КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

імені ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»

(шифр і назва спеціальності)

«Прикладне програмування»

(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Чат-бот для надання освітніх консультацій»

Виконав _____ _____

(Підпис)

Шимотюк Євген Олександрович

(прізвище, ім'я, по батькові)

Керівник Броварець Олександр Олександрович

(прізвище, ім'я, по батькові)

_____

(Резолюція «До захисту»)

Попередній захист:

_____

(Висновок: "До захисту в екзаменаційній комісії ")

Завідувач кафедри _____ Плескач В.Л.

(Підпис) (Прізвище, ініціали) (Дата)

Київ – 2021

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра прикладних інформаційних систем

Освітня програма: Прикладне програмування

Спеціальність: Комп'ютерні науки

Назва теми: «Чат-бот для надання навчальних консультацій»

ПІБ	**Підпис**
Шимотюк Євген Олександрович	

Назва роботи українською та англійською мовами

Чат-бот для надання навчальних консультацій Chatbot for providing educational consultations

Мета бакалаврської роботи, завдання

Мета бакалаврської роботи: Реалізація чат-бота для надання навчальних консультацій.

План роботи:

1. Проаналізувати сучасні підходи до розроблення і впровадження чат-ботів
2. Аналіз архітектурних рішень і вибір програмних засобів для реалізації чат-бота
3. Програмна реалізація чат-бота для надання навчальних консультацій

ПІБ, ступінь, звання наукового керівника роботи: Броварець Олександр Олександрович, доцент, кандидат технічних наук

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Номер	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	26.10.2020	
2.	Видача завдання кваліфікаційної роботи бакалавра	23.11.2020	заява
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	01.12.2020	
4.	Затвердження плану кваліфікаційної роботи бакалавра	18.02.2021	
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	25.02.2021	
6.	Підготовка і подання науковому керівнику першого варіанту I розділу роботи	05.03.2021	
7.	Підготовка і подання науковому керівнику першого варіанту II розділу роботи	09.04.2021	
8.	Підготовка і подання науковому керівнику першого варіанту III розділу роботи	07.05.2021	
9.	Подання роботи у першому варіанті	11.05.2021	
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	12.05.2021	
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**24.05.2021**	
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	28.05.2021	

13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	11.06.2021	
14.	Захист кваліфікаційної роботи бакалавра	16.06.2021	

Здобувач вищої освіти _____ _____(підпис)
Керівник _____(підпис)

АНОТАЦІЯ

Дипломна робота: 55 с., 6 рис., 2 табл., 8 джерел, 6 додатків

Ця дипломна робота присвячена проектуванню та розробленню чат-бота для надання навчальних консультацій.

Метою дипломної роботи є ефективне отримання необхідної інформації за допомогою чат-бота для надання навчальних консультацій.

Поставлену мету мети можна реалізувати вирішивши наступні **завдання**:

- дослідити загально-теоретичні засади понять чат-ботів та класифікації тексту;
- здійснити аналіз програмно-технологічних рішень побудови чат-ботів для надання інформаційних консультацій;
- спроектувати, реалізувати, впровадити чат-бота для надання навчальних консультацій з урахуванням інженерії вимог;

Об’єкт дослідження.

Процеси надання навчальних консультацій.

Предмет дослідження.

Програмні продукти, технічні засоби, організаційні заходи щодо побудови чат-бота для надання навчальних консультацій.

Методи дослідження.

Теорія для дослідження теоретичних аспектів надання навчальних консультацій, емпіричний аналіз і синтез чат-ботів, що застосовувався при вивченні прикладів сучасних методів побудови чат-ботів, аналогії залучені в процесі проектування, розробки та побудови власного чат-бота, метод порівняння, щоб проаналізувати наявні ресурси та чат-боти.

Ключові слова: чат-бот, текст, класифікація, дані.

SUMMARY

Thesis: 55 pages, 6 figures, 2 tables, 8 sources, 6 appendices

This thesis is about the design and implementation of a chatbot for training consultations.

The purpose of the thesis is to effectively obtain the necessary information through a chatbot to provide training advice.

To achieve this goal must solve the following **tasks**:

- explore the general theoretical foundations of the concepts of chatbots and text classification;
- to do the analysis of technological and software considerations of construction of chatbots for rendering of information consultations;
- design, implement, implement a chatbot to provide training advice based on engineering requirements;

Object of study.

Processes of providing training consultations.

Subject of study.

Technical and software, organizational principles, approaches of building a chatbot for training.

Research methods.

Theory for the study of theoretical aspects of training, empirical analysis and synthesis of chatbots, used in the study of examples of modern methods of building chatbots, analogies involved in the design, development and construction of your own chatbot, the method of comparison used for analysis available resources and chatbots.

Keywords: chatbot, text, classification, data.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

Скорочення:

CSS – Cascading Style Sheets –каскадні таблиці стилів.

HTML – HyperText Markup Language –мова гіпертекстової розмітки.

СУБД – Система управління базами даних.

API – Application Programming Interface–інтерфейс програмування.

Терміни:

Ендпоінт – кінцева точка, до якої відбувається звернення віддаленим HTTP методом.

Фреймворк – це програмне оточення спеціального призначення, каркас, який полегшує процес об’єднання компонентів при створенні програми.

Утиліта – невеличка прикладна программа.

ВСТУП

Чат-бот — програма, яка відповідає на повідомлення в чаті певного месенджера з метою забезпечення інформаційних потреб користувачів. Насьогодні відбувається загальний перехід до інформаційного суспільства, розвиток якого невід’ємно пов'язаний із інтенсифікацією процесів обміну даними, необхідністю збору, обробки і передавання величезних обсягів інформації.

Чат-боти є важливими, тому що це сильно спрощує роботу персоналу, робить зручнішим здійснення покупок користувачами та зменшує кількість часу, необхідного для цього. Також це дає можливість забезпечувати свої потреби, не покидаючи додаток свого улюбленого месенджера.

Актуальність теми цієї бакалаврської роботи зумовлено популярністю месенджерів та кількістю інформаційних потреб користувачів.

Мета бакалаврської роботи - це побудова ефективного чат-бота для надання навчальних консультацій. Для цього необхідно виконати наступні завдання:

- Дослідити теоретичні засади реалізації чат-бота для надання інформаційних консультацій;
- Проаналізувати технічні рішення та різні види забезпечення чат-бота;
- Спроектувати і впровадити чат-бота для надання навчальних консультацій.

Об’єктом дослідження бакалаврської роботи є чат-бот для надання навчальних консультацій, також його рішення з точки зору функціональності, засади та підходи для дизайну чат-бота з визначеною функціональністю.

Предметом дослідження бакалаврської роботи є технічні, організаційні засади, принципи, концептуальні підходи до побудови чат-бота для надання навчальних консультацій.

У процесі виконання бакалаврської роботи за основу було взято мову програмування Python, необхідні бібліотеки для цієї мови програмування та їх документацію.

Дипломна робота складається зі вступу, трьох розділів, висновку і списку використаних джерел.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ПОБУДОВИ ЧАТ-БОТА ДЛЯ НАДАННЯ НАВЧАЛЬНИХ КОНСУЛЬТАЦІЙ

1.1 Огляд і аналіз програмних систем, принципів і засобів створення чат-бота для надання навчальних консультацій

Насьогодні чат-боти є популярними по всьому світу, тому що вони спрощують комунікацію клієнта з компанією, необхідну для отримання певних послуг. Їх використовують для досягнення різних цілей: отримання консультації на певну тему, здійснення оплати товарів, отримання оповіщень і т.д. Чат-бот для надання навчальних консультацій надає інформацію у вигляді тексту (або інших медіа ресурсів) або посилання на сайт із потрібною інформацією у відповідь на повідомлення користувача. Існує багато ботів, які надають навчальні консультації. Усі вони мають свої переваги та недоліки. Аналіз деяких із них значно допоміг у створенні власного бота. Для проведення аналізу було взято наступні чат-боти:

- “Andy English Bot” (https://t.me/andyrobot)
- “Imperium Romanum Bot” (https://t.me/imperiumromanumbot)

1. “Andy English Bot”

Телеграм-бот, метою якого є допомогти у вивченні англійської мови. Він пропонує різні ігри для збільшення словникового запасу та практикування граматики. Деякі ігри та граматика доступні лише у відповідному мобільному додатку, тому я розгляну функціональність для вивчення нових слів. Після відправлення повідомлення з командою “vocabulary”, бот відповідає із інструкцією щодо того, як зупинити гру. Після чого він надсилає слово його значення та вимову окремим голосовим повідомленням. Також бот пропонує варіанти відповіді на його повідомлення, а саме: прохання пояснити слово детальніше, перейти до наступного слова або пропустити це слово, якщо ти його вже знаєш.

