAlertCharactersCount;
                        var length =
message.Text.Length - startIndex + 1
<MaxAlertCharactersCount
                            ?
message.Text.Length - startIndex
```

```
                                    :
MaxAlertCharactersCount;

       spoilerComments.Add(new
SpoilerComment
                            {
                                    ChatId =
message.Chat.Id.ToString(),
                                    MessageId
= message.MessageId,

       MessagePart = i,
                                    Message =
message.Text.Substring(startIndex, length)
                            });
                     }

                     return spoilerComments;
              }

              private static
List<InlineKeyboardButton>GetButtons(List<Sp
oilerCallback>spoilerCallbacks)
              {

       List<InlineKeyboardButton>inlineKeybo
ardButtons =
                            new
List<InlineKeyboardButton>(spoilerCallbacks.
Capacity);
                     foreach (var
spoilerCallback in spoilerCallbacks)
                     {

       inlineKeyboardButtons.Add(InlineKeybo
ardButton.WithCallbackData(
                                    $"See
Spoiler [pt.{spoilerCallback.MessagePart}]",
JsonConvert.SerializeObject(spoilerCallback)
));
                     }

                     return
inlineKeyboardButtons;
              }

              private static async Task
BotOnCallbackQueryReceived(CallbackQuerycall
backQuery)
              {
                     var spoilerCallbackData
=
JsonConvert.DeserializeObject<SpoilerCallbac
k>(callbackQuery.Data);


       SpoilerCommentspoilerComment;
                     using (var dbContext =
new SpoilerContext())
                     {
                            spoilerComment =
dbContext.SpoilerComments.Find(spoilerCallba
ckData.ChatId,

       spoilerCallbackData.MessageId,
spoilerCallbackData.MessagePart);
                     }

                     string originalComment;
                     originalComment =
spoilerComment?.Message ?? "sorry, can't
find original message";

                     try
                     {
                            await
Bot.AnswerCallbackQueryAsync(

       callbackQuery.Id,

       originalComment,

       showAlert: true
                            );
                     }
                     catch(Exception e)
                     {

                     }
              }

              public static async Task
HandleErrorAsync(ITelegramBotClientbotClient
, Exception exception,
CancellationTokencancellationToken)
              {
                     var ErrorMessage =
exception switch
                     {

       ApiRequestExceptionapiRequestExceptio
n => $"Telegram API
Error:\n[{apiRequestException.ErrorCode}]\n{
apiRequestException.Message}",
                                    _
=>exception.ToString()
                     };


       Console.WriteLine(ErrorMessage);
                     }
       }
}
```