МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Інтелектуальний чат-бот для фільтрації контенту телеграм-каналу компанії»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ.м-01 Волін Вадим Володимирович

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «___» грудня 2021 р.

Науковий керівник _____ к.т.н., доц., Ващенко С. М.
(підпис)

Голова комісії _____ Шифрін Д.М.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2021

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. кафедри ІТ

_____ В. В. Шендрик
«___» _____ 2021 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Волін Вадим Володимирович
(прізвище, ім'я, по батькові)

1 Тема проекту *Інтелектуальний чат-бот для фільтрації контенту телеграм-каналу компанії*

затверджена наказом по університету від «29» 10 2021 р. № 0787-IV

2 Термін здачі студентом закінченого проекту «10» грудня 2021 р.

3 Вхідні дані до проекту запит на створення даного проекту

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз предметної області, постановка задачі та методи дослідження, проектування чат-боту, практична реалізація чат-боту

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність роботи, мета та задачі, аналіз аналогів, порівняння додатків-аналогів, функціональні вимоги до чат-боту, засоби реалізації, структурно-функціональне моделювання роботи додатку, діаграма ВВ, проектування моделі мережі, аналіз алгоритмів, практична реалізація, висновки

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Аналіз предметної області	Ващенко С.М.		
Постановка задачі та методи дослідження	Ващенко С.М.		
Проектування чат-боту	Ващенко С.М.		
Практична реалізація чат-боту	Ващенко С.М.		

Дата видачі завдання 10.12.2020.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Ініціалізація	26.08.21-03.09.21	
2	Проектування	06.09.21-13.10.21	
3	Реалізація	14.10.21-30.11.21	
4	Завершення	01.12.21-13.12.21	

Магістрант _____ Волін В.В.

Керівник роботи _____ к.т.н., доц. Ващенко С.М.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Інтелектуальний чат-бот для фільтрації контенту телеграм-каналу компанії».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 36 найменуваньта двох додатків. Загальний обсяг роботи − 72 сторінок, у тому числі 47 сторінок основного тексту, 4 сторінки списку використаних джерел, 16 сторінок додатків.

Кваліфікаційну роботу магістра присвячено розробці чат-боту для фільтрації телеграм-чату компанії.

У першій частині аналізується предметна область. Працюючи над цим розділом, ми переглянули дослідження та публікації в цій галузі, проаналізували чат-боти аналоги, їх особливості, сформулювали проблему, пояснили цілі та завдання дослідження та вибрали засоби реалізації.

Друга частина описує дизайн чат-бота. Під час роботи в цьому розділі було виконано структурно-функціональне моделювання чат-бота, вибір архітектури чат-бота, показ IDEF-схеми основного процесу, розбирання цього процесу та побудова варіанти використання чат-бота.

У третій частині чат-бот був активований на практиці. В ході роботи було описано структуру програмного модуля, описаний прототип чат-бота, який відображав функціональність чат-бота, а також навчальні моделі, активацію програмного забезпечення та використання чат-бота.

Результатом проведеної роботи є розроблений інтелектуальний чат-бот для автоматизації процесу модерації недоцільного контенту.

Практичне значення роботи полягає у автоматизації процесу модерації недоцільного контенту телеграм-чату компанії та дослідження доцільності використання машинного навчання при роботі з ботами.

Ключові слова: чат-бот, повідомлення, машинне навчання, алгоритм, штучний інтелект, середовище розробки, зображення, звіт.

ЗМІСТ

ВСТУП

Кожна компанія рано чи пізно стикається з пошуком онлайн-сервісу, який би об'єднав команду. Великий бізнес обирає власні розробки, які потім обслуговує відділ програмістів. А якщо бюджет обмежений, то найчастіше спілкуються у звичних месенджерах. Але це небезпечно і не завжди практично. Тому компанії вводять додаткові системи регулювання у чатах, де можна вести все робоче листування [4, 20].

Корпоративний чат допомагає спілкуватися співробітникам усередині компанії, а також підтримує зв'язок із клієнтами, підрядниками та партнерами. Тут можна розмовляти з колегами, призначати зустрічі, обговорювати хід робіт та обмінюватися файлами, не переходячи в сторонні месенджери або електронну пошту. Це безкоштовно та доступно. Можна сказати, що чат має як переваги, так і недоліки, але справедливіше висловитися по-іншому: чат має властивості, які можуть ставати і перевагами, і недоліками. Якщо користуватися властивостями чату нерозумно, то до кінця робочого дня можна сильно втомитися від нескінченного потоку повідомлень [1, 4, 20].

При огляді роботи корпоративних чатів було виявлено, що 2/3 чатів регулюються вручну, за допомогою адміністраторів та ін. Кожен чат при недоцільному використанні стає джерелом проблем, бо його використовують у цілях, для яких він не підходить. Тоді він із місця спілкування перетворюється на джерело інформаційного шуму та цвинтар ідей. При невеликих розмірах чату ручне регулювання на викликає великого навантаження на людей, але недоцільний контент може нанести шкоду для користувачів, якщо буде невчасно промодерований. Окрім того, у карантинних умовах, чати мають підвищене навантаження у вигляді великої кількості повідомлень при віддаленій роботі. Вирішенням такої проблеми є чат-бот, який буде слугувати інструментом для фільтрації повідомлень користувачів [4, 5, 11, 12, 14].

Об’єктом досліджень є застосування методів аналізу медіа та текстових повідомлень за допомогою машинного навчання та інтелектуальних чат-ботів [2, 3, 4].

Предмет досліджень – технології аналізу зображень та текстових даних з використанням машинного навчання та чат-ботів [7, 8, 9].

В результаті основною метою даної роботи було обрано створення інтелектуального чат-боту, який буде виконувати роль автоматизованого модератора чату [4].

Для створення основного ядра проекту було обрано такі ключові задачі:

— Створення послідовного плану розробки чат-боту [4, 18];

— Аналіз та створення контекстної діаграми та діаграми декомпозиції головного процесу;

— створенння діаграми варіантів використання чат-боту;

— детальне планування проекту, структури та діаграм WBS/OBS[17, 18, 19];

— створення календарного плану з використанням діаграми Ганта [17];

— проведення аналізу основних ризиків [17].

Розроблений інтелектуальний чат-бот полегшить процес фільтрування повідомлень та модерацію чату, завдяки автоматизації на основі штучного інтелекту та машинного навчання [10, 13, 15].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Огляд останніх досліджень і публікацій

Предметна область - набір сервісів та інструментів. Створення яких має бути передбачене у рамках поточного проекту.

З початку 2016 року світ поринув у справжні перегони в області розробки нейронних мереж, свої алгоритми у даній області демонстрували такі гіганти індустрії, як Google (мережа-гравець у AlphaGo), Microsoft (ряд сервісів для ідентифікації зображень), стартапи MSQRD, Prisma, Meta(колишній Facebook) та інші [2].

Нейронні мережі є одним із основних компонентів розвитку штучного інтелекту. Ідея полягає в тому, щоб максимально імітувати дії людини, тобто її здатність вчитися та виправляти помилки. Це найважливіша особливість будь-якої мережі - вона може бути незалежною і працювати відповідно до попереднього досвіду, кожного разу роблячи менше помилок[9, 10].

Нейромережа імітує як діяльність, так і структуру нервової системи людини. Така мережа складається з великої кількості окремих обчислювальних елементів («нейронів»). Найчастіше кожен «нейрон» належить певному шару мережі. Вхідні дані послідовно проходять обробку по всіх шарах мережі. Параметри кожного нейрона можуть змінюватися залежно від результатів, отриманих на попередніх наборах вхідних даних, змінюючи таким чином і порядок роботи всієї системи [13].

Керівник напряму «Пошук Mail.ru» в Mail.Ru Group Андрій Калінін зазначає, що нейронні мережі здатні вирішувати такі самі завдання, як і інші алгоритми машинного навчання, різниця полягає лише у підході до навчання.

Всі завдання, які можуть вирішувати нейронні мережі, так чи інакше пов'язані із навчанням. Серед основних областей застосування нейронних мереж – прогнозування, прийняття рішень, розпізнавання образів, оптимізація, аналіз даних.

Директор програм технологічного співробітництва Microsoft в СНД Влад Шершульський зауважує, що зараз нейромережі застосовуються повсюдно. Нейромережі лежать в основі більшості сучасних систем розпізнавання та синтезу мови, а також розпізнавання та обробки зображень. Вони застосовуються в деяких системах навігації, таких як промислові роботи або безпілотні автомобілі. Алгоритми на основі нейромереж захищають інформаційні системи від атак зловмисників та допомагають виявляти незаконний контент у мережі [20].

Вчені займаються розробкою штучних нейронних мереж понад 70 років. Першу спробу формалізувати нейронну мережу відносять до 1943 року, коли два американські вчені (Уоррен Мак-Каллок і Уолтер Піттс) представили статтю про логічне обчислення людських ідей та нервової активності.

Трудомістка та тривала частина процесу розробки нейронної мережі – її навчання. Щоб нейронна мережа могла коректно вирішувати поставлені завдання, потрібно «запускати» її роботу на десятках мільйонів наборів вхідних даних.

Серед основних гравців ринку слід зазначити Google та її підрозділ Google DeepMind, який створив мережу AlphaGo, та Google Brain. Власні розробки в цій галузі є у Microsoft – ними займається лабораторія Microsoft Research. Створенням нейронних мереж займаються в IBM, Facebook(підрозділ Facebook AI Research), Baidu(Baidu Institute of Deep Learning) та інші. Багато розробок ведеться в технічних університетах по всьому світу [21].

Згорткова нейронна мережа - нейронна мережа, в якій присутній шар згортки (Convolutional layer). Зазвичай у сверточних нейронних мережах також присутні шар субдискретизації (pooling layer) та повнозв'язний шар (fully connected layer). Згорткові нейронні мережі застосовуються для оптичного розпізнавання, класифікації зображень, детектування предметів, семантичної сегментації та інших завдань [10].

Основи сучасної архітектури згорткових нейронних мереж було закладено в одній із перших широко відомих згорткової нейронної мережі - LeNet-5 Яна ЛеКуна у 1998 році. У згорткових нейронних мережах шари згортки та субдискретизації складаються з кількох «рівнів» нейронів, званих картами ознак (feature maps), або каналами (channels). Кожен нейрон такого шару з'єднаний з невеликою ділянкою

попереднього шару, званим рецептивним полем. У разі зображення картка ознак є двовимірним масивом нейронів, або просто матрицею. Інші вимірювання можуть бути використані, якщо на вхід приймається інший вид даних, наприклад, аудіо дані (одномірний масив) або об'ємні дані (тривимірний масив).

У 2012 році на конкурсі ILSVRC з класифікації зображень вперше перемогла нейронна мережа - AlexNet, досягнувши top-5 помилки 15,31%. Для порівняння, метод, який не використовує згорткові нейронні мережі, отримав помилку 26,1%. В AlexNet були зібрані нові на той момент техніки для покращення роботи мережі. Навчання AlexNet через кількість параметрів мережі відбувалося на двох GPU, що дозволило скоротити час навчання в порівняно з навчанням на CPU. Також виявилося, що використання функції активації ReLU(Rectified Linear Unit) замість більш традиційних функцій сигмоїди та гіперболічного тангенсу дозволило знизити кількість епох навчання у шість разів [20].

Широка популярність соціальних медіа спільнот, як відомо, призводить спаду продуктивності діяльності людини, адже їх використання відволікає увагу, займає величезні шматки цінного робочого часу та призводить до зниження продуктивності [9].

Обмеження доступу допомагає значно підвищити продуктивність. Однак деякі галузі застосовують соціальні мережі та месенджери у роботі. У цих галузях різні набори сайтів, чатів чи каналів стають збудниками спаду продуктивності, наприклад, онлайн-магазини та потокові сайти, канали з розважальним контентом та ін. Немає абсолютно ніяких причин, чому працівники повинні дивитися фільми під час роботи або обмінюватися мемами чи посиланнями на розважальний контент. Окрім обмеження доступу до такого контенту, який відволікає увагу, важливо розглянути й інші важливі проблеми, які можуть становити загрозу, наприклад, модерація контенту з невідповідним вмістом чи контенту з образливим вмістом. Для фільтрування таких даних почали створюватись різні рівні фільтрів, одним з яких є модератори чатів чи груп. Вони виконують обробку медіа та текстових даних вручну [3, 8] .

Використання штучного інтелекту та розробки віртуальних ботів надали значну перевагу у створенні нових рівнів фільтрації. Вони дозволяють динамічно аналізувати

вміст чату, групи чи каналу, що підвищує ефективність боротьби з невідповідним контентом та ін. [3, 10, 17, 18].

З цього можна зробити висновки, що впровадження ботів з розумною фільтрацією контенту дозволить підвищити рівень продуктивності та зберегти учасників чату від образливого контенту та контенту, який завадить у роботі [1, 10].

Практичне значення даної роботи полягає у проведенні досліджень та розробці чат-боту з інтсрументами для ефективного аналізу та класифікації текстових даних та зображень. Розроблений чат-бот є повноцінним інструментом для аналізу та фільтрації текстового контенту чи зображень у чатах.

Провівши аналіз існуючих бібліотек та алгоритмів для фільтрації, алгоритмів згорткових нейронних мереж та загальної області дослідження, було визначено такі характеристики:

— Навчання згорткової нейронної мережі для класифікації зображень;

— Використання алгоритму Левенштейна для аналізу тексту з вирахуванням дистанції між співвідношеннями слів;

— Побудова чат-бота для перехоплення повідомлень у чаті;

— Можливість видалення повідомлення та нотифікації адміністратора;

— Можливість додаткового тренування моделі під час роботи боту;

— Можливість заміни моделі за необхідністю.

У наш час месенджерами користується майже кожна людина, вони набули широкої популярності у різних організаціях, як інструмент для спілкування, управління підприємствами віддаючи накази та ін.

На даний момент більшість крупних компаній використовують чат-ботів як інструмент для оперування контентом у каналах, спілкуванні з клієнтами та ін [21].

При теперішній карантинній ситуації, велика кількість робітників перейшли на віддалену работу, але організовувати роботу при масивах відволікаючого контенту дуже складно. На протязі дня багато збудників можуть відволікати від роботи, одним з таких збудників є канали з зображеннями розважального чи шкідливого характеру та текстовими даними відповідно. Працівники відволікаються переглядяючи такий контент та відволікають колег від праці пересилаючи такі дані до корпоративних

чатів та іншого. Тому вдалим вирішення даної проблеми буде інтеграція чат-боту до чатів для проведення інтелектуальної фільтрації контенту, що допоможе оптимізувати роботу працівників, завадити відволіканню від роботи та провести дослідження доцільності використання навчаних моделей при роботі з месенджерами та ін.

З цього можна зробити висновок, що створення такого боту та проведення дослідження має перспективи у розвитку та використанні навчаних моделей у поєднанні з месенджерами у майбутньому. Метою даної роботи є, розробка інтелектуального чат-боту, який буде відповідати вимогам предметної області. Вирішувати поставлені перед ним питання. Чат-бот буде слугувати інструментом автоматизації фільтрації контенту корпоративних чатів.

1.2. Аналіз програмних продуктів – аналогів

В мережі Інтернет ми можемо знайти інтелектуальних чат-ботів, боти для месенджера «Telegram» не є винятком. Як правило ці боти виконують роль провідників або інструментів для збору заказів чи звітів.

Під час аналізу програмних ботів-аналогів було проведено аналіз декількох розповсюджених ботів для платформи «Telegram», функціонал яких найбільш наближений до нашої майбутньої реалізації. На даному етапі ми провели аналіз конкурентів та виявили сильні та слабкі сторони.

Для аналізу ми обрали інтелектуальні чат-боти, які виконують роль антиспаму, антимату чи інші наближені функції.

Розглянемо перший аналог, антиспам-бот «ChatKeeperBot».

ChatKeeperBot — це багатофункціональний бот-модератор для тих, хто цінує легкість спілкування та свій особистий час. У ньому є всі необхідні інструменти для керування групою «Telegram» [5].

Бот має такі функції:

- потужна анти-спам система, що включає 28 різних фільтрів, face-контроль, набір інструментів для перевірки нових користувачів;
- різні обмеження для новачків групи;
- система репутації
- великий перелік команд для адміністраторів;
- статистика:
  - загальна за чатом;
  - по користувачам;
  - з рефералів;
  - аналітика та теплові карти;
- тригери, що гнучко налаштовуються, за допомогою яких можна отримати будь-яку реакцію за заданими вами умовами [5].

Недоліком боту є відсутність автомодерації тексту, основною вимогою є введення фільтрів для пошуку по тексту чи посиланням.

Нижче розглянуто приклад використання бота (рис. 1.1 – 1.5). Доступ до боту знаходиться за наступним посиланням: https://t.me/ChatKeeperBot [5].

