Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

СТВОРЕННЯ ЧАТ-БОТУ ДЛЯ ФАКУЛЬТЕТУ ІНФОРМАТИКИ

Текстова частина до курсової роботи

за спеціальністю „Інженерія програмного забезпечення” 121

Керівник курсової роботи
Доктор. тех. наук, доцент
Глибовець А.М.
(прізвище та ініціали)

_____
(підпис)
“____” _____ 2020 р.

Виконав студент 4 курсу
Кундік К.В.
(прізвище та ініціали)
“____” _____ 2020 р.

Київ 2020

ЗМІСТ

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри мережних технологій,
проф., д.ф.-м.н.
_____Г. І. Малашонок
(підпис)
“____”_____202_ р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студенту Кундіку Кирилу
4 курсу факультету інформатики
ТЕМА: Створення чат-боту для факультету інформатики
Вихідні дані:

Зміст ТЧ до курсової роботи:

Вступ
Розділ 1. Загальні відомості щодо розвитку чат-ботів в сучасному світі
Розділ 2. Дослідження існуючих технологій для чат-ботів
Розділ 3. Дослідження практичного застосування чат-ботів
Розділ 4. Опис практичної частини роботи
Висновки
Список джерел
Додатки
Дата видачі “___” _____ 202_ р.

Керівник _____ Завдання отримано _____

Календарний план виконання курсової роботи

Тема: Створення чат-боту для факультету інформатики

№ п/п	Назва етапу курсового проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	листопад 2019 р.	
2.	Огляд літератури за темою роботи	листопад-грудень 2019 р.	
3.	Оволодіння навичками роботи із Flask	грудень 2019 р.	
4.	Розробка та реалізація алгоритму роботи бота	січень-березень 2020 р.	
5.	Написання пояснювальної роботи	квітень 2020 р.	
6.	Створення слайдів для доповіді та написання доповіді	квітень 2020 р.	
7.	Надання роботи керівнику для перевірки	квітень 2020 р.	
8.	Корегування роботи за результатами перевірки керівником	квітень 2020 р.	
9.	Остаточне оформлення пояснювальної роботи та слайдів	квітень 2020 р.	
10.	Подання роботи на кафедру для перевірки на плагіат	квітень 2020 р.	
11.	Захист курсової роботи	квітень 2020 р.	

Студент _____Кундік К.В._____

Керівник _____Глибовець А.М._____

“_____” _____ _____ р.

ВСТУП

Актуальність, наукове та практичне значення обраної теми

З розвитком інтернет-технологій бізнеси та організації зазнали значних змін в інтеракції з клієнтами та користувачами. Наявність веб-сайту, рейтинг при пошуку, можливість комунікації онлайн вже не є перевагою чи привілеєм, а є необхідністю. З популяризацією месенджерів прийшло поняття чат-боту. Бот - це комп'ютерна програма, яка симулює людську комунікацію завдяки голосовим або текстовим повідомленням. Теперішній розвиток алгоритмів машинного навчання дозволяє створювати чат-боти, які симулюють та замінюють спілкування з технічною підтримкою користувачів, автоматизують рутинну роботу та спрощують отримання послуг.

Згідно з дослідженням Juniper Research до 2022 року, до 90% банківських операцій буде здійснюватись завдяки чат-ботам. Дослідження Spiceworks показує, що 40% великих компаній наймали понад 500 людей для розробки чат-ботів у 2019 році. Високі рейтинги використання та імплементації ботів пояснюються швидкістю та простотою у користуванні для клієнтів, зменшенням часу та економією коштів для компаній.

Доступність ботів полягає у тому, що не потрібне додаткове програмне забезпечення для користування. Інтерфейс є інтуїтивно зрозумілим усім користувачам, оскільки є інтерфейсом будь-якого іншого чату у месенджері. Долучитись до бота можна перейшовши за посиланням, QR-кодом або ввівши ім'я боту у пошуку. Головна перевага чат-боту - це задоволення вимог користувача миттєво. Відтепер пошук інформації, виконання операцій банкінгу, замовлення чи купівлі товарів не вимагає додаткових дій.

Мета та завдання курсової роботи

Мета: Створити чат-бот для факультету інформатики, який спростить пошук відповідей на питання та мінімізує витрату часу.

Завдання: Розробити сервіс, який стане швидким інтерактивним довідником для студентів, абітурієнтів та інших користувачів. Реалізувати пошук у книзі про факультет, інформації від Бадді та веб-сайту факультету.

Об’єкт дослідження

Розробка веб-сайту з використанням мікрофреймворку Flask на Python.

Предмет дослідження

Сучасні системи, представлені чат-ботами, які вважаються найкращими за своїм наповненням, швидкістю та з використанням AI.

Практичне значення одержаних результатів

Розроблений чат-бот буде використовуватись під час вступної кампанії для популяризації університету серед абітурієнтів, подальша підтримка допоможе зменшити кількість питань, які виникають у студентів до Бадді, викладачів та адміністрації університету.

Джерела дослідження

Під час написання курсової роботи було досліджено наукові статті про чат-боти та їх частку ринку за останні роки з прогнозами на майбутнє. Проаналізовано інтернет-ресурси, які містять інформації про рейтинги найкращих чат-ботів. Під час роботи було вивчено аналоги для визначення можливих проблем та аналізу шляхів їх уникнення.

Структура роботи

Курсова робота складається з анотації, вступу, чотирьох основних розділів та підрозділів, висновків, списку використаних джерел, додатків.

Перший розділ курсової роботи присвячений проблемам проєкту, які вирішує розробка чат-боту. Детально розглянуто наявні види ботів, їх поширення та використання. Вивчено приклади успішних рішень та шляхи їх реалізації. Додатково проведено та описано дослідження щодо актуальності використання чат-ботів у НаУКМА та на факультеті інформатики.

Другий розділ акцентує увагу на алгоритмах, покладених в основу чат-ботів. Описано принципи, які є основними для створення чат-ботів. Проведено аналіз для вибору алгоритму за яким буде створюватись продукт. Розглянуто який результат буде досягнуто за допомогою обраних принципів.

У третьому розділі вивчено використання нейронних мереж для ботів на основі прикладів. Додатково описано процес, як буде реалізовано нейронну мережу для чат-бота. Описано теорію, яка лягла в основу реалізації.

Четвертий розділ є детальним описом практичної частини роботи. У розділі детально описано як було вибрано середовище розробки, мову

програмування. Міститься детальний опис використаних технологій з перевагами та недоліками на основі схожих рішень. Розглянуто та описано процес розробки з кодом. Додатково представлено скріншоти інтерфейсу та інструкція щодо користування чат-ботом.

РОЗДІЛ 1

1.1 Поширення та види чат-ботів

Найпоширеніша класифікація ботів полягає у їхній основі. Боти поділяюсь на скриптові та AI [1]. Бот в основі якого лежить скрипт заздалегідь визначається потоком. Коли користувач надсилає запит до бота, він отримує у відповідь заздалегідь визначений сценарій з бібліотеки. Такий вид потребує менших зусиль у реалізації, але від наповнення бібліотеки буде залежати його ефективність.

AI боти використовують NLP (Natural Language Processing) [2]. Цей вид ботів є швидким та відповідь, яку отримає користувач, буде більш влучною та доцільною. Бот навчається на запитах користувачів, тому є більш популярним наразі. Реалізація AI боту потрібна, якщо число запитів у день перевищує декілька сотень, потрібно додатковий канал комунікації, існують піки активності питань користувачів.

Іншим популярним видом є бот на основі кнопок [3]. Цей вид ідеально підходить для ситуацій, де чітко визначено сценарії. Наприклад, для відділу продажів, відділу підтримки, онбордингу або опитувань. Бот пропонує вибір кнопок і запитує дані. Створення боту не вимагає високої професійності, легкі та інтуїтивні. Однак, якщо потрібно з часом охопити більшу кількість сценаріїв, то бот може не підійти.

Гібридний бот - бот на основі кнопкою, але з додатковою можливістю додати питання. Зазвичай бот дає декілька варіантів кнопок і поле, у яке можна ввести питання, якщо серед кнопок немає потрібного. Цей вид є перехідним етапом між кнопковим та AI ботом. Цей бот допомагає користувачам знайти відповіді на поширені питання, але також стає певним дослідженням що додатково потрібно реалізувати. Також такий бот може стати історією логів, якщо її ще немає.

Найпрогресивнішим видом боту - є голосовий асистент, в основі якого розмова. Розробка такого боту потребує високої експертизи та залучення великої команди спеціалістів. За прогнозами голосові боти будуть найпопулярнішими серед інших видів.

Незалежно від відмінностей в інтерфейсі чи реалізації, боти приносять користь організаціям для яких створені завдяки прискоренню процесів, зменшенню витрат та покращенню сервісів.

1.2 Приклади успішних рішень

Чат-бот для факультету інформатики буде реалізовано українською мовою, тому під час аналізу наявних рішень, було обрано найпопулярніші та найуспішніші боти українських продуктів. Серед найпопулярніших потреб, які можуть задовольнити боти, є взаємодія з іншими додатками та сервісами.

