МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ІНТЕЛЕКТУАЛЬНА СИСТЕМА НАДАННЯ КОНСУЛЬТАЦІЙ З ТЕОЛОГІЧНИХ ТА АПОЛОГЕТИЧНИХ ПИТАНЬ

Спеціальність 122 «Комп'ютерні науки»

122 – КРБ – 402.22010220

Виконав студент 4-го курсу, групи 402
_____ І. О. Рибаков
«17» червня 2024 р.

Керівник: канд. техн. наук, доцент
_____Є. В. Сіденко
«17» червня 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**

Спеціальність **122 «Комп'ютерні науки»**

(шифр і назва)

Галузь знань **12 «Інформаційні технології»**

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.

_____ Ю. П. Кондратенко

«____» _____2024 р.

З А В Д А Н Н Я

на виконання кваліфікаційної роботи

Видано студенту групи 402 факультету комп'ютерних наук Рибакову Івану Олександровичу.

1. Тема кваліфікаційної роботи «Інтелектуальна система надання консультацій з теологічних та апологетичних питань».

Керівник роботи Сіденко Євген Вікторович, канд. техн. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «28» грудня 2024 р. № 271

2. Строк представлення кваліфікаційної роботи студентом «17» червня 2024 р.

3. Вхідні (початкові) дані до роботи: матеріали апологетичного та теологічного змісту.

Очікуваний результат: інтелектуальна система надання консультацій з теологічних та апологетичних питань

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– аналіз сучасного стану задачі вибору технології розпізнавання та обробки мови, пошуку відповідей на запитання за допомогою цих технологій;

– огляд існуючих методів розпізнавання людської мови за допомогою штучного інтелекту;

– експертне оцінювання технологій розпізнавання та обробки мови з використанням штучного інтелекту;

– розробка програмного забезпечення для надання відповідей на теологічні та апологетичні питання.

5. Перелік графічного матеріалу: рисунки, презентація.

6. Завдання до спеціальної частини: «Забезпечення вимог охорони праці у офісному приміщенні, роботи за комп'ютером та у надзвичайних ситуаціях»

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Алексєєва А. О., доцент кафедри екології	

Керівник роботи __канд. техн. наук, доцент Сіденко Євген Вікторович

(наук. ступінь, вчене звання, прізвище та ініціали)

_____ *(підпис)*

Завдання прийнято до виконання ___Рибаков І. О._____

(прізвище та ініціали)

_____ *(підпис)*

Дата видачі завдання «_14_» _____січня____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Інтелектуальна система надання консультацій з теологічних та апологетичних питань_____

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників КРБ	10.11.2023	15.11.2023	**ВИКОНАНО**
2	Отримання завдання на виконання КРБ	10.01.2024	15.01.2024	**ВИКОНАНО**
3	Складання календарного плану роботи на весь період виконання КРБ	16.01.2024	30.01.2024	**ВИКОНАНО**
4	Отримання завдання на переддипломну практику	15.04.2024	29.04.2024	**ВИКОНАНО**
5	Проходження переддипломної практики, збір та аналіз матеріалів до КРБ	29.04.2024	11.05.2024	**ВИКОНАНО**
6	Розробка звіту з переддипломної практики	12.05.2024	15.05.2024	**ВИКОНАНО**
7	Виконання КРБ: аналіз сучасного стану задачі комп'ютерного розпізнавання тексту та систем відповідей на питання на основі штучного інтелекту, огляд існуючих технологій, розробка ПЗ	13.05.2024	22.06.2024	**ВИКОНАНО**
8	Перший попередній захист КРБ на засіданні комісії кафедри	27.05.2024	27.05.2024	**ВИКОНАНО**
9	Доробка та остаточне оформлення КРБ	28.05.2024	09.06.2024	**ВИКОНАНО**
10	Другий попередній захист КРБ на засіданні комісії кафедри	10.06.2024	10.06.2024	**ВИКОНАНО**
11	Подання КРБ рецензенту	13.06.2024	13.06.2024	**ВИКОНАНО**
11	Подання КРБ, її електронної копії та інших документів (відгуку, рецензії) до захисту	17.06.2024	21.06.2024	**ВИКОНАНО**
12	Захист КРБ перед екзаменаційною комісією (ЕК)	24.06.2024	28.06.2024	**ВИКОНАНО**

Розробив студент __Рибаков І. О._____ _____
(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи __ канд. техн. наук, доцент Сіденко Є. В. _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«_29_» _____01_____ 2024 р.

АНОТАЦІЯ

кваліфікаційної роботи бакалавра студента 402 ЧНУ ім. Петра Могили Рибакова Івана Олександровича

Тема: «ІНТЕЛЕКТУАЛЬНА СИСТЕМА НАДАННЯ КОНСУЛЬТАЦІЙ З ТЕОЛОГІЧНИХ ТА АПОЛОГЕТИЧНИХ ПИТАНЬ»

Актуальність теми теології Християнства не можна переоцінити у всі часи, це місце, де люди знаходять справжній сенс, надію та мир, особливо ця потреба існує в людей під час війн, криз та скрутних обставин. Інтелектуальні системи для надання консультацій з теологічних та апологетичних питань облегшать отримання інформації на теологічні та апологетичні питання.

Об'єкт роботи - процеси надання інформаційних консультацій.

Предмет роботи - технології обробки природної мови та алгоритми машинного навчання для створення інтелектуальної системи надання консультацій з теологічних та апологетичних питань.

Метою кваліфікаційної роботи є спрощення процесу отримання відповідей на теологічні та апологетичні питання на платформі Telegram.

Пояснювальна записка складається зі вступу, чотирьох розділів, висновків та додатків.

У першому розділі розглядається сучасний стан задачі надання консультацій з теологічних та апологетичних питань, огляд наявних рішень цієї проблеми, публікацій.

Другий розділ містить у собі огляд існуючих технологій та алгоритмів, проведено аналіз моделей для поставленої задачі.

У третьому розділі описується підхід до розробки системи, зокрема процес підготовки власного набору даних, навчання моделі ШІ DistilBERT на підготовленому наборі даних.

У четвертому розділі описано деталі програмної реалізації системи, розробки користувацького інтерфейсу на базі Telegram, а також тестування застосунку.

Кваліфікаційна робота бакалавра містить 69 сторінки, 37 рисунки, 28 літературних джерела.

Ключові слова: теологія, апологетика, машинне навчання, CDQA, Telegram, BERT.

ABSTRACT

To the bachelor's qualification work by student of group 402 of Petro Mohyla Black Sea National University

Ivan Rybakov

"INTELLIGENT SYSTEM FOR PROVIDING CONSULTATIONS ON THEOLOGICAL AND APOLOGETIC QUESTIONS"

The relevance of Christian theology cannot be overstated throughout the ages; it is a place where people find true meaning, hope, and peace, especially during times of war, crises, and difficult circumstances. Intelligent systems for providing consultations on theological and apologetic questions facilitate access to information on these topics.

Object of the study: the processes of providing information consultations.

Subject of the study: natural language processing technologies and machine learning algorithms for creating a question-answer system in the field of theology and apologetics.

The goal of the qualification work is to simplify the process of obtaining answers to theological and apologetic questions on the Telegram platform.

The explanatory note consists of an introduction, four chapters, conclusions, and appendices.

In the first chapter, the current state of the task of providing consultations on theological and apologetic questions is considered, including a review of existing solutions and publications on this problem.

The second chapter contains an overview of existing technologies and algorithms, and an analysis of models for the set task.

In the third chapter, the approach to developing the system is described, including the process of preparing a custom dataset and training the DistilBERT AI model on the prepared dataset.

The fourth chapter describes the details of the system's software implementation, the development of the user interface based on Telegram, and the testing of the application.

The bachelor's qualification work contains 69 pages, 37 figures, and 28 literary sources.