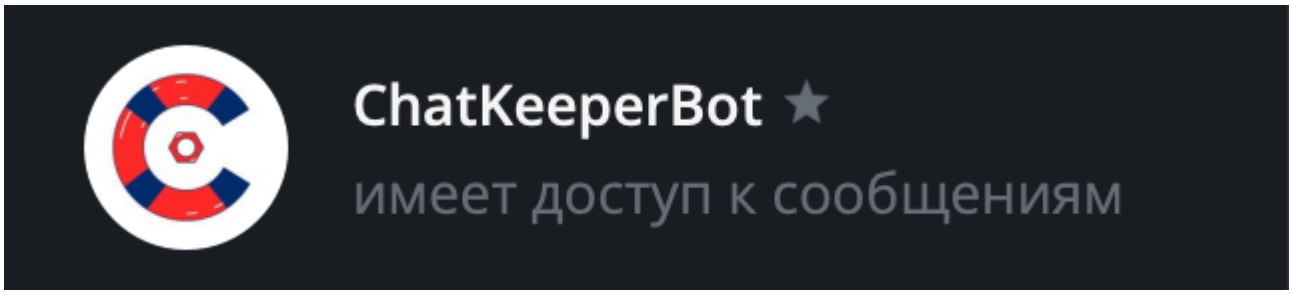


Рисунок 1.1 – Бот доданий до чату

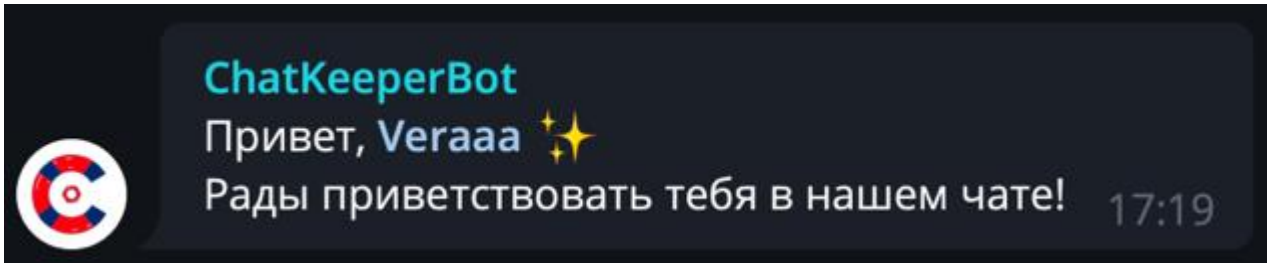


Рисунок 1.2 – Бот надсилає привітання для нового користувача

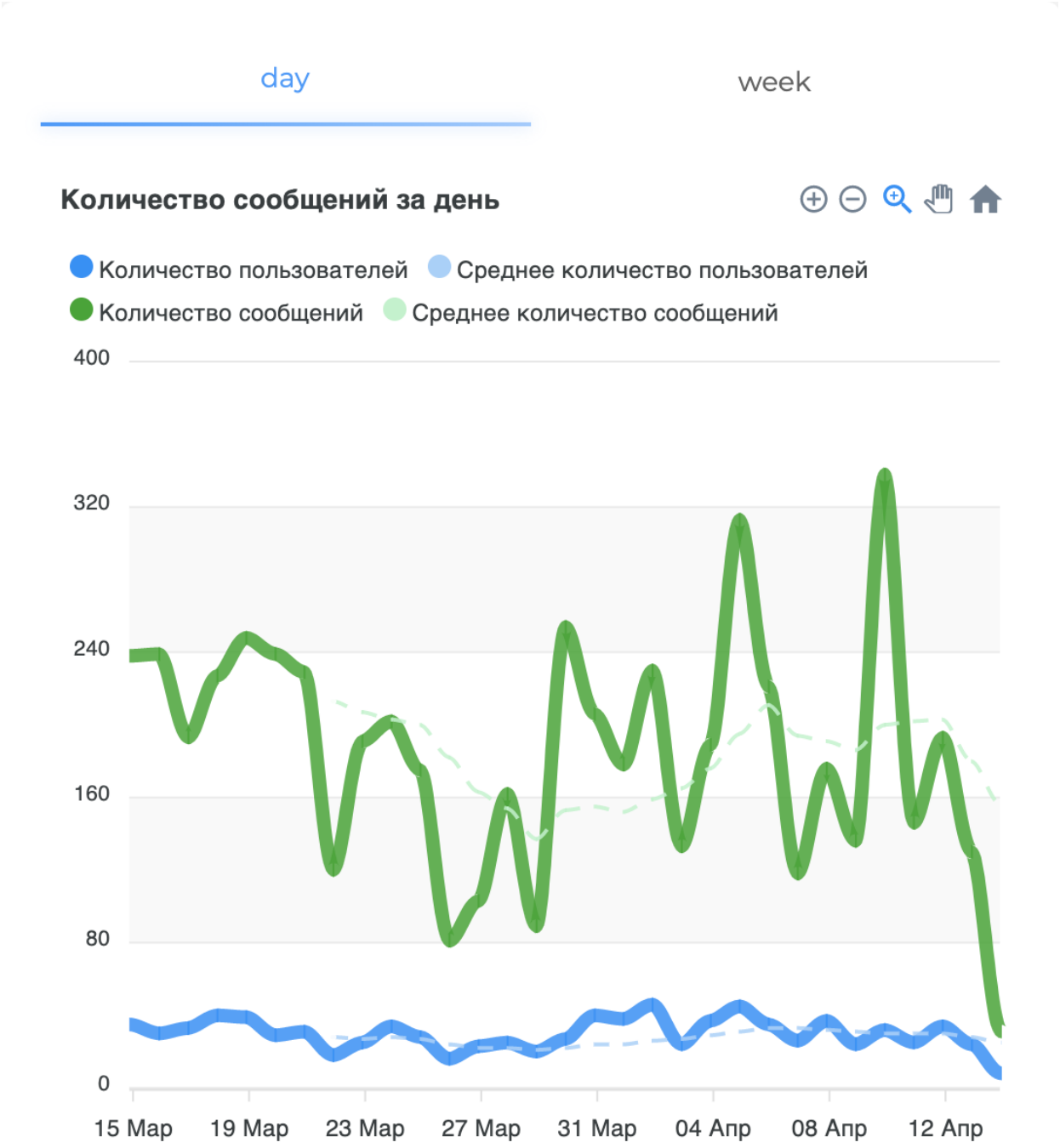


Рисунок 1.3 – Розділ «Статистика»

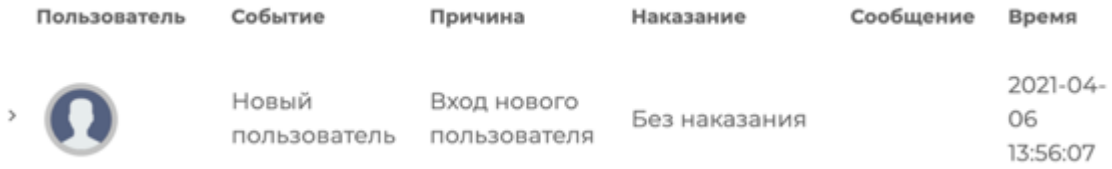


Рисунок 1.4 – Журнал, щоб знати, який користувач, коли, за що і як покараний

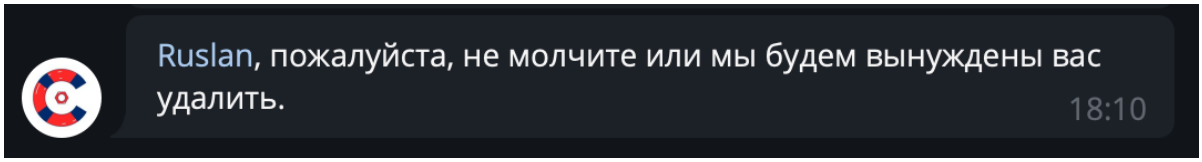


Рисунок 1.5 – Приклад модерації

Розглянемо другий аналог, бот «WatchDog» (рис. 1.6 – 1.8). Доступ до боту знаходиться за наступним посиланням: https://t.me/watchdog_robot [6].

Watchdog видаляє повідомлення типів чорного списку: посилання, наклейки, gif, голосові вкладення, вкладення файлів та інше. Також він може заборонити інших ботів [6].

Недоліком боту є відсутність автомодерації тексту, основною вимогою є введення фільтрів для пошуку по тексту чи посиланням. Недолік є критичним, бо адміністратор повинен додавати кожне слово вручну, що забирає багато часу [6].

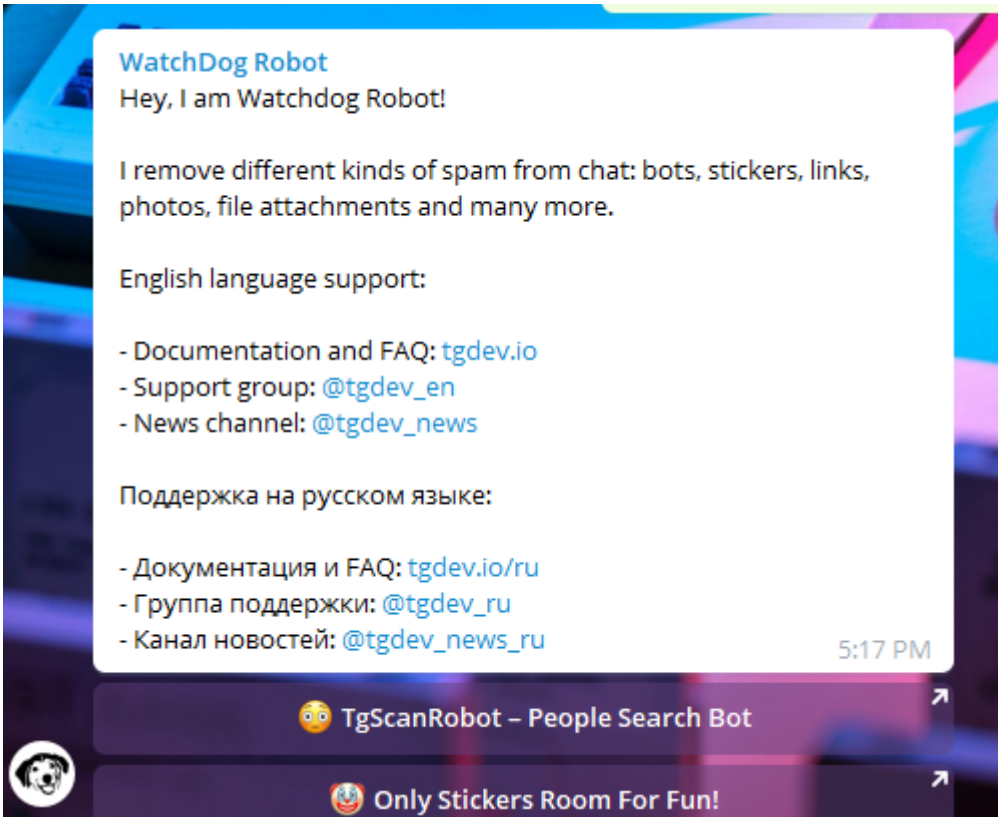


Рисунок 1.6 – Додавання боту до чату

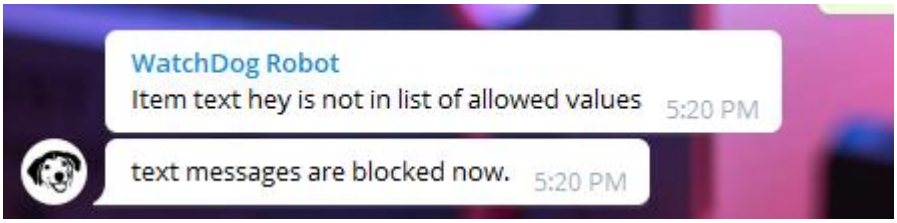


Рисунок 1.7 – Видалення повідомлення

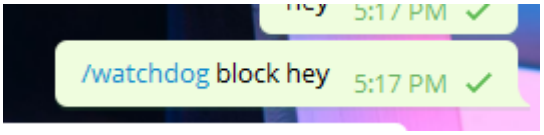


Рисунок 1.8 – Додавання фільтру

Розглянемо наступний аналог-бот «Daysandbox» (рис. 1.9 – 1.11). Доступ до інформаційної системи знаходиться за наступним посиланням: https://t.me/daysandbox_bot [7] .

Daysandbox Bot — це простий, але ефективний бот-анти-спам. Ідея полягає в тому, щоб видалити посилання, медіа та переслані повідомлення, написані користувачами, які приєдналися до чату менше ніж 24 години тому. Цей метод дозволяє видалити більшість спам-повідомлень від нових користувачів. Недоліком цього підходу є те, що будь-який учасник чату старше 1 дня може публікувати будь-що. Ви отримуєте безкоштовно інструмент, який автоматично видаляє багато спам-повідомлень, але не всі. Ви можете встановити список доменів і груп/каналів, які ніколи не повинні блокуватися. Ви можете додати нового учасника чату до білого списку, якщо хочете дозволити йому публікувати посилання до закінчення безпечного періоду [7].

Недоліком є те, що видалення медіа-повідомлень вимкнено за замовчуванням, також бот видаляє лише посилання та відмітки користувачів чи інших каналів як спам [7].

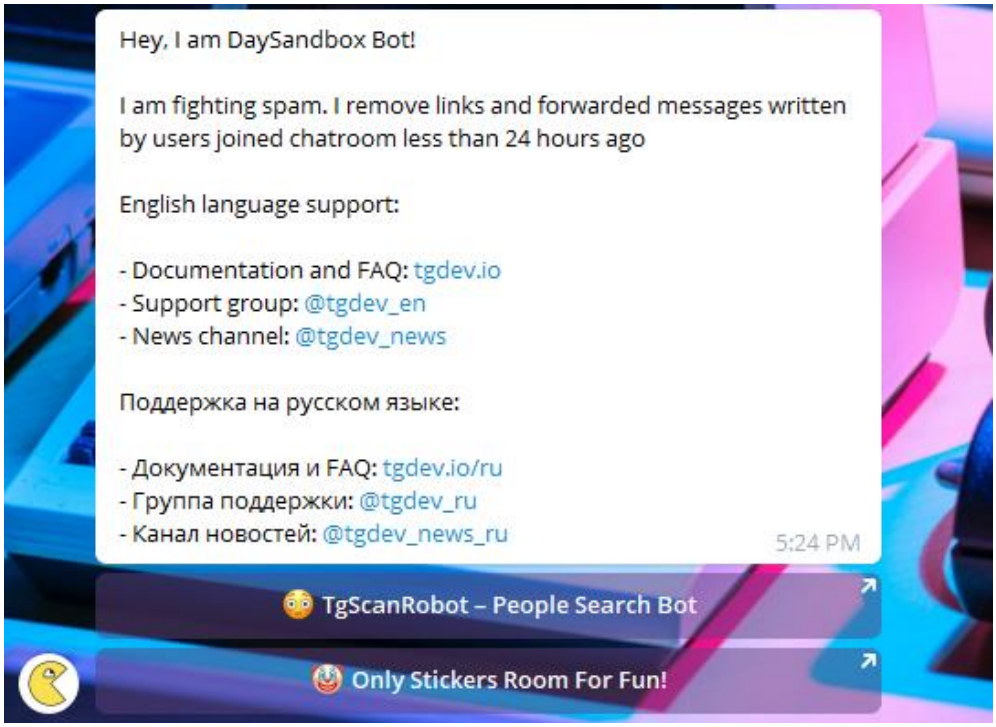


Рисунок 1.9 – Додавання боту до чату

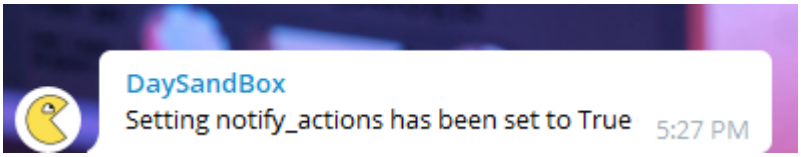


Рисунок 1.10 – Налаштування бота

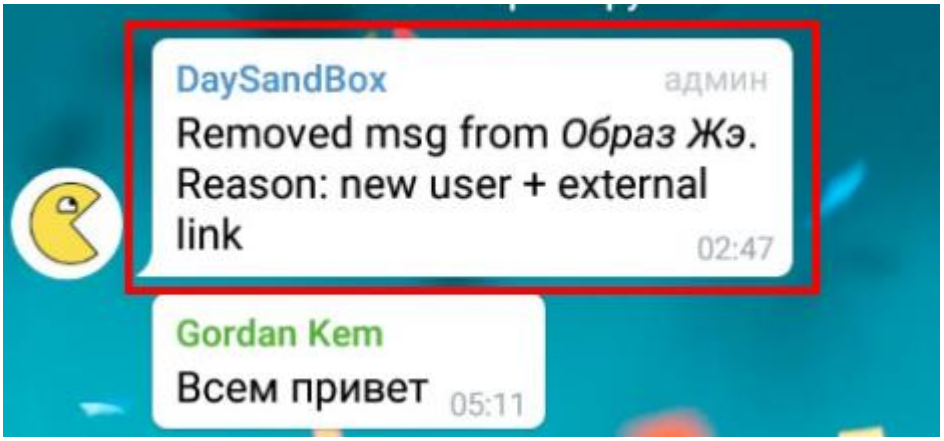


Рисунок 1.11 – Видалення повідомлення

Результати викладеного розгляду програмних аналогів зведено до порівняльної таблиці характеристик ботів-аналогів (табл. 1.1).

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерії	Bot «ChatKeeperBot»	Bot «WatchDog»	Bot «Daysandbox»	Bot «ASC»
Швидке встановлення до чату	+	+	+	+
Автоматичний аналіз тексту	-	-	-	+
Аналіз зображень	-	-	-	+
Видалення повідомлень	-	+	+	+
Бан користувачів	+	+	+	+
Повідомлення для адміністратора	-	-	-	+

Отже, проведений огляд актуального стану розвитку інформаційних технологій в сфері реалізацій функції модерації контенту дозволяє констатувати факт, що обрана тема дослідження є актуальною. Також актуальним є питання створення інтелектуального власного програмного продукту – чат-боту «ASC», для фільтрації чатів організації. Основне його призначення – це автоматизація процесу модерації та фільтрування повідомлень учасників чатів та груп організації.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1. Мета та задачі дослідження

Дана дипломна робота присвячена реалізації інтелектуального чат-боту для автоматизації модерації та фільтрації контенту у групах і чатах компанії, також мета роботи присвячена дослідженню доцільності використання технологій машинного навчання при роботі з масивами медіа та текстових даних. Чат-бот повинен забезпечити виконання аналізу повідомлень, що надходять від користувачів, у випадку виявлення повідомлень з невідповідним контентом, забезпечити видалення повідомлення, попередження адміністратора групи чи чату та блокування роботи користувача.

Для досягнення поставленої мети було визначено задачі:

— провести аналіз реалізованих чат-ботів зі схожим функціоналом;

— визначити послідовність етапів побудови інтелектуального чат-боту;

— провівши детальний аналіз обрати технології, необхідні для побудови чат-боту;

— за допомогою uml провести моделювання компонентів чат-боту;

— спроектувати та навчити інструменти для аналізу повідомлень;

— розробка чат-боту;

— тестування чат-боту.

Інтелектуальний чат-бот повинен бути реалізований мовою Python, з використанням вбудованих бібліотек для аналізу тексту та зображень [10, 13, 17, 18]. Даний бот повинен бути розміщений в мережі Інтернет. [11] Реалізований чат-бот повинен мати змогу навчатись на отриманих повідомленнях використовуючи компоненти бібліотек TensorFlow, Keras, NumPy [12, 14 - 16].

Бот має нотифікувати адміністраторів чату. [19] Банити користувачів [10].

Для підтримки роботи чат-боту необхідні мати базові навички роботи з чат-ботами у Telegram, користувач повинен мати загальні навички роботи з персональним комп'ютером, вміти використовувати браузер чи месенджер.

Програмна реалізація повинна задовольняти версію месенджера Telegram 2.0 і вище.

Чат-бот повинен задовольняти наступні функціональні вимоги:

— аналіз повідомлень користувача;

— видалення повідомлення користувача;

— бан користувачів

— нотифікація адміністратора.

Детальніше планування робіт описано у додатку А.