Прикладом такого боту є RailwayBot, який допомагає моніторити, шукати та купувати квитки на потяг. За допомогою боту можна поставити моніторинг на певний потяг, дату, часу та тип місця. Як тільки квиток з’явиться в наявності - бот надішле сповіщення. Алгоритм дії схожий на купівлю квитків на uz.booking.gov.ua. Бот реалізовано на основі кнопок з введенням міст відправлення та прибуття.

PrivatBankBot - бот від ПриватБанку, який спрощує систему платежів. Це легкий спосіб відправити кошти, не відкриваючи додаток. Для надсилання коштів іншому користувачу, у нього теж має бути активований бот.

NovaPoshtaBot - бот для моніторингу відправлень, їх статусу, пошуку найближчого відділення. “Індекс якості повітря в Києві” - надає інформацію по двох обраних районах. Наприклад, Позняки - дім, Поділ - робота. Окрім

інформації про забруднення повітря доступні ще рекомендації чи потрібен додатковий захист. “Коронавірус в Україні” - бот, який надає актуально статистику щодо кількості заражених, тих, хто одужав, та тих, хто загинув від коронавірусу в Україні та Світі.

Більшість ресторанів та кафе мають власні боти для замовлення та доставки у популярних месенджерах. Проаналізувавши готові рішення, було взято до уваги, що швидкість боту є ключовою вимогою. Якщо час очікування перевищує декілька секунд, бот втрачає свою перевагу над пошуком у Google чи відкриттям додатку. AI боти більше нагадують живе спілкування, тому мають перевагу над скриптовими ботами. Водночас, гібридні боти є більш зручними у користуванні. Надання готових варіантів спрощує комунікацію, вимагає менше дій від клієнта.

1.3 Дослідження залученості у месенджері telegram на основі НаУКМА

Основними користувачами боту є студенти НаУКМА, тому невіддільною частиною дослідження було вивчення поширення telegram ботів серед студентства. Провівши індивідуальні опитування, було виявлено, що студенти активно користуються ботами в telegram. Серед ботів є зазначені у попередньому пункті. Найпопулярнішим ботом серед студентства є реалізований раніше KMASchedulerBot. Кількість зареєстрованих користувачів перевищує 4559. Бот дозволяє завжди мати свій розклад пар під рукою, реалізовано ряд команд для розкладу на тиждень, день, час. Зроблена інтеграція з САЗ, яка дозволяє дізнатися про зміни у розкладі без постійного моніторингу веб-сайту.

Успіх KMAScheduler та активність студентів у месенджері telegram доводить, що вид та середовище проєкту було обрано правильно.

РОЗДІЛ 2

2.1 Дослідження основи чат-ботів (алгоритми)

Чат-боти схожі на віртуальних асистентів [4]. Вони часто називаються “комунікаційними агентами”, оскільки їхня основна задача - це взаємодія та діалог з реальними людьми. Чат-боти можуть базуватись на веб-додатках або бути окремими додатками. Інтерфейс чат-ботів - це чат для спілкування. Швидке зростання кількості ботів пояснюється спрямованістю на клієнта, легкістю у користуванні та зручному інтерфейсі. Зараз з’явилось багато конструкторів, які допомагають створити чат-бот для будь-якого бізнесу, а не лише для технічних компаній.

Боти розуміють текст завдяки використанню технологій NLP (Natural Language Processing). Технологія обробляє текст та визначає відповідь, яка ситуативно підходить. На сьогодні, технологія досягла настільки високого рівня, що під час взаємодії у чаті інколи важко визначити чи спілкування відбувається з ботом чи з представником служби підтримки.

NLP покращує свої можливості у розумінні нюансів людської мови [5]. Боти активно використовуються у службах підтримки, маркетингу, продажах, але можуть бути налаштовані під будь-який аспект бізнесу.

Розглянемо, як саме чат-боти обробляють людську мову. На перший погляд, бот виглядає як звичайна програма. До її складу входить прикладний рівень, база даних та API для зовнішніх служб. Інтерфейс боту - це інтерфейс чату. Чат-боти легкі у використанні для користувачів, але складність полягає у самій програмі.

Розробники використовують наявні логи спілкування боту з клієнтами, для визначення питань та мотивів. Це допомагає покращити наявну модель через навчання. Комбінація для покращення складається з моделі машинного навчання та вбудованих інструментів. Розробники

поєднують питання клієнтів з найбільш вдалими відповідями. Основна задача, яка стоїть перед розробниками, це навчити модель розпізнавати два різних питання, з точки зору мови, але які насправді, матимуть одну й ту саму відповідь. Якщо ж внутрішніх даних немає у наявності, то для навчання ботів можна використовувати дані API.

Тренування ботів відрізняється від тренування людей швидкістю та масштабом даних. Наприклад, представник служби підтримки повинен прочитати та зрозуміти навчальну документацію. В той час як бот служби підтримки наповнюється тисячами логів діалогів і з цих логів, розуміє який тип питання вимагає якої відповіді.

Після того, як бот запущено та він взаємодії з користувачами, можна реалізувати цикли зворотного зв’язку для подальшого навчання боту. Під час розмови з ботом, коли клієнт надсилає питання, бот може надавати варіанти відповідей і запитувати що саме мав на увазі клієнт. У такий спосіб, клієнт самостійно визначає свої наміри і ця інформація допомагає перенавчати модель машинного навчання, щоб покращити точність боту. У цьому випадку, залишається можливість помилки, якщо користувачі обирають варіанти, що направляють бота у неправильний бік.

2.2 Принципи створення ботів для месенджерів

Для створення боту важливо включати три методи класифікацій [6]. Перший метод - це зіставлення з зразком. Боти використовують цей метод щоб класифікувати текст та надати відповідну відповідь для клієнта. Стандартною структурою для таких патернів є Artificial Intelligence Markup Language (AIML).

Зразком такого патерну буде:

```
<aiml version = "1.0.1" encoding = "UTF-8"?>
  <category>
    <pattern> WHO IS ABRAHAM LINCOLN </pattern>
    <template>Abraham Lincoln was the US President during American civil war.</template>
  </category>

  <category>
    <pattern>DO YOU KNOW WHO * IS</pattern>
    <template>
      <srai>WHO IS <star/></srai>
    </template>
  </category>
</aiml>
```

Відповідь комп’ютера:

Людина: Do you know who Abraham Lincoln is?

Комп’ютер: Abraham Lincoln was the US President during American civil war.

Відповідно, для кожного питання має існувати унікальний патерн в базі даних, щоб надати відповідний результат. У зв’язку з великою кількістю комбінацій патернів, створюється ієрархічна структура. Використовуються алгоритми щоб зменшити класифікатори та згенерувати легші в управлінні структури. Комп’ютерні інженери називають “урізаючим” підходом: для отримання спрощеного рішення, алгоритм зменшує проблему.

Наївний байєсівський алгоритм - це класичний алгоритм для класифікації тексту та NLP. Наприклад, сет речень відповідає певному класу. З введенням нового речення, кожне слово рахується числом його появи. Від числа буде залежати його загальність. Кожен клас має свою оцінку. До класу з найвищою оцінкою буде додано нове речення. Такий підхід не надає стовідсоткової впевненості у точності підбору класу, а лише приблизний результат.

Нейронні мережі - це спосіб обчислення виходу з вхідних даних за допомогою зважених з'єднань, які обчислюються з повторних ітерацій під

час навчання даних. Кожен крок через дані тренувань коригує ваги, що призводять до точного результату.

Як було вказано вище, кожне речення розбивається на різні слова, кожне слово використовується як вхідні дані для нейронної мережі. Зважені з'єднання обчислюються за допомогою різних ітерацій тренувальних даних. Кожного разу покращуються ваги для збільшення точності.

NLU (Natural Language Understanding) складається з трьох основних концепцій: структури, наміри та контекст [6]. Структура представляє концепцію у чат-боті. Намір - це дія боту, яку він має виконувати, коли користувач щось надсилає. Контекст - це коли NLU алгоритм аналізує речення без історії спілкування з користувачем. Якщо буде отримано відповідь на щойно надіслане питання, то бот не буде пам'ятати питання.

NLP (Natural Language Processing) бот здійснює певну комбінацію кроків для перетворення тексту чи мови клієнта в структуровані дані, які використовуються для вибору відповідної відповіді.

2.3 Використання NLTK для чат-ботів

NLTK (Natural Language Toolkit) - це провідна платформа для побудови програм на мові Python для роботи з даними людської мови [7]. Платформа надає прості у використанні інтерфейси для понад 50 корпоративних та лексичних ресурсів, таких як WordNet, а також набір бібліотек для обробки тексту для класифікації, токенізації, стримування, тегування, розбору та семантичних міркувань, обгортки для промислових NLP-бібліотек. NLTK - інструментом для навчання та роботи в обчислювальній лінгвістиці за допомогою Python та бібліотекою для роботи з природною мовою.

Основна проблема з текстовими даними полягає в їхньому текстовому форматі, тобто форматі рядка. Однак алгоритмам машинного навчання потрібен певні числові векторні ознаки для виконання завдання. Тому перед тим, як розпочати будь-який проект з NLP, потрібно попередньо обробити данні, щоб підготувати їх до роботи. Основна попередня обробка тексту включає перетворення тексту у великі чи малі регістри, щоб алгоритм не розглядав одні й ті самі слова в різних випадках як різні.