Keywords: theology, apologetics, machine learning, CDQA, Telegram, BERT.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API	– Application Programming Interface
BERT	– Bidirectional Encoder Representations from Transformers
BM25	– Best Matching 25
CDQA	– Closed-Domain Question Answering
CNN	– Convolutional Neural Network
JSON	– JavaScript Object Notation
NER	– Named Entity Recognition
NLP	– Natural Language Processing
PaaS	– Platform as a Service
RNN	– Recurrent Neural Network
SQuAD	– Stanford Question Answering Dataset
TF-IDF	– Term Frequency-Inverse Document Frequency
UI	– User Interface

ВСТУП

Актуальність теми: надання консультацій з теологічних та апологетичних питань є важливим аспектом для багатьох віруючих та дослідників. В умовах сучасного світу, де інформаційні технології грають ключову роль у поширенні знань, використання інтелектуальних систем стає все більш актуальним задля отримання швидкого доступу до важливої інформації з великої купи неопрацьованої інформації. Створення інтелектуальної системи, здатної надавати точні та компетентні відповіді на питання з теології та апологетики на зручній для користувача платформі, може значно підвищити доступність та якість консультаційних послуг у цій галузі.

Об’єктом роботи - процеси надання інформаційних консультацій.

Предметом роботи - технології обробки природної мови та алгоритми машинного навчання для створення інтелектуальної системи надання консультацій з теологічних та апологетичних питань.

Метою кваліфікаційної роботи є спрощення процесу отримання відповідей на теологічні та апологетичні питання на платформі Telegram.

Для досягнення цієї мети необхідно створити власний набір даних, на якому буде навчатися обрана модель нейронної мережі.

Для вирішення проблеми необхідно виконати наступні задачі:

1) проаналізувати сучасні технології для вирішення задачі обробки природної мови та надання відповідей на запитання, зокрема в сфері теології та апологетики. Дослідити існуючі матеріали, методи та технології для розробки систем питань-відповідей у галузі теології та апологетики;

2) створити власний набір даних, навчити на ній нейронну мережу для аналізу запитань користувачів та надання точних відповідей на питання;

3) протестувати навчену модель нейронної мережі за для визначення точності наданих відповідей на прикладі теологічних та апологетичних запитань;

4) інтегрувати цю систему на популярну платформу «Telegram» у вигляді чат-боту для зручного для користувача доступу.

1 ОГЛЯД ІНТЕЛЕКТУАЛЬНОЇ CDQA СИСТЕМИ НА ОСНОВІ NLP ТА ЙОГО ЗАСТОСУВАННЯ

1.1 Основні поняття та визначення

Інтелектуальна CDQA система – це інформаційна система, яка призначена для автоматичного пошуку та надання відповідей на запитання користувачів в обмеженій предметній області. Вона використовує технології обробки природної мови (NLP) для аналізу та розуміння запитань користувачів і знаходить відповіді в текстових джерелах інформації, таких як бази даних, документація, веб-сторінки тощо [1-2].

Інтелектуальна CDQA система використовує NLP для аналізу та розуміння мовленнєвих запитань користувачів. Це включає в себе сегментацію тексту, розпізнавання іменованих сутностей, аналіз синтаксису тощо. Система повинна мати доступ до бази знань, в якій зберігається інформація, з якої можуть бути отримані відповіді на запитання користувачів [3, 5].

Інтелектуальна CDQA система використовує різні алгоритми пошуку для знаходження найбільш відповідних відповідей на запитання користувачів. Після пошуку система вибирає найбільш відповідні відповіді та надає їх користувачам. Деякі системи можуть підтримувати діалогову взаємодію з користувачем, дозволяючи задавати додаткові запитання та отримувати пов'язані відповіді [7-8]. Розглядаючи принципи роботи, варто зазначити основні етапи роботи системи - система спочатку аналізує запитання користувача з використанням технологій NLP для розуміння його змісту. За допомогою алгоритмів пошуку система знаходить відповіді в базі знань або інших джерелах інформації. Система вибирає найбільш відповідні відповіді на запитання користувача, враховуючи його контекст та специфіку. Вибрані відповіді надаються користувачу у зрозумілій формі [1].

Обробка природної мови (NLP) - це галузь комп'ютерної науки, що займається розробкою методів та технік для розуміння, аналізу та генерації людської мови. NLP включає в себе такі задачі, як сегментація тексту,

розпізнавання іменованих сутностей, аналіз синтаксису, семантичний аналіз тощо [6]. Одним з основних завдань NLP є сегментація тексту. Це процес розділення тексту на окремі частини або фрази для подальшого аналізу. Сегментація тексту може бути за словами, реченнями або навіть тематичними блоками [7]. Другим важливим завданням є розпізнавання іменованих сутностей (NER). Це завдання полягає в ідентифікації та класифікації іменованих сутностей в тексті, таких як особи, місця, організації, дати тощо [9]. Далі, аналіз синтаксису - це вивчення структури речень та їхніх складових частин, таких як підмети, присудки та об'єкти [10].

Аналіз синтаксису допомагає розуміти граматичну структуру мовлення та встановлювати зв'язки між словами. Семантичний аналіз є ще одним важливим завданням. Він полягає в розумінні значення слова та його контексту в реченні або тексті [6]. Семантичний аналіз допомагає визначити смислові відношення між словами та розуміти загальний зміст тексту. Крім того, машинний переклад - це процес автоматичного переводу тексту з однієї мови на іншу за допомогою комп'ютерних алгоритмів та моделей [7].

Цей процес дозволяє автоматизувати процес перекладу текстів між різними мовами. Також, генерація мовлення є важливим завданням. Це створення тексту або мовлення комп'ютерною системою на основі вхідних даних. Генерація мовлення може бути використана для автоматичного створення текстового контенту, генерації відповідей або створення діалогів [5].

Теологія - це наука про Бога, вірування та вчення, яка вивчає питання віри та істини. У широкому розумінні теологія охоплює вивчення Божественного, релігійних практик, святих писань та інших релігійних текстів, а також різні аспекти релігійного життя. Теологія має велике значення для суспільства, оскільки вона допомагає людям знайти сенс та значення у своєму житті. Вона сприяє формуванню моральних цінностей, підтримує духовний розвиток індивідуумів [3]. Теологія є ключовою галуззю духовного та інтелектуального розвитку. Вона досліджує глибинні питання про природу Бога, віру, мораль, справедливість та

екзистенцію. Теологія вивчає різноманітні аспекти релігії, включаючи її історію, священні тексті, ритуали, традиції та доктрини. Вона допомагає людям розуміти своє місце у світі, вирішувати моральні дилеми та знаходити значення у своєму житті [22].

Апологетика - це галузь теології, яка займається захистом віри та виправданням релігійних переконань. Апологетика використовує наукові факти з різних галузей (такі, як історія, археологія, культурологія, тощо.), аргументацію, логіку для вивчення та відповіді на сумніви та виклики щодо віри [2]. Вона має за мету зміцнити віру вірян та сприяти розумінню та прийняттю релігійних істин.

1.2 Огляд останніх публікацій та аналоги існуючих програм

Останні дослідження в області розвитку інтелектуальних систем CDQA на основі обробки природної мови (NLP) акцентують увагу на вдосконаленні технологій пошуку та аналізу інформації для використання в сфері теології та апологетики. З розвитком NLP з’явилася можливість автоматизованого аналізу та інтерпретації текстів релігійних та філософських творів, що відкриває нові можливості для дослідження та розвитку теологічних концепцій [1-2]. Останні публікації, присвячені розробці використанню систем штучного інтелекту та їх застосуванню в сфері теології та апологетики для різних цілей, включають наступне:

"AI and data-driven Christian Theology" (2020) – у цій статті описано використання обробки природної мови (NLP) та штучного інтелекту (AI) у дослідженнях з гуманітарних наук, зокрема в теології. Автори звертають увагу на успішні застосування NLP, такі як індексація веб-сторінок пошуковими системами, аналіз настроїв відгуків на електронних торговельних сайтах, автоматизоване збирання розвідувальної інформації на основі текстових даних тощо. Вони пропонують алгоритмічний підхід для застосування штучного інтелекту та NLP в дослідженнях теології, що дозволяє автоматизовано аналізувати великі обсяги

тексти та виділяти інформацію, що є значущою для конкретної теми дослідження [21].