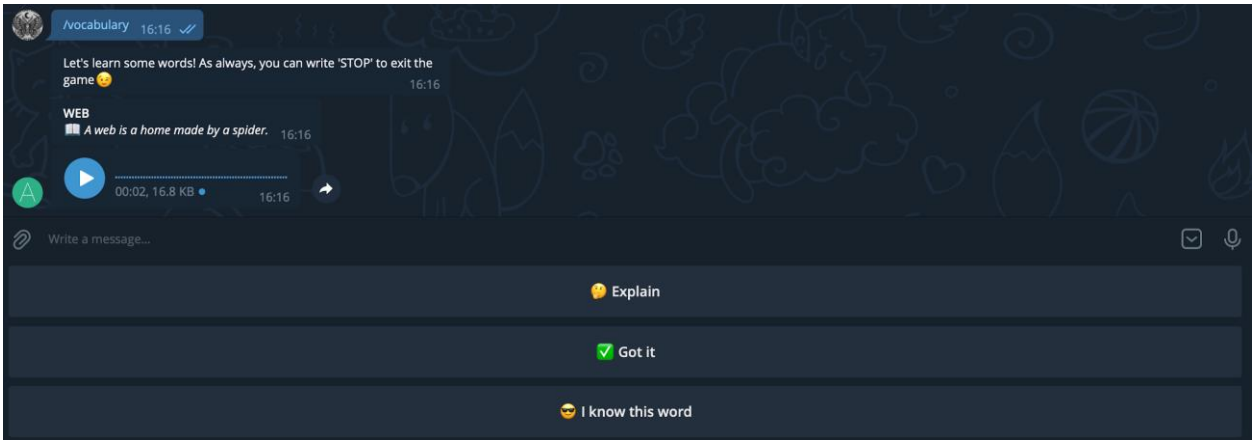

Рисунок 1.1.1. “Andy English Bot”

Переваги:

- простий у використанні

Недоліки:

- мало можливостей, якщо ти хочеш використовувати лише чат-бота, а не мобільний додаток

2. “Imperium Romanum Bot”

Телеграм-бот, який допоможе вам отримати доступ до даних з https://vici.org. vici.org - це безкоштовна база даних археологічних місць та римських артефактів. Бот дає можливість шукати археологічні місця за ключевим словом. Після відправлення повідомлення з командою “search” та ключевим словом ви отримаєте результати пошуку та кнопки вибору необхідного місця у відповідь. Також можна отримати інформацію щодо об'єкту з допомогою його id використовуючи команду “f<id>”.

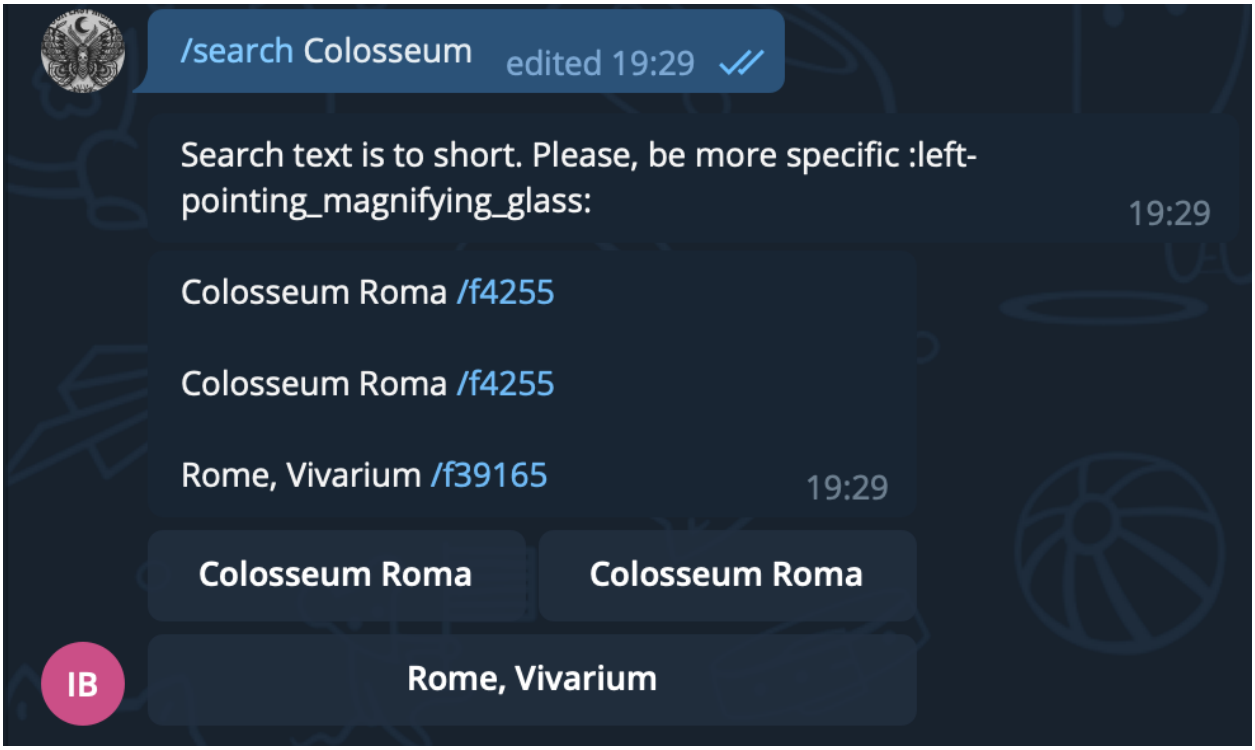

Рисунок 1.1.2. (а) “Imperium Romanum Bot” початок пошуку

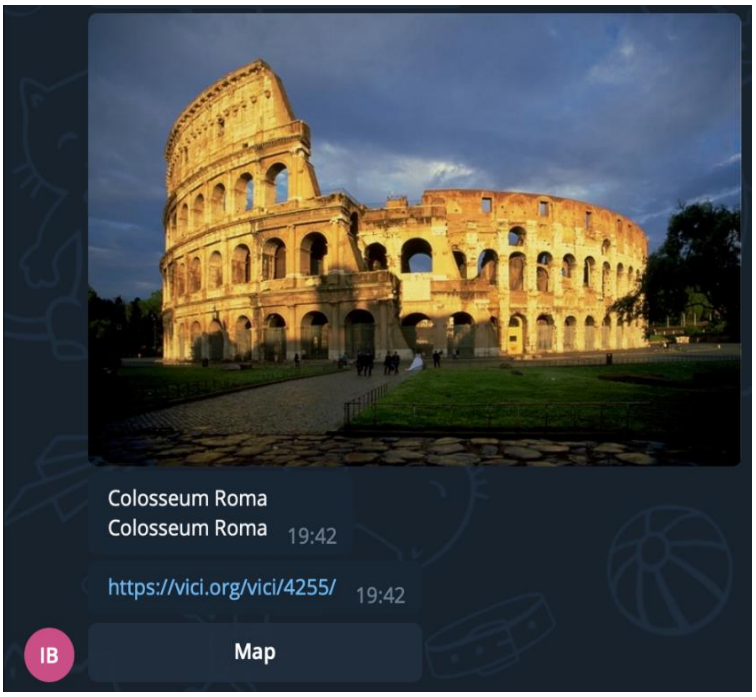

Рисунок 1.1.2. (б) “Imperium Romanum Bot” результат пошуку

Переваги:

- простий у використанні

Недоліки:

- насьогодні не підтримується

1.2 Обґрунтування мети вирішення поставленої задачі

На сучасному етапі розвитку технологій часто виникає необхідність спілкуватися з чат-ботами, оскільки вони є дуже зручним та одним із найбільш популярних способів отримати корисну інформацію на певну тему. У тому числі часто використовуються і чат-боти для надання навчальних консультацій, які допомагають людям отримати консультацію на певну конкретну тему. Наприклад, коли людина звертаються до консультанта банку щодо її картки. Замість консультанта, набагато простіше звернутися до чат-бота, який швидше допоможе та буде готовим надавати інформацію цілодобово.

Метою є створення чат-бота для надання навчальних консультацій, впровадження повної функціональності, аналіз схожих ботів, які існують, а також додавання нових можливостей для покращення роботи звичайного користувача − мета цієї бакалаврської роботи.

1.3 Особливість програмного забезпечення

Хоча чат-боти і є дуже популярними, але зазвичай вони здатні відповідати лише на конкретні питання (точні фрази або команди). Особливість цього програмного забезпечення є те, що бот не прив’язаний до конкретних фраз, а аналізує текст повідомлення користувача і надає відповідну інформацію.

Основною ідеєю є реалізація бота, який здатен відповідати на повідомлення з будь-яким текстом, надаючи необхідну інформацію користувачеві.