Іншим важливим кроком є токенізація - термін, який використовується для опису процесу перетворення звичайних текстових рядків у список лексем. Токенизатор речення може бути використаний для пошуку списку речень, а токенізатор Word може бути використаний для пошуку списку слів у рядках.

Пакет даних NLTK включає попередньо підготовлений токенизатор Punkt для англійської мови, який видаляє “шум”, тобто все, що не стандартна цифра чи літери. Додатково здійснюється видалення стоп слів. Іноді деякі надзвичайно поширені слова, які мають мало значення при доборі документів, що відповідають потребі користувача, повністю виключаються з лексики.

2.4 Дослідження відмінностей платформ для чат-ботів: IBM Watson, Dialogflow, Wit.ai, Azure Bot Service

При пошуку інсрументу для реалізації чат-боту, було досліджено три платформи: IBM Watson, Dialogflow, Wit.aim, Wit.ai, Azure Bot Service. Розглянемо детальніше кожну із них.

IBM Watson - це набір програм та інструментів готових для використання для AI-сервісів, найкраще підходить для підприємств [8]. Watson легко вбудовується у канал, пристрій чи програму та підтримує

пошук відповіді у базі знань. Платформа дозволяє два канали для комунікації: текст та голос. IBM Watson Assistant легкий у користуванні завдяки зручному інтерфейсу, доступний широкий вибір відео для навчання і темлпейтів для швидкого старту роботи.

Розробники Watson зазначають, що їхній продукт відрізняється від інших тим, що не робить спроб наслідувати людську мову, що допомагає уникнути непорозумінь під час комунікації з користувачем. Watson Assistant знає коли потрібно шукати відповідь у базі знань, коли потрібно надіслати уточнювальне запитання і коли направити клієнта до людини.

Доступні інтеграції зі Voice Agent, Slack, Facebook Messenger, Wordpress та програмами через API. Завдяки можливості розгорнути чат у хмарі або локально - продукт став доступним для усіх потреб. З недоліків платформи варто зазначити вартість користування, що і стало фінальним рішенням щодо використання даної платформи.

Dialogflow - продукт від Google, що раніше відомий як API.ai, який повністю орієнтований на клієнта [9]. Він базується на стандартних процесах під час користування клієгнта сервісом. Розглянемо, приклад Dominos. Бот “Dom” вбудований у систему та дозволяє зробити замовлення базуючись на можливих сценаріях. Компанія стала однією з перших, хто користувався вбудованим асистентом у власну систему.

Dialogflow інтегрується з Google Assistant, веб-сайтами, Slack, Facebook Messenger, Skype, Twitter. Dialogflow надає клієнтам можливість взаємодіяти з сервісами або продуктами компаній за допомогою голосового або текстового розмовного інтерфейсу. Розробка боту можлива завдяки зручному веб-інтерфейсу. Не потрібно мати високого рівня технічної підготовки, оскільки поняття намір, сутність і дія є інтуїтивно зрозумілими у системі і легко налаштовуються.

Платформа підходить для ботів “питання-відповідь”, але більш орієнтована на чіткий процес для користувача. Серед недоліків для

використання у роботі - платна версія і складніша логіка, якої не вимагає створення чат-боту у нашому випадку.

Wit.ai - дуже простий у використанні, але складний у наповненні його бази знань. Платформа орієнтується на самонавчання, а не на готові матеріали. Обробка природних мов (NLP) дозволяє зрозуміти значення слів користувача. Прямих інтеграцій через веб-інтерфейс немає, всі інтеграції проводяться через HTTP API та бібліотеки, доступні в Node js, Python, Ruby та Go. Wit.ai - безкоштовний, але не підходить для нашого проекту, оскільки націлений на розуміння та трактування слів користувача, в той час як нам потрібно сфокусуватись на швидкому наповненні та пошуку у базі знань.

Wit.ai - це API, який дозволяє розробникам дуже легко створювати додатки чи пристрої, за допомогою яких можна спілкуватися з будь-яким додатком або будь-яким пристроєм, наприклад смарт-годинником, Google Glass, Nest, навіть автомобілем, передавати аудіо в програму Wit API, і отримувати відповідні дані натомість. Сервіс допомагає перетворювати мовлення в дії.

Розглянемо Azure Bot Service, який надає інтегроване середовище, яке спеціалізується на розробці ботів. Служба дозволяє будувати, підключати, тестувати, розгортати та керувати інтелектуальними ботами в одному місці. Azure Bot використовує SDK Bot Framework з підтримкою C# та JavaScript.

Microsoft Bot Framework дозволяє створювати ботів з можливістю говорити, слухати, розуміти та навчатись від користувачів. Веб-інтерфейс для створення та запуску ботів є легким та доступним для розуміння. Чат-бот, створений за допомогою служби Azure Bot, може публікуватися на різних каналах, таких як Web, Facebook Messenger, Skype та Skype for Business, Microsoft Teams, Slack тощо [10].

На жаль, Azure Bot Service не підійшов за декількома критеріями. Найголовнішим недоліком платформи є те що, для написання простого боту вимагається багато коду, що на інших платформах є більш оптимізованим.

Підсумовуючи, варто зазначати що жодна із наведених платформ цілком не задовольняє потреби проекту. Наступним кроком буде дослідження альтернативних платформ.

РОЗДІЛ 3

3.1 Боти як інструмент пошуку інформації

Чат-боти є корисним інструмент у навчанні, наприклад, для вивчення нової мови чи математики. Учні використовують бот для розв’язання алгебраїчних задач, а викладачі - для пошуку. Дослідники пошуку інформації також відзначають, що техніки для знаходження відповідей на питання з набору документів, використовуються далеко поза сферою навчання.

У Лідському університеті було створено програму FAQ по мові Java [11]. До цього було розроблено схожу систему по системі Linux. Розглянемо роботу таких систем. Розробка FAQ має перевагу над іншими тренувальними сетами у тому, що має чіткі визначення для користувача - питання, та для чат-боту - відповідь, що робить модель більш зрозумілою.

Результати, які повертає FAQ-чат є схожими до результатів пошукового двигуна Google. Вони подаються у виді посилань до відповідних або майже точно відповідних веб-ресурсів. У зв’язку з цією подібністю, інтерфейс був побудований для отримання питання користувача і надсилав дві відповіді. Перша відповідь була згенерованою з FAQ-чату, інша - з Google після фільтрування FAQ.

Після проведеного дослідження щодо думки студентів та адміністрації університету, чат отримав перевагу через дві причини. По-перше, можливість отримувати відповідь напряму, в той час я Google просто надає посилання. По-друге, економія часу, бо FAQ-чат надає меншу кількість інформації на опрацювання.

Користувачі, які надали перевагу використанню Google, пояснили свій вибір тим, що він їм уже знайомий, а відповідно і зручний. Також

існувала думка, що чатом важче керувати підбираючи ключові слова. Це виникло оскільки чат давав відповіді по певним визначним словам. Якщо ж користувач переформульовує питання, то відповідь залишається такою ж. В той самий час, Google міг надавати різні відповіді при такому варіанті сценарію.

Незалежно від того, що користувачі обирала для користування, усі вони надали позитивний зворотний зв’язок про чат-бот, оскільки доступ до FAQ здійснювався природною мовою. Кожен третій користувач зумів знайти відповідь на своє питання. Ціллю дослідження було показати, що такий чат є альтернативою пошуковим системам. Така технологія допомагає надати легкий доступ до бази даних з FAQ.

3.2 Боти як асистенти

Розглянемо асистентські чат-боти на прикладі боту Happy Assistant [12]. Happy Assistant - це навігаційна система, яка базується на природній мові діалогів, яка допомагає користувачам отримати доступ до електронної комерції, щоб знайти відповідну інформацію про продукти та сервіси. Система складається з трьох основних модулів: менеджер презентацій, діалогу та дій.

Менеджер презентацій застосовує техніку неглибокого розбору для виявлення семантичної та синтаксичної інформації, яка отримана від користувача. Потім питання користувача перекладається у сформоване XML повідомлення. Менеджер діалогу несе відповідальність за узгодження понять із запиту користувача та правил, знайдених в області знань. Правила складаються з переліку понять разом з деякими метаданими про товар чи послугу. Якщо збіг знайдено, веб-сторінка, пов’язана з цим правилом, буде представлена користувачеві. Якщо інформація відсутня, то система надішле

питання користувачу для отримання деталей. Далі менеджер дій, який отримує доступ до продукту, що відповідає запиту. Якщо користувач надає додаткові вимоги, алгоритм сортування застосовується для отримання відповідного списку продуктів.

Щоб отримати довіру користувачів, система повинна запропонувати певні пояснення, перш ніж створювати результат, тому система узагальнює запит користувача, перефразовуючи його.

Результати показують, що така система відповідає вимогам користувачів. Клієнтам подобається ідея, що вони надають свої вимоги у природній для них спосіб. Щобільше, було зазначено, що така система економить витрачений час.

3.3 Опис логіки розробленого чат-бота

Чат-бот матиме змішаний тип, окрім FAQ-чату, він надаватиме мінімальний асистентський функціонал. Основне завдання, яке буде виконувати програма, полягає у швидкому доступі до відповідей про університет, факультет, контакти викладачів. Бот допоможе зменшити витрату часу на пошук та комунікацію з адміністрацією, студентами.