"ChatGPT’s Significance for Theology" (2023) – в цій роботі розглядається вплив штучного інтелекту на богословську освіту та дослідження. Автори аналізують конкретний приклад системи ChatGPT та його потенційні наслідки для богословської освіти, теології культури, систематичної теології та богословського дослідження. Вони розглядають якість відповідей ChatGPT на запити порівняно з попередніми системами обробки природної мови (NLP), а також обговорюють питання етики та філософії, пов'язані з розвитком штучного інтелекту, такі як поняття особистості та агентності. Завершується стаття закликом до богословів і академічної громадськості критично сприймати ці питання та розвивати відповідні стратегії використання технологій у своїй роботі [23].

“Artificial Intelligence’s Understanding of Religion: Investigating the Moralistic Approaches Presented by Generative Artificial Intelligence Tools” (2024) – у цій статті автори пропонують цікавий погляд на спосіб, яким штучний інтелект розуміє та представляє релігійні поняття. Автори аналізують декілька розмов з різними штучними інтелектуальними інструментами щодо трьох релігійних традицій: юдаїзму, ісламу та християнства. З їхнього аналізу випливає три важливі аспекти: штучний інтелект має проблеми з представленням складних релігійних питань; він акцентує різноманітність поглядів; і він закликає читачів вести діалог з повагою та чутливістю щодо релігійних питань. Таким чином, автори вважають, що штучний інтелект приймає аксіологічно орієнтований підхід до представлення релігійних понять [22].

“Impact of AI-Powered Technology on Religious Practices and Ethics: The Road Ahead” (2023) - ця стаття досліджує вплив технології на релігійні практики та етику, зокрема штучного інтелекту (ШІ). Автор звертає увагу на проблему затримки, з якою людство встигає наздогнати швидкість розвитку технологій, включаючи ШІ.

Стаття ставить перед собою завдання дослідити позитивні та негативні наслідки використання технологій, ШІ та мобільних додатків у релігійних практиках, а також етичні аспекти цього питання. Дослідження зосереджується на тому, як ШІ впливає на релігійні практики та етику, а також як релігійні установи можуть реагувати на етичні питання та інші виклики. Автор стверджує, що релігійні установи можуть відігравати активну роль у просуванні етичних стандартів у використанні ШІ в релігійних практиках [25].

Подібно до інших інноваційних сервісів, як Bible Ai, інтелектуальні CDQA системи використовують передові технології машинного навчання та AI, щоб надавати користувачам глибше розуміння релігійних текстів та допомагати їм застосовувати ці знання у повсякденному житті [4, 10].

Використання таких систем у сфері теології та апологетики відкриває шлях до нових можливостей дослідження та розуміння релігійних текстів, а також дозволяє індивідуальним користувачам отримати більш глибоке й контекстуалізоване розуміння вчень своєї віри [24].

1.3 Огляд існуючих технологій, методів, підходів для розробки CDQA системи на мові програмування Python

Під час розробки CDQA системи в Python використання методів машинного навчання та обробки природної мови (NLP) відіграє важливу роль. Машинне навчання дозволяє моделі аналізувати великий обсяг даних та вивчати закономірності між запитаннями та відповідями. Зокрема, використання глибокого навчання та рекурентних нейронних мереж (RNN) дозволяє створити модель, яка може розпізнавати та аналізувати текстові дані з біблійних текстів для зрозуміння контексту та семантики запитань. Такий підхід дозволяє системі надавати точні та зрозумілі відповіді на запитання користувачів щодо біблійних пасажів та принципів віри [7].

Для обробки природної мови (NLP) біблійних текстів використовуються спеціалізовані методи, які дозволяють аналізувати текст та зв'язки між словами та

реченнями. Це може включати в себе розпізнавання іменованих сутностей, аналіз синтаксичних зв'язків та використання лексичних аналізаторів. Застосування цих методів дозволяє системі впізнавати ключові поняття та контекстуалізувати запитання користувачів для надання інформативних відповідей [8].

Розробка користувацького інтерфейсу є важливим аспектом створення CDQA системи, оскільки вона повинна бути зручною та інтуїтивно зрозумілою для користувачів. У Python існують багато бібліотек та інструментів для розробки веб-додатків та інших інтерфейсів, таких як Flask або Django. Ці інструменти дозволяють створювати потужні та ефективні користувацькі інтерфейси з використанням мінімальної кількості коду, що полегшує їх розробку та підтримку [5, 19].

Для розробки CDQA системи можна використовувати готові бібліотеки та фреймворки для машинного навчання та обробки природної мови. Наприклад, TensorFlow, PyTorch, spaCy та NLTK є популярними бібліотеками, які надають широкі можливості для розробки та налаштування моделей машинного навчання та NLP. Використання цих інструментів дозволяє забезпечити високу точність та ефективність CDQA системи [12].

Навчання моделі на великих обсягах біблійних текстів є важливим етапом для досягнення високої точності CDQA системи. Для цього можна використовувати різні джерела текстів, такі як Інтернет або спеціалізовані корпуси текстів, щоб навчити модель розпізнавати та аналізувати широкий спектр запитань про біблійні тексти. Застосування різноманітних джерел текстів дозволяє системі збільшити обсяг знань та розширити свій словниковий запас для кращого розуміння та відповіді на запитання користувачів.

Висновки до розділу 1

Під час аналізу інтелектуальних систем CDQA на основі обробки природної мови (NLP) та їх застосування в сфері теології та апологетики, було виявлено, що такі системи можуть відігравати важливу роль у збагаченні релігійного досвіду та

розумінні біблійних принципів. Огляд існуючих технологій та методів для розробки CDQA систем в Python показав, що використання машинного навчання та NLP дозволяє створити потужні інструменти для аналізу та інтерпретації біблійних текстів.

З метою досягнення поставленої мети дослідження, яка полягає у створенні CDQA системи для теологічних цілей, було визначено низку завдань. По-перше, необхідно провести детальний огляд існуючих методів обробки природної мови та їх застосування в області теології. Далі, потрібно розробити методи аналізу біблійних текстів та інтерпретації духовних принципів для їх використання в CDQA системі. Також важливо провести експерименти з різними моделями машинного навчання та визначити найефективніші підходи до розробки CDQA системи для теологічних цілей.

Постановка задачі передбачає створення CDQA системи, яка буде здатна надавати користувачам точні та зрозумілі відповіді на запитання з тематики біблійних текстів та доктрини віри. Для цього необхідно використовувати передові методи машинного навчання та NLP для розпізнавання контексту запитань та пошуку відповідей у біблійних текстах. Такий підхід дозволить створити інтерактивну та корисну інструментальну систему для поглибленого вивчення біблійних принципів та духовного розвитку.

Отже, на основі проведеного аналізу та постановки задачі можна зробити висновок, що розробка і впровадження CDQA системи для сфери теології та апологетики є перспективним та важливим напрямком. Така система може стати корисним інструментом для дослідження біблійних текстів, поглибленого розуміння релігійних принципів та навіть вирішення духовних питань.

2 МОДЕЛІ, МЕТОДИ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

2.1 Методи для вирішення задачі

У даному розділі розглянуто основні методи та підходи, які будуть використані для реалізації інтелектуальної CDQA системи на основі обробки природної мови (NLP) для задач теології та апологетики. Давайте розглянемо основні методи.