1.4 Постановка задачі. Технічне завдання на розробку

Основним завданням бакалаврської роботи є створення чат-бота для надання навчальних консультацій, який матиме повну функціональність для надання

користувачеві необхідної інформації відповідно до його потреби. Для досягнення поставленої мети необхідно послідовно виконати такі завдання:

- здійснити аналіз наявних додатків, які мають функціональність для надання навчальних консультацій
- знайти переваги наявних програм та додати їх до майбутньої
- функціональності чат-бота
- знайти недоліки в наявних додатків та звернути на них увагу або
- вилучити ці недоліки із чат-бота
- створення діаграми з готовою функціональністю чат-бота

РОЗДІЛ 2. ПРОЕКТНІ ТА ТЕХНІЧНІ РІШЕННЯ. ЗАБЕЗПЕЧЕННЯ ПРОЕКТОВАНОГО ЧАТ-БОТА

2.1 Інформаційне забезпечення проектованого чат-бота

Для реалізації чат-бота, була використана мова програмування Python та відповідні бібліотеки для створення телеграм-ботів, машинного навчання, BigQuery, машинного навчання та обробки природньої мови. Ці технології дають можливість виконати всі необхідні вимоги поставленого технічного завдання. Із допомогою основних відомостей про дані технології будуть надані відповіді, як та задля чого вони були задіяні.

Python - це інтерпретована мова програмування високого рівня та загального призначення. Дизайн мови Python має на мені значно покращити читабельність коду шляхом використання великої кількості пробілів. Його мовні конструкції та об’єктно-орієнтований підхід мають на меті допомогти програмістам писати чіткий та логічний код для малих та великих проектів. [1]

Python - мова програмування з багатьома парадигмами. Мова повністю підтримує об'єктно-орієнтовану та структурну парадигми програмування, багато її функцій підтримують функціональну парадигму програмування та аспекто-орієнтоване програмування (враховуючи метапрограмування). Багато інших парадигм підтримуються, використовуючи різні модулі (розширення), враховуючи контрактний дизайн та логічне програмування.

Наприкінці 1980-х років історія мала бути написана. Це був той час, коли почалася робота над Python. Незабаром після цього, Гвідо Ван Россум почав виконувати роботу, засновану на програмах, у грудні 1989 року в Центрі Віскунде та Інформатика (CWI), який знаходиться в Нідерландах. Його почали спочатку як хобі-проект, оскільки він шукав цікавий проект, який би зайняв його під час Різдва. Мовою програмування, яку, як кажуть, вдалося досягти Python, є ABC Programming Language, яка мала взаємодію з операційною системою Amoeba і мала функцію

обробки винятків. Мова вже допомогла створити ABC на початку своєї кар'єри, і він бачив деякі проблеми з ABC, але йому подобалася більшість функцій. Після цього те, що він зробив, було дуже розумним. Він взяв синтаксис ABC та деякі його хороші риси. Він також мав багато скарг, тому він повністю виправив ці проблеми і створив гарну мову сценаріїв, яка усунула всі недоліки. Натхненням для назви послужило телевізійне шоу Бі-Бі-Сі - «Летючий цирк Монті Пайтона», оскільки йому дуже подобалися різні телешоу та він також хотів коротке, унікальне і трохи загадкове ім’я для свого проекту, тому назвав його Python! Він був «Доброзичливим диктатором на все життя» (BDFL), поки 12 липня 2018 року він не пішов з посади лідера. Довгий час він працював у Google, але зараз працює в Dropbox.

Python був нарешті випущений у 1991 році. Коли він вийшов, він використовував набагато менше кодів для вираження концепцій, коли ми порівнюємо його з Java, C ++ і C. Його філософія дизайну теж була досить хорошою. Його головна мета - забезпечити читабельність коду та вдосконалену продуктивність розробників. Коли він був випущений, він мав більш ніж достатньо можливостей забезпечити класи успадкуванням, обробкою винятків кількох основних типів даних та функціями.

Мова використовує текст, уведений динамічно, та комбінацію підрахунку посилань та сміттєзбирач, що визначає цикл, для управління пам’яттю. Також вона має динамічну здатність розділення імен (пізнє зв'язування), яка прив'язує імена методів та змінних при виконанні програми.

Дизайн і філософія Python вплинули на багато інших мов програмування:

- Boo використовує відступ, подібний синтаксис та подібну об'єктну модель.
- Cobra використовує відступи та подібний синтаксис, а в його документі «Подяка» Python перераховано першим серед мов, які вплинули на нього.
- CoffeeScript, мова програмування, яка перешикується до JavaScript, має синтаксис, натхненний Python.

- ECMAScript / JavaScript запозичив ітератори та генератори у Python.
- GDScript, мова сценаріїв, дуже схожа на Python, вбудована в ігровий движок Годот.
- Go розроблений для "швидкості роботи на такій динамічній мові, як Python" і має однаковий синтаксис для нарізування масивів. Groovy було мотивовано бажанням перенести філософію дизайну Python на Java.
- Julia була розроблена, щоб бути "такою ж придатною для загального програмування, як Python".
- Nim використовує відступи та подібний синтаксис.
- Творець Рубі, Юкіхіро Мацумото, сказав: "Я хотів, щоб мова сценаріїв була потужнішою, ніж Perl, і більш об'єктно-орієнтованою, ніж Python. Ось чому я вирішив створити власну мову".
- Swift, мова програмування, розроблена Apple, має певний синтаксис, натхненний Python.

Практики розробки Python також наслідували інші мови. Наприклад, практика вимагання документа, що описує обгрунтування та проблеми, пов’язані зі зміною мови (на Python, PEP), також використовується в Tcl, Erlang, та Swift.

Мова програмування Python спочатку не була розроблена для чисельних обчислень, але на початку привернула увагу наукової та інженерної спільноти. У 1995 році була заснована matrix-sig групи спеціальних груп інтересів (SIG) з метою визначення пакету обчислень масивів; серед його членів був дизайнер та супровідник Python Гвідо ван Россум, який розширив синтаксис Python (зокрема синтаксис індексації), щоб полегшити обчислення масивів.

Реалізація пакету матриць була завершена Джимом Фултоном, потім узагальнена Джимом Гугуніним і названа Numeric (також по-різному відома як "Розширення числових Python" або "NumPy"). Гугунін, аспірант Массачусетського технологічного інституту (MIT), приєднався до Корпорації національних

дослідницьких ініціатив (CNRI) у 1997 році, щоб працювати над JPython, залишивши Пола Дюбуа з Національної лабораторії Лоуренса Лівермора (LLNL), щоб взяти на себе роль супровідника. Серед інших ранніх авторів - Девід Ашер, Конрад Хінсен та Тревіс Оліфант.Новий пакет під назвою Numarray був написаний як більш гнучка заміна Numeric. Як і Numeric, він також застарів. Numarray мав більш швидкі операції для великих масивів, але був повільнішим, ніж Numeric для малих, тому певний час обидва пакети використовувались паралельно для різних випадків використання. Остання версія Numeric (v24.2) була випущена 11 листопада 2005 року, тоді як остання версія numarray (v1.5.2) вийшла 24 серпня 2006 року.Було бажання потрапити Numeric до стандартної бібліотеки Python, але Гвідо ван Россум вирішив, що тоді код не підтримувався у своєму стані.На початку 2005 року розробник NumPy Тревіс Оліфант хотів об'єднати спільноту навколо єдиного пакету масивів і переніс функції Numarray на Numeric, випустивши результат як NumPy 1.0 у 2006 році. Цей новий проект був частиною SciPy. Щоб уникнути встановлення великого пакету SciPy лише для отримання об’єкта масиву, цей новий пакет був відокремлений і названий NumPy. Підтримка Python 3 була додана в 2011 році з NumPy версії 1.5.0.У 2011 році PyPy розпочав розробку впровадження API NumPy для PyPy. Він ще не повністю сумісний з NumPy.

NumPy націлений на стандартну реалізацію CPython, яка є інтерпретатором байт-коду без оптимізацій. Алгоритми математики які, написані для даної версії Python, зазвичай працюють значно повільніше, ніж вибрані еквіваленти. Бібліотека NumPy вирішує різні проблеми повільності частково, шляхом надання багатовимірних масивів, функцій та операторів, які ефективно співпрацюють із масивами. Для їх використання необхідно переписати частину коду, переважно внутрішні цикли, за допомогою модулю NumPy.