2.2. Методи дослідження

За методи дослідження було обрано методи машинного навчання для класифікації зображень та алгоритми аналізу текстових данних на збіги.

Провівши аналіз різних технологій для створення чат-ботів, серед яких мови програмування Go, Java, Python та JavaScript, було вирішено зупинитися на мові програмування Python та наближених до неї компонентів та бібліотек.

При аналізі було виявлено, що мова має деякі недоліки через використання інтерпретатора у процесі роботи, але серед переваг слід виділити швидкість створення додатків та велику кількість бібліотек для машинного навчання та роботи з масивами даних. Тому, є очевидним, що мова Python є несумнівним лідером у цьому напрямку.

Також було проведено аналіз методів зчитування та класифікації зображень серед великої кількості дослідів з використанням бібліотек PyTorch, TensorFlow, Keras та статистикою швидкості та якості використання алгоритмів машинного навчання було вирішено зупинитися на TensorFlow у поєднанні з компонентами

Keras. Перевагою даних бібліотек є відкритий доступ до компонентів та широкий набір документації та зразків для навчання та аналізу використання даних інструментів при машинному навчанні. TensorFlow має широкий ряд прикладів для створення та навчання моделі класифікатору зображень, дозволяє швидко створювати аналізатори зображень завдяки гнучкості налаштування шарів та наборам дата-сетів для навчання моделей. Також слід зазначити, що дана бібліотека підтримується компанією Google та завдяки цьому має великий набір супутніх інструментів. Даний інструмент дозволить швидко будувати згорткову мережу для проведення навчання класифікації зображень.

Для класифікації зображень було проведено дослідження у навчанні згорткової нейронної мережі.

Для початку слід зрозуміти, що таке нейронна мережа та в чому суть її використання. Нейронна мережа — це математична модель програмного або апаратного забезпечення, яка заснована на принципі нейробіологічних мереж, які діють як нейрони в живому організмі. Ця концепція походить від спроби моделювати процеси, що відбуваються в мозку. Першим таким експериментом були нейронні мережі У. Маккалока та У. Піттса. Використовувалися моделі, отримані після розробки алгоритмів навчання, у цілях прогнозування проблем, розпізнавання моделі, задачі керування тощо.

Зі сторони машинного навчання, нейронні мережі є унікальними методами класифікації зображень, методами упередженого аналізу, кластерізації або агрегації.

Нейронні мережі не мають якогось ключового алгоритму як програма, завдяки своїй будові вони навчаються так можуть покращувати показники своєї роботи, а не лише виконують операції з базових блоків. Здатність до навчання є однією з найважливіших переваг нейронних мереж порівняно з традиційними алгоритмами. Технічно суть навчання полягає в тому, щоб знайти правильні з’єднання для нейронів використовуючи коефіцієнти. Під час процесу навчання мережа може виявляти та аналізувати складні зв’язки між вхідними та вихідними даними. Це означає, що успішне навчання дозволить мережі отримати точний результат на основі великої

кількості даних, які спочатку можуть бути відсутніми, показуючи навчання або частково неправильні дані.

Згорткова нейронна мережа є алгоритмом «глубокого» навчання, який може прийняти як вхідні дані зображення, призначати важливі коефіцієнти(корегуючі ваги та відхилення) на зображенні та мати змогу їх відрізняти. Необхідна попередня обробка менше, ніж для інших алгоритмів класифікації. Зазвичай, фільтри створюються вручну, якщо є достатньому рівні підготовки, мережі вивчають ці фільтри та проводять аналіз. Зображення – це не що інше, як матриця значень пікселів. У випадках надзвичайно простих двійкових зображень метод прямих переміщень у звичайній мережі може показати середню оцінку точності під час виконання передбачення класів, але не матиме практичної точності, коли справа дійде до складних зображень із залежністю від пікселів.

Конвульсивні мережі можуть успішно фіксувати залежності у просторі чи часі в зображенні застосовуючи відповідні фільтри. Набір архітектури добре поєднується з даними зображення, з можливістю зменшення кількості включених параметрів і повторного використання масштабу. Це означає, що ви можете краще навчити свою мережу та виявити унікальність свого зображення.

При визначенні найбільш вдалого інструменту для аналізу тексту було проведено аналіз статистики навчань моделей для класифікації компонентів текстових масивів даних. При аналізі було виявлено, що використання машинного навчання для аналізу природньої мови потребує великих ресурсів та масивів даних, чого у свою чергу розробник не має для проведення досліджень. Також, одним з основних тригерів, що стали збудниками до відхилення ідеї використання машинного навчання при класифікації тексту була швидкість навчання моделей та швидкість і ціна використання при роботі з віддаленими серверами. Прикладом може слугувати процес навчання алгоритму GPT-3, який потребував великого масиву часу та приблизно 600 Гігабайт текстових даних. Тому, для аналізу тексту було вирішено дивитися у бік алгоритмів для спрощенного аналізу та виявлення необхідних даних у частинах тексту за допомогою комбінацій літер природньої мови. Одним з таких алгоритмів було обрано алгоритм «Редакційної» відстані або відстані Левенштейна.

Редакційна відстань, або відстань Левенштейна - метрика, що дозволяє визначити подібність двох рядків, проводячи мінімум операцій, видалення і заміни одного символу на інший, перетворюючи один рядок в інший. Даний алгоритм широко використовується у теорії інформації та базовій компьютерній лінгвістиці, а у поєднанні з аугментаційним алфавітом, який забезпечить можливістю боротьби з навмисним спотворенням слова для запобігання його вдалому аналізу, надасть швидку класифікацію слів згідно з заданим словником слів.

Для створення чат-боту було обрано вільний текстовий редактор коду Visual Studio Code від компанії Microsoft. Редактор має переваги у швидкості роботи та гнучкості у налаштуванні у порівнянні з повноцінним середовищем розробки, що є плюсом, якщо розробник має недостатньо потужну електронно-обчислювальну машину. Розповсюджується безкоштовно, розробляється як програмне забезпечення з відкритим вихідним кодом.

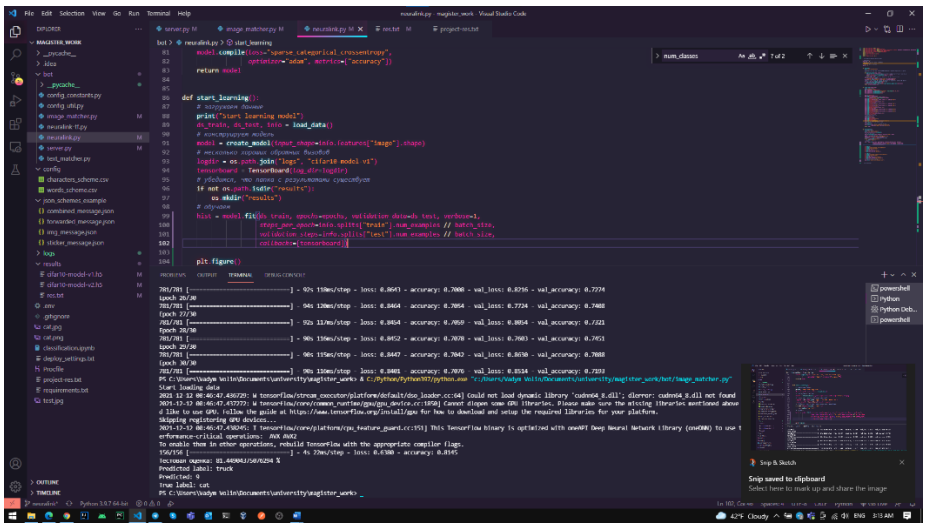


Рисунок 2.1 – Вікно редактора VS Code

Переваги VS Code:

— Розумне завершення;

— Аналіз потоку даних;

— Мовна ін'єкція;

— Багатомовне переробництво;

— Виявлення дублікатів;

— Перевірки та швидкі виправлення;

— Контроль версій;

Мають вбудовані функції для систем управління версіями. Основними функціями є редактор даних, об'єкти даних, текстовий редактор, навігацію по компонентах коду, автодоповнення коду, історію змін файлу, термінал.

3 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Архітектура додатку

Для початку необхідно проаналізувати функції нашого чат-боту та побудувати майбутню архітектуру, і розробити структуру для планування майбутнього продукту та проектування чат-боту.

Для базової роботи інтелектуального чат-боту потрібна реєстрація у месенджері Telegram, наявність віртуальної машини чи серверу, де буде розміщено компоненти чат-боту, для цього було використано ресурси власного комп'ютеру, через відсутність безкоштовних серверів чи хмарних віртуальних машин з дисковим масивом більше ніж 512 Мегабайт, для збільшення можливостей боту та підтримки його роботи при великих масивах повідомлень необхідно обрати якісний сервер чи хмарний диск з достатніми характеристиками.

Архітектура інтелектуального чат-боту представлена на зображенні (рис. 3.1).

Для роботи з чат-ботом користувачу потрібно відкрити веб-додаток месенджеру використавши веб-браузер або відкрити десктоп чи мобільний додаток «Telegram».

Користувач через веб-, десктоп- чи мобільний-додаток надсилає повідомлення до чату, чат-бот буде перехоплювати повідомлення та проводити перевірку на наявність невідповідного контенту за допомогою навчаної моделі, яка у свою чергу зберігається на дисковому масиві серверу.

Архітектуру взаємодії з чат-ботом розподілено на такі складові:

— Користувачі чату;

— Веб-додаток;

— Мобільний-додаток;

— Десктоп-додаток;

— Чат-бот;

— Навчана модель;

— Дата-сервер.

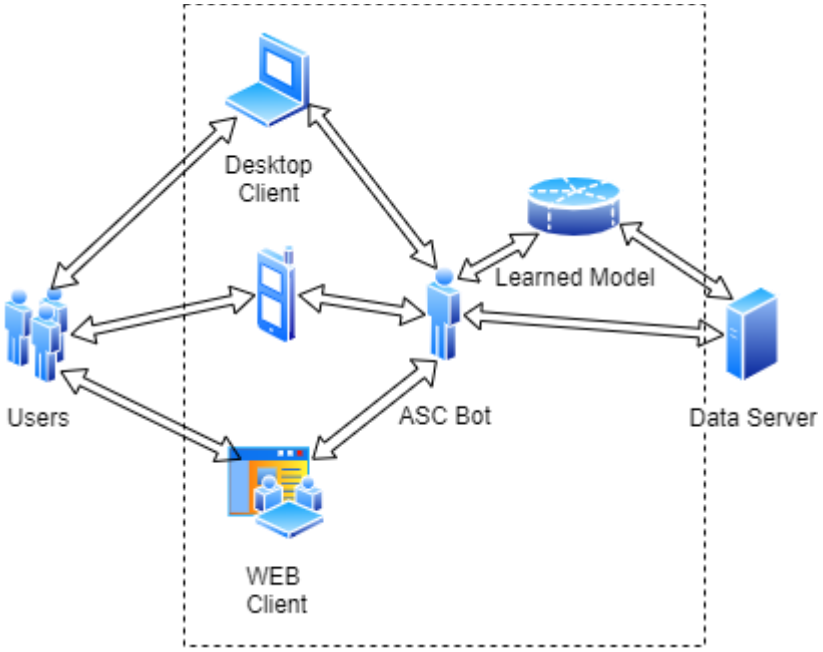


Рисунок 3.1 – Детальна архітектура чат-боту

3.2.Структурно-функціональне моделювання роботи чат-боту

Спершу необхідно створити контекстну діаграму, що є функціональною моделлю, діаграма є центром системи побудови та поєднує процес роботи чат-боту [18, 19].

Проміжна структура – це ще одна обчислювальна альтернатива, яка відображається в одиниці реалізації та відображає основну логіку роботи чат-боту.

Вхідними даними до функції «Модерація повідомлень користувача» є:

— Повідомлення користувача.

Вихідними даними є:

— Результат видалення повідомлення;

— Повідомлення без змін.

Для даної діаграми необхідна наявність принаймні одного елементу управління, який під’єднаний згори до моделі з функціонального блоку. З даної діаграми видно, що елементом управління представлено:

— Правила фільтрації повідомлень [17, 18].

Механізмами моделі є:

— Чат-бот(виконує автоматизовану модерацію контенту);

— Адміністратор чату(виконує модерацію контенту за необхідності вручну).

Контекстна діаграма інтелектуального чат-боту та її декомпозиція, зображені на рисунках 3.2-3.3.

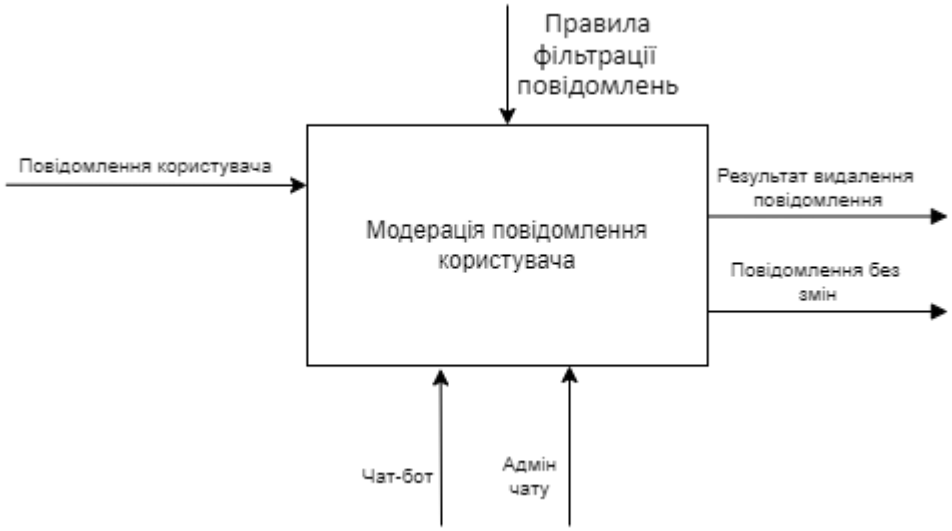


Рисунок 3.2 – Діаграма IDEF0

Щоб детальніше провести аналіз основних функціональних процесів у роботі чат-боту потрібно спроектувати декомпозицію контекстної діаграми.

В результаті роботи було розроблено декомпозицію головного процесу.

Після виконаної роботи схема була декомпозована на такі процеси:

— Перехоплення повідомлення;

— Модерація контенту;

— Схвалення повідомлення;

— Видалення повідомлення.

Процес «Перехоплення повідомлення» має такі властивості:

— Вхідні дані:
    — повідомлення користувача.
— Вихідні дані:
    — контент повідомлення.
— Механізми:
    — Чат-бот.

Процес «Модерація контенту» має такі властивості:

— Вхідні дані:
    — Контент повідомлення.
— Вихідні дані:
    — Результат модерації.
— Елементи управління:
    — Ф-ції аналізу чат-боту.
— Механізми:
    — Адміністратор чату;
    — Чат-бот.

Процес «Схвалення повідомлення» має такі властивості:

— Вхідні дані:
    — Результат модерації.
— Вихідні дані:
    — Повідомлення без змін.

Процес «Процес видалення повідомлення» має такі властивості:

— Вхідні дані:
    — Результат модерації.
— Вихідні дані:
    — Результат видалення повідомлення.
— Механізми:
    — Адміністратор чату;
    — Чат-бот.

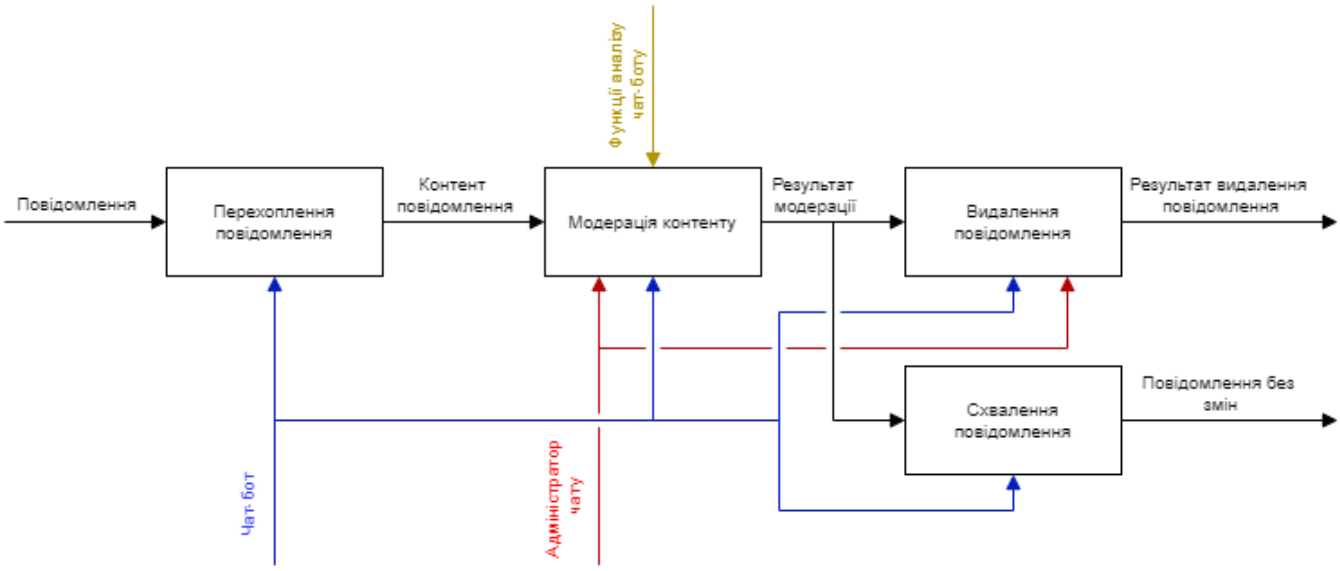


Рисунок 3.3 – Діаграма декомпозиції

3.3.Проектування та навчання моделі класифікації

Для проектування моделі класифікації було обрано конволюційну нейронну мережу для класифікації зображень CIFAR-10.

Нейронна мережа , в якій щонайменше один шар являє собою згортковий шар. Типова згорткова нейронна мережа складається з певної комбінації наступних шарів:

— згорткові шари;

— об'єднання шарів;

— щільні шари.

Згорткові нейронні мережі мали великий успіх у деяких видах проблем, таких як розпізнавання зображень.

Набір даних CIFAR-10 містить 60 000 кольорових зображень у 10 класах, по 6 000 зображень у кожному класі. Набір даних поділено на 50 000 навчальних зображень та 10 000 тестових зображень. Класи є взаємовиключними, і з-поміж них немає перекриття.