Першим справжнім тестом для бота стане час вступної комісії, коли абітурієнти активно спілкуються з департаментом “Бадді” НаУКМА та шукають інформацію про університет та його життя у пошукових двигунах типу Google, соціальних мережах Facebook та телеграм. Для вирішення зазначених вище проблем, база знань боту буде містити книгу М.М. Глибовця “Справа мого життя - факультет інформатики НаУКМА”, статті підібрані “Бадді НаУКМА”, інформацію зібрану за 4 роки діяльності організації про внутрішні системи та інші джерела.

Перевага такої системи - це можливість проводити контроль питань, які будуть надходити у чати та поступово заповнювати відповідні прогалини у системі. Реалізація самонавчання чат-ботів описана у попередньому розділі роботи. Інтерфейс програми реалізована як чат у телеграмі. Розглянемо приклад ймовірного сценарію боту.

Користувач: Привіт! Які вступні бали на факультет інформатики?

Чат-бот: Привіт! Вкажи, будь ласка, яка спеціальність(-ості) тебе цікавить?

а) Комп'ютерні науки та інформаційні технології;

б) Інженерія програмного забезпечення;

в) Прикладна математика.

Користувач: Варіанти а та б.

Чат-бот: Середні бали зарахованих на КНІТ: 192,3 та на ІПЗ: 193,8.

Інформація наведена у відповіді боту вище взята з діаграм, що були підготовлені Бадді, переглянути діаграми можна у додатку №1.

РОЗДІЛ 4

4.1 Аналіз технічного завдання

Постановка задачі – розробити чат-бот факультету інформатики для швидких відповідей на запитання студентів (будь-якого курсу; запитання про навчання, розміщення аудиторій, історію факультету, навчальний процес тощо) та абітурієнтів (запитання про вступну кампанію, спеціальності та їх різницю, гуртожитки тощо).

Структура вхідних даних складається з бази знань, що містить в собі: питання та відповіді, які були зібрані спеціально для бота за допомогою анкетування студентів; з електронної версії книги М.М. Глибовця «Справа мого життя – факультет інформатики НаУКМА». Вихідні дані – перелік відповідей на конкретне запитання від студента або абітурієнта.

Основними параметрами програми є база знань, питання користувачів та відповіді, що генеруються на основі бази знань або знаходяться в базі знань.

Програма надає можливість користувачеві в соціальному месенджері Telegram отримати відповідь на будь-яке запитання, що стосується факультету інформатики та університету в цілому, написавши це запитання спеціальному боту, яким керує сама програма. Після отримання відповіді у користувача є можливість оцінити її.

Кожна оцінка впливає на якість та навчання моделі, що генерує відповіді. Якщо отримана відповідь для користувача з будь-яких причин вважатиметься їм неправильною і такою, що не задовольняє його потреби, він матиме можливість поставити повторно своє запитання команді підтримки. Для цього було створено спеціальний чат Telegram, куди можна долучити студентів для команди підтримки. Запитання користувача буде надіслано туди автоматично, а відповісти на нього можна просто

відправивши нове повідомлення з посиланням на питання. Після чого користувач отримає цю відповідь у приватні повідомлення з ботом.

В основі реалізації – асинхронні відповіді сервера, що керує чат-ботом, асинхронні запити до веб-сервера, який взаємодіє з моделлю машинного навчання, пошуковим двигуном Elastic Search, базою даних для збереження всієї необхідної інформації по роботі чат-боту.

Чат-бот може працювати у будь-якому середовищі з використанням ОЗУ більше 4Гб та встановленою технологією Docker [14]. З боку користувача необхідний лише встановлений додаток Telegram та підключення до мережі інтернет.

4.2 Вибір засобів розробки чат-боту

4.2.1 Мова програмування та середовище розробки

Python – високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника та читання коду. Ядро синтаксису Python мінімалістичне, однак стандартна бібліотека включає в собі великий обсяг корисних функцій [15].

Перша задумка про створення мови програмування Python була в кінці 1980-х років, а перші кроки в створенні повноцінної мови програмування вже були зроблені наприкінці грудня 1989 року Гуйдо ван Россумом в Нідерландах. Першочергове призначення мови були такі як здатність обробляти винятки та взаємодіяти з операційною системою Amoeba. Назва Python походить від телевізійного шоу, яке транслювалося на каналі BBC у Великобританії, Літаючий цирк Monty Python’а.

Python має динамічну типізацію, збирача сміття. Він підтримує декілька парадигм програмування, включаючи структуроване (зокрема,

процедурне), об'єктно-орієнтоване, функціональне та імперативне програмування, а також деякі принципи рефлекції. Має в собі дуже велику, всебічно розвинену систему вбудованих бібліотек та модулів, що встановлюються «з коробки» разом з інтерпретатором.

Також мова програмування Python використуває комбінацію підрахунку посилань та циклічне виявлення сміття спеціальним збірником для управління пам'яттю. Він також реалізує функцію пізнього зв'язування, яке пов'язує імена методів та змінних безпосередньо під час виконання програми [16].

Середовище розробки PyCharm - це інтегроване середовище розробки, яке використовується в комп'ютерному програмуванні, спеціально для мови Python. Його розробляє чеська компанія JetBrains. Він забезпечує аналіз коду, графічний відлагоджувач, інтегрований плагін для швидкої розробки та відтворення unit-тестів, інтеграцію з системами управління версіями Git, Mercurial, BitBucket та підтримує веб-розробки з декількома веб-фреймворками, на приклад, Django. Також доступний наукові режими для роботи в сфері Data Science, Big Data, аналітики з Anaconda. Середовище розробки є кросплафторменним, тому підтримується на операційних системах Windows, MacOS та Linux-подібних системах.

Особливості роботи з PyCharm це допомога з написання та аналіз із доповненням коду, виділенням синтаксису та помилок інтерпретації, інтеграцією linter (або lint) та швидкими виправленням та рефакторингом кода. Також дуже спрощена система навігації по проєктах та файлах: спеціалізовані перегляди проєктів, перегляди структури файлів та швидкий перехід між файлами, класами, методами та місцями, де вони використовуються або викликаються. Середовище також дозволяє налагодити процес створення продуктової версії продукту за допомогою використання Google App Engine та декількох наукових інструментів, що

дуже необхідні для створення моделей машинного навчання та роботи з великими даними такі, як matplotlib, numpy та scipy.

4.2.2 Розробка фронтенду (серверу чат-боту)

В основі розробки серверу чат-боту було покладено використання Telegram Bot API, що дозволяє легко і швидко створювати інтегровані чат-боти в месенджері Telegram. Це API можна використовувати на основі HTTP запитів на сервер телеграму. Чат-бот в свою чергу створюється за допомогою іншого боту від месенджера, який називається BotFather, що і є батьківським класом в системі телеграму для інших ботів [17].

За допомогою запитів можна легко отримувати інформацію щодо оновлення боту (отримання нових повідомлень, змінення повідомлень, відправлення різних даних мультимедіа тощо).

На основі використання Telegram Bot API було побудовано асинхронний фреймворк aiogram, що був використаний під час розробки чат-боту [18]. Фреймворк aiogram було обрано за його простоту у використанні та за те, що він повністю асинхронний. Побудований за допомогою Python стандартної бібліотеки asyncio та асинхронного веб-фреймворку aiohttp [19].

Найголовнішою особливістю є те, що він може витримувати дуже велике навантаження при використанні невеликих машинних ресурсів (фізичних або віртуальних), має вбудований скінчений автомат станів для проектування більш складних чат-ботів, підтримує особливість Telegram API вебхук (можливість робити запити у відповідь на оновлення).

Для запитів до Telegram Bot API фреймворк використовує саме клієнтську частину веб-фреймворку aiohttp, що має перевагу над

використання вбудованої синхронної біблотеки urllib3 більше ніж в 10 разів (при використанні urllib3 в одному процесі) [20].

Клієнт для запитів заснований на Python Event Loop, що є нескінченим циклом, який має в собі багато обробників подій. Кожна подія – це виконання якогось коду, функції тощо, що використовує системні ресурси. Деякі ресурси такі як запит по мережі чи до бази даних блокуються (та званий Input/Output block) в Python за допомогою GIL (Global Interpreter Lock), що змушує всю програму чекати на відповідь мережі або бази даних наприклад.

Проте з використанням циклу подій та асинхронності програма попадаючи в ділянки коду, де відбувається I/O block замість того, аби чекати на відповідь від пристрою блокування, передає контроль інтерпретатору, який в цей час може виконувати іншу роботу не пов’язану з I/O, але пов’язану з використанням процесорного часу.

Потім коли інший обробник подій натрапляє на таку саму ділянку коду і віддає контроль інтерпретатору, він проходить по циклу і повертається до попереднього обробника, той в свою чергу перевіряє чи була отримана відповідь від I/O пристроїв. Таким чином відбувається пришвидшення програми, яка має багато ділянок коду з I/O блоками в декілька разів.

4.2.3 Розробка бекенду

На проектування та розробку бекенду було виділено більшу частину часу. А основним веб-фреймворком було обрано Flask, що є мікрофреймворком для створення клієнт-серверних додатків. Та pyTorch – фреймворк для створення та роботи з моделями машинного навчання.