2.1.1 Метод обробки природної мови (NLP)

Обробка природної мови (NLP) є ключовою технологією, яка дозволяє аналізувати, розуміти та генерувати людську мову за допомогою комп'ютерів. У контексті CDQA системи, методи NLP використовуються для розуміння запитань користувачів, пошуку відповідей у текстових джерелах та формування зрозумілих відповідей.

Основні завдання NLP включають:

– сегментація тексту: розділення тексту на окремі частини або фрази для подальшого аналізу;

– розпізнавання іменованих сутностей (NER): ідентифікація та класифікація іменованих сутностей у тексті, таких як особи, місця, організації, дати тощо [7];

– аналіз синтаксису: вивчення структури речень та їхніх складових частин, таких як підмети, присудки та об'єкти;

– семантичний аналіз: розуміння значення слів та їхнього контексту в реченні або тексті [6].

Сегментація тексту є важливим першим кроком в обробці природної мови, оскільки дозволяє розділити текст на менші частини, які легше аналізувати. Це може бути виконано на рівні речень, абзаців або тематичних блоків. Важливим аспектом є також розпізнавання іменованих сутностей (NER), що дозволяє визначати важливі об'єкти в тексті та їх категорії, що особливо корисно для біблійних текстів, де часто зустрічаються імена людей, місця, події тощо [8].

Аналіз синтаксису допомагає зрозуміти граматичну структуру речень, що є критичним для правильного розуміння контексту запитання. Наприклад, визначення підмету та присудка дозволяє системі точно зрозуміти, про що йдеться у запитанні. Семантичний аналіз, в свою чергу, зосереджується на розумінні значення слів та фраз у їхньому контексті, що дозволяє системі краще зрозуміти запитання та надавати релевантні відповіді.

2.1.2 Метод машинного навчання

Машинне навчання є невід'ємною частиною створення CDQA системи. Використання алгоритмів машинного навчання дозволяє моделі аналізувати великий обсяг даних, виявляти закономірності та вчитися знаходити відповіді на запитання користувачів [5].

Основні підходи машинного навчання, які будуть використані:

– глибоке навчання (Deep Learning): використання нейронних мереж для навчання моделей, здатних аналізувати складні структури даних;

– рекурентні нейронні мережі (RNN): спеціалізовані нейронні мережі, які ефективно працюють з послідовними даними, такими як текст [12].

На рисунку 2.1 схематично відображена різниця між глибоким навчанням та машинним навчанням.

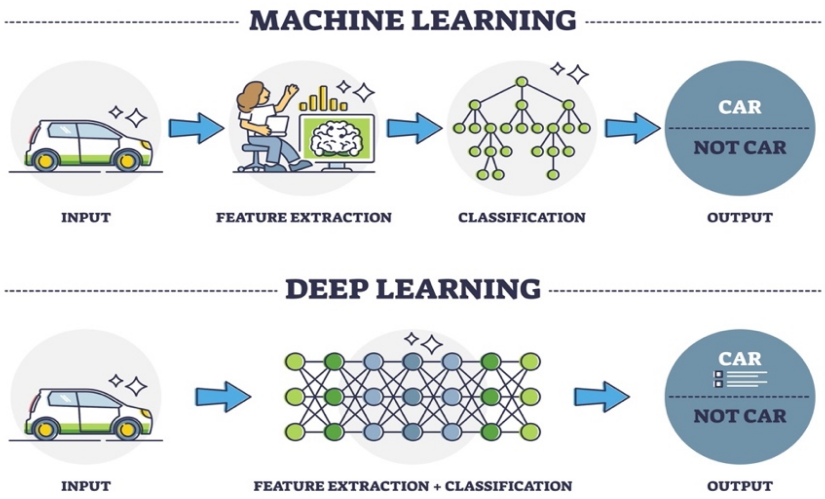


Рисунок 2.1 – Глибоке навчання проти звичайного МН [12]

Глибоке навчання передбачає використання багатошарових нейронних мереж, які здатні вивчати складні закономірності у великих наборах даних. Це дозволяє моделі навчатись на різноманітних прикладах запитань та відповідей, що покращує її здатність надавати точні та релевантні відповіді. Рекурентні нейронні мережі (RNN) є особливо корисними для роботи з послідовними даними, такими як текстові документи, оскільки вони здатні враховувати контекст попередніх слів у реченні, що підвищує точність розуміння запитань та відповіді на них.

Одним з найбільш відомих прикладів глибокого навчання в NLP є модель BERT, яка була розроблена компанією Google. BERT використовує двонаправлені трансформери для обробки тексту, що дозволяє моделі враховувати контекст як зліва, так і справа від кожного слова. Це значно покращує точність розуміння запитань та знаходження відповідей [7].

2.1.3 Метод пошуку інформації

Метод пошуку інформації полягає у виявленні релевантних документів та їх частин на основі запитів користувачів (рис. 2.2). Це включає в себе:

– індексування документів: створення структури даних для швидкого пошуку інформації;

– алгоритми пошуку: використання алгоритмів для знаходження найбільш відповідних документів та їх частин на основі запиту [20].

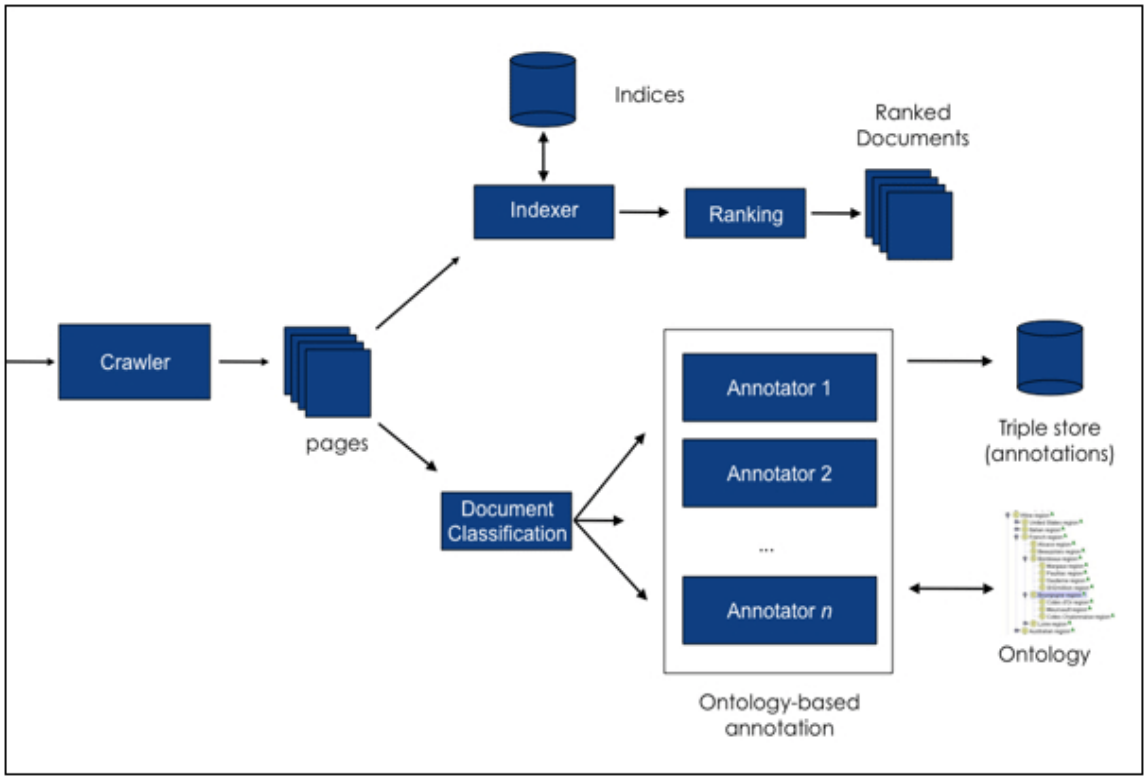


Рисунок 2.2 – Загальна схема індексування документів для пошуку інформації [13]

Індексування документів є критичним етапом, який дозволяє системі швидко знаходити релевантні документи у великому обсязі текстових даних. Це здійснюється шляхом створення спеціальних структур даних, які містять інформацію про те, де саме у документах знаходяться певні слова та фрази.