Задієння модулю NumPy у мову Python дає функціональність, яку можна порівняти з MATLAB, оскільки обидва варіанти інтерпретуються, і обидва вони

дають можливість користувачеві писати програми які працюють дуже швидко, якщо майже всі операції здійснюються над масивами або матрицями замість скалярних типів даних. Задля порівняння, MATLAB має можливість похвалитися значною кількістю нестандартних наборів інструментів, враховуючи Simulink, тим часом як NumPy є невід'ємно інтегрованою з мовою Python, більш сучасною та повноцінною мовою програмування. Більш того, у доступі є додаткові пакети для Python; Модуль SciPy - це бібліотека, яка надає більше подібних до MATLAB функціональних можливостей, а бібліотека Matplotlib - це пакет, який використовується для побудови графіків функцій, який надає функціональність, схожу з MATLAB.

Із середини MATLAB і NumPy покладаються на BLAS і LAPACK для ефективного обчислення операцій лінійної алгебри. Зв'язки Python до широко використовуваного модулю комп'ютерного бачення OpenCV використовують масиви даних NumPy для того, щоб зберігати та працювати з даними. Оскільки цифрові зображення із кількома каналами є завди просто представленими у вигляді трьохвимірних масивів, індексація, нарізування чи маскування використовуючи інші масиви є дуже ефективними варіантами доступу до певних пікселів у зображенні. Масив даних NumPy є універсальною структурою даних в OpenCV для зображень, які є витягнутими точками об’єктів, фільтрів та багато іншого, він сильно спрощує процес роботи програмування та налагодження.

python-telegram-bot - це бібліотека, яка забезпечує чистий Python інтерфейс для Telegram Bot API. У додаток до реалізації API, ця бібліотека має ряд класів високого рівня, що роблять розробку ботів простою та зрозумілою. [2]

NumPy - це основний пакет, необхідний для наукових обчислень з Python. [3] Він забезпечує:

- потужний N-вимірний об'єкт масиву
- складні (мовні) функції

- інструменти для інтеграції коду C / C ++ та Fortran
- корисна лінійна алгебра, перетворення Фур'є та можливості випадкових чисел

NumPy - це основний пакет, необхідний для наукових обчислень із Python. Це модуль для мови Python, який забезпечує тип даних багатовимірний масив, надає різні інші похідні об'єкти (масиви з масками та матриці), варіанти функцій для дуже швидких операцій із масивами, враховуючи математичні, логічні, дії з фігурами, сортування даних, вибір, введення та виведення даних, різні дискретні перетворення, основи лінійної алгебри, основні операції статистики, рандомне моделювання та багато інших операцій.

У базі модулю NumPy знаходиться об'єкт масиву ndarray. Він включає масиви з n вимірів однакових типів даних, та багато які операції виконуються в коді, який уже є скомпілюованим, для ефективності та продуктивності. Є декілька дуже важливих відмінностей, масивами NumPy від стандартних списків мови Python:

Масиви даних модулю NumPy мають статичний розмір після створенні, на відміну від послідовностей мови Python (які можуть динамічно змінювати розмір). Оновлення розміру масиву ndarray створить новий масив даних і видалить попередній.

Усі елементи масиву даних модулю NumPy змушені мати однакові типи даних, і, таким чином, мають однакові розміри в пам'яті. Є винято, що можна мати масиви даних об’єктів (Python або NumPy), які дозволяють мати масиви даних елементів різного розміру.

Масиви даних модулю NumPy сприяють вдосконаленню математичних та інших типів операцій щодо великої кількості даних. Зазвичай ці операції виконуються більш ефективно та з меншою кількістю кодом, ніж якщо це можливо використовуючи вбудовані списки мови Python.

Постійно збільшується кількість наукових та математичних модулів для мови Python, які використовують статичні масиви даних із NumPy. Зазвичай вони дають

можливітьс використовувати введення стандартних списків мови Python, але вони після введення перетворюють ці списки в масиви даних модулю NumPy перед тим, як оброблювати їх, також ці модулі дуже часто дають можливість виведення масивів NumPy. Інакше кажучи, для більш ефективного використання великої частини (зазвичай більшості) сучасного наукового та математичного програмного забезпечення для мови програмування Python, недостатньо просто знати, як можна використовувати вбудовані списки та структури даних - необхідно також мати бачення того, як треба використовувати статичні масиви даних бібліотеки NumPy.

pandas - це пакет для мови програмування Python, який забезпечує структури даних, які є швидкими та ефективними для роботи, гнучкими у використанні, робить операції з “реляційними” або “позначеними” даними одночасно простими та інтуїтивно зрозумілими. Він має на меті стати фундаментальним елементом, який дає можливість практично аналізувати дані на високому рівні. Крім того, він має більш широку мету - стати найпотужнішим та найгнучкішим інструментом аналізу / маніпулювання даними з відкритим кодом, доступним будь-якою мовою. Вона вже на шляху до цієї мети. [4] Ось декілька речей, які pandas добре виконує:

- Легка обробка відсутніх даних (представлених як NaN, NA або NaT) як з плаваючою точкою, так і з даними без плаваючої точки
- Змінюваність розміру: стовпці можна вставляти та видаляти з DataFrame та об’єктів більшого розміру
- Автоматичне та явне вирівнювання даних: об’єкти можуть бути вирівняними явно за набором міток, або проігнорованими користувачем, який має можливість дозволити структурам Series, DataFrame та іншим автоматично здійснювати вирівнювання даниї для вас під час обчислень
- Потужна, гнучка група за функціональністю для виконання операцій розділення-застосування-об'єднання над наборами даних, як для агрегування, так і для перетворення даних

- Спрощення перетворення нерівних, по-різному індексованих даних в інших структурах даних Python та NumPy в об’єкти DataFrame
- Інтелектуальне нарізування на основі етикеток, вигадливе індексування та підмножина великих наборів даних
- Інтуїтивне об’єднання наборів даних
- Гнучка зміна форми та обертання наборів даних
- Ієрархічне маркування осей (можна мати кілька міток на галочку)
- Надійні інструменти введення / виводу для завантаження даних із плоских файлів (CSV та з роздільниками), табличних файлі програми Excel, різних баз даних і завантаження або збереження даних використовуючи найбільш швидкий формат HDF5
- Функціональні можливості рядків з типом даних часу: генерування діапазонів дат або різні перетворювання частот, статистика відставання дат, переміщення або перетворення вікон

pandas-gbq - це пакет, що забезпечує інтерфейс до Google BigQuery API із pandas. Модуль “pandas_gbq” забезпечує обгортку веб-служби аналітики BigQuery від Google, щоб спростити отримання результатів із таблиць BigQuery за допомогою SQL-подібних запитів. [5] Набори результатів аналізуються з pandas.DataFrame з формою та типами даних, отриманими з вихідної таблиці. Крім того, DataFrames можна вставляти в нові таблиці BigQuery або додавати до існуючих таблиць. [6]

Beautiful Soup - це бібліотека для мови Python для витягування даних із файлів HTML та XML. Вона працює з синтаксичним аналізатором, щоб забезпечити ідіоматичні способи навігації, пошуку та модифікації дерева синтаксичного аналізу. Це зазвичай економить програмістам години або дні роботи. [7]

2.1.1 Класифікація тексту

Класифікація тексту використовується для передбачення, у чому зацікавлений користувач, на основі тексту його повідомлення. Далі буде надано інформацію про бібліотеки, які використовуються для її досягнення.

scikit-learn - це модуль для мови програмування Python для машинного навчання, побудований поверх SciPy. [8]

Загалом, проблема машинного навчання розглядає набір з n вибірок даних, а потім намагається передбачити властивості невідомих даних. Якщо кожна вибірка більше, ніж одне число, і, наприклад, багатовимірний запис (він же багатовимірні дані), кажуть, що він має кілька атрибутів або ознак.

Проблеми машинного навчання можна поділити на кілька категорій:

- контрольоване навчання, в якому дані постачаються з додатковими атрибутами, які ми хочемо передбачити (натисніть тут, щоб перейти на сторінку наукового контролю з науковим управлінням scikit-learn). Ця проблема може бути:
- класифікація: зразки належать до двох або більше класів, і ми хочемо навчитися з уже позначених даних, як передбачити клас немічених даних. Прикладом проблеми класифікації може бути розпізнавання рукописних цифр, при якому метою є присвоєння кожного вхідного вектора одній із кінцевої кількості дискретних категорій. Інший спосіб мислити класифікацію - це дискретна (на відміну від безперервної) форма контрольованого навчання, коли одна має обмежену кількість категорій, і для кожної із запропонованих зразків потрібно спробувати позначити їх відповідною категорією або класом .
- регресія: якщо бажаний результат складається з однієї або декількох безперервних змінних, то завдання називається регресією. Прикладом

проблеми регресу може бути передбачення довжини лосося як функції його віку та ваги.

- навчання без нагляду, при якому дані навчання складаються з безлічі вхідних векторів x без будь-яких відповідних цільових значень. Метою таких проблем може бути виявлення груп подібних прикладів у наборах даних, де це має назву “кластеризація”, або також метою може бути визначення розподілів наборів даних у вхідному просторі, яке має назву “оцінка щільності”, або проектування даних із простору з високою ймовірністю до двох-трьох вимірів з метою візуалізації.

spaCy - це бібліотека для передової обробки природньої мови для мов програмування Python та Cython. Вона побудована на основі найновіших досліджень і була розроблена з першого дня для використання в реальних продуктах.