Flask написаний на Python та класифікується як мікрофреймворк, оскільки не потребує конкретних інструментів чи бібліотек. У нього немає вбудованих інструментів готових для роботи з базою даних, її міграцією, підтримання моделей або перевірки та валідації форм користувача. Однак Flask підтримує розширення, які можуть додавати ці функції та інструменти в програму, що розробляється, так, ніби вони були реалізовані в самому фреймворку. Розширення існують для об'єктно-реляційних моделей, перевірки форм, обробки завантажень, різних технологій відкритої аутентифікації (oauth) та декількох інших загальних інструментів. Розширення оновлюються набагато частіше, ніж основна програма Flask [21].

Для розробки було використані такі розширення фреймворку, як Flask-Migrate та Flask-SQLAlchemy. Flask-Migrate використовується для створення, обробки та контролю міграції бази даних, тобто відповідає за коректність перенесення моделей програми в DDL (Data Definition Language) відповідно до обраної розробником СКБД. Тобто розробник згодом зможе змінити СКБД на будь-яку іншу (але реляційну) не переписуючи жодну стрічку коду, лише встановивши відповідні Python драйвери для роботи з новою СКБД. Також це розширення дозволяє створювати коміти міграції, на кшталт Git комітів. Це дозволяє дотримуватись чіткої логіки під час проектування та розширення об’єктно реляційних моделей програми.

Розширення Flask-SQLAlchemy використовується для побудови об’єктно-реляційних моделей та роботи з ними під час роботи самої програми. Воно так само дозволяє легко змінювати СКБД, не переписуючи програму, тому що само генерує запити до бази мовою DML (Data Manipulation Language). З переваг використання цього розширення – це простота створення моделей та їх зв’язків та головне, що всі запити, що робляться безпосередньо до бази даних самостійно оптимізовуються та

спрощуються декількома етапами. Це значно полегшує розробку та читабельність коду та підвищує продуктивність програмного забезпечення.

PyTorch - це бібліотека машинного навчання з відкритим кодом, заснована на бібліотеці Torch, яка використовується для таких програм класу computer vision та natural language processing. В першу чергу, він розроблявся лабораторією дослідження штучного інтелекту Facebook. Це безкоштовне та відкрите програмне забезпечення, що випускається під ліцензією Modified BSD. Хоча інтерфейс Python є основним напрямком розвитку, PyTorch також має інтерфейс для C ++.

На PyTorch було побудовано ряд програм Deep Learning, включаючи Pyro Uber, трансформери HuggingFace та Catalyst. PyTorch на противагу іншому популярному фреймворку машинного навчання Tensorflow більш відполірований та має більш гнучке та доступне API, що дозволяє працювати на Python в першу чергу як з мовою програмування, а не інтерфейсом командного рядка, яким забезпечує розробників Tensorflow API.

PyTorch надає дві функції високого рівня: тензорні обчислення (наприклад, NumPy) з сильним прискоренням за допомогою графічних процесорів (GPU) та глибокі нейронні мережі, побудовані на tape-based autodiff системі [22].

Також в роботі використано бібліотеки pdfminer.six та gunicorn. Gunicorn "Зелений Єдиноріг" – це сервер для HTTP-сервісів Python WSGI для UNIX. Це модель була портована від проекту Ruby’s Unicorn. Сервер Gunicorn широко сумісний з різними веб-фреймворками, просто реалізований, легкий для ресурсів фізичного сервера та досить швидкий.

Основною особливістю його – це те, що він відноситься до класу production-ready серверів, тобто таких, що готові «з-під коробки» працювати в режимі production. Його інші особливості: нативна підтримка WSGI, Django, Paster; автоматичне управління робочими процесами

програми; проста конфігурація через інтерфейс командного рядка або Python коду; підтримка хуків для розширення серверу; підтримка режиму роботу в декілька воркерів (master/slave або master/worker парадигма); сумісний зі всіма версіями Python починаючи з 3.5 [23].

Pdfminer.six бібліотека для швидкої обробки PDF документів за допомогою мови програмування Python. Було обрано для використання під парсингу (обробки тексту) текстової частини бази знань чат-боту, що зберігалася в форматі PDF. З особливостей цієї бібліотеки варто відмітити: перетворення кожного елементу документа на об’єкт в Python з унікальним інтерфейсом для кожного класу об’єктів; аналіз та групування тексту у вигляді зрозумілому людині (in a human-readable way); можливість обробки тексту, таблиць, зображень, вмісту тегів; підтримання усіх функцій специфікацій PDF версій 1.7 та вище; підтримання різних типів шрифтів та шифрувань RC4 та AES [24].

4.2.4 Робота з базою даних

Під час розробки основною СКБД було обрано реляційну базу даних Postgres та для реалізації швидкого пошуку було використано нереляційну базу даних Elasticsearch зі встановленим плагіном ingest.

PostgreSQL - це потужна об'єктно-реляційна база даних із відкритим кодом, яка використовує та розширює мову SQL у поєднанні з багатьма функціями, які безпечно зберігають та масштабують найскладніші навантаження даних. Витоки PostgreSQL відносяться до 1986 року в рамках проекту POSTGRES в Каліфорнійському університеті в Берклі та мають більш ніж 30 років активного розвитку на основній платформі.

PostgreSQL заслужив потужну репутацію за свою перевірену архітектуру, надійність, цілісність даних, надійний набір функцій,

розширюваність. PostgreSQL працює на всіх основних операційних системах, сумісний з ACID (atomicity, consistency, isolation, durability) з 2001 року і має потужні додатки, такі як популярний розширювач геопросторових баз даних PostGIS [25].

PostgreSQL оснащений багатьма функціями, спрямованими на допомогу розробникам у створенні програм, адміністраторам для захисту цілісності даних та побудови середовищ, стійких до відмов, та допомагають керувати даними незалежно від того, наскільки великий чи малий набір даних. А також можна визначати власні типи даних, створювати власні функції. Ці всі переваги й стали вирішальними під час вибору основної СКБД.

Elasticsearch - це розподілений пошук та аналітика для всіх типів даних, включаючи текстові, числові, геопросторові, структуровані та неструктуровані, що має відкритий код. Elasticsearch побудований на Apache Lucene і вперше був випущений у 2010 році. Відомий своїми простим REST API інтерфейсом запитів, розподіленістю, швидкістю та масштабованістю.

Elasticsearch є центральним компонентом Elastic Stack - набору інструментів з відкритим кодом для прийому даних, збагачення, зберігання, аналізу та візуалізації. Зазвичай називається стек ELK (Elasticsearch, Logstash і Kibana). Тепер стек включає в собі багату колекцію легких агентів, відомих як Beats для надсилання даних Elasticsearch.

Використовується для реалізації пошуку в додатках, веб-сайтах, корпоративних сервісах, ведення та аналіз логування систем, різних показників інфраструктур та моніторингу контейнерів, в аналізі геопросторових даних та їх візуалізації, аналітиці безпеки, бізнес-аналітиці. Архітектуру використання Elasticsearch можна описати таким чином. Сирі дані надходять у Elasticsearch з різних джерел, включаючи журнали, системні показники та веб-додатки.

Прийом даних - це процес, за допомогою якого ці необроблені дані аналізуються, нормалізуються та збагачуються до того, як вони будуть індексовані в Elasticsearch. Після індексації в Elasticsearch, користувачі можуть запускати складні запити щодо своїх даних та використовувати функції агрегації для отримання складних резюме своїх даних. Від використання Kibana користувачі можуть створювати потужні візуалізації своїх даних, обмінюватися своєю інформацією та керувати Elastic Stack'ом [26].

Ingest attachment плагін було використано для пришвидшення роботи Elasticsearch з документами загального формату (наприклад, PPT, XLS та PDF), використовуючи бібліотеку вилучення тексту Apache Tika. Також можна використовувати будь-який пристрій для зчитування даних напряму замість використання зчитування документів. Цей плагін дозволяє індексувати велику кількість документів будь-якого розміру та об'єму. При чому, для розробника API не змінюється, використовуються ті самі запити та функції, що і для роботи з основним ядром пошуку.

Джерело даних для плагіна повинно бути закодованим двійковим кодом base64. Також є можливість використання формату CBOR (для того аби не робити перетворення base64 кодування туди і назад), замість JSON і вказати поле як масив байтів замість представлення рядків. Потім процесор пропустить до декодування base64 автоматично [27].