Набір даних складається з 10 класів зображень, які мають мітки від 0 до 9:

— 0 : літак.
— 1 : автомобільний.
— 2 : птах.
— 3 : кіт.
— 4 : олень.
— 5 : собака.
— 6 : жаба.
— 7 : кінь.
— 8 : корабель.
— 9 : вантажівка.

50000 зразків для навчальних даних і 10000 зразків для даних тестування.

Кожен зразок є зображенням розміром 32x32x3 пікселів (ширина і висота 32 і глибина 3, що є значеннями RGB).

Завдяки використанню API Keras Sequential та TensorFlow побудова шарів моделі та згорткової основи проходить швидко з компонентами Conv2D та MaxPooling2D шарів, які представленні на рисунках 3.3-3.4 нижче.

Conv2D – шар 2D згортки (наприклад, просторова згортка над зображеннями). Цей шар створює ядро згортки, яке згортається з вхідним шаром для створення тензора вихідних даних. Використовуючи цей шар як перший шар у моделі, надайте аргумент ключового слова input_shape (кортеж цілих чисел або None, не включає вісь зразка), наприклад, input_shape=(128, 128, 3)для зображень 128x128 RGB у форматі data_format="channels_last". Ви можете використовувати, None коли розмір має змінний розмір.

MaxPooling2D – максимальна операція об’єднання для 2D просторових даних. Зменшує дискретизацію вхідного сигналу за його просторовими розмірами (висота та ширина), приймаючи максимальне значення у вікні введення (розміром, визначеним pool_size) для кожного каналу входу. Вікно зміщується на strides уздовж кожного виміру.

Activation – застосовує функцію активації до виходу.

Dropout – застосовується до входу. Рівень Dropout випадково встановлює вхідні одиниці на 0 з частотою rate на кожному кроці протягом часу навчання, що допомагає запобігти переобладнанню. Входи, не встановлені на 0, збільшуються на 1/(1 - швидкість), так що сума по всіх входах не змінюється.

Зауважте, що рівень Dropout застосовується лише тоді, коли для training встановлено значення True, щоб під час висновку жодні значення не скидалися. При використанні model.fit, training буде відповідним чином встановлене значення True автоматично, а в інших контекстах ви можете встановити kwarg явно на True під час виклику шару.

Flatten – вирівнює вхідні дані. Не впливає на розмір партії.

Dense – просто ваш звичайний щільно зв’язаний шар NN. Dense реалізує операцію: output = activation(dot(input, kernel) + bias), де activation– поелементна функція активації, передана як activation аргумент, kernel– матриця ваг, створена шаром, і bias вектор зміщення, створений шаром (застосовується лише, якщо use_biasє True). Це все атрибути Dense.

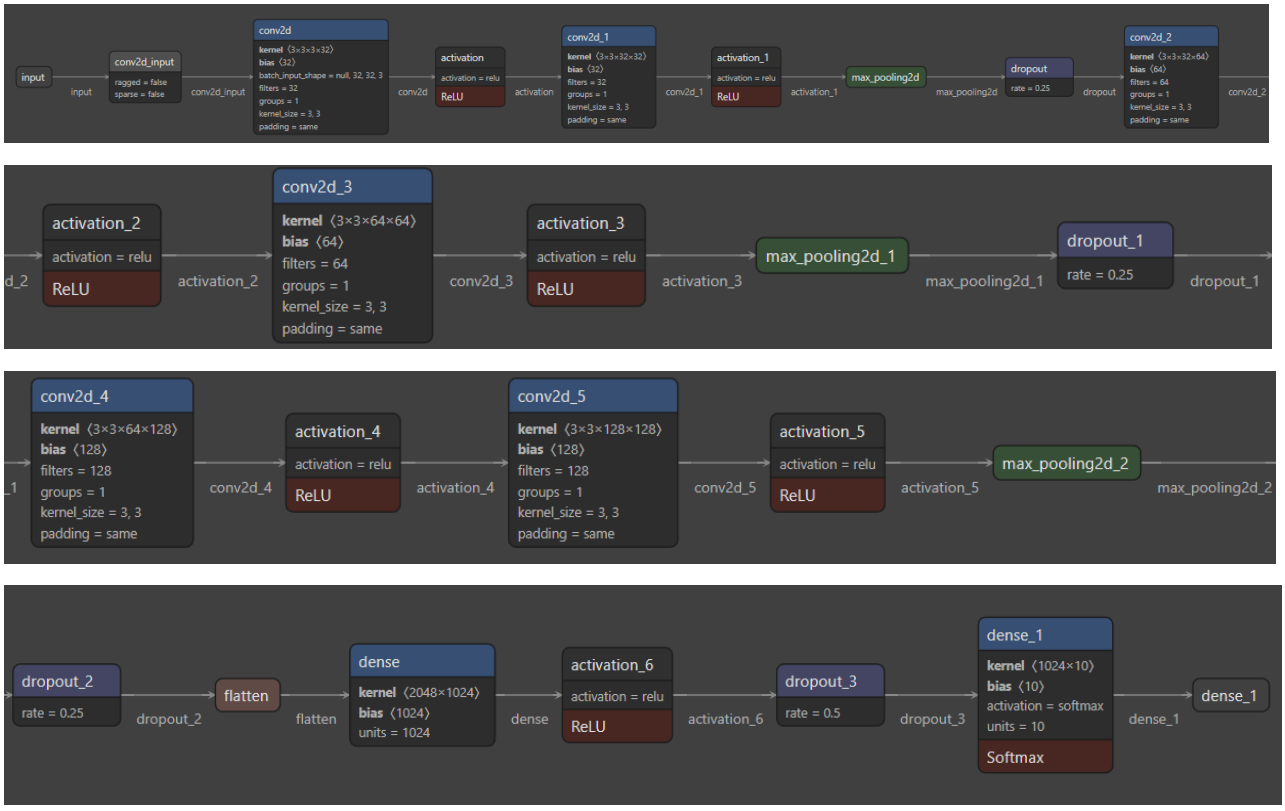


Рисунок 3.4 – Структурна діаграма нейронної мережі

Отриманий результат при використанні параметра "valid" заповнення має просторову форму (кількість рядків або стовпців): out_shape = math.floor((in_shape - p_size) / strides) + 1 (якщо in_shape >= p_size).

Результуюча вихідна форма при використанні параметра "same" відступу виглядає так: out_shape = math.floor((in_shape - 1) / strides) + 1.

Це 3 шари з 2 ConvNet з максимальним об’єднанням і функцією активації ReLU, а потім повністю з’єднані з 1024 одиницями. Це відносно невелика модель у порівнянні з ResNet50 або Xception, які є найсучаснішими.Для більш детального аналізу мережі було створено діаграму з детальними вагами та іншими коефіцієнтами.

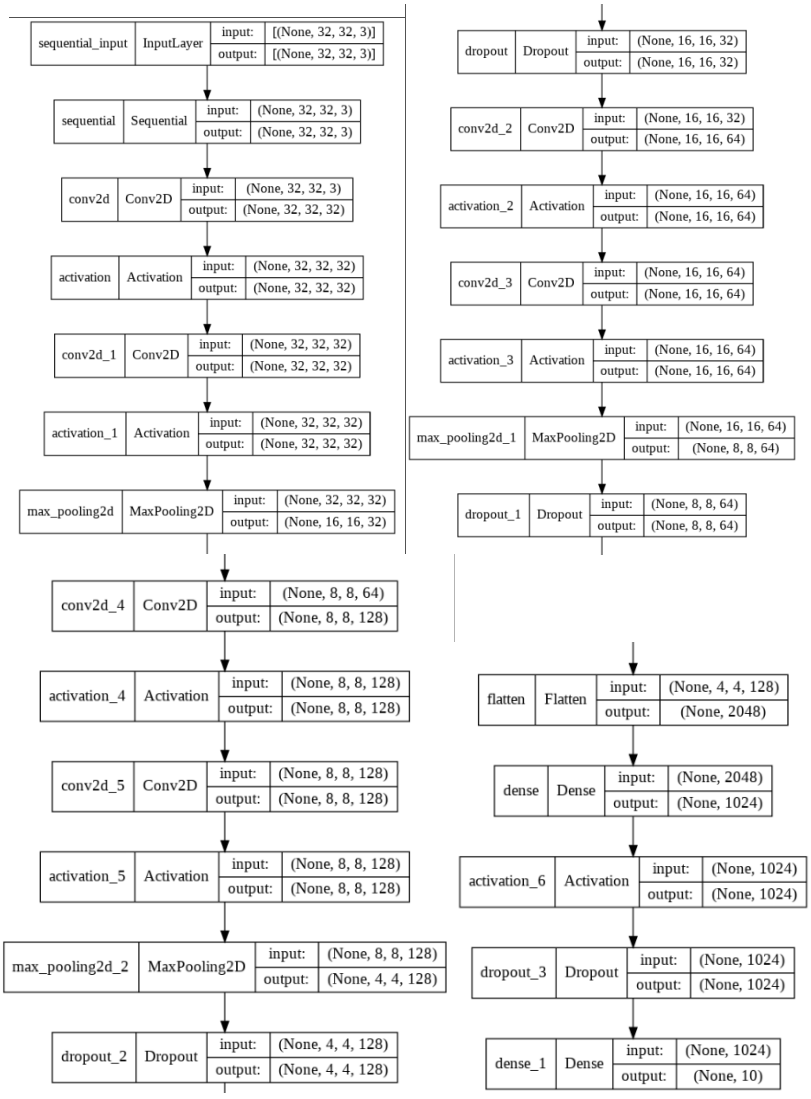


Рисунок 3.5 – Детальна діаграма з шарами нейронної мережі та вагами

Для аналізу тексту було вирішено використовувати алгоритми пошуку схожого тексту у масиві текстових даних. Було прийнято таке рішення, тому що, аналіз тексту навіть на добре навченій моделі займає дуже багато часу, таку оцінку ми отримали після навчання моделі для класифікації зображень. При аналізі зображення на навченій моделі для отримання результату модель витрачає велику кількість часу. В результаті цього було вирішено використовувати алгоритм відстані Левенштейна або редакційної відстані для знаходження близькості двох рядків.

Даний алгоритм аналізує скільки дій з редагування одного рядка потрібно зробити, щоб отримати інший рядок. Дії бувають такими: вставка символу, видалення символу та заміна одного символу іншим.

Поодинокі рядки мають нульову відстань: не потрібно нічого редагувати, перша і так дорівнює другій. Рядки «Добре» та «Доре» мають відстань 1 (пропущена одна літера, інше не змінилося). «Добре» та «добті» мають відстань 3 (одна заміна «Д» на «д», видалення літери «е» та вставка «і» на кінці та «р» на «т»).

Відстань Левенштейна та його узагальнення використовується для знаходження та виправлення помилок у словах (активно використовуються для пошукових систем, баз даних, аналізу набраного тексту, для автоматичного розпізнаванні отриманого тексту чи мови), у біоінформатиці використовується для аналізу і порівняння генів, білків чи хромосом.

На прикладі можна представити, що S1 та S2 це два рядки, з довжиною M і N, тоді редакційна відстань чи (відстань Левенштейна) D(S1, S2) можна вирахувати за наступною рекурентною формулою D(S1, S2) = D(M, N). Формулу алгоритму представлено на рисунку 3.6.

$$\mathrm{d}(S_1, S_2) = D(M, N)\text{ , где}$$

$$D(i,j) = \begin{cases} 0, & i = 0,\ j = 0 \\ i, & j = 0,\ i > 0 \\ j, & i = 0,\ j > 0 \\ \min\{ & \\ \quad D(i, j-1) + 1, & \\ \quad D(i-1, j) + 1, & j > 0,\ i > 0 \\ \quad D(i-1, j-1) + \mathrm{m}(S_1[i], S_2[j]) & \\ \} & \end{cases},$$

Рисунок 3.6 – Рекурентна формула відстані Левенштейна

Як алфавіт для прикладу було обрано базові літери та літери, які можуть бути використані для хибної зміни тексту з метою спотворення тексту для алгоритму.

Таблиця 3.1 – Алфавіт для співвідношення літер

а	а,a,@
б	б,6,b
в	в,b,v
г	г,r,g
д	д,d
е	е,e
ё	ё,e
ж	ж,zh,*
з	з,3,z
и	и,u,i
й	й,u,i
к	к,k,i{,|{
л	л,l,ji
м	м,m
н	н,h,n
о	о,o,0
п	п,n,p
р	р,r,p
с	с,c,s
т	т,m,t
у	у,y,u
ф	ф,f
х	х,x,h,}{
ц	ц,c,"u,"
ч	ч,ch
ш	ш,sh
щ	щ,sch
ь	ь,b
ы	ы,bi
ъ	ъ

э	э,е
ю	ю,іо
я	я,уа

3.4.Моделювання варіантів використання

Після проектування функціональних складових чат-боту, переходимо до побудови діаграми варіантів використання, що демонструє варіативність дій при роботі з чат-ботом [19].

UML - це мова моделювання, що складається з набору діаграм, створених аби допомогти учасникам проекту ідентифікувати та переглядати системи. UML надає компоненти для побудови графічного, архітектурного дизайну програмного забезпечення [19].

Діаграму варіантів використання необхідно створювати під час проектування системи, щоб описати графічно основні складові, прецеденти та специфікації, як опис послідовних дій, які виконуються при роботі чат-боту.

Компонентами зовнішнього зв’язку на діаграмі(рис. 3.7) виділено сутність користувача, який надсилає повідомлення. Дана Use Case diagram зображена на рисунку 3.7.

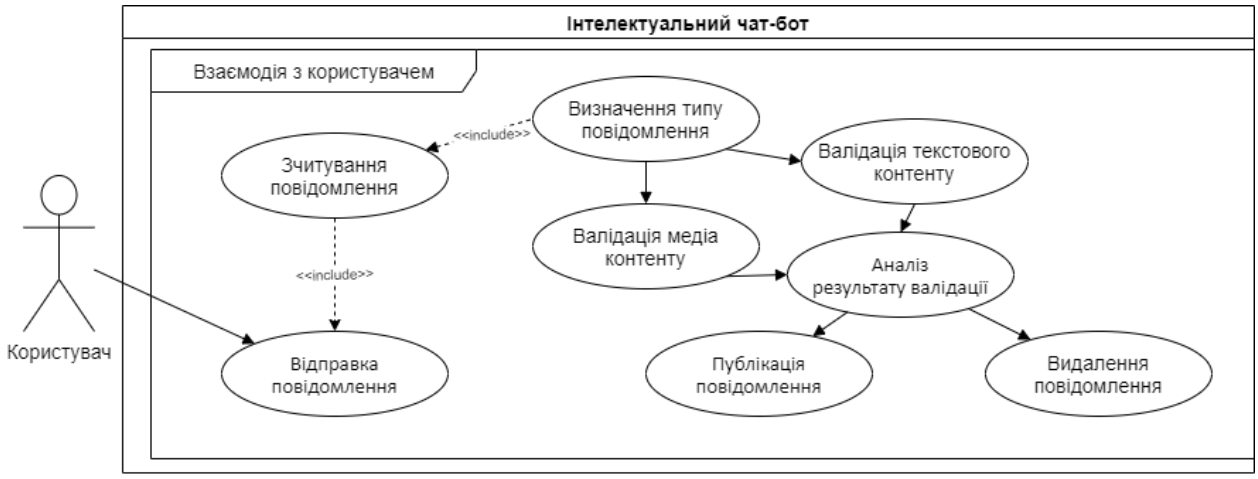


Рисунок 3.7 – Діаграма варіантів використання

Актором на діаграмі представлено користувача, який надсилає повідомлення у чат та ініціює функцію аналізу.

Варіантами використання виділено:

— ВВ – Відправка повідомлення – користувач надсилає повідомлення до чату;

— ВВ – Зчитування повідомлення – перехоплення повідомлення від користувача;

— ВВ – Визначення типу повідомлення – аналіз типу повідомлення та формування елементу дослідження;

— ВВ – Валідація текстового контенту – аналіз текстового елементу дослідження;

— ВВ – Валідація медіа контенту – аналіз медіа елементу дослідження;

— ВВ – Аналіз результату валідації – формування результату аналізу;

— ВВ – Публікація повідомлення – затвердження повідомлення при задовільному результаті;

— ВВ – Повідомлення видалено – видалення повідомлення при негативному результаті.

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ ДОДАТКУ

4.1.Програмна реалізація

Реалізація програмного забезпечення – це процес розробки додатків із використанням обраних технологій та інструментів відповідно до плану та структури проекту.

На етапі практичної реалізації проходить підключення створених компонентів до методів перехоплення повідомлень чат-боту, створення обробників повідомлень користувача до чату. Для реалізації буде описано ключові компоненти даного процесу:

— пакети для розмежування програмних модулів;
— скрипти, як основний програмний код;
— допоміжні файли для чат-боту;
— підключення чат-бота до мережі «Telegram».

Загальний код чат-боту та компонентів можна знайти у додатку Б.

Visual Studio Code використовувався як текстовий редактор для програмної розробки боту.

Для початку нам потрібно показати використовуючи каталог ключові компоненти проекту, які ми розробили, це допоможе нам швидко знайти необхідні файли та інше, каталог проекту представлений на рисунку 4.1. Основні компоненти цієї програми розташовані в локальних пакетах, а пакети призначені для зберігання файлів .py формату.

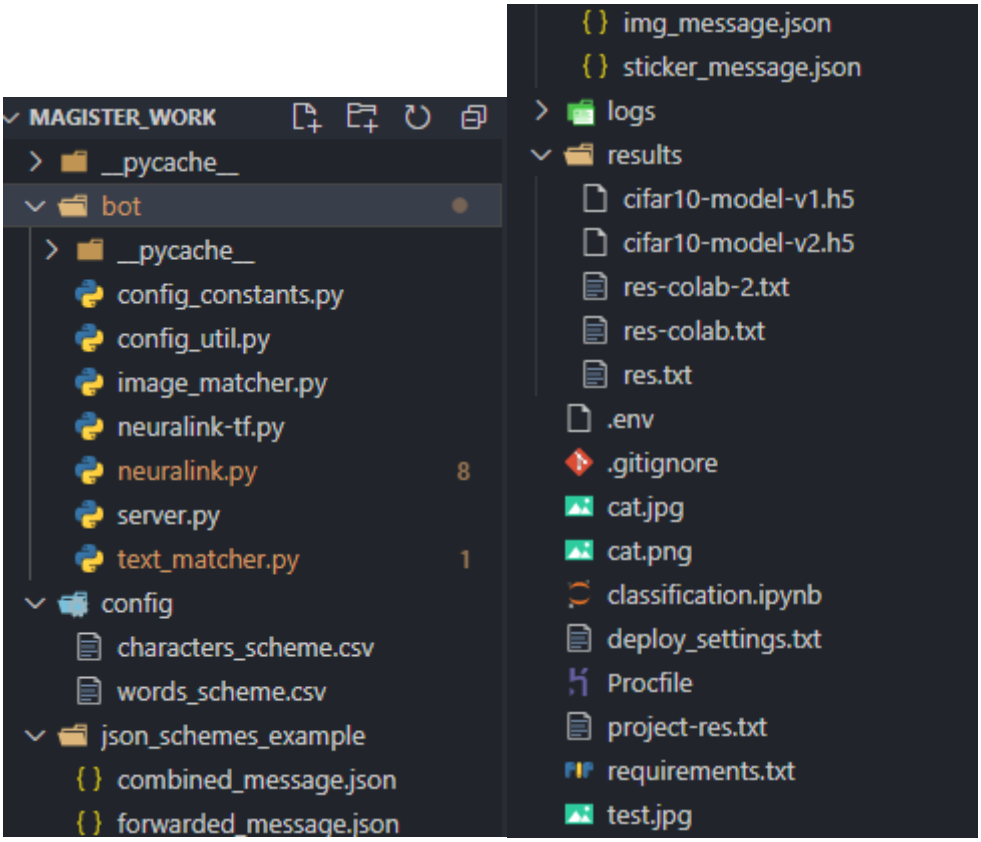


Рисунок 4.1 – Каталог проекту

Пакет ***bot*** – це пакет в якому знаходяться файли модулів чат-боту.