Індексування дозволяє значно скоротити час пошуку інформації та підвищити ефективність системи.

Алгоритми пошуку використовують ці індекси для швидкого знаходження документів, які відповідають запиту користувача. Це можуть бути різні алгоритми, такі як TF-IDF (term frequency-inverse document frequency), BM25, або сучасніші

методи, які використовують нейронні мережі для пошуку релевантних текстів [20]. Після знаходження відповідних документів, система може розбити їх на абзаци та відправити на подальший аналіз.

Ці методи є основою для розробки інтелектуальної CDQA системи, яка зможе ефективно обробляти запитання користувачів та надавати точні та зрозумілі відповіді. Вони забезпечують можливість глибокого аналізу текстових даних, розуміння контексту та надання релевантних відповідей, що є критично важливим для задач теології та апологетики [19, 26].

2.2 Технології розробки системи

Для розробки інтелектуальної CDQA системи для теології та апологетики буде використано кілька ключових технологій, які забезпечать ефективність та надійність системи. Ці технології включають бібліотеку CDQA, Telegram Bot API з використанням Aiogram, а також інші допоміжні інструменти. Нижче наведено детальний опис кожної з цих технологій.

2.2.1 Бібліотека CDQA на Python

Бібліотека CDQA (Closed-Domain Question Answering) є ключовим компонентом для реалізації системи відповіді на питання з використанням штучного інтелекту. Вона побудована на базі бібліотеки HuggingFace transformers і складається з двох основних блоків: cdQA, cdQA-annotator [10].

cdQA - це легкий у використанні Python-пакет, який дозволяє розробникам реалізувати QA-пайплайн для системи відповіді на питання в обмеженій предметній області. Основне призначення cdQA - забезпечити інструментарій для створення системи, яка може автоматично відповідати на запитання користувачів, аналізуючи текстові джерела інформації.

Інструментарій:

– cdQA library: основний модуль, який містить всі необхідні компоненти для реалізації QA-системи;

– retriever: компонент, відповідальний за пошук релевантних документів, що можуть містити відповіді на запитання користувачів [8, 10];

– reader: компонент, який використовує модель глибокого навчання (зокрема BERT) для аналізу відібраних документів та знаходження найбільш ймовірних відповідей [8, 10].

CDQA дозволяє створювати інтегровані системи, які можуть обробляти запитання користувачів, здійснювати пошук релевантних документів та надавати точні відповіді на основі аналізу тексту. CDQA була обрана завдяки її здатності ефективно відповідати на запитання в обмеженій предметній області, що є критичним для задач теології та апологетики. Використання BERT забезпечує високу точність та глибоке розуміння тексту, що дозволяє створити надійну та точну систему. На рисунку 2.3 показана схема роботи архітектури CDQA бібліотеки.

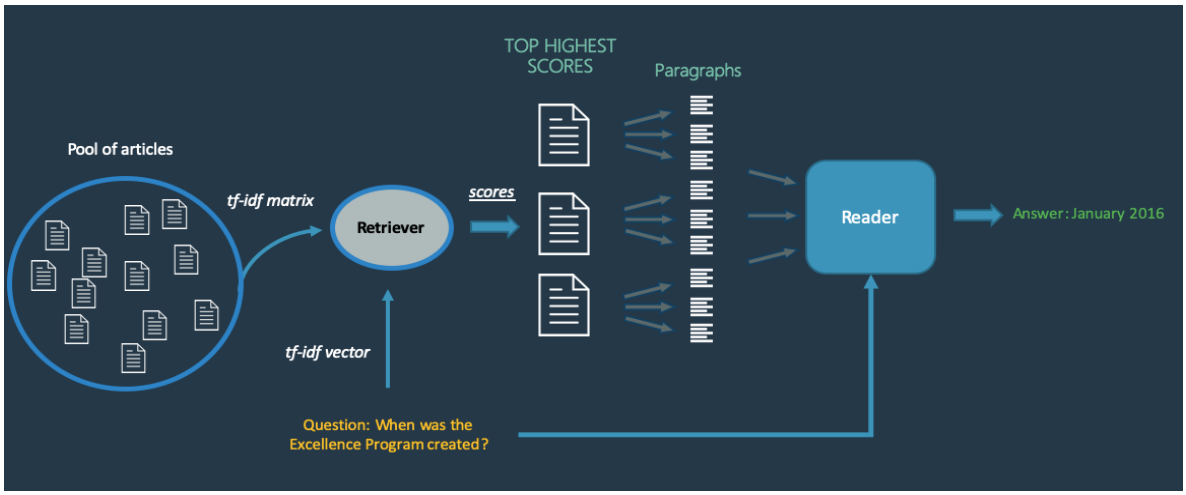


Рисунок 2.3 – Схема роботи архітектури CDQA бібліотеки [8]

cdQA-annotator.

Сутність та призначення: cdQA-annotator - це інструмент, створений для полегшення анотації наборів даних питань-відповідей. Він використовується для

оцінки та налаштування моделей QA, що є важливим етапом у розробці та вдосконаленні системи. На рисунку 2.4 показано інтерфейс cdQA-annotator [10].

Інструментарій:

– інтерфейс анотації: надає зручний користувацький інтерфейс для маркування даних, що дозволяє дослідникам легко створювати та керувати анотованими наборами даних;

– інтеграція з моделями: підтримка безпосередньої інтеграції з моделями глибокого навчання, що дозволяє швидко оцінювати та налаштовувати моделі на основі анотованих даних.

Призначення: cdQA-annotator полегшує процес створення та управління наборами даних, необхідними для навчання та налаштування моделей QA, що значно підвищує точність та ефективність системи.

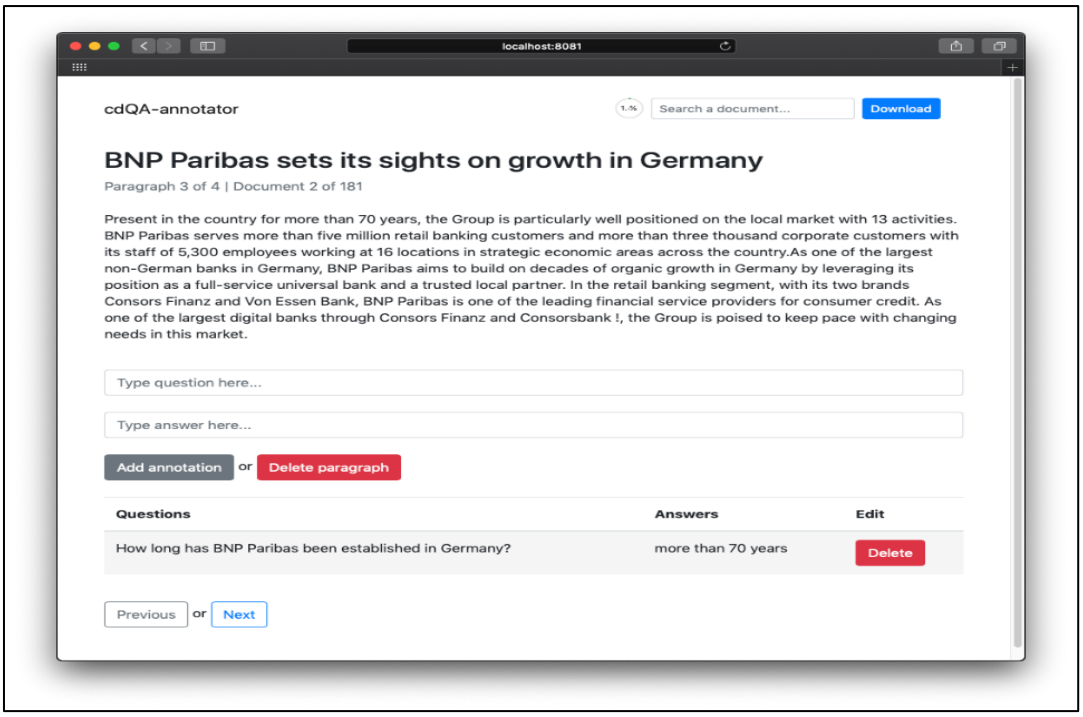


Рисунок 2.4 – Інтерфейс cdQA-annotator [11]

Retriever.