NLTK - це набір модулів для мови програмування Python із відкритим кодом, наборів даних та навчальних посібників, що підтримують дослідження та розробки в галузі обробки природньої мови.

Обробку природньої мови (NLP) можна вважати підрозділом лінгвістики, інформатики та штучного інтелекту, пов’язаного із взаємодією комп’ютерів та людської мови; зокрема, як запрограмувати комп’ютери на обробку та аналіз великих обсягів даних природною мовою.

Завдяки таким цікавим програмам, як класифікація тексту, аналіз настроїв, машинний переклад, перетворення мовлення в текст, перетворення тексту в мовлення тощо, за останні кілька десятиліть NLP розвинувся від підходів, заснованих на правилах, статистичних технік до додатків на основі штучного інтелекту найближчого минулого.

Випадки використання NLP:

- Машинний переклад - це завдання автоматичного перетворення однієї природньої мови в іншу, зберігаючи значення вхідного тексту та створюючи вільний текст мовою виводу. Однак це завдання машинного перекладу має має багато невід'ємних підзавдань.
- Класифікація тексту - це процес присвоєння тегів або категорій тексту відповідно до його змісту. Це фундаментальна проблема NLP. Її можна вирішити або вручну (нудно, трудомістко та сприйнятливо до людських помилок), або використовуючи методи машинного начання.
- Аналіз настрою - це контекстний аналіз певного тексту, займається ідентифікаціє6 та виділенням суб’єктивної інформації у вихідному тексті, такої як: визнання полярності (позитивної, негативної, нейтральної), виявлення емоцій, тощо. Типовий приклад застосування аналізу настрою - це галузь електронної комерції, де необхідним є видобуток та аналіз різних оглядів для отримання уявлення щодо задоволеності та досвіду клієнтів, визначення потенційних напрямків щодо вдосконалення є важливими.
- Віртуальні голосові помічники, такі як: Siri, Alexa та Cortana. Перекладач Google, перетворення мови в текст та перетворювачі тексту в мовлення - це програми NLP, які дуже багато людей постійно використовують у своєму житті.

Природня мова людей є дуже різноманітною, і кожна мова має власну багату граматику та значну унікальність. Далі наведено деякі проблеми та виклики, які виникають під час виконання завдань NLP.

- Неоднозначність - це стандартна внутрішня характеристика людських розмов, яка є дуже складною в сценаріях розуміння природньої мови, коли може існувати безліч різних форм, які мають значення у природній мові та в системі штучного інтелекту, яку ми запрограмували. У теорії штучного

інтелекту процес вирішення проблемо, коли є два або більше значень, називається неоднозначністю.

- Синонімічність з'являється тому, що ми можемо висловити одну і ту ж ідею різними термінами, схожими зазначенням словами, які також залежать від певного контексту; Наприклад, "великий" і "величезний" мають подібне значення, коли йдеться про розміри, тоді як "великий" не має сенсу, коли використовується як прикметник, ознака до слова "сестра".
- Взаємозв’язок - це процес пошуку всіх виразів, які посилаються на одну й ту саму сутність у тексті. Можливість спільного посилання є дуже важливим кроком для багатьох завдань NLP вищого рівня, які передбачають розуміння природньої мови, і часто сприяють поліпшенню роботи архітектур нейронних мереж, таких як RNN та LSTM.

Стандартний робочий процес для проблеми NLP включає наступні кроки:

- Першим етапом зазвичай є знаходження тексту та попередня його обробка в корпусі документів. Потім проводяться синтаксичний та базовий дослідницький аналізи даних.
- Наступним кроком є те, що розглядається представлення тексту зі вбудованими словами і подальшою інженерною розробкою функцій. Далі вибирається модель залежно від того, чи розглядається проблема навчання під наглядом, чи розглядається навчання без нагляду.
- Завершальний етап передбачає оцінку та розгортання моделі, як і в будь-якому іншому процесі машинного навчання.

Для того, щоб отримати модель для класифікації, ми повинні здійснити наступні кроки:

- оптимізація
    - видалення пунктуації
    - видалення стоп-слів

- лематизація
- токенізація (векторизація)
- тренування

1. Оптимізація

Спочатку текст необхідно оптимізувати, а саме привести до певного вигляду, щоб спростити роботу з ним у подальшому (Додаток А). Для цього необхідно здійснити наступні кроки:

A. Видалення пунктуації

Необхідно видалити всі знаки пунктуації, адже вони не є потрібними для аналізу мови

B. Видалення стоп-слів

Необхідно видалити стоп-слова, які можуть зустрічатися майже в будь-яких контекстах і тому негативно впливають на точність результату аналізу. Наприклад, в англійській мові це: ‘ourselves’, ‘hers’, ‘between’, ‘yourself’, ‘but’, ‘again’, ‘there’ і т.д.

C. Лематизація

Лематизація в лінгвістиці - це процес групування змінюваних форм слова таким чином, щоб була можливість проаналізувати їх як єдиний елемент, ідентифікований лемою слова або словниковою формою

Наприклад:

- Слово "краще" має лему "добре". Ця логіка зазвичай пропускається лематизаторами, оскільки вона вимагає пошук у словнику.
- Слово "ходити" є базовою формою слова "ходьба", і, отже, це і є його лемою.
- Слово "зустріч" може бути або базовою формою іменника, або формою дієслова ("зустрітися") залежно від контексту;

наприклад, "на нашій останній зустрічі" або "Ми збираємось знову завтра".

2. Токенізація (векторизація)

Токенізація - це розбиття тексту на слова або фрази для полегшення його аналізу. Отже, необхідно розбити текст на слова для подальшого аналізу кожного слова окремо (Додаток А).

3. Тренування

Далі необхідно використати одну із заздалегідь тренованих моделей векторів слів із бібліотеки spaCy, яка містить вектори коефіцієнтів вживання слів у певних контекстах та SVC (C-Support Vector Classification) моделі із бібліотеки scikit-learn для того, щоб створити власну модель для класифікації тексту (Додатки Б, Г).

2.2 Програмне забезпечення

Програмне забезпечення для створення чат-бота для надання навчальних консультацій повинно давати можливість достатньо просто працювати із великим об’ємом даних, оброблювати їх, тренувати та використовувати моделі машинного навчання, а саме обробки природньої мови та взаємодіяти із месенджером, у якому буде відбуватися спілкування з ботом.

Для створення чат-бота використано мову Python та необхідні бібліотеки для обробки даних, машинного навчання та API месенджера. Для написання коду застосовано текстовий редактор Emacs.

2.2.1 Опис структури системи

На схемі на рис. 2.2.1 продемонстровано структуру чат-бота на етапі розробки.