Пакет ***config*** – цей пакет містить .csv файли, які використовуються для налаштування боту.

Пакет ***results*** – в цьому пакеті знаходяться навчені моделі.

Для полегшення розуміння програмних складових чат-боту детально опишемо файли каталогу. У таблиці 4.1 описані файли проекту.

Таблиця 4.1 – Опис файлів у каталозі проекту

Пакет (и)	Файл (и)	Метод (и)
bot	***config_constants.py***	Константні значення для конфігурації
	config_util.py	***def get_words_scheme() –*** метод зчитує з .csv файлів схему слів

Продовження таблиці 4.1

Пакет (и)	Файл (и)	Метод (и)
bot	***config_util.py***	***def get_characters_scheme()*** – метод зчитує з .csv алфавіт для порівнянь
	image_matcher.py	***def check_image(image_path)*** – метод виклику перевірки зображення
	neuralink.py	***def load_data():*** – метод завантаження даних до мережі ***def create_model(input_shape)*** – метод створення моделі ***def start_learning()*** – метод початку навчання

Для навчання моделі підготуємо датасети з компоненту CIFER-10 за допомогою TensorFlow Datasets. Приклад завантаження представлений на рисунку 4.2.

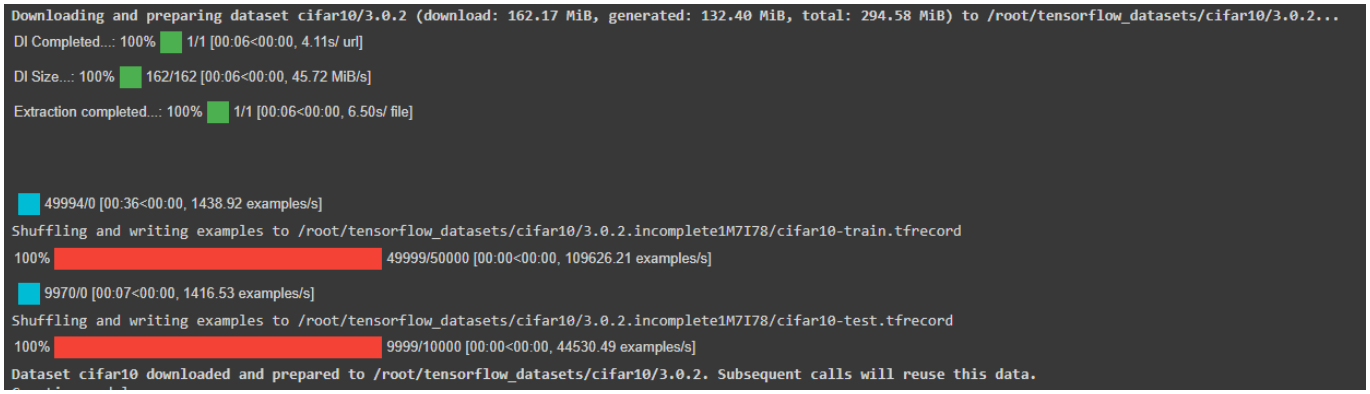


Рисунок 4.2 – Створення датасету

Далі створюємо модель та виводимо шари нейронної мережі. Готова структура мережі представлена на рисунку 4.3.

```
Creating model
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (None, 32, 32, 3)         0

 conv2d (Conv2D)             (None, 32, 32, 32)        896

 activation (Activation)     (None, 32, 32, 32)        0

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        9248

 activation_1 (Activation)   (None, 32, 32, 32)        0

 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
 )

 dropout (Dropout)           (None, 16, 16, 32)        0

 conv2d_2 (Conv2D)           (None, 16, 16, 64)        18496

 activation_2 (Activation)   (None, 16, 16, 64)        0

 conv2d_3 (Conv2D)           (None, 16, 16, 64)        36928

 activation_3 (Activation)   (None, 16, 16, 64)        0

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 dropout_1 (Dropout)         (None, 8, 8, 64)          0

 conv2d_4 (Conv2D)           (None, 8, 8, 128)         73856

 activation_4 (Activation)   (None, 8, 8, 128)         0

 conv2d_5 (Conv2D)           (None, 8, 8, 128)         147584
```

Рисунок 4.3 – Створення нейронної мережі

```
=================================================================
Total params: 2,395,434
Trainable params: 2,395,434
Non-trainable params: 0
```

Рисунок 4.4 – Сумарна кількість параметрів у моделі

Далі починаємо навчання нашої моделі з використанням підготовлених даних.

На рисунку 4.5 представлено стектрейс процесу навчання мережі та валідації даних.

```
_________________________________________________________________
Epoch 1/100
781/781 [==============================] - 90s 115ms/step - loss: 1.6481 - accuracy: 0.3872 - val_loss: 1.2412 - val_accuracy: 0.5576
Epoch 2/100
781/781 [==============================] - 90s 116ms/step - loss: 1.1685 - accuracy: 0.5806 - val_loss: 1.0007 - val_accuracy: 0.6512
Epoch 3/100
781/781 [==============================] - 89s 115ms/step - loss: 0.9808 - accuracy: 0.6530 - val_loss: 0.8218 - val_accuracy: 0.7138
Epoch 4/100
781/781 [==============================] - 89s 115ms/step - loss: 0.8683 - accuracy: 0.6952 - val_loss: 0.7773 - val_accuracy: 0.7332
Epoch 5/100
781/781 [==============================] - 89s 114ms/step - loss: 0.7889 - accuracy: 0.7212 - val_loss: 0.7410 - val_accuracy: 0.7398
Epoch 6/100
781/781 [==============================] - 89s 114ms/step - loss: 0.7353 - accuracy: 0.7436 - val_loss: 0.7191 - val_accuracy: 0.7533
Epoch 7/100
781/781 [==============================] - 90s 115ms/step - loss: 0.6899 - accuracy: 0.7557 - val_loss: 0.6604 - val_accuracy: 0.7739
Epoch 8/100
781/781 [==============================] - 88s 113ms/step - loss: 0.6538 - accuracy: 0.7709 - val_loss: 0.6348 - val_accuracy: 0.7837
Epoch 9/100
781/781 [==============================] - 88s 113ms/step - loss: 0.6255 - accuracy: 0.7826 - val_loss: 0.6342 - val_accuracy: 0.7870
Epoch 10/100
781/781 [==============================] - 87s 111ms/step - loss: 0.6002 - accuracy: 0.7900 - val_loss: 0.6254 - val_accuracy: 0.7898
Epoch 11/100
781/781 [==============================] - 88s 112ms/step - loss: 0.5801 - accuracy: 0.7961 - val_loss: 0.6063 - val_accuracy: 0.7971
Epoch 12/100
781/781 [==============================] - 89s 114ms/step - loss: 0.5637 - accuracy: 0.8030 - val_loss: 0.6256 - val_accuracy: 0.7921
Epoch 13/100
781/781 [==============================] - 89s 114ms/step - loss: 0.5469 - accuracy: 0.8084 - val_loss: 0.6131 - val_accuracy: 0.7923
Epoch 14/100
781/781 [==============================] - 88s 113ms/step - loss: 0.5295 - accuracy: 0.8138 - val_loss: 0.5954 - val_accuracy: 0.8013
Epoch 15/100
781/781 [==============================] - 88s 113ms/step - loss: 0.5168 - accuracy: 0.8181 - val_loss: 0.6435 - val_accuracy: 0.7912
Epoch 16/100
781/781 [==============================] - 88s 113ms/step - loss: 0.5057 - accuracy: 0.8220 - val_loss: 0.5928 - val_accuracy: 0.8061
Epoch 17/100
781/781 [==============================] - 88s 113ms/step - loss: 0.4953 - accuracy: 0.8253 - val_loss: 0.5967 - val_accuracy: 0.8082
Epoch 18/100
781/781 [==============================] - 90s 115ms/step - loss: 0.4811 - accuracy: 0.8302 - val_loss: 0.5964 - val_accuracy: 0.8064
Epoch 19/100
781/781 [==============================] - 89s 114ms/step - loss: 0.4798 - accuracy: 0.8327 - val_loss: 0.5754 - val_accuracy: 0.8140
Epoch 20/100
781/781 [==============================] - 90s 116ms/step - loss: 0.4660 - accuracy: 0.8357 - val_loss: 0.5903 - val_accuracy: 0.8100
Epoch 21/100
781/781 [==============================] - 90s 115ms/step - loss: 0.4580 - accuracy: 0.8372 - val_loss: 0.5975 - val_accuracy: 0.8091
Epoch 22/100
781/781 [==============================] - 91s 116ms/step - loss: 0.4606 - accuracy: 0.8381 - val_loss: 0.5900 - val_accuracy: 0.8104
Epoch 23/100
781/781 [==============================] - 91s 116ms/step - loss: 0.4474 - accuracy: 0.8421 - val_loss: 0.5914 - val_accuracy: 0.8135
```

Рисунок 4.5 – Навчання моделі

По стектрейсу можна побачити, що з кожною епохою коефіцієнт похибки зменшується, а коефіцієнт вдачі збільшується. Більш детально про прогрес навчання можна дізнатися з графіків представлених на рисунках 4.6-4.7.

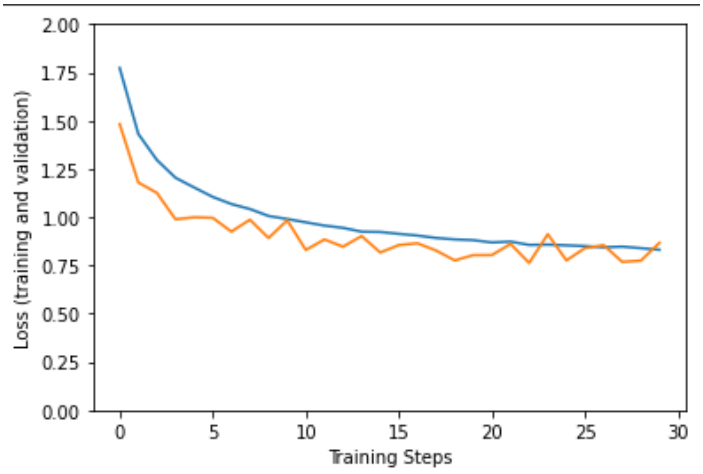


Рисунок 4.6 – Коефіцієнт похибки під час навчання моделі

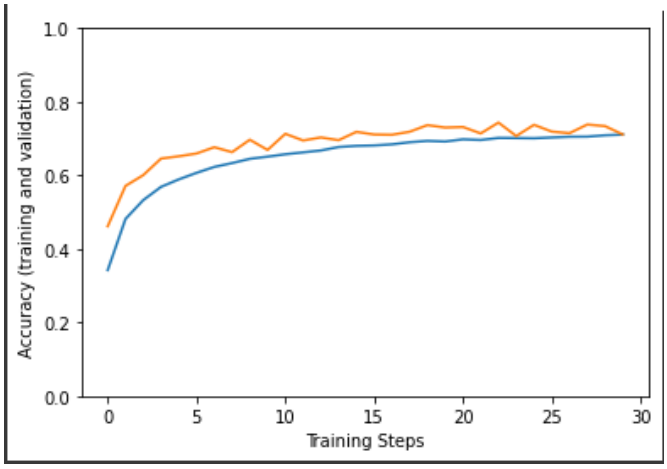


Рисунок 4.7 – Коефіцієнт вдачі під час навчання моделі

Результатом навчання є файл *cifar10-model-v1.h5* модель збережена у форматі .h5. Результат можна побачити на рисунку 4.8.

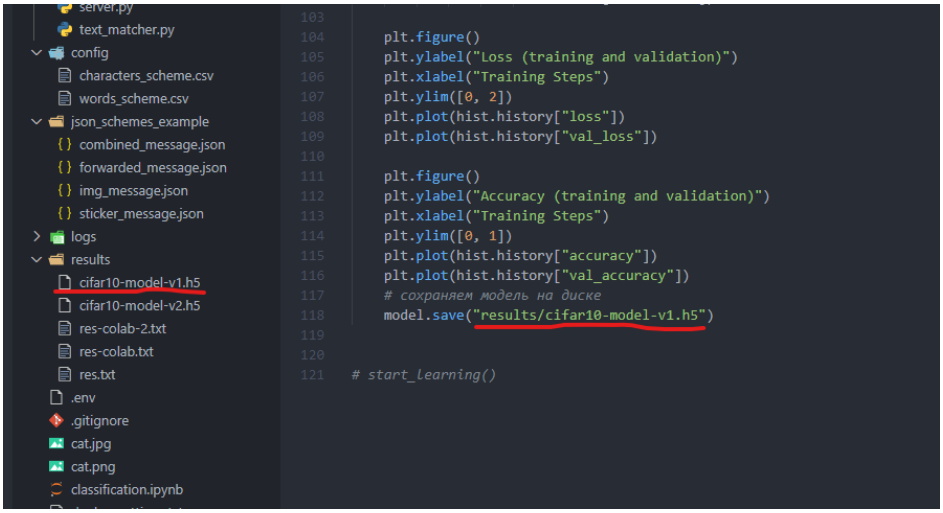


Рисунок 4.8 – Збереження результату навчання моделі

Далі необхідно інтегрувати алгоритм Левенштейна для класифікації тексту.

Створимо алфавіт літер, які можуть бути навмисно спотворені та словник слів, які можуть бути використані для порівняння у класифікації. Ці дані для зручності були збережені в csv файли, з яких потім завантажуються до програми. Приклади цих файлів представлені на рисунках 4.9-4.10.

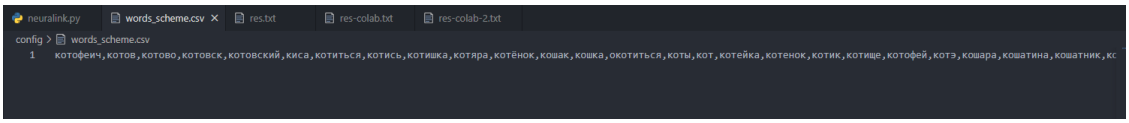


Рисунок 4.9 – Словник класифікаційних слів

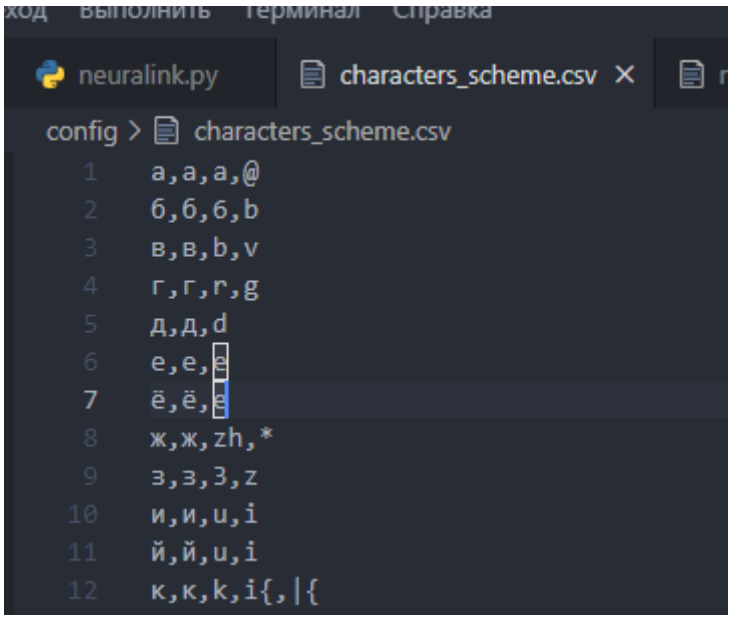


Рисунок 4.10 – Алфавіт з спотвореними літерами

Наступним етапом будуємо алгоритм та тестуємо його. Приклад алгоритму та результат тестування представлений на рисунку 4.11-4.12.

```python
def distance(a, b):
    "Calculates the Levenshtein distance between a and b."
    n, m = len(a), len(b)
    if n > m:
        # Make sure n <= m, to use O(min(n, m)) space
        a, b = b, a
        n, m = m, n

    # Keep current and previous row, not entire matrix
    current_row = range(n + 1)
    for i in range(1, m + 1):
        previous_row, current_row = current_row, [i] + [0] * n
        for j in range(1, n + 1):
            add, delete, change = previous_row[j] + \
                1, current_row[j - 1] + 1, previous_row[j - 1]
            if a[j - 1] != b[i - 1]:
                change += 1
            current_row[j] = min(add, delete, change)

    return current_row[n]
```

Рисунок 4.11 – Алгоритм обчислення відстані

```
                count += 1

    return count > 0

check_text('Когда я играю со своей кошкой, кто знает, не забавляется ли скорее она мною, нежели я ею.')
```

```
Найдено кошкой
Похоже на кошкой
True
```

Рисунок 4.12 – Тестування алгоритму

Далі створимо телеграм-бот та зареєструємо його у мережі. Для створення телеграм-боту потрібно знайти бота @BotFather , написати йому команди /start або /newbot , заповнити поля, які він запитає (назва бота і його коротке ім'я), і отримати повідомлення з токеном бота і посиланням на документацію. Токен потрібно зберегти, бажано надійно, тому що це єдиний ключ для авторизації робота і взаємодії з ним. Процес створення боту представлений на рисунку 4.13.