Призначення: Вибір найбільш ймовірних документів, що містять відповіді на запитання користувачів.

Принцип роботи: Використовує різні алгоритми пошуку, такі як TF-IDF або BM25, для швидкого індексування та пошуку релевантних документів. Після отримання запиту користувача, retriever аналізує базу даних та вибирає документи, що найбільше відповідають запиту [10].

Reader

Призначення: Аналіз відібраних документів для знаходження найбільш ймовірних відповідей.

Принцип роботи: Використовує модель BERT для розуміння тексту та знаходження відповідей. Reader ділить документи на абзаци та аналізує кожен абзац окремо, визначаючи найбільш ймовірні відповіді на запитання користувачів [10].

Кроки роботи системи:

1) запит користувача: користувач вводить запитання через інтерфейс cdQA-ui;

2) пошук документів: retriever здійснює пошук релевантних документів у базі даних;

3) аналіз тексту: reader аналізує відібрані документи, використовуючи модель BERT;

4) формування відповіді: система порівнює можливі відповіді та вибирає найбільш ймовірну, яка відповідає запиту користувача;

5) відповідь користувачу: користувач отримує відповідь через інтерфейс cdQA-ui [9-10].

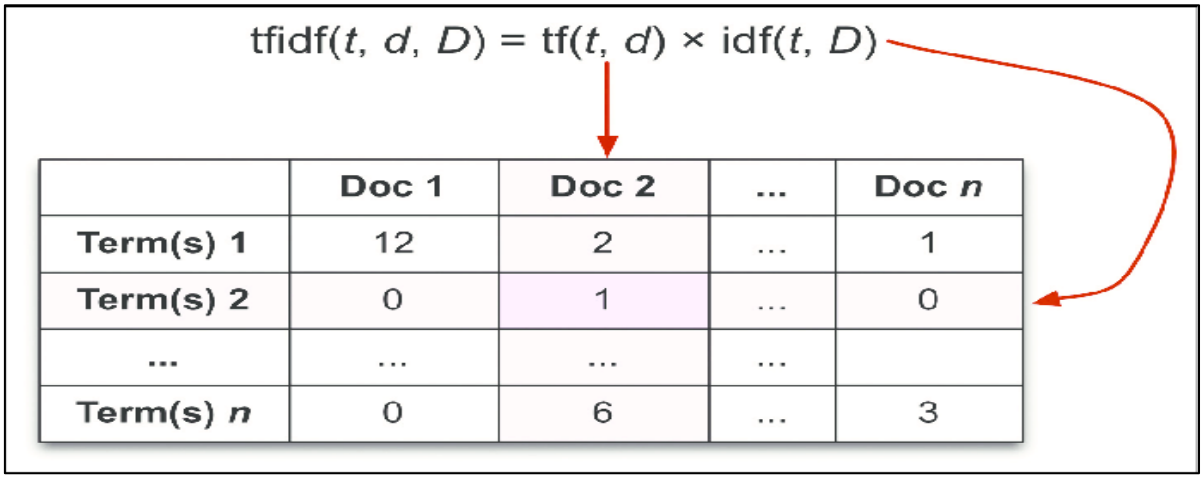
Алгоритми TF-IDF та BM25.

TF-IDF (Term Frequency-Inverse Document Frequency) та BM25 (Best Matching 25) є двома широко використовуваними алгоритмами для інформаційного пошуку та ранжування документів. TF-IDF був розроблений у 1970-х роках та став основою багатьох систем інформаційного пошуку. BM25 є більш сучасною модифікацією

моделі TF-IDF, розробленою у 1990-х роках в рамках досліджень в університеті Глазго.

TF-IDF (Term Frequency-Inverse Document Frequency).

TF-IDF є статистичним методом, що використовується для оцінки важливості слова в документі відносно до колекції документів. Він враховує як частоту слова в документі (Term Frequency), так і зворотну частоту документів (Inverse Document Frequency), в яких це слово з'являється. TF-IDF обчислює значення для кожного слова в документі, що дозволяє визначити його важливість (2.5). Це значення є добутком частоти слова у документі та зворотної частоти документів, в яких це слово зустрічається. В результаті слова, які часто зустрічаються в документі, але рідко в інших документах, отримують високу оцінку [20].



	Doc 1	Doc 2	...	Doc *n*
Term(s) 1	12	2	...	1
Term(s) 2	0	1	...	0
...	...	...	...	
Term(s) *n*	0	6	...	3

Рисунок 2.5 – Схема роботи методу TF-IDF [20]

Переваги: легкість реалізації та використання; ефективність у виділенні ключових слів у текстах.

Недоліки: не враховує контекст слів; може бути менш ефективним для довгих документів.

Приклади використання: TF-IDF використовується у багатьох системах інформаційного пошуку, включаючи пошукові системи, системи рекомендацій та аналізу тексту. Наприклад, пошукові системи використовують TF-IDF для ранжування результатів пошуку за релевантністю до запиту користувача [20].

BM25 (Best Matching 25).

BM25 є більш складним алгоритмом ранжування документів на основі термінів, що використовується в інформаційному пошуку. Він є варіацією моделі TF-IDF і враховує як частоту термінів у документі, так і їх частоту у всій колекції документів. BM25 обчислює оцінки для кожного документа на основі запиту, враховуючи частоту термінів, довжину документів та інші фактори. Це дозволяє більш точно ранжувати документи за релевантністю до запиту.

Переваги: більш точне ранжування у порівнянні з TF-IDF; врахування довжини документа та інші параметри для підвищення точності.

Недоліки: більш складна реалізація; потребує налаштування параметрів для оптимальної роботи.

Приклади використання: BM25 широко використовується в сучасних пошукових системах та інформаційних системах, таких як Elasticsearch та Solr, для забезпечення точного та ефективного пошуку релевантних документів.

BERT (Bidirectional Encoder Representations from Transformers).

BERT - це модель глибокого навчання для обробки природної мови, розроблена дослідниками Google та представлена у 2018 році. Вона стала революційною завдяки здатності обробляти текст у двох напрямках, враховуючи контекст з обох боків кожного слова. BERT швидко здобула популярність та стала основою для багатьох сучасних NLP-додатків. BERT - це модель глибокого навчання для обробки природної мови, розроблена компанією Google. Вона використовує трансформери для двонаправленого кодування тексту, що дозволяє моделі враховувати контекст як зліва, так і справа від кожного слова.

Принцип роботи:

– двонаправлене кодування: BERT аналізує текст у двох напрямках, що дозволяє враховувати контекст з обох боків кожного слова. Це значно підвищує точність моделі у порівнянні з однонаправленими моделями [7, 14];

– трансформери: BERT використовує архітектуру трансформерів, що складається з механізму уваги, який дозволяє моделі зосереджуватись на найбільш важливих частинах тексту під час обробки.

Методи роботи:

– попереднє навчання: BERT попередньо навчається на великому корпусі текстів, таких як Вікіпедія, для розуміння загальних мовних закономірностей;

– тонке налаштування: після попереднього навчання модель BERT налаштовується на конкретних завданнях, таких як відповіді на питання, шляхом додаткового навчання на спеціалізованих наборах даних.

Переваги:

– висока точність завдяки двонаправленому кодуванню контексту;

– гнучкість у застосуванні до різних завдань обробки природної мови;

– можливість тонкого налаштування для специфічних завдань.