4.3 Алгоритм та структура програми (опис файлів та інтерфейсу)

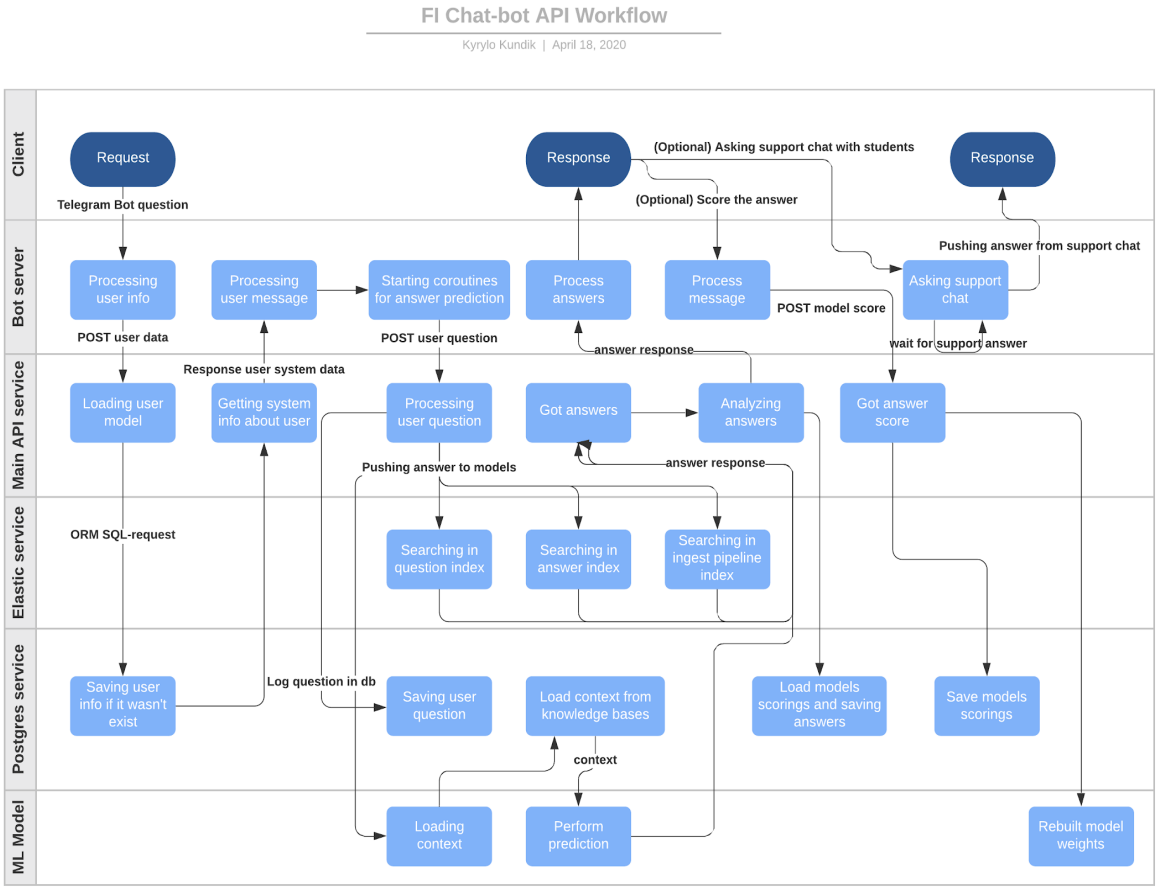

Рисунок 1.1 Опис процесу роботи чат-боту та його серверної частини

В основі розробки закладено мікросервісний підхід. Тобто розроблявся кожен сервіс окремо, незалежно один від одного та все запускається так само ізольовано одне від одного (кожен сервіс нічого не знає про інший та виконує суто свою функцію).

В першу чергу було спроектовано базу даних чат-бота та почала наповнюватися база знань (проведення анкетування, заповнення відповідей на найчастіші запитання студентів, підбір літератури та іншої необхідної інформації).

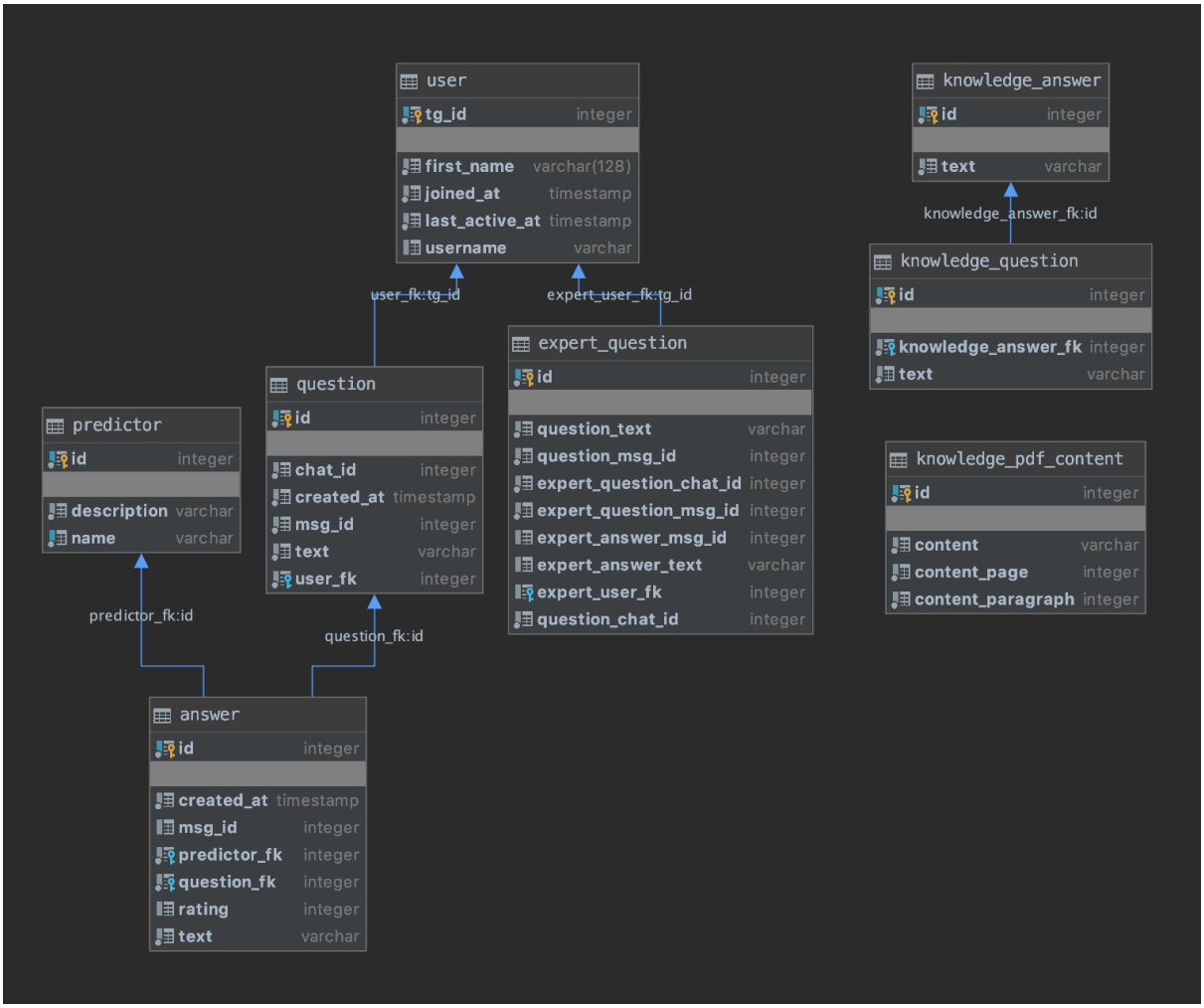

Рисунок 1.2 Модель даних для API сервісу

Наступним етапом було проектування та навчання моделі машинного навчання. Для цього було використано передтреновану модель BERT-Base, Multilingual Cased. Ця передтренована модель NLP розроблена Google та опублікована у 2018 році. Google використовує BERT для кращого розуміння пошукових запитів користувачів.

Для дотренування моделі за допомогою фреймворку pyTorch було спроектовано спеціальний клас, який і представляє цю мережу та надає інтерфейс для взаємодії з нею та отримання результатів її роботи.

```
def predict(self, passage: str, question: str):
    example = input_to_squad_example(passage, question)
    features = squad_examples_to_features(example, self.tokenizer, self.max_seq_length, self.doc_stride,
                                          self.max_query_length)
    all_input_ids = torch.tensor([f.input_ids for f in features], dtype=torch.long)
    all_input_mask = torch.tensor([f.input_mask for f in features], dtype=torch.long)
    all_segment_ids = torch.tensor([f.segment_ids for f in features], dtype=torch.long)
    all_example_index = torch.arange(all_input_ids.size(0), dtype=torch.long)
    dataset = TensorDataset(all_input_ids, all_input_mask, all_segment_ids,
                            all_example_index)
    eval_sampler = SequentialSampler(dataset)
    eval_dataloader = DataLoader(dataset, sampler=eval_sampler, batch_size=1)
```

Рисунок 1.3 Частина функції, що на основі контексту відповідає на запит користувача.

Після відлагодження та тестування моделі машинного навчання було спроектовано безпосередньо головний веб-сервіс з використанням архітектурного підходу розробки програмного забезпечення RESTful API. Тобто веб-сервіс дозволяє запитуючим системам отримувати доступ та маніпулювати текстовими поданнями веб-ресурсів за допомогою заздалегідь визначеного набору операцій без стану. Інші види веб-сервісів, такі як веб-сервіси SOAP, піддають власні довільні набори операцій.

```python
@main_bp.route("/search")
def perform_search():
    try:
        predictor = request.args.get("predictor")
        query = request.args.get("query")
        msg_id = int(request.args.get("msg_id"))
        user_id = int(request.args.get("user_id"))
        chat_id = int(request.args.get("chat_id"))
    except (KeyError, ValueError):
        abort(400)  # will raise an error
        return

    question_id = publish_question(query, user_id, chat_id, msg_id)

    try:
        answer = current_app.predictors_table[predictor](query, question_id)
        if answer:
            return jsonify({"success": True, "answer": answer.serialize})
        else:
            return jsonify({"success": False})
    except KeyError:
        abort(404)  # will raise an error
        return
```

Рисунок 1.4 Приклад реалізації ендпоінту пошуку відповіді на запитання

Після реалізації всіх необхідних ендпоінтів та моделей даних була розпочата робота над підключенням Elastic стеку та всього необхідного для його роботи. Для цього була використання бібліотека elasticsearch-python , що дозволяє у синхронному режимі робити запити до інтерфейсів Elasticsearch’а та його плагінів.