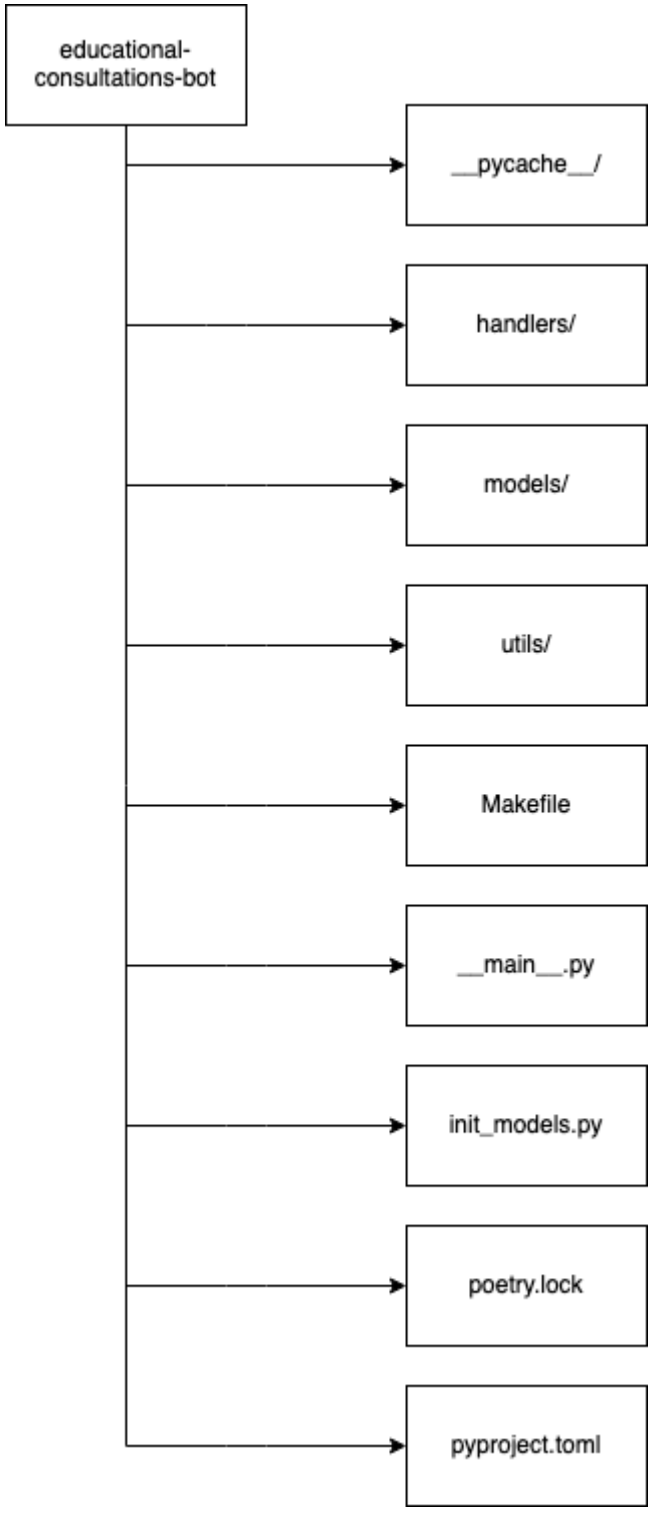

Рисунок 2.2.1. Структура проекту чат-бота для надання навчальних консультацій

Саме ці директорії виконують головну роль у створенні чат-бота. Кожна з них відповідає за окрему частину проекту. Детальний опис директторій можна побачити в таб. 2.2.1.

Назва директорії/файла	Опис
__pycache__	Директорія, яка містить Python файли, скомпільовані в байт-код
handlers	Директорія, яка містить модулі з функціями для обробки запитів оновлень Telegram API
models	Директорія, яка містить треновані моделі та модулі для роботи з ними
utils	Директорія, яка містить додаткові модулі, інструменти, які використовуються як доповнення до стандартної бібліотеки
Makefile	Файл, що містить набір команд, який використовується інструментом автоматизації make для досягнення певної цілі
__main__.py	Файл, який використовується для запуску сервера бота
init_models.py	Скрипт, який використовується для тренування моделей

poetry.lock	Файл, який описує точні версії Python пакетів, встановлених із допомогою інструмента Poetry
pyproject.toml	Файл, який містить список Python пакетів, встановлених із допомогою інструмента Poetry та загальну інформацію про проект

Таблиця 2.2.1. Опис директорій чат-бота для надання навчальних консультацій

РОЗДІЛ 3. ОПИС РОБОТИ СИСТЕМИ

3.1 Інструкція користувача

Даний бот - це реалізація чат-бота для надання навчальних консультацій. Для прикладу використано чат-бота, який надає консультації щодо вивчення тем, пов’язаних із комп’ютерними науками.

Для початку роботи з чат-ботом необхідно надіслати йому повідомлення із текстом стосовно того, що ви хочете вивчити. Бот спробує передбачити категорію, яка вас цікавить, та надасть вам можливість або підтвердити її, або вибрати іншу з передбачених варіантів (рис 3.1.1).

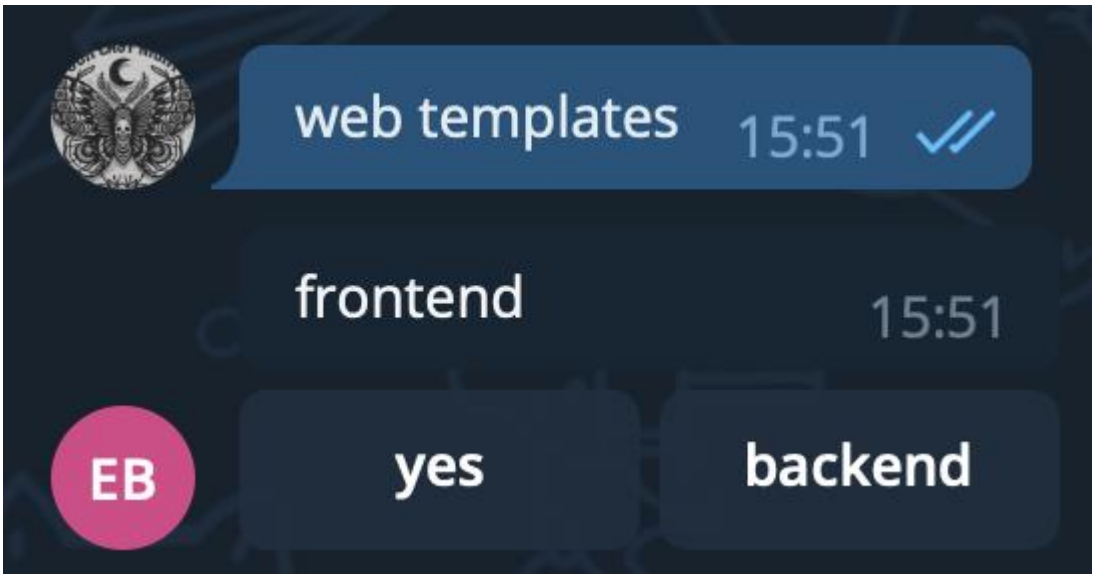

Рисунок 3.1.1. Початок роботи з чат-ботом (вибір категорії)

Після того, як ви вибрали категорію, бот спробує передбачити підкатегорію, яка вас цікавить, та надасть вам можливість або підтвердити її, або вибрати іншу з передбачених варіантів (рис 3.1.2).

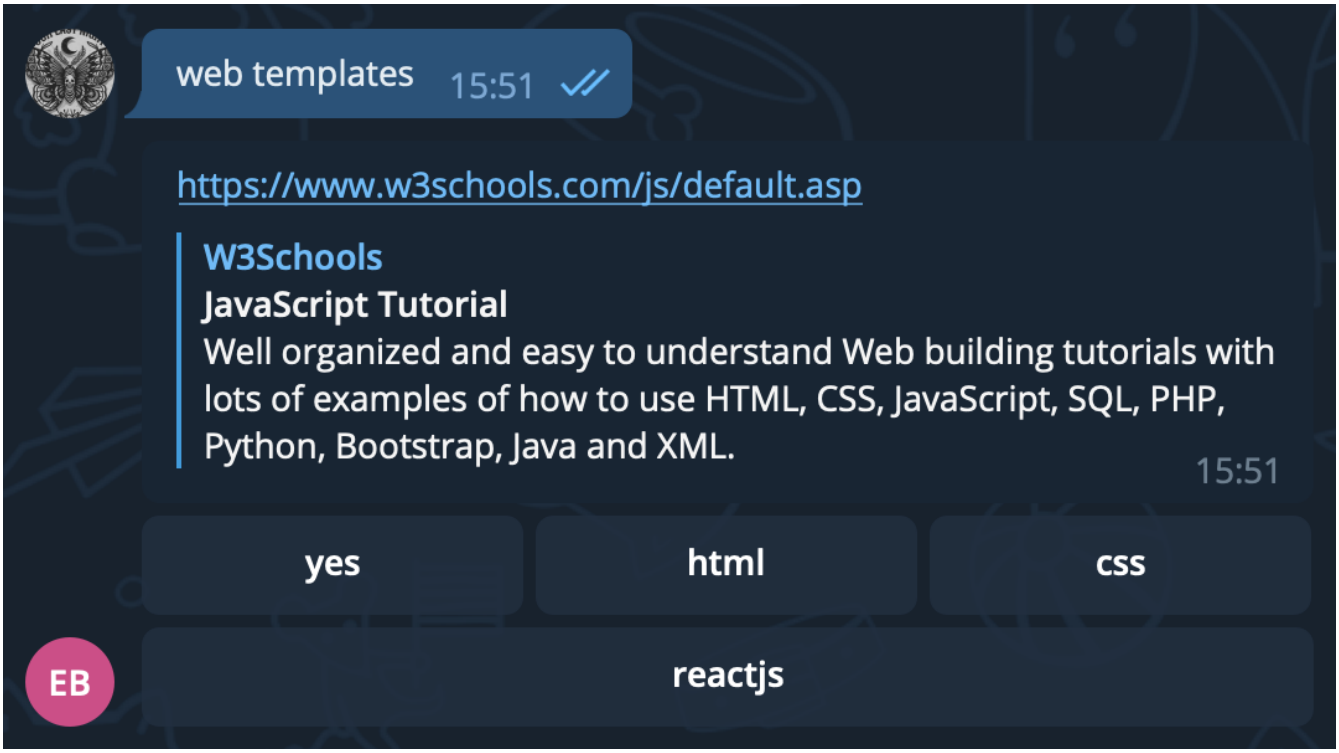

Рисунок 3.1.2. Вибір підкатегорії

Після того, як ви вибрали підкатегорію, або просто зникнуть кнопки вибору (якщо ви підтвердили підкатегорію), або посилання на ресурс зміниться відповідно до категорії, на назву якої ви натиснули (рис. 3.1.3).