Недоліки:

– велика обчислювальна складність та потреба в значних ресурсах для навчання та використання моделі;

– складність у налаштуванні та інтеграції для нових користувачів.

BERT використовується у багатьох додатках обробки природної мови, включаючи системи відповіді на питання, машинний переклад, аналіз настроїв та інші завдання. Наприклад, Google використовує BERT для покращення результатів пошуку, забезпечуючи більш точні та релевантні відповіді на запити користувачів [7][19]. BERT була обрана для використання в системі CDQA завдяки її здатності забезпечувати високу точність та глибоке розуміння тексту. Двонаправлене кодування дозволяє моделі враховувати контекст з обох боків, що є критичним для точного розуміння та інтерпретації складних текстових запитів [8].

2.2.2 Telegram Bot API та Aiogram

Для реалізації користувацького інтерфейсу буде використовуватись Telegram Bot, написаний на Python з використанням бібліотеки Aiogram 3.0. Aiogram - це потужна бібліотека для створення Telegram ботів, яка використовує асинхронність Python та надає багато інструментів для взаємодії з Telegram Bot API [4].

Сутність та призначення

Telegram Bot API дозволяє створювати інтерактивні чат-боти, які можуть взаємодіяти з користувачами у реальному часі. Aiogram є високорівневою бібліотекою, яка спрощує розробку таких ботів завдяки підтримці асинхронних операцій та зручного API. Основне призначення Aiogram - забезпечити інструментарій для створення високопродуктивних та ефективних Telegram ботів, які можуть обробляти велику кількість запитів користувачів одночасно [4][5].

Інструментарій Aiogram.

Aiogram надає широкий спектр інструментів та функціональності для розробки ботів, включаючи [4]:

– асинхронна обробка запитів: Aiogram використовує асинхронні функції Python для обробки запитів користувачів. Це дозволяє одночасно обробляти велику кількість запитів, забезпечуючи високу продуктивність та швидкість відповіді;

– фреймворк для роботи з командами: Aiogram підтримує визначення команд, що дозволяє легко створювати різні функціональності для бота, такі як обробка команд користувачів та інтерактивні меню;

– механізм фільтрів: бібліотека надає механізми фільтрації для обробки різних типів повідомлень та подій, що дозволяє більш гнучко налаштовувати логіку обробки запитів;

– менеджери станів: Aiogram включає інструменти для управління станами користувачів, що дозволяє створювати складні діалоги та процеси взаємодії.

Патерни програмування в Aiogram

Aiogram підтримує кілька патернів програмування, що підвищують гнучкість та ефективність розробки ботів [4]:

– асинхронне програмування: використання асинхронних функцій та бібліотек Python для забезпечення високої продуктивності та одночасної обробки запитів;

– декоратори: Aiogram використовує декоратори для визначення обробників команд та повідомлень, що спрощує структуру коду та робить його більш читабельним;

– менеджери контексту: використання менеджерів контексту для управління ресурсами та забезпечення коректного завершення операцій.

Принципи роботи та взаємодія з Telegram сервером, методи обробки запитів [4]**:**

1) webhooks: Aiogram підтримує роботу з webhooks, що дозволяє боту отримувати оновлення безпосередньо від сервера Telegram. Це забезпечує більш швидку обробку запитів та зменшує навантаження на сервер;

2) long polling: Aiogram також підтримує метод long polling, коли бот періодично запитує сервер Telegram на наявність нових повідомлень. Цей метод простіший у налаштуванні та використовується за замовчуванням.

Процес обробки запитів [4]**:**

– отримання оновлень: бот отримує оновлення (нові повідомлення та події) від сервера Telegram через webhooks або long polling;

– фільтрація оновлень: Aiogram використовує механізми фільтрації для визначення типу повідомлення та вибору відповідного обробника;

– обробка повідомлень: обробники (функції або методи) виконують необхідні дії для обробки повідомлення, такі як відправка відповіді користувачу або виконання певних команд;

– управління станами: якщо бот підтримує складні діалоги або процеси, Aiogram використовує менеджери станів для збереження та відновлення стану користувача між повідомленнями.

Приклади використання

Telegram боти на базі Aiogram широко використовуються для різних задач, включаючи автоматизацію бізнес-процесів, підтримку користувачів, інтерактивні ігри, освітні проекти та багато іншого. Наприклад, боти можуть автоматично відповідати на запити клієнтів, надсилати сповіщення про нові події, проводити опитування та голосування, надавати інформаційні послуги тощо [4-5].

Переваги використання Aiogram [4-5]**:**

– висока продуктивність: завдяки асинхронній обробці запитів, боти можуть одночасно обробляти велику кількість повідомлень без зниження продуктивності;

– гнучкість та зручність розробки: Aiogram надає зручний API та підтримку сучасних патернів програмування, що робить розробку ботів швидкою та ефективною;

– широка функціональність: бібліотека підтримує різні механізми взаємодії з користувачами, такі як команди, меню, форми та діалоги.

Недоліки:

– потреба в налаштуваннях: задля забезпечення безпеки та стабільності роботи бота, необхідно правильно налаштувати сервер та вебхуки;

– обмеження Telegram API: незважаючи на широкі можливості, деякі функції можуть бути обмежені API Telegram.

Причини вибору: Aiogram була обрана завдяки її здатності забезпечувати високу продуктивність та зручність розробки, що є важливим для створення інтерактивного інтерфейсу користувача. Асинхронні можливості бібліотеки дозволяють ефективно обробляти запити користувачів у реальному часі, а підтримка сучасних патернів програмування підвищує гнучкість та ефективність розробки [4-5].

2.2.3 Інші допоміжні інструменти

Для підтримки роботи системи будуть використані також Environs - бібліотека Python для парсингу (екстракції) змінних середовища з .env файлу, що спрощує конфігурацію та налаштування системи. Це дозволяє зберігати конфігураційні параметри в окремому файлі, забезпечуючи зручність їх зміни без необхідності редагування коду.

Причини вибору: Environs спрощує управління конфігураційними параметрами, забезпечуючи гнучкість та зручність у налаштуванні системи.

Висновки до розділу 2

У цьому розділі було детально розглянуто моделі, методи та інформаційні технології, що застосовуються для вирішення задачі створення інтелектуальної системи питань-відповідей (CDQA) в галузі теології та апологетики. Використання бібліотеки CDQA, яка побудована на базі HuggingFace transformers, дозволяє ефективно вирішувати завдання пошуку відповідей на запитання користувачів, використовуючи передові моделі глибокого навчання, такі як BERT [8-9].

Розгляд компонентів бібліотеки CDQA, включаючи cdQA, cdQA-annotator та cdQA-ui, показав їх важливість у створенні комплексної системи, здатної здійснювати пошук релевантних документів (Retriever) та аналіз тексту для знаходження відповідей (Reader). Використання алгоритмів TF-IDF та BM25 для пошуку документів дозволяє забезпечити високу точність та швидкість обробки запитів, що є критичним для створення ефективної системи [8-9].

Детальний огляд моделі BERT підкреслив її важливість для обробки природної мови, завдяки здатності враховувати контекст з обох боків кожного слова, що значно підвищує точність відповіді на запитання. Використання BERT в системі CDQA забезпечує глибоке розуміння тексту та дозволяє надавати більш точні та релевантні відповіді.

Також було розглянуто використання Telegram Bot API та бібліотеки Aiogram для створення інтерактивного користувацького інтерфейсу. Aiogram надає можливості асинхронної обробки запитів, що забезпечує високу продуктивність та зручність розробки ботів. Розгляд патернів програмування, підтримуваних Aiogram, таких як асинхронне програмування та використання декораторів, показав їх ефективність у створенні гнучкої та масштабованої системи.