Для більш гнучкої розробки та подальшого перевикористання коду було вирішено зробити окремі моделі даних, що “під капотом” використовуватимуть індекси elasticsearch’а. Це дозволить веб-сервісу напряму працювати з моделями, а сама реалізація доступу до інтерфейсів пошуку буде прихована.

Для цього в Python’і широко використовуються так звані міксини (mixins). Mixins - це невеликі класи, які зосереджені на наданні невеликого набору специфічних функцій, які згодом можна поєднувати з кодом, який живе в інших класах. Це означає, що міксин завжди очікується використовувати разом з іншим кодом, який він буде покращувати або

налаштовувати, в свою чергу міксин не повинен використовуватись сам по собі.

```
class SearchableMixin:
    @classmethod
    def search(cls, expression):
        ids = query_index(cls.__tablename__, expression)

        if len(ids) == 0:
            return None

        when = []
        for i in range(len(ids)):
            when.append((ids[i], i))
        res = cls.query.filter(cls.id.in_(ids)).order_by(
            db.case(when, value=cls.id)
        ).all()
        return res[0] if len(res) > 0 else None
```

Рисунок 1.5 Фрагмент міксину, що впроваджує функції пошуку по elasticsearch індексам для об’єктно-реляційних моделей.

Після закінчення етапу розробки бекенду, тобто всіх необхідних сервісів для коректної роботи серверу була розпочата розробка фронтенд частини, тобто безпосередньо самого серверу, що відповідав за керування чат-боту через Telegram Bot API.

Цей сервер може також поділити на різні компоненти такі, як модуль функцій-обробників повідомлень користувача, модуль функцій підбору висловів у відповідь користувачу, клієнт для роботи з основним веб-сервісом та моделі запитів та відповідей веб-сервісу.

```
bot, dispatcher = setup_bot(os.getenv("API_TOKEN"))

phrase_handler = PhraseHandler()
api_client = ApiClient(
    api_url=f"http://{os.getenv('API_HOST')}:{os.getenv('API_PORT')}",
    loop=bot.loop
)
support_chat_id = int(os.getenv("SUPPORT_CHAT_ID"))

predictors = bot.loop.run_until_complete(api_client.get_all_predictors())

answer_cb = CallbackData('answer_callback', 'id', 'rating', 'predictor')


@dispatcher.message_handler(commands=["start"])
async def send_welcome(message: types.Message):
    await bot_typing(bot, message.chat.id)

    await bot.send_message(
        message.chat.id,
        phrase_handler.get_phrase(PhraseTypes.WELCOME_PHRASE),
        parse_mode="Markdown"
    )
```

Рисунок 1.6 Фрагмент коду початкової ініціалізації боту та оброблювач команди телеграму /start

Для підвищення швидкодії фреймворку чат-бота було використані такі доповнення для бібліотеки, яку використовує фреймворк для клієнтських HTTP запитів, а саме: uvloop (замінює вбудований цикл подій Python asyncio event loop на інший швидкіший, написаний мовою C), ujson (бібліотека для прискорення завантаження тексту формату JSON у Python тип даних словник, також написаний мовою C), aiohttp[speedups] (інші бібліотеки для прискорення клієнтських запитів, по типу aiohttp-dns, що замінює вбудовані функції обробки DNS таблиць).

```python
async def update_answer(self, answer: Answer) -> bool:
    await self.fetch(
        method=f"/answer/{answer.id_}",
        verb="PUT",
        payload={
            "msg_id": answer.msg_id,
            "rating": answer.rating,
        }
    )

    return True
```

Рисунок 1.7 Фрагмент коду для оновлення відповіді після отримання оцінки від користувача.

Завершальним етапом було створення Docker-контейнерів та використання docker-compose стеку для розгортання сервісів на дроплеті DigitalOcean з версією Ubuntu 18.04 (bionic).

```
bot:
  build:
    context: ./bot
  container_name: chatbot
  restart: always
  links:
    - web
  depends_on:
    - web
  environment:
    API_TOKEN: $BOT_API_TOKEN
    API_HOST: $API_HOST
    API_PORT: $WEB_APP_PORT
    SUPPORT_CHAT_ID: $SUPPORT_CHAT_ID
  networks:
    - backend
```

```
web:
  build:
    context: ./web_service
    args:
      PORT: $WEB_APP_PORT
      HOST: "0.0.0.0"
  container_name: chatbot_web
  environment:
    APP_HOST: "0.0.0.0"
    APP_PORT: $WEB_APP_PORT
    PG_HOST: $PG_HOST
    PG_PORT: $POSTGRES_PORT
    PG_DB: $POSTGRES_DB
    PG_USER: $POSTGRES_USER
    PG_PASS: $POSTGRES_PASS
    ES_HOST: $ES_HOST
    ES_PORT: $ELASTIC_PORT
    ES_USER: $ES_USER
    ES_PASS: $ES_PASS
```

Рисунок 1.8 Фрагменти опису docker-compose сервісів: сервер чат-боту (зліва), веб-сервіс (справа).

4.4 Тестування програми та результати її виконання

Для більш детального ознайомлення з результатом практичної частини роботи, розглянемо декілька сценаріїв взаємодії користувача з чат-ботом.

В описі чат-боту поданий короткий опис його можливостей. Для зображення боту використано дизайн, розроблений для факультету інформатики НаУКМА. Знайти бот у пошуці в месенджері Telegram можна ввівши його назву: «FIChatbot».

Для початку роботи з ботом, користувач повинен натиснути на «Send message».

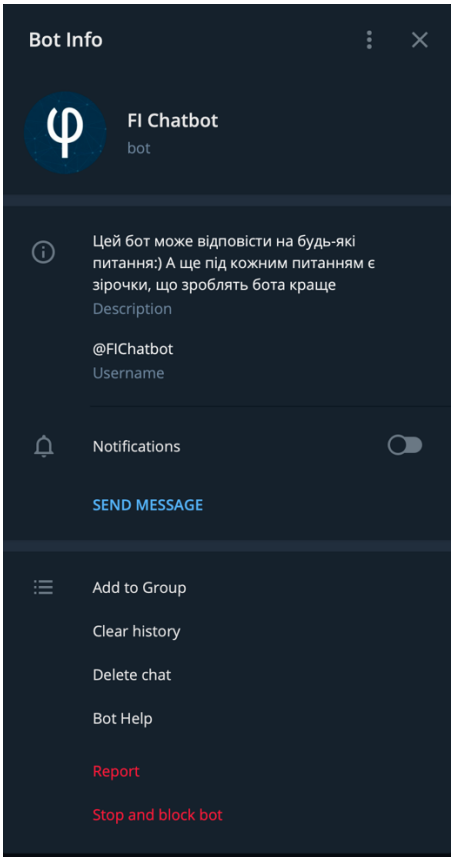

Рисунок 1.9 Опис чат-бота

Розглянемо перший сценарій: користувач вітається з ботом.

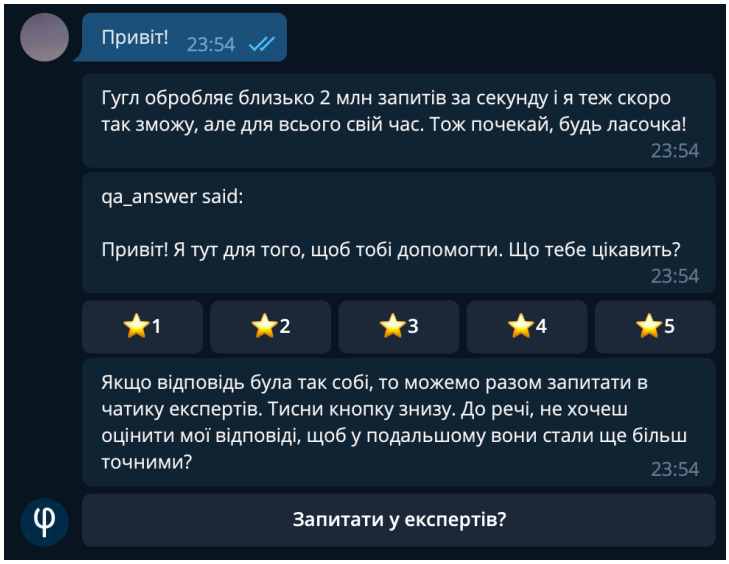

Рисунок 1.10 Сценарій спілкування користувача з ботом №1

Відповідь говорить про те, що бот підлаштовується під комунікацію з людиною, надаючи відчуття живого спілкування. Перша відповідь (див. перше повідомлення від боту на рис.1.10) є загальною і несе в собі інформацію про те, що бот досі на стадії навчання та розробки. Друга відповідь – це власне відповідь на питання користувача. Відразу надається можливість оцінити наскільки відповідь була корисною. Третє повідомлення – це пояснення до оцінки і можливість зв'язатись із експертом.