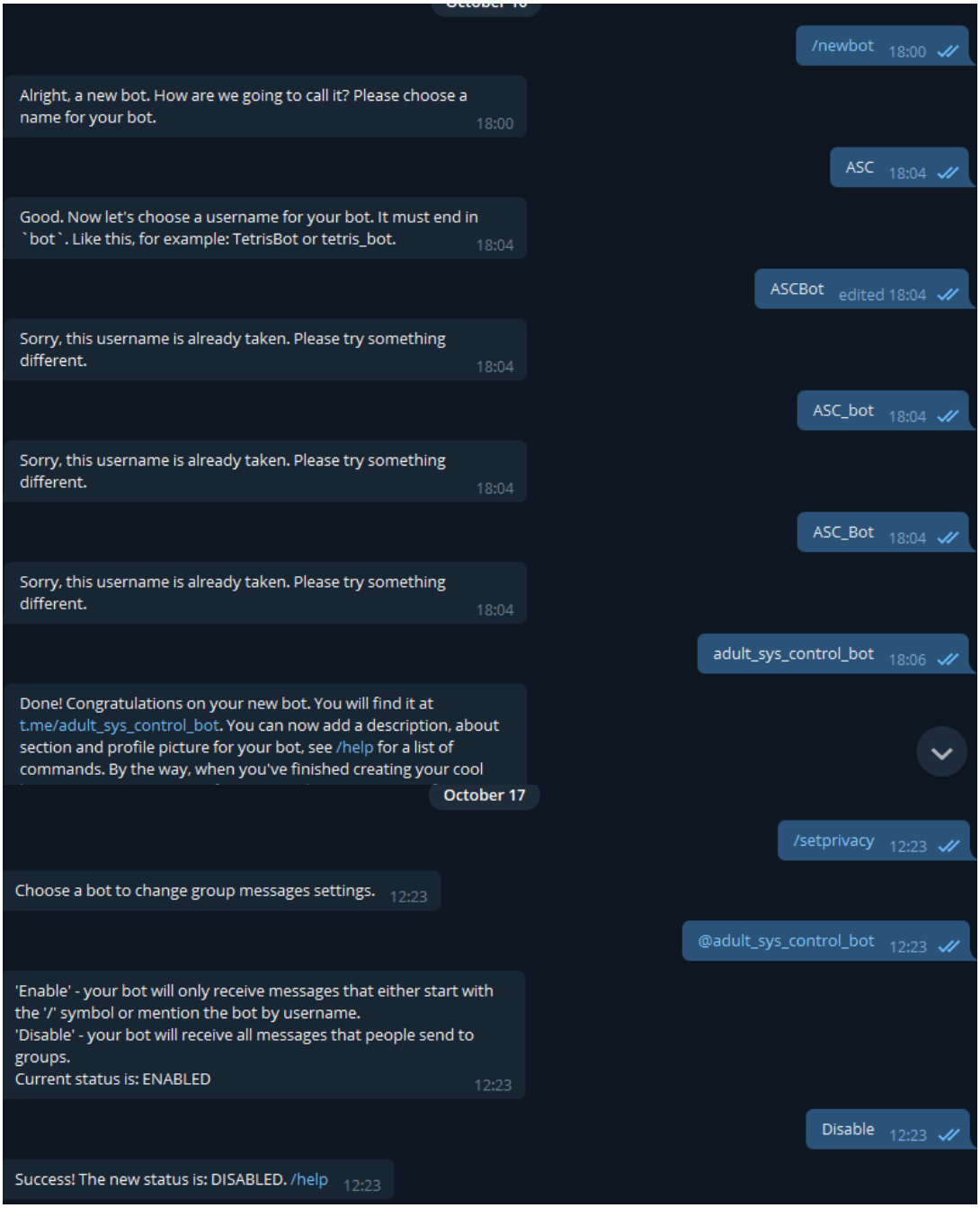


Рисунок 4.13 – Реєстрація боту

Додамо обробники повідомлень до телеграм боту для обробки повідомлень у групових чатах, каналах та приватних чатах. Створимо нотифікацію адміністраторів групи при виявленні невідповідного контенту та поєднаємо модель та нашого бота.

Обробник повідомлень представлений на рисунку 4.14.

```python
def notify_admins(chat_id, message):
    admins = bot.get_chat_administrators(chat_id).wait()
    for admin in admins:
        bot.send_message(admin.user.id, message)


@bot.message_handler(commands=['start'])
def startCmd(message):
    start_learning()
    bot.reply_to(message, "Hey, let's start")

@bot.message_handler(chat_types=['private'])
def command_help(message):
    check_image_result = False
    check_text_result = False
    invalid_text_segment = ''

    message_text = message.caption
    message_photo = message.photo
    message_sticker = message.sticker

    if message_text:
        check_text_result = check_text(message_text)

    if check_text_result:
        invalid_text_segment = "Text not valid"
        bot.delete_message(message.chat.id, message.message_id)
        result_text = "Received message with adult content from user: " + message.from_user.firs
            "(@" + message.from_user.username + \
            "): [" + invalid_text_segment + \
                     "]. We notified admin's of the current chat."
        notify_admins(message.chat.id, result_text)

    if (not check_text_result) and message_photo:
        img_data = bot.get_file(message_photo[-1].file_id).wait()
```

Рисунок 4.14 – Приклад обробнику повідомлень

4.2.Використання програмного додатку

Під час даного етапу демонструється робота чат-боту на прикладі тестового чату.

Для початку надішлемо зображення вантажівки до чату та подивимось на реакцію нашого боту.

Рисунок 4.15 – Надсилання вантажівки до чату

Нижче наведено результат аналізу зображення, як бачимо модель класифікувала зображення спершу вірно.

```
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
156/156 [==============================] - 4s 23ms/step - loss: 0.6328 - accuracy: 0.8149
Тестовая оценка: 81.48885369300842 %
Predicted label: truck
Predicted: 9
True label: cat
```

Рисунок 4.16 – Аналіз зображення

Спробуємо надіслати зображення кота, яке є тригером для нашого бота та повинне бути видалене з нотифікацією адміністраторів, та переглянемо результат.

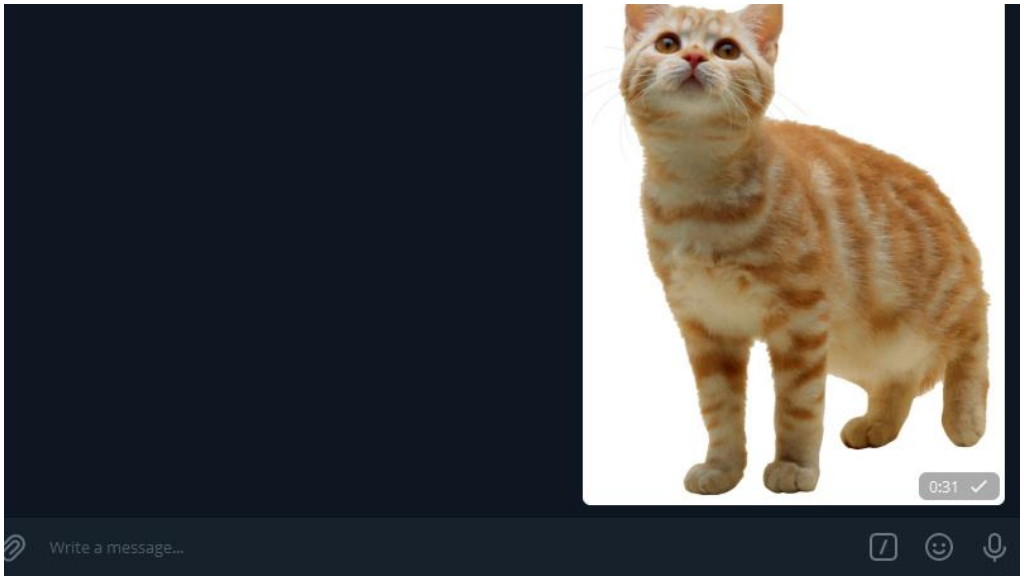


Рисунок 4.17 – Надсилаємо зображення кота

Модель класифікувала кота, провела видалення повідомлення та нотифікувала адміністратора чату.

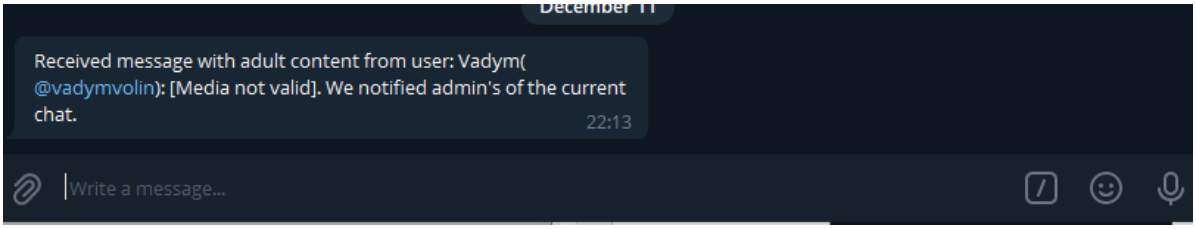


Рисунок 4.18 – Отримана нотифікація

Спробуємо повторити процес з новим зображенням та переглянемо результат аналізу зображення.

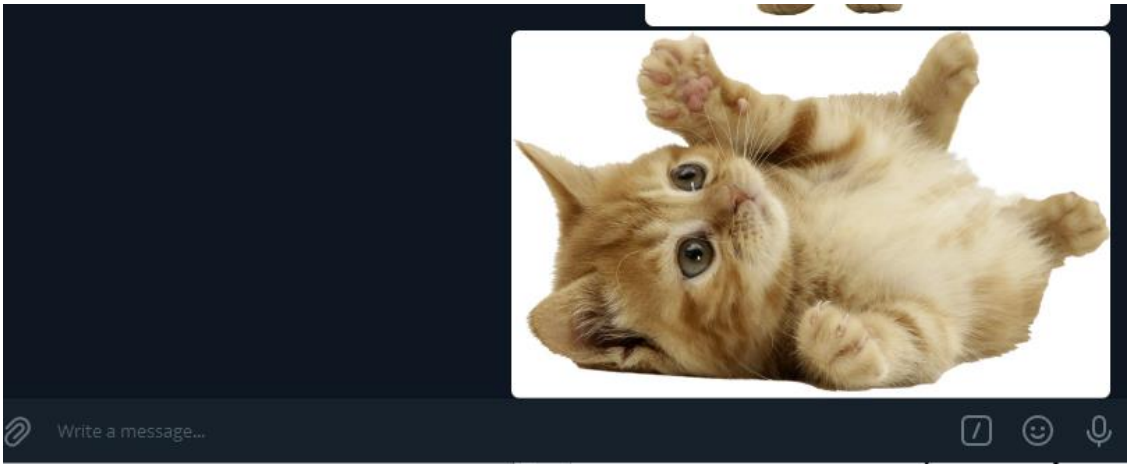


Рисунок 4.19 – Надіслане зображення

```
Start loading data
156/156 [==============================] - 4s 23ms/step - loss: 0.8511 - accuracy: 0.7196
Тестовая оценка: 71.96456789970398 %
Predicted label: horse
Predicted: 7
True label: cat
```

Рисунок 4.20 – Результат аналізу

Як можна побачити, мережа ідентифікувала зображення невірно, з цього необхідно зробити висновок, що модель потрібно тренувати на більшій кількості даних та погіршити умови тренувань завдяки модифікації зображень.

Наступним етапом протестуємо класифікацію тексту у чаті. Приклад та результат тестування представлений на рисунках 4.21-4.22.

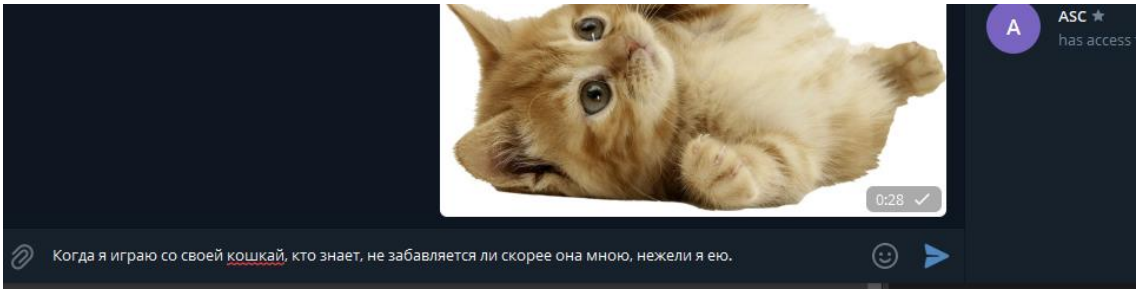


Рисунок 4.21 – Надіслане зображення

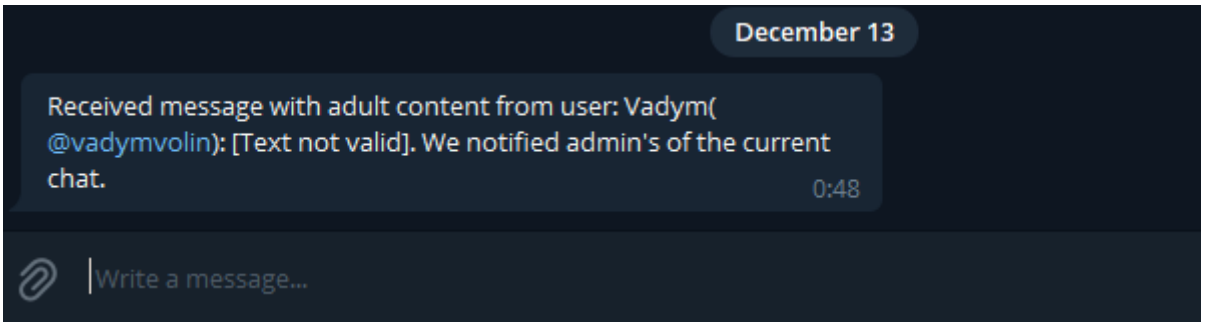


Рисунок 4.22 – Результат аналізу

Даний алгоритм аналізує текст без затримки та миттєво надсилає результат до адміністраторів групи на відміну від навчаної моделі для класифікації зображень, яка має затримку у середньому на 3-4 секунди у залежності від потужності серверу.

ВИСНОВКИ

При створенні кваліфікаційної роботи магістра було проведено аналіз досліджень в області класифікації зображень та текстового аналізу.

Знайдено аналогічні програмні чат-боти для аналізу та дослідження їх функціоналу, щоб виділити переваги і недоліки у порівнянні з нашим ботом.

Після аналізу аналогів було проведено формування проблеми, як постановки головної задачі, описаний основний функціонал чат-бота та підібрано технології та засоби реалізації.

Для підтримки структурованості та систематизації чат-боту було проведено детальне проектування чат-бота.

Спершу створена структурно-функціональна модель і контекстні діаграми, щоб забезпечити повний огляд ключових робочих процесів.

Було створено та проведено навчання моделі класифікації зображень на прикладі датасетів CIFAR-10.

Також було проведено побудову діаграми варіантів використання даного чат-боту та детально описує послідовність дій.

Під час реалізації було проаналізовано процес навчання нейронної мережі на обмеженому датасеті з обмеженим технічним обладнанням. Також було визначено, що використання моделі у поєднанні з чат-ботом може збільшити час затримки між надходженням повідомлення та прийняттям рішень через процес класифікації зображення у моделі.

Було реалізовано алгоритм знаходження відстані Левенштейна, що дозволив проводити аналіз тексту, та знаходити відповідні частини з умисними відхиленнями.

З реалізованими компонентами було проведено програмну реалізацію чат-боту, який повністю відповідає предметній області та відповідає висунутим вимогам.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Modrzyk N. Building telegram bots: develop bots in 12 programming languages using the telegram bot API / Nicolas Modrzyk. – [S. l.] : Apress, 2018. – 296 p.

2. Доценко С. І. Про природний та штучний інтелект кібернетичних систем [Електронний ресурс] / С. І. Доценко // Radioelectronic and computer systems. – 2019. – № 3. – С. 4–18. – Режим доступу: https://doi.org/10.32620/reks.2019.3.01 (дата звернення: 24.11.2021). – Назва з екрана.

3. Пчелянський Д. П. Штучний інтелект: перспективи та тенденції розвитку [Електронний ресурс] / Д. П. Пчелянський, С. А. Воінова // Automation of technological and business processes. – 2019. – Т. 11, № 3. – С. 59–64. – Режим доступу: https://doi.org/10.15673/atbp.v11i3.1500 (дата звернення: 24.11.2021). – Назва з екрана.

4. Чубатюк Ю. "Штучний інтелект" змінить ринок праці / Ю. Чубатюк // День. – 2018. – 4 лип. – С. 8.

5. Averkyna M. Artificial intelligence in business: advantages and obstacles in implementation [Electronic resource] / Maryna Averkyna, Liubov Sydorchuk // Market infrastructure. – 2019. – No. 37. – Mode of access: https://doi.org/10.32843/infrastruct37-23 (date of access: 24.11.2021). – Title from screen.

6. Chollet F. Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek / François Chollet. – [S. l.] : MITP Verlags GmbH, 2018.

7. Chollet F. Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek / François Chollet. – [S. l.] : MITP Verlags GmbH, 2018.

8. Cussens J. Machine learning [Electronic resource] / J. Cussens // Computing & control engineering journal. – 1996. – Vol. 7, no. 4. – P. 164–168. – Mode of access: https://doi.org/10.1049/cce:19960402 (date of access: 24.11.2021). – Title from screen.

9. Derevianko S. Artificial intelligence and emotional artificial intelligence as phenomena of modern cognitive psychology [Electronic resource] / Svitlana Derevianko,

Yulia Prymak, Irina Yushchenko // Scientific notes of ostroh academy national university: psychology series. – 2020. – Vol. 1. – P. 115–119. – Mode of access: https://doi.org/10.25264/2415-7384-2020-11-115-119 (date of access: 24.11.2021). – Title from screen.

10. Karim M. R. TensorFlow: Powerful Predictive Analytics with TensorFlow: Predict valuable insights of your data with TensorFlow / Md Rezaul Karim. – [S. l.] : Packt Publishing, 2018. – 164 p.

11. Louridas P. Machine learning [Electronic resource] / Panos Louridas, Christof Ebert // IEEE software. – 2016. – Vol. 33, no. 5. – P. 110–115. – Mode of access: https://doi.org/10.1109/ms.2016.114 (date of access: 24.11.2021). – Title from screen.