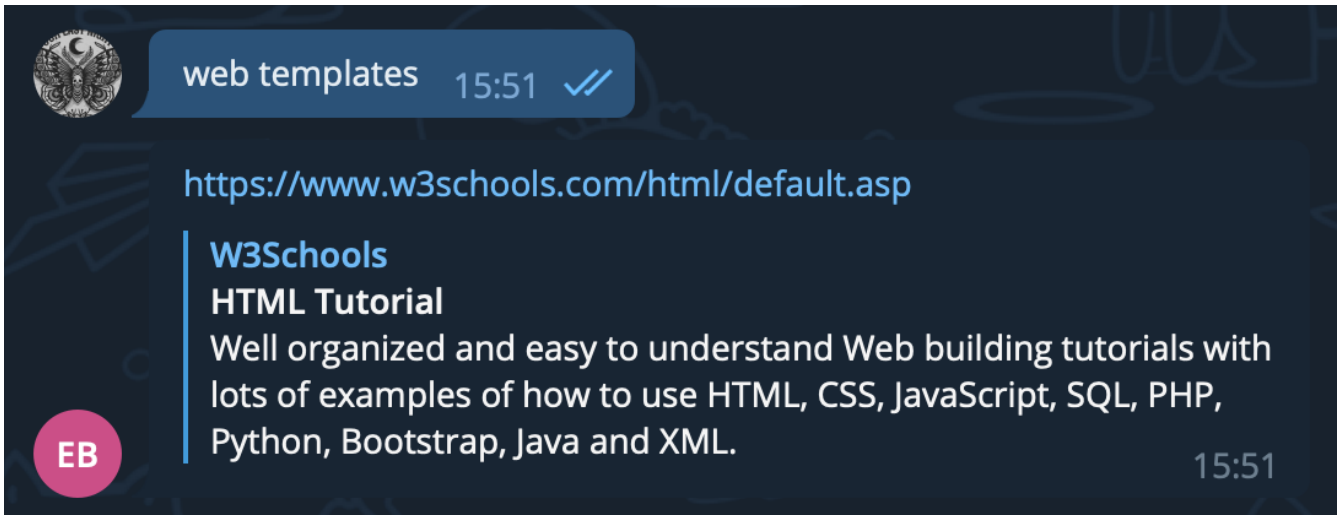

Рисунок 3.1.3. Підтвердження (зміна передбаченої) підкатегорії

Роботу з ботом закінчено! Ви отримали посилання на необхідний вам ресурс, відповідно до того, яким було ваше початкове повідомлення та які варіанти ви вибирали.

ВИСНОВОК

У результаті виконання бакалаврської було розроблено чат-бот для надання навчальних консультацій. Особливістю розробленого чат-бота для надання освітніх послуг є:

- можливість отримувати необхідні навчальні консультації або іншу інформацію у відповідь на ваші повідомлення;
- реалізована класифікація тексту повідомлення користувача;
- реалізовано декілька моделей машинного навчання, які використовуються залежно від передбаченої ботом (вибраної користувачем) категорії інформації
- реалізована можливість або підтвердити категорію інформації, яку надав бот, або вибрати іншу категорію зі списку

Для дослідження теоретичних основ побудови чат-бота було взято існуючу документацію, книги та інформацію про роботу подібних застосунків на інших сервісах у мережі інтернет.

За час виконання даної бакалаврської роботи було досліджені теоретичні та практичні основи та засоби для побудови чат-бота для надання навчальних консультацій. Завдяки різноманітним фреймворкам, бібліотекам та архітектурним рішенням застосованим та дослідженим в процесі виконання роботи було розроблено всю необхідну функціональність застосунку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Wikipedia - Python (programming language). URL: https://en.wikipedia.org/wiki/Python_%28programming_language%29
2. GitHub - python-telegram-bot. URL: https://github.com/python-telegram-bot/python-telegram-bot#introduction
3. GitHub - NumPy. URL: https://github.com/numpy/numpy
4. GitHub - pandas. URL: https://github.com/pandas-dev/pandas#what-is-it
5. GitHub - pandas-gbq. URL: https://github.com/pydata/pandas-gbq,
6. Readthedocs –pandas-gbq. URL: https://pandas-gbq.readthedocs.io/en/latest/
7. Readthedocs - Beautiful Soup. URL: https://beautiful-soup-4.readthedocs.io/en/latest/
8. GitHub - scikit-learn. URL: https://github.com/scikit-learn/scikit-learn
9. Джейк Вандерплас. Підручник із науки про дані в мові програмування Python, 2017. С. 58-62
10. Луціано Рамалго. Вільний Python, 2015. С. 120-122
11. Марк Лутц. Кишенькове посилання на Python, 2014. С. 201-205

ДОДАТКИ

Додаток А (оптимізація тексту)

```
def optimized_phrase(phrase):
  stopwords_ = stopwords.words("english")
  lemmatizer = WordNetLemmatizer()

  words = word_tokenize(phrase)
  optimized_words = [
lemmatizer.lemmatize(word.lower())
for word in words
if word not in punctuation and word.lower() not in stopwords_
  ]

  return " ".join(optimized_words)
```

Додаток Б (тренування моделі)

```
def train(dataframe):
  nlp = load_nlp()
  docs = [nlp(optimized_phrase(text)) for text in dataframe["tex
t"]]
  word_vectors = [x.vector for x in docs]

  X_train, X_test, y_train, y_test = train_test_split(
word_vectors, dataframe["category"], test_size=0.35
  )

  model = svm.SVC(kernel="linear")
  model.fit(X_train, y_train)

  score = model.score(X_test, y_test)
  print(f"Model score: {score}")

  return model
```

Додаток В (підготовка даних)

```
import os
import pickle

from bs4 import BeautifulSoup
from pandas_gbq import read_gbq


def fetch_questions_query(categories):
  def category_conditions(categories):
conditions = ""

for c in categories:
if conditions:
conditions += "\n\tOR "

positive_conditions = f"""(tags LIKE '%|{c}|%'
OR tags LIKE '%|{c}'
OR tags LIKE '{c}|%'
OR tags LIKE '{c}')"""
negative_conditions = " AND ".join(
f"""(tags NOT LIKE '%|{oc}|%'
AND tags NOT LIKE '%|{oc}%'
AND tags NOT LIKE '%{oc}|%')"""
for oc in [x for x in categories if x != c]
)
conditions += (
f"{positive_conditions}\n\t\tAND {negative_conditions}"
)

return conditions

  query = f"""
  SELECT title, body, tags
  FROM `bigquery-public-data.stackoverflow.posts_questions`
  WHERE {category_conditions(categories)}
  LIMIT 15000
```

```
  """

  return query


def category(categories):
def f(row):
for c in categories:
if c in row["tags"]:
value = c
break

return value

  return f


def load_df(categories, file_path):
  if os.path.isfile(file_path):
with open(file_path, "rb") as f:
df = pickle.load(f)
  else:
query = fetch_questions_query(categories)
df = read_gbq(query, project_id=os.getenv("GOOGLE_PROJECT_ID"))
with open(file_path, "wb") as f:
pickle.dump(df, f)

  df["text"] = df["title"] + [
BeautifulSoup(body, "html.parser").get_text() for body in df["bo
dy"]
  ]
  df["category"] = df.apply(category(categories), axis=1)
  df = df.drop(columns=["title", "body", "tags"])

  return df
```

Додаток Г (ініціалізація моделей)

```
def init():
  nltk.download("wordnet")
  nltk.download("stopwords")
  nltk.download("punkt")

  categories = Category.values()
  category_df = load_df(
categories, os.getenv("CATEGORY_DATAFRAME_FILE_PATH")
  )
  category_model = train(category_df)

  frontend_subcategories = [
c.value for c in SUBCATEGORIES[Category.FRONTEND]
  ]
  frontend_df = load_df(
frontend_subcategories,
os.getenv("FRONTEND_SUBCATEGORY_DATAFRAME_FILE_PATH")
)
  frontend_model = train(frontend_df)

  backend_subcategories = [
c.value for c in SUBCATEGORIES[Category.BACKEND]
]
  backend_df = load_df(
backend_subcategories,
os.getenv("BACKEND_SUBCATEGORY_DATAFRAME_FILE_PATH"),
  )
  backend_model = train(backend_df)

  dump(category_model, os.getenv("CATEGORY_MODEL_FILE_PATH"))
  dump(
frontend_model, os.getenv("FRONTEND_SUBCATEGORY_MODEL_FILE_PATH"
)
)
  dump(
backend_model, os.getenv("BACKEND_SUBCATEGORY_MODEL_FILE_PATH")
```

```
)
```