Можливість отримати зворотний зв'язок від людини, а не лише від боту, є дуже важливою функцією. Оскільки, не усі питання можуть бути легко розв'язані за допомогою пошуку або навчання.

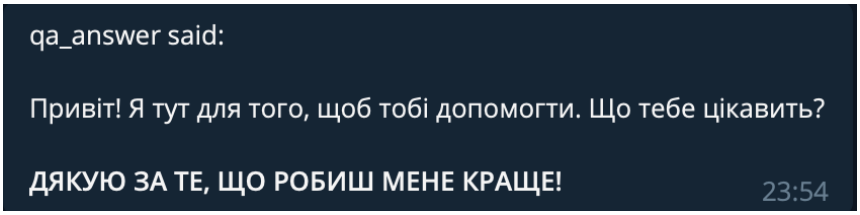

Рисунок 1.11 Оцінка відповіді бота

Після оцінки бота, з'являється повідомлення з словами вдячності. Такі деталі створюють позитивний досвід клієнта і знову ж таки – імітують спілкування з людиною. Завдяки оцінкам, відповіді бота будуть покращуватись. Але в той же час, не можна відкидати похибку неправильної оцінки.

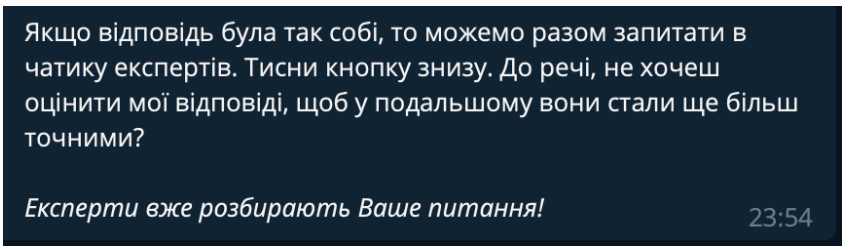

Рисунок 1.12 Запит до експерта

Якщо відповідь на питання не підходить, користувач може натиснути «Запитати у експертів». Ця функція надає можливість отримати відповідь від експерта, тобто людини. Легкий у користуванні функціонал не потребує додаткового трактування чи створення інструкцій.

Варто також відзначити швидкість із якою надходять відповіді від бота. Середня швидкість відповідей від бота складає близько 5 секунд. Поки система підбирає потрібні відповіді, користувач бачить повідомлення про те, що бот «друкує».

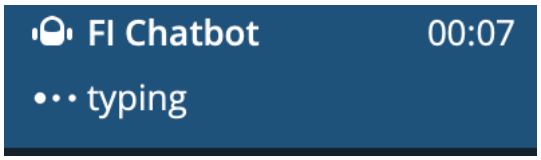

Рисунок 1.13 Очікування відповіді від бота

Бот не завжди працює точно, але у більшості випадків, відповідь можна знайти прочитавши підібрані фрагменти тексту. Приклад цього, можна побачити на рисунку 1.14.

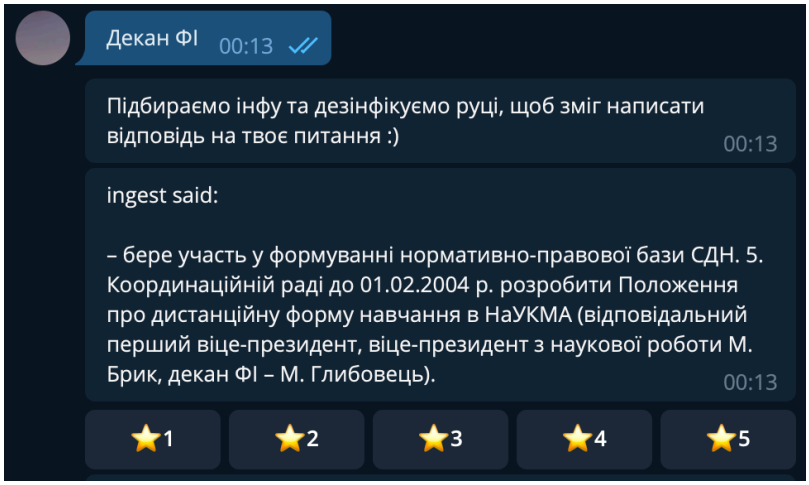

Рисунок 1.14 Відповідь бота на «Декан ФІ»

ВИСНОВКИ

Результатом курсової роботи став чат-бот для месенджеру Telegram, основною ціллю якого є легкий та швидкий доступ до інформації про факультет інформатики НаУКМА та університет.

Під час написання курсової роботи було досліджено типи та види чат-ботів, алгоритмів, що лежать в їх основі та принципи розробки чат-ботів для месенджерів. Вивчено основи NLTK та розглянуто вже готові системи. Окремий розділ присвячено дослідженню ботів, які розроблено в цілях онлайн-довідників для пошуку інформації та ботів асистентів. Описано логіку чат-боту.

У четвертому розділі курсової роботи детально описано та обґрунтовано вибір використаних технологій. Описано процес розробки фронтенду та бекенду програми, розглянуто роботу з базою даних. Додатково подано алгоритм роботи чат-боту та структуру програми. Проведено тестування чат-боту.

В перспективі, бот допоможе мінімізувати багато рутинних процесів, які зараз існують під час виборчої кампанії та на початку навчального року. Бот стане швидким способом дізнатись інформацію без залучення студентів, викладачів чи адміністрації НаУКМА.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. [Електронний ресурс] Five Different Types of Chatbot https://medium.com/voiceui/five-different-types-of-chatbot-17bb255b23b4
2. [Електронний ресурс] Natural Language Processing with Python http://www.nltk.org/book/
3. [Електронний ресурс] A brief history of Chatbots https://chatbotslife.com/a-brief-history-of-chatbots-d5a8689cf52f
4. [Електронний ресурс] What are Chatbots and Why are They Becoming so Popular? https://www.uctoday.com/contact-centre/what-are-chatbots-and-why-are-they-becoming-so-popular/
5. [Електронний ресурс] Chatbot Report 2019: Global Trends and Analysis https://chatbotsmagazine.com/chatbot-report-2019-global-trends-and-analysis-a487afec05b
6. [Електронний ресурс] What Are The Inner Workings of a Chatbot? https://chatbotsmagazine.com/what-is-the-working-of-a-chatbot-e99e6996f51c
7. [Електронний ресурс] Building a Simple Chatbot from Scratch in Python (using NLTK) https://medium.com/analytics-vidhya/building-a-simple-chatbot-in-python-using-nltk-7c8c8215ac6e
8. [Електронний ресурс] Watson Assistant https://www.ibm.com/cloud/watson-assistant/
9. [Електронний ресурс] Dialogflow Case Studies https://dialogflow.com/case-studies/
10. [Електронний ресурс] Dialogflow vs Lex vs Watson vs Wit vs Azure Bot | Which Chatbot Service Platform To Use? https://www.kommunicate.io/blog/dialogflow-vs-lex-vs-watson-vs-wit-vs-azure-bot/

11. [Електронний ресурс] Chatbots: Are they Really Useful? https://jlcl.org/content/2-allissues/20-Heft1-2007/Bayan_Abu-Shawar_and_Eric_Atwell.pdf
12. [Електронний ресурс] Deep Learning Chatbots: Everything You Need to Know https://hackernoon.com/deep-learning-chatbot-everything-you-need-to-know-r11jm30bc
13. [Електронний ресурс] Recipes for apps you can talk to https://wit.ai/docs/recipes
14. [Електронний ресурс] What is a Container? https://www.docker.com/resources/what-container
15. [Текстовий ресурс] Кундік К.В. Розробка веб-сервісу інтернет оголошень / Кундік К.В., Гречко А.В. – К.: Києво-Могилян. акад., 2019. – 19 с.
16. [Електронний ресурс] Welcome to python.org https://www.python.org/
17. [Електронний ресурс] API Telegram https://api.telegram.org/bot<token>/METHOD_NAME
18. [Електронний ресурс] Telegram Bot API - https://core.telegram.org/bots/api
19. [Електронний ресурс] Welcome to aiogram's documentation!- https://aiogram.readthedocs.io/
20. [Електронний ресурс] Python and fast HTTP clients https://julien.danjou.info/python-and-fast-http-clients/
21. [Електронний ресурс] Welcome to Flask – Flask Documentation https://flask.palletsprojects.com/en/1.1.x/
22. [Електронний ресурс] PyTorch https://pytorch.org/
23. [Електронний ресурс] Welcome to pdfminer.six's documentation! https://pdfminersix.readthedocs.io/en/latest/]
24. [Електронний ресурс] Gunicorn - WSGI server https://docs.gunicorn.org/en/stable/

25. [Електронний ресурс] PostgreSQL: The World's Most Advanced Open Source Relational Database https://www.postgresql.org/
26. [Електронний ресурс] Get Started with Elasticsearch https://www.elastic.co/
27. [Електронний ресурс] Ingest Attachment Processor Plugin https://www.elastic.co/guide/en/elasticsearch/plugins/master/ingest-attachment.html

ДОДАТКИ

Додаток №1