12. Machine learning [Electronic resource] / Kaizhu Huang [et al.]. – Berlin, Heidelberg : Springer Berlin Heidelberg, 2008. – Mode of access: https://doi.org/10.1007/978-3-540-79452-3 (date of access: 24.11.2021). – Title from screen.

13. Mooney R. J. Machine learning [Electronic resource] / Raymond J. Mooney ; ed. by R. Mitkov. – [S. l.] : Oxford University Press, 2012. – Mode of access: https://doi.org/10.1093/oxfordhb/9780199276349.013.0020 (date of access: 24.11.2021). – Title from screen.

14. Nandy A. Reinforcement learning: with open AI, tensorflow and keras using python / Abhishek Nandy, Manisha Biswas. – [S. l.] : Apress, 2017. – 167 p.

15. Pattanayak S. Pro Deep Learning with TensorFlow [Electronic resource] / Santanu Pattanayak. – Berkeley, CA : Apress, 2017. – Mode of access: https://doi.org/10.1007/978-1-4842-3096-1 (date of access: 24.11.2021). – Title from screen.

16. Scarpino M. TensorFlow / Matthew Scarpino. – Hoboken, New Jersey : John Wiley & Sons, Inc., 2018. – 336 p.

17. Work breakdown structure (WBS) and WBS dictionary: prepared in accordance with DRD# 784MA-004. – Cambridge, MA : Smithsonian Astrophysical Observatory, 1994.

18. Aryanto D. V. Development of WBS dictionary and checklist based on WBS of railway construction for safety planning [Electronic resource] / D. V. Aryanto, L. S. R. Supriadi, Y. Latief // IOP conference series: materials science and engineering. – 2020. – Vol. 1007. – P. 012007. – Mode of access: https://doi.org/10.1088/1757-899x/1007/1/012007 (date of access: 24.11.2021). – Title from screen.

19. Institute P. M. A guide to the project management body of knowledge (pmbok guide): official russian translation / Project Management Institute. – [S. l.] : Project Management Institute, 2004. – 388 p.

20. Nicoletti P. Content filtering [Electronic resource] / Pete Nicoletti // Computer and information security handbook. – [S. l.], 2013. – P. 1073–1074. – Mode of access: https://doi.org/10.1016/b978-0-12-394397-2.00066-0 (date of access: 24.11.2021). – Title from screen.

21. Блог "АгроКебети Агрообразование будущего". Коммуникации внутри компании: что это, как работает и как улучшить коммуникации в организации [Електронний ресурс] / АгроКебети – Режим доступу до ресурсу: https://blog.agrokebety.com/kommunikatsii-vnutri-kompanii.

22. Chatbots. What are they? And how can you use them in conjunction with analytics? [Електронний ресурс] / Bolen A. – Режим доступу до ресурсу: https://www.sas.com/ru_ru/insights/articles/analytics/what-are-chatbots.html

23. What is Web Content Filtering? [Електронний ресурс] / CYBER EDU – Режим доступу до ресурсу: https://www.forcepoint.com/cyber-edu/web-content-fitlering

24. Офіційна документація для створення чат-боту у "Telegram" [Електронний ресурс] / Telegram – Режим доступу до ресурсу: https://core.telegram.org/bots/api#available-methods

25. Чат-бот "ChatKeeperBot" [Електронний ресурс] / ChatKeeperBot – Режим доступу до ресурсу: https://docs.chatkeeper.app/ru/instruction/#chatkeeperbot

26. Чат-бот "WatchDog_Robot" [Електронний ресурс] / WatchDog – Режим доступу до ресурсу: https://tgdev.io/bot/watchdog_robot

27. Чат-бот "DaySandBox" [Електронний ресурс] / DaySandBox – Режим доступу до ресурсу: https://tgdev.io/bot/daysandbox_bot

28. Офіційна документация бібліотеки для аналізу даних Pandas [Електронний ресурс] / Pandas – Режим доступу до ресурсу: https://pandas.pydata.org/docs/reference/index.html

29. Джанарсанам С. Розробка чат-ботів та розмовних інтерфейсів / Сріні Джанарсанам., 2016. – 340 с.

30. Офіційна документация бібліотеки для аналізу даних PyTorch.text [Електронний ресурс] / Pandas – Режим доступу до ресурсу: https://pytorch.org/text/stable/index.html

31. Офіційна документация бібліотеки для аналізу даних NumPy [Електронний ресурс] / NumPy – Режим доступу до ресурсу: https://numpy.org/learn/

32. Офіційна документация бібліотеки для аналізу даних TensorFlow [Електронний ресурс] / TensorFlow – Режим доступу до ресурсу: https://www.tensorflow.org/api_docs/python/tf

33. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython / Wes McKinney, 2016. – 450 с. – (вип. 2).

34. VanderPlas J. Python Data Science Handbook: Essential Tools for Working with Data / Jake VanderPlas., 2016. – 460 с. – (вип. 1).

35. Trask A. Grokking Deep Learning. / Andrew Trask., 2020. – 320 с.

36. Gugger S. Deep Learning for Coders with fastai and PyTorch: AI Applications Without a PhD / S. Gugger, J. Howard., 2020.

ДОДАТОК А

ПЛАНУВАННЯ РОБІТ

Деталізація мети проекту методом SMART. Мета проекту: є розробка інтелектуального чат-боту з елементами штучного інтелекту, який буде автоматизувати процес модерації чату від недоцільного контенту за допомогою використання штучного інтелекту [5, 6, 7]. Мета є досяжною, підґрунтям створення додатку є навички отримані на курсі з дисципліни Intoduction to Data Science, ООР, веб-розробка, Системи штучного інтелекту. Проект буде виконано вчасно, що підтверджується календарним планом проекту. Результати деталізації методом SMART розміщені у табл. А.1.

Таблиця А.1 − Деталізація мети методом SMART

Specific (конкретна)	Створити інтелектуальний чат-бот з елементами штучного інтелекту.
Measurable (вимірювання)	Результатом роботи проекту є робочий інтелектуальний чат-бот з виконаними поставленими технічними вимогами.
Achievable (досяжна, узгоджена)	Ціль створення чат-боту є цілковито реальною, бо розробник має відповідні навички та знання, а структура проекту погоджена з дипломним керівником.
Relevant (реалістична)	Цілі проекту є реалістичними та досяжними, також, розробник має необхідне програмне та технічне забезпечення для реалізації проекту.
Time-framed (обмежена в часі)	Виконання проекту відбувається поступово, дотримуючись строків, запланованих згідно з календарним планом

Планування змісту структури робіт. Зручним інструментом для планування робіт є WBS діаграма − представлення проекту. Діаграма виконуєтьсяму вигляді ієрархічної структури робіт, яка досягається за побудови послідовної декомпозиції [17, 18].

Як правило, на верхньому рівні вказується сам проект, під ним основні результати, в свою чергу, кожен результат деталізується, наступний рівень завжди менше попереднього за обсягом робіт, зазвичай, включає 2 і більше елементів робіт [18, 19].

Під час планування етапу розробки чат-боту було створено WBS діаграми, де перший рівень структури має назву – «Інтелектуальний чат-бот для фільтрації контенту телеграм-каналу компанії», який поділяється на декілька гілок декомпозиції. Повна структура WBS наведена на рисунку А.1.

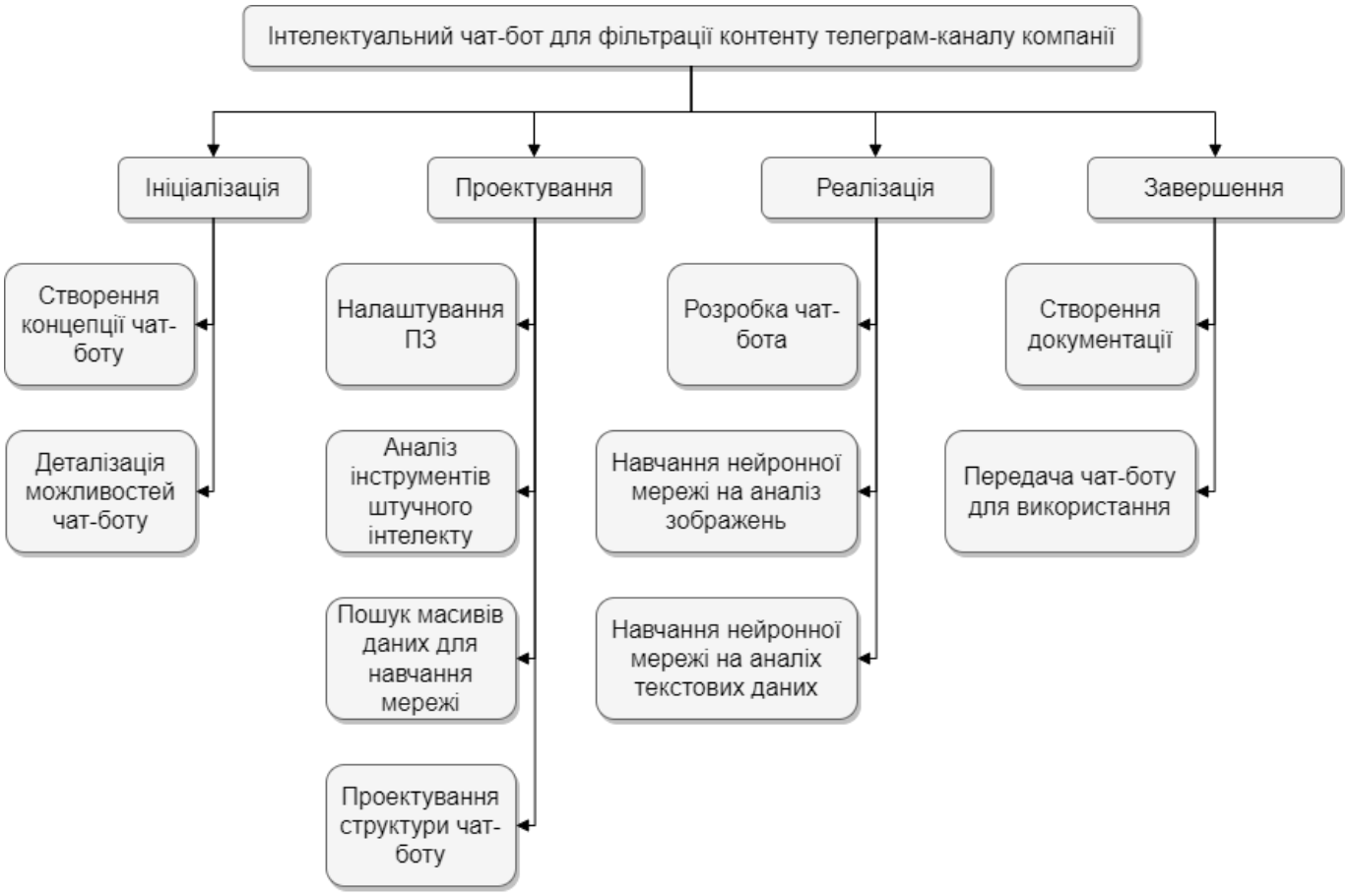


Рисунок А.1 – WBS структура проекту

Після того, як була побудована WBS структура проекту наступним етапом є розроблення OBS - склад, підпорядкованість, взаємодія і розподіл робіт по підрозділах і органам управління, між якими встановлюються певні відносини з приводу реалізації владних повноважень, потоків команд і інформації. Організаційна структура проекту стосується тільки внутрішньої організаційної структури проекту і не стосується відносин проектних груп чи учасників з батьківськими організаціями [18, 19].

Список виконавців, що функціонують в проекті представлений в таблиці А.2 та на рисунку А.2.

Таблиця А.2 – Виконавці проекту

Роль	Ім’я	Проектна роль
Виконавець	Волін В.В.	Виконує розробку функціоналу системи та її інтерфейсчат-боту.
Керівник проекту	Ващенко С.М.	Формує завдання на виконання проекту, корегує процес створення проекту та перевіряє виконання.

Діаграма OBS представлена на рисунку А.2.

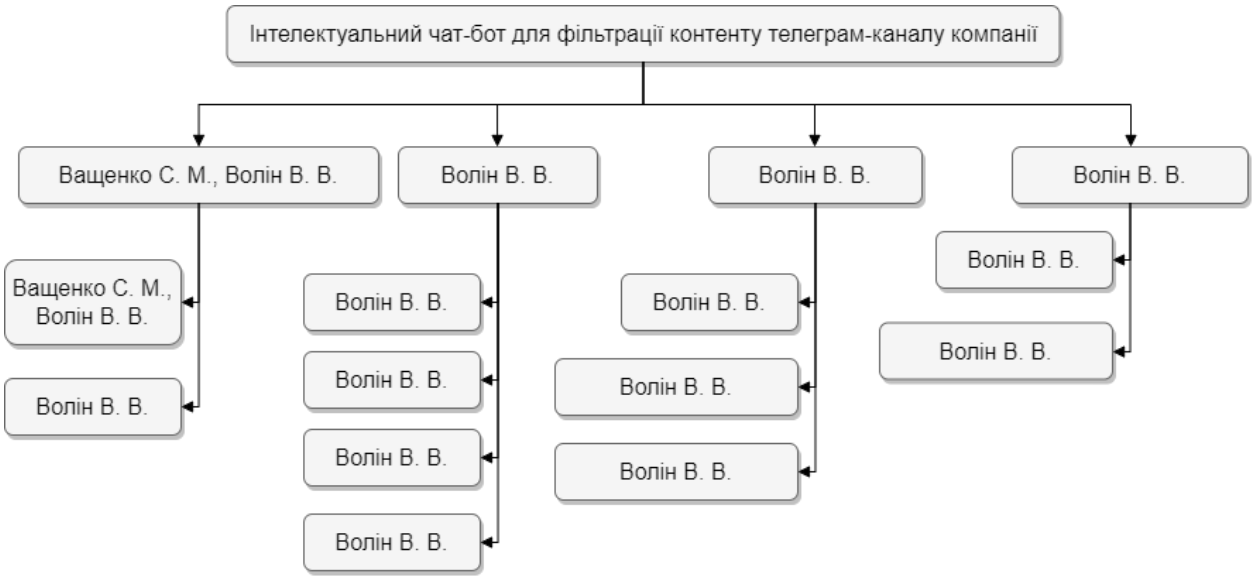


Рисунок А.2 – OBS структура проекту

Діаграма Ганта. Діаграма Ганта вважається якісним інструментом для відображення цілей та задач. Основна відмінність цих інструментів - в способі демонстрації і відображення інформації.

На відміну від PDM – мережі, що пропонує мережеву модель, управління проектами з діаграмами Ганта засноване на форматі гістограм. Це допомагає відслідковувати відсоток робіт, виконаних по кожному завданню. Керівникам проектів дуже важливо правильно розподілити завдання і бути впевненими в тому, що проект буде завершений вчасно.

Основна увага діаграм Ганта зосереджено на процентному завершенні кожного завдання. Крім того, діаграми Ганта краще для проектів з невеликою кількістю взаємопов'язаних завдань.

Завдяки засобам програмного продукту Smartsheet була розроблена діаграма Ганта, яка у вигляді гістограми відображає тривалість кожного процесу, що був визначений на етапі формування WBS. Діаграма Ганта представлена на рисунку А.3.

Имя задачи	Длительнос	Начало	Готово	Предшест
Ініціалізація	7д	26.08.21	03.09.21	
Створення концепції ча	4д	26.08.21	31.08.21	
Деталізація можливост	3д	01.09.21	03.09.21	2
Проектування	28д	06.09.21	13.10.21	
Налаштування ПЗ	3д	06.09.21	08.09.21	3
Аналіз інструментів шт	7д	09.09.21	17.09.21	5
Пошук масивів даних дл	4д	20.09.21	23.09.21	6
Проектування структур	14д	24.09.21	13.10.21	7
Реалізація	34д	14.10.21	30.11.21	
Розробка чат-бота	14д	14.10.21	02.11.21	8
Навчання нейронної ме	10д	03.11.21	16.11.21	10
Навчання нейронної ме	10д	17.11.21	30.11.21	11
Завершення	9д	01.12.21	13.12.21	
Створення документаці	8д	01.12.21	10.12.21	12
Передача чат-боту для	1д	13.12.21	13.12.21	14

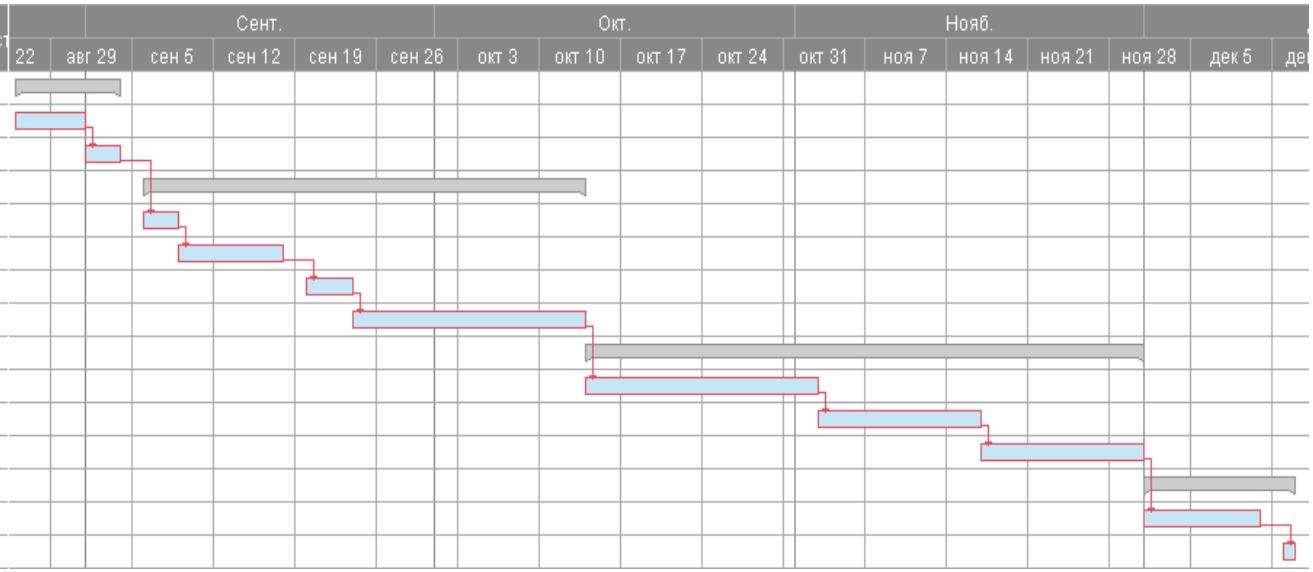


Рисунок А.3 – Діаграма Ганта проекту

Аналіз ризиків. Ризик – ймовірнісна подія, яка може позитивно чи негативно вплинути на проект. Причиною виникнення ризиків є невизначеності, існуючі в кожному проекті. Ризики можуть бути «відомі» - ті, які визначені, оцінені, для яких можливе планування. Ризики «невідомі» - ті, які не ідентифіковані і не можуть бути прогнозовані. Хоча специфічні ризики і умови їх виникнення не визначені, але більшу частину ризиків можна передбачити.