Додаток Д (обробка повідомлень та інших оновлень Telegram API)

```
import logging
import os

import models
from models import Category
from .callback_query import button_handler_wrapper
from .command import start_handler
from .message import category_handler_wrapper


def add_handlers(dispatcher):
nlp = models.load_nlp()
category_model = models.load(os.getenv("CATEGORY_MODEL_FILE_PATH
"))
frontend_subcategory_model = models.load(
os.getenv("FRONTEND_SUBCATEGORY_MODEL_FILE_PATH")
)
backend_subcategory_model = models.load(
os.getenv("BACKEND_SUBCATEGORY_MODEL_FILE_PATH")
)

dispatcher.add_handler(category_handler_wrapper(nlp, category_mo
del))
dispatcher.add_handler(start_handler)
dispatcher.add_handler(
button_handler_wrapper(
nlp,
{
Category.FRONTEND: frontend_subcategory_model,
Category.BACKEND: backend_subcategory_model,
},
)
)
 logging.info("Handlers were added")
```

```
from telegram import Update
from telegram.ext import CallbackQueryHandler, CallbackContext

from models import predict
from models.category import Category, Subcategory
from utils.telegram import reply_markup_from_categories


def button_wrapper(nlp, models):
def button(update: Update, _context: CallbackContext) -> None:
query = update.callback_query
query.answer()

yes, category, subcategory, text = query.data.split("|")
if yes:
if subcategory:
query.edit_message_reply_markup(reply_markup=None)
else:
subcategory = predict(nlp, models[Category(category)], text)
reply_markup = reply_markup_from_categories(
Category(category).subcategories,
category,
subcategory,
text,
)
query.edit_message_text(text=Subcategory(subcategory).link)
query.edit_message_reply_markup(reply_markup=reply_markup)
elif subcategory:
query.edit_message_text(text=Subcategory(subcategory).link)
else:
subcategory = predict(nlp, models[Category(category)], text)
reply_markup = reply_markup_from_categories(
Category(category).subcategories, category, subcategory, text)
query.edit_message_text(text=Subcategory(subcategory).link)
query.edit_message_reply_markup(reply_markup=reply_markup)

return button
```

```
button_handler_wrapper = lambda nlp, models: CallbackQueryHandle
r(
button_wrapper(nlp, models)
)

from telegram import Update
from telegram.ext import CommandHandler, CallbackContext


def start(update: Update, context: CallbackContext) -> None:
context.bot.send_message(
chat_id=update.effective_chat.id,
text="Hello, I'm educational consultations bot. Ask me, what you
 want to
know",
)


start_handler = CommandHandler("start", start)


import sys

from telegram.ext import MessageHandler, Filters

from models import predict
from models.category import Category
from utils.telegram import reply_markup_from_categories


def category_wrapper(nlp, model):
def category(update, context):
chat_id = update.effective_chat.id
text = update.message.text

if sys.getsizeof(text) >= 64:
return context.bot.send_message(
chat_id=chat_id,
text="Your message is too long. Try something shorter!",
)
```

```
category = predict(nlp, model, text)
reply_markup = reply_markup_from_categories(
Category, category, "", text
)

context.bot.send_message(
chat_id=chat_id,
text=category,
reply_markup=reply_markup,
)

return category


category_handler_wrapper = lambda nlp, model: MessageHandler(
Filters.text & (~Filters.command), category_wrapper(nlp, model)
)
```

Додаток Е (конфігурація проекту)

```
[tool.poetry]
name = "educational-consultations-bot"
version = "0.1.0"
description = ""
authors = ["Yevhen Shymotiuk <yevhenshymotiuk@gmail.com>"]

[tool.poetry.dependencies]
python = "^3.8"
python-telegram-bot = "^13.3"
scikit-learn = "^0.24.1"
nltk = "^3.5"
numpy = "^1.20.1"
pandas-gbq = "^0.14.1"
beautifulsoup4 = "^4.9.3"
spacy = "^3.0.5"

[tool.poetry.dev-dependencies]
ipython = "^7.21.0"
```

```
matplotlib = "^3.3.4"
ipdb = "^0.13.7"
pylint = "^2.8.2"

[build-system]
requires = ["poetry-core>=1.0.0"]
build-backend = "poetry.core.masonry.api"
```

Додаток Є (інструменти)

```
from telegram import InlineKeyboardButton, InlineKeyboardMarkup

from .iterable import batch
from models.category import Subcategory


def reply_markup_from_categories(
    categories, category="", subcategory="", text=""
):
    sub = isinstance(list(categories)[0], Subcategory)
    buttons = [
        InlineKeyboardButton(
            "yes", callback_data=f"yes|{category}|{subcategory}|
{text}"
        )
    ]
    for c in categories:
        if c.value != (subcategory if sub else category):
            category_value = category if sub else c.value
            subcategory_value = c.value if sub else subcategory
            button = InlineKeyboardButton(
                c.value,
                callback_data=f"|{category_value}|{subcategory_v
alue}|{text}",
            )
            buttons.append(button)

    keyboard = batch(buttons, 3)
    reply_markup = InlineKeyboardMarkup(keyboard)
```

```
    return reply_markup


def batch(iterable, n=1):
    l = len(iterable)
    for ndx in range(0, l, n):
        yield iterable[ndx : min(ndx + n, l)]
```

Додаток Ж (категорії та підкатегорії)

```
from enum import Enum


class ValuesMixin:
    @classmethod
    def values(cls):
        return list(map(lambda c: c.value, cls))


class Category(ValuesMixin, Enum):
    BACKEND = "backend"
    FRONTEND = "frontend"

    @property
    def subcategories(self):
        return SUBCATEGORIES[self]


class Subcategory(ValuesMixin, Enum):
    SQL = "sql"
    ASP = "asp.net"
    NODE_JS = "nodejs"
    RASPBERRY_PI = "raspberry-pi"
    HTML = "html"
    CSS = "css"
    JAVASCRIPT = "javascript"
    REACT = "reactjs"
```

```
    @property
    def link(self):
        return LINKS[self]


SUBCATEGORIES = {
    Category.BACKEND: [
        Subcategory.SQL,
        Subcategory.ASP,
        Subcategory.NODE_JS,
        Subcategory.RASPBERRY_PI,
    ],
    Category.FRONTEND: [
        Subcategory.HTML,
        Subcategory.CSS,
        Subcategory.JAVASCRIPT,
        Subcategory.REACT,
    ],
}

LINKS = {
    Subcategory.SQL: "https://www.w3schools.com/sql/default.asp"
,
    Subcategory.ASP: "https://www.w3schools.com/asp/default.asp"
,
    Subcategory.NODE_JS: "https://www.w3schools.com/nodejs/defau
lt.asp",
    Subcategory.RASPBERRY_PI: "https://www.w3schools.com/nodejs/
nodejs_raspberrypi.asp",
    Subcategory.HTML: "https://www.w3schools.com/html/default.as
p",
    Subcategory.CSS: "https://www.w3schools.com/css/default.asp"
,
    Subcategory.JAVASCRIPT: "https://www.w3schools.com/js/defaul
t.asp",
    Subcategory.REACT: "https://www.w3schools.com/react/default.
asp",
}
```

Додаток З (Docker)

```
FROM python:3.8

SHELL ["/bin/bash", "-c"]

ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

WORKDIR /code

COPY . /code/

RUN apt-get -y update
RUN apt-get -y upgrade
RUN apt-get -y install cmake

RUN pip install poetry \
    && poetry config virtualenvs.create false \
    && poetry install --no-dev

RUN pip install "en_core_web_lg-3.0.0.tar.gz"

RUN sed "1,2d" .envrc > .env

ENTRYPOINT [ "bash", "entrypoint.sh" ]
```

```
#!/usr/bin/env bash

# add variables to environment
source .env

# download nltk data
python -c "import models;models.download_nltk_data()"

# start bot server
python .
```

Додаток И (Makefile)

```
SPACY_MODEL=en_core_web_lg
```

```
.PHONY: spacy models docker

spacy:
  wget -nc "https://github.com/explosion/spacy-
models/releases/download/$(SPACY_MODEL)-3.0.0/en_core_web_lg-
3.0.0.tar.gz"

models: spacy
  python -
c "import importlib;importlib.util.find_spec('$(SPACY_MODEL)') o
r exit(1)"
  [[ "$?" -eq 1 ]] \
    && pip install "$(SPACY_MODEL)-3.0.0.tar.gz" \
    || echo "SpaCy model is already installed"
  python init_models.py

docker: spacy
  docker build -t ec-bot .
```