Ідентифікація ризиків - визначення ризиків, здатних вплинути на проект, і документування їх характеристик.

Ідентифікація ризиків визначає, які ризики здатні вплинути на проект, і документує характеристики цих ризиків. Ідентифікація ризиків не буде ефективною, якщо вона не буде проводитися регулярно протягом реалізації проекту.

Ідентифікація ризиків повинна залучати якомога більше учасників: менеджерів проекту, замовників, користувачів, незалежних фахівців.

Класифікація ризиків:

1. За імовірністю виникнення:
   - слабо ймовірнісні (Negligible) ;
   - мало ймовірнісні (Low);
   - імовірні (Medium);
   - досить імовірні (High);
   - майже імовірні (Critical).
2. За величиною впливу:
   - Мінімальна (Negligible);
   - Низька (Low);
   - Середня (Medium);
   - Висока (High);
   - Максимальна (Critical).

Після проведеного аналізу виявлено деякі ризики, які можливі при створенні даного проекту:

- R1 – Невідповідність вимогам;
- R2 – Недотримання термінів календарного плану;

— R3 – Проблеми з апаратно-програмним забезпеченням;
— R4 – Халатне тестування чат-боту;
— R5 – Людський фактор;
— R6 – Перепланування проекту під час реалізації;
— R7 – Непередбачувані обставини.

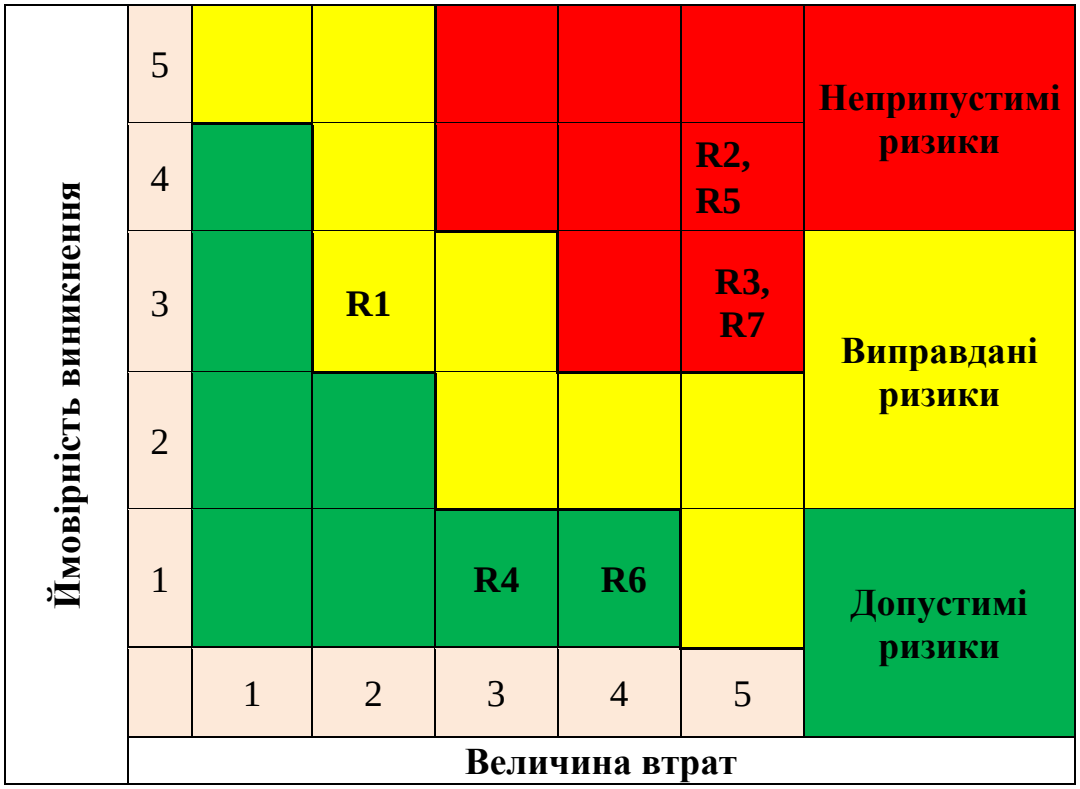
На основі цих даних була проведена класифікація ризиків для даного проекту, що наведена в таблиці А.3.

Таблиця А.3 – Класифікація ризиків

№	Назва ризику	Ймовірність	Величина втрат
1	Невідповідність вимогам	3	2
2	Недотримання термінів календарного плану	4	5
3	Проблеми з апаратно-програмним забезпеченням	3	5
4	Халатне тестування чат-боту	1	3
5	Людський фактор	4	5
6	Перепланування проекту під час реалізації	1	4
7	Непередбачувані обставини	3	5

Визначену вище інформацію було для побудови матриці ризику, для відображення відношення ризиків до відповідних категорій їх класифікації, розподіл представлений в таблиці А.4.

Таблиця А.4 - Матриця ризиків



Далі необхідно визначити рівні для ризиків та ступінь для їх дії.

Рівні поділяються на:

– допустимі 1<R<4;
– виправдані 5<R<10;
– недопустимі 11<R<25.

Ступені дії ризиків поділяються на:

– ті, що можна проігнорувати 1<R<4;
– незначні 5<R<8;
– помірні 9<R<10;
– істотні 11<R<16;
– критичні 17<R<25.

На основі цих даних виконаємо оцінку для ступенів та рівнів кожного ризику в проекті. Результати роботи представлені в таблиці А.5.

Таблиця А.5 – Оцінка ступенів та рівнів для кожного ризику

№	Ризик	Ймовір-ність	Величина втрат	Індекс ризику	Рівень ризику	Ступінь дії
1	Невідповідність вимогам	3	2	6	Виправданий	Незначний
2	Недотримання термінів календарного плану	4	5	24	Неприпустимий	Істотний
3	Проблеми з апаратно-програмним забезпеченням	3	5	15	Неприпустимий	Істотний
4	Халатне тестування чат-боту	1	3	3	Допустимий	Проігнорувати
5	Людський фактор	4	5	20	Неприпустимий	Істотний
6	Перепланування проекту під час реалізації	1	4	4	Допустимий	Проігнорувати
7	Непередбачувані обставини	3	5	15	Неприпустимий	Істотний

ДОДАТОК Б

Б.1 Побудова чат-боту

```
import os
import telebot

from neuralink import start_learning
from image_matcher import check_image
from text_matcher import check_text
from config_constants import BASE_IMAGE_URL

API_KEY = os.getenv('API_KEY')
bot = telebot.AsyncTeleBot(API_KEY)


def notify_admins(chat_id, message):
    admins = bot.get_chat_administrators(chat_id).wait()
    for admin in admins:
        bot.send_message(admin.user.id, message)


@bot.message_handler(commands=['start'])
def startCmd(message):
    start_learning()
    bot.reply_to(message, "Hey, let's start")

@bot.message_handler(chat_types=['private'])
def command_help(message):
    check_image_result = False
    check_text_result = False
    invalid_text_segment = ''

    message_text = message.caption
    message_photo = message.photo
    message_sticker = message.sticker

    if message_text:
        check_text_result = check_text(message_text)

    if check_text_result:
        invalid_text_segment = "Text not valid"
        bot.delete_message(message.chat.id, message.message_id)
        result_text = "Received message with adult content from user: " + message.from_user.first_name + \
            "(@" + message.from_user.username + \
            "): [" + invalid_text_segment + \
                    "]. We notified admin's of the current chat."
        notify_admins(message.chat.id, result_text)
```

```
    if (not check_text_result) and message_photo:
        img_data = bot.get_file(message_photo[-1].file_id).wait()
        img_url = BASE_IMAGE_URL.format(API_KEY, img_data.file_path)
        check_image_result = check_image(img_url)

    if (not check_text_result) and message_sticker:
        img_data = bot.get_file(message_sticker.file_id).wait()
        img_url = BASE_IMAGE_URL.format(API_KEY, img_data.file_path)
        check_image_result = check_image(img_url)

    if check_image_result:
        bot.delete_message(message.chat.id, message.message_id)
        invalid_text_segment = "Media not valid"
        result_text = "Received message with adult content from user: " + message.from_user.first_name + \
            "(@" + message.from_user.username + \
            "): [" + invalid_text_segment + \
                    "]. We notified admin's of the current chat."
        notify_admins(message.chat.id, result_text)


@bot.message_handler(func=lambda message: True, content_types=['text'])
def default_command(message):
    check_text_result = False
    invalid_text_segment = ''
    message_text = message.text
    if message_text:
        check_text_result = check_text(message_text)

    if check_text_result:
        invalid_text_segment = "Text not valid"
        bot.delete_message(message.chat.id, message.message_id)
        result_text = "Received message with adult content from user: " + message.from_user.first_name + \
            "(@" + message.from_user.username + \
            "): [" + invalid_text_segment + \
                    "]. We notified admin's of the current chat."
        notify_admins(message.chat.id, result_text)


# Handle all other messages.

@bot.message_handler(func=lambda message: True, content_types=['photo', 'sticker'])
def receive_img(message):
    check_image_result = False
    check_text_result = False
    invalid_text_segment = ''

    message_text = message.caption
    message_photo = message.photo
    message_sticker = message.sticker

    if message_text:
        check_text_result = check_text(message_text)
```

```
    if check_text_result:
        invalid_text_segment = "Text not valid"
        bot.delete_message(message.chat.id, message.message_id)
        result_text = "Received message with adult content from user: " + message.from_user.first_name + \
            "(@" + message.from_user.username + \
            "): [" + invalid_text_segment + \
                    "]. We notified admin's of the current chat."
        notify_admins(message.chat.id, result_text)

    if (not check_text_result) and message_photo:
        img_data = bot.get_file(message_photo[-1].file_id).wait()
        img_url = BASE_IMAGE_URL.format(API_KEY, img_data.file_path)
        check_image_result = check_image(img_url)

    if (not check_text_result) and message_sticker:
        img_data = bot.get_file(message_sticker.file_id).wait()
        img_url = BASE_IMAGE_URL.format(API_KEY, img_data.file_path)
        check_image_result = check_image(img_url)

    if check_image_result:
        bot.delete_message(message.chat.id, message.message_id)
        invalid_text_segment = "Media not valid"
        result_text = "Received message with adult content from user: " + message.from_user.first_name + \
            "(@" + message.from_user.username + \
            "): [" + invalid_text_segment + \
                    "]. We notified admin's of the current chat."
        notify_admins(message.chat.id, result_text)

# @bot.message_handler(func=lambda message: True, content_types=['audio', 'photo', 'voice', 'video',
'document', 'text', 'location', 'contact', 'sticker'])
# def default_command(message):
#     bot.send_message(message.chat.id, "This is the default command handler.")


bot.infinity_polling()
```

Б.2 Класифікатор тексту

```
from config_util import get_characters_scheme, get_words_scheme

words = get_words_scheme()
characters_scheme = get_characters_scheme()


def distance(a, b):
    "Calculates the Levenshtein distance between a and b."
```

```
    n, m = len(a), len(b)
    if n > m:
        # Make sure n <= m, to use O(min(n, m)) space
        a, b = b, a
        n, m = m, n

    # Keep current and previous row, not entire matrix
    current_row = range(n + 1)
    for i in range(1, m + 1):
        previous_row, current_row = current_row, [i] + [0] * n
        for j in range(1, n + 1):
            add, delete, change = previous_row[j] + \
                1, current_row[j - 1] + 1, previous_row[j - 1]
            if a[j - 1] != b[i - 1]:
                change += 1
            current_row[j] = min(add, delete, change)

    return current_row[n]


def check_text(phrase):
    phrase = phrase.lower().replace(" ", "")
    for key, value in characters_scheme.items():
        # Проходимся по каждой букве в значении словаря. То есть по вот этим спискам ['a', 'a', '@'].
        for letter in value:
            # Проходимся по каждой букве в нашей фразе.
            for phr in phrase:
                # Если буква совпадает с буквой в нашем списке.
                if letter == phr:
                    # Заменяем эту букву на ключ словаря.
                    phrase = phrase.replace(phr, key)

    # Проходимся по всем словам.
    count = 0
    for word in words:
        # Разбиваем слово на части, и проходимся по ним.
        for part in range(len(phrase)):
            # Вот сам наш фрагмент.
            fragment = phrase[part: part+len(word)]
            # Если отличие этого фрагмента меньше или равно 25% этого слова, то считаем, что они
равны.
            if distance(fragment, word) <= len(word) * 0.1:
                print("Найдено", word, "\nПохоже на", fragment)
                count += 1

    return count > 0
```

Б.3 Класифікатор зображень

```
from neuralink import load_data, batch_size
```

```
import tensorflow as tf
from tensorflow.keras.models import load_model
import matplotlib.pyplot as plt
import numpy as np
# example of converting an image with the Keras API
from keras.preprocessing.image import load_img

# CIFAR-10 classes
categories = {
    0: "airplane",
    1: "automobile",
    2: "bird",
    3: "cat",
    4: "deer",
    5: "dog",
    6: "frog",
    7: "horse",
    8: "ship",
    9: "truck"
}


def check_image(image_path):
    # загрузим тестовый набор
    ds_train, ds_test, info = load_data()
    # загрузим итоговую модель с весовыми коэффициентами
    model = load_model("./results/cifar10-model-v2.h5")

    # оценка
    loss, accuracy = model.evaluate(
        ds_test, steps=info.splits["test"].num_examples / batch_size)
    print("Тестовая оценка:", accuracy*100, "%")

    # получить прогноз для этого изображения
    data_sample = next(iter(ds_test))

    image_path = tf.keras.utils.get_file('Test_image', origin=image_path)

    img = load_img(path=image_path, target_size=(32, 32))
    img_array = tf.keras.utils.img_to_array(img)

    sample_label = "cat"
    prediction = np.argmax(model.predict(
        img_array.reshape(-1, *img_array.shape))[0])
    print("Predicted label:", categories[prediction])
    print("Predicted:", prediction)
    print("True label:", sample_label)

    # show the image
    # plt.axis('on')
    # plt.imshow(img)
    # plt.show()
```

```
    return prediction == 3 and categories[prediction] == sample_label
```

Б.4 Нейромережа

```
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, RandomFlip, RandomRotation,
RandomZoom
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.callbacks import TensorBoard
import tensorflow as tf
import tensorflow_datasets as tfds
import os
import matplotlib.pylab as plt


# tf.config.list_physical_devices('GPU')

# Гиперпараметры
batch_size = 64
# 10 категорий для изображений  (CIFAR-10)
num_classes = 10
# количество эпох для обучения
epochs = 30


def load_data():
    print("Start loading data")
    """
    Эта функция загружает набор данных CIFAR-10 dataset и делает предварительную обработку
    """
    def preprocess_image(image, label):
        # преобразуем целочисленный диапазон [0, 255] в диапазон действительных чисел [0, 1]
        image = tf.image.convert_image_dtype(image, tf.float32)
        return image, label
    # загружаем набор данных CIFAR-10, разделяем его на обучающий и тестовый
    ds_train, info = tfds.load(
        "cifar10", with_info=True, split="train", as_supervised=True)
    ds_test = tfds.load("cifar10", split="test", as_supervised=True)
    # повторять набор данных, перемешивая, предварительно обрабатывая, разделяем по пакетам
    ds_train = ds_train.repeat().shuffle(1024).map(
        preprocess_image).batch(batch_size)
    ds_test = ds_test.repeat().shuffle(1024).map(preprocess_image).batch(batch_size)
    return ds_train, ds_test, info


def create_model(input_shape):
    # построение модели
    print("Creating model")
    data_augmentation = Sequential()
```

```
    data_augmentation.add(RandomFlip("horizontal", input_shape=(32, 32, 3)))
    data_augmentation.add(RandomRotation(0.1))
    data_augmentation.add(RandomZoom(0.1))

    model = Sequential()
    model.add(data_augmentation)
    model.add(Conv2D(filters=32, kernel_size=(3, 3),
            padding="same", input_shape=input_shape))
    model.add(Activation("relu"))
    model.add(Conv2D(filters=32, kernel_size=(3, 3), padding="same"))
    model.add(Activation("relu"))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Conv2D(filters=64, kernel_size=(3, 3), padding="same"))
    model.add(Activation("relu"))
    model.add(Conv2D(filters=64, kernel_size=(3, 3), padding="same"))
    model.add(Activation("relu"))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Conv2D(filters=128, kernel_size=(3, 3), padding="same"))
    model.add(Activation("relu"))
    model.add(Conv2D(filters=128, kernel_size=(3, 3), padding="same"))
    model.add(Activation("relu"))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    # сглаживание неровностей
    model.add(Flatten())
    # полносвязный слой
    model.add(Dense(1024))
    model.add(Activation("relu"))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation="softmax"))
    # печатаем итоговую архитектуру модели
    model.summary()
    # обучение модели с помощью оптимизатора Адама
    model.compile(loss="sparse_categorical_crossentropy",
                optimizer="adam", metrics=["accuracy"])
    return model


def start_learning():
    # загружаем данные
    print("Start learning model")
    ds_train, ds_test, info = load_data()
    # конструируем модель
    model = create_model(input_shape=info.features["image"].shape)
    # несколько хороших обратных вызовов
    logdir = os.path.join("logs", "cifar10-model-v1")
    tensorboard = TensorBoard(log_dir=logdir)
    # убедимся, что папка с результатами существует
    if not os.path.isdir("results"):
        os.mkdir("results")
```

```
# обучаем
hist = model.fit(ds_train, epochs=epochs, validation_data=ds_test, verbose=1,
                 steps_per_epoch=info.splits["train"].num_examples // batch_size,
                 validation_steps=info.splits["test"].num_examples // batch_size,
                 callbacks=[tensorboard])

plt.figure()
plt.ylabel("Loss (training and validation)")
plt.xlabel("Training Steps")
plt.ylim([0, 2])
plt.plot(hist.history["loss"])
plt.plot(hist.history["val_loss"])

plt.figure()
plt.ylabel("Accuracy (training and validation)")
plt.xlabel("Training Steps")
plt.ylim([0, 1])
plt.plot(hist.history["accuracy"])
plt.plot(hist.history["val_accuracy"])
# сохраняем модель на диске
model.save("results/cifar10-model-v1.h5")
```