Таким чином, вибір та використання описаних технологій та методів є виправданим та ефективним для розробки інтелектуальної CDQA системи, спрямованої на надання консультацій з теологічних та апологетичних питань. Вони забезпечують високу точність, продуктивність та зручність користування системою, що дозволяє досягти поставлених цілей дослідження.

3 СТВОРЕННЯ НАБОРУ ДАНИХ. НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ

3.1 Створення набору даних для навчання нейронної мережі

Набір даних є ключовим елементом при навчанні нейронних мереж QA типу, оскільки саме з набору даних модель буде брати дані, на основі яких зможе надавати користувачеві відповіді. Перед навчанням нейронної мережі насамперед потрібно мати власний набір даних, або завантажити з відкритих ресурсів який підходить для обраної задачі.

Для виконання задачі цієї кваліфікаційної роботи потрібно мати дані теологічного (напр. Біблійні тексти тощо.) та апологетичного характерів (напр. факти з наукових статей, які вивчають Біблійну тему і інше).

Спершу звернемося за даними теологічного характеру, а саме – Біблією. За цим набором даних я звернувся до популярного ресурсу Kaggle, де знайшов «The King James Bible», в якому міститься вся Біблія від Буття (Старий Завіт) до Одкровення Івана Богослова (Новий Завіт) англійською мовою. Структура документу наступна: кожен рядок це окремий Біблійний вірш, підписаний книгою, главою та порядковим номером у цій главі (рис. 3.1) [16].

Було вирішено використати тільки Євангельську цього набору задля звуження об’єму інформації, на якій буде навчатися модель, а саме – Євангелію від Матвія; це дозволить більш точно навчити модель та більш якісно оцінити точність її відповіді, залишаючи поле для зростання на майбутнє.

Для використання цього набору даних в системі CDQA, спершу, потрібно привести цей набір до очікуваного формату, цим форматом є Pandas фрейму даних з наступними колонками: ‘title’ та ‘paragraphs’, при цьому кожен окремий параграф має йти через кому, утворюючи лист з параграфів в кожному рядку (рис. 3.2) [11].

7940	Revelation 22:	Revelatio	22	3	And there shall be no more curse: but the throne of God and of the Lamb shall be in it; and his servants shall serve him:
7941	Revelation 22:	Revelatio	22	4	And they shall see his face; and his name shall be in their foreheads.
7942	Revelation 22:	Revelatio	22	5	And there shall be no night there; and they need no candle, neither light of the sun; for the Lord God giveth them light: and they shall reign for ever and ever.
7943	Revelation 22:	Revelatio	22	6	And he said unto me, These sayings are faithful and true: and the Lord God of the holy prophets sent his angel to shew unto his servants the things which must shortly be
7944	Revelation 22:	Revelatio	22	7	Behold, I come quickly: blessed is he that keepeth the sayings of the prophecy of this book.
7945	Revelation 22:	Revelatio	22	8	And I John saw these things, and heard them. And when I had heard and seen, I fell down to worship before the feet of the angel which shewed me these things.
7946	Revelation 22:	Revelatio	22	9	Then saith he unto me, See thou do it not: for I am thy fellowservant, and of thy brethren the prophets, and of them which keep the sayings of this book: worship God.
7947	Revelation 22:	Revelatio	22	10	And he saith unto me, Seal not the sayings of the prophecy of this book: for the time is at hand.
7948	Revelation 22:	Revelatio	22	11	He that is unjust, let him be unjust still: and he which is filthy, let him be filthy still: and he that is righteous, let him be righteous still: and he that is holy, let him be holy still
7949	Revelation 22:	Revelatio	22	12	And, behold, I come quickly; and my reward is with me, to give every man according as his work shall be.
7950	Revelation 22:	Revelatio	22	13	I am Alpha and Omega, the beginning and the end, the first and the last.
7951	Revelation 22:	Revelatio	22	14	Blessed are they that do his commandments, that they may have right to the tree of life, and may enter in through the gates into the city.
7952	Revelation 22:	Revelatio	22	15	For without are dogs, and sorcerers, and whoremongers, and murderers, and idolaters, and whosoever loveth and maketh a lie.
7953	Revelation 22:	Revelatio	22	16	I Jesus have sent mine angel to testify unto you these things in the churches. I am the root and the offspring of David, and the bright and morning star.
7954	Revelation 22:	Revelatio	22	17	And the Spirit and the bride say, Come. And let him that heareth say, Come. And let him that is athirst come. And whosoever will, let him take the water of life freely.
7955	Revelation 22:	Revelatio	22	18	For I testify unto every man that heareth the words of the prophecy of this book, If any man shall add unto these things, God shall add unto him the plagues that are writte
7956	Revelation 22:	Revelatio	22	19	And if any man shall take away from the words of the book of this prophecy, God shall take away his part out of the book of life, and out of the holy city, and from the thin
7957	Revelation 22:	Revelatio	22	20	He which testifieth these things saith, Surely I come quickly. Amen. Even so, come, Lord Jesus.
7958	Revelation 22:	Revelatio	22	21	The grace of our Lord Jesus Christ be with you all. Amen.

SUM 230 882 AVERAGE 14,50810607 MIN 1 MAX 80 COUNTA 39 790

Рисунок 3.1 – Біблійний набір даних Нового Завіту, взятий з Kaggle

У результаті було вирішено зробити кожну главу зі всіма віршами окремим рядком фрейму даних, присвоївши йому певне ім’я, відповідне до змісту Біблійної глави, щоб потім привести кожен параграф (тобто, главу) до потрібного формату.

title	paragraphs
The Article Title	[Paragraph 1 of Article, ... , Paragraph N of Article]

Рисунок 3.2 – Очікуваний формат набору даних для бібліотеки CDQA [9]

Для вирішення цього питання створюємо наступний скрипт, який приймає .csv файл з віршами з Біблії в описаному вище форматі та об’єднує ці вірші в глави в потрібному для роботи форматі, а саме – лист з параграфів, далі кожній главі Біблії скрипт надає визначене в ім’я та записує це все в новий .csv файл, який буде використовуватися для навчання BERT моделі відповідно до задач кваліфікаційної роботи [7].

```
import pandas as pd
titles = {...} # here are specified titles for each of the chapters of the Gospels
def split_paragraphs(text):
    return [paragraph.strip() for paragraph in text.strip().split('\n') if paragraph.strip()]
# Function to create the new dataframe with combined paragraphs and titles
def create_combined_dataframe(df):
    combined_data = []
    # Group by book and chapter
    grouped = df.groupby(['book', 'chapter'])
    for (book, chapter), group in grouped:
        paragraphs = group['text'].apply(split_paragraphs).sum()
        # Construct the title based on book and chapter
        title = titles[book][chapter]
        combined_data.append({
            'title': title,
            'paragraphs': paragraphs
        })
    # Create a new dataframe with the combined data
    combined_df = pd.DataFrame(combined_data)
    return combined_df
# Read the CSV file with semicolon delimiter
df = pd.read_csv('files/gospels.csv', delimiter=';')
# Create the combined dataframe
combined_df = create_combined_dataframe(df)
# Save the new dataframe to a CSV file
combined_df.to_csv('files/combined_gospels.csv', index=False)
```

У результаті маємо .csv документ зі всіма віршами з Євангелії від Матвія, розбитими по параграфам у необхідному нам форматі, до кожного параграфу документу призначені назви цих частин тексту. Саме цей файл буде використовуватися бібліотекою CDQA для знаходження відповідей на теологічні питання Євангельського характеру [11].

Тепер перейдемо до підготовки набору даних апологетичного характеру. Такий специфічний набір даних для навчання моделі не було знайдено на відповідних ресурсах, тому було вирішено використати інформацію та докази, які

були зібрані в книзі «Не просто тесля» Джоша Макдауелла, в якій було зібрано відповіді на найрозповсюдженіших питання та сумніви стосовно Християнських доктрин з точки зору апологетики, використовуючи різноманітніші джерела наукового та історичного змісту, що сильно облегшить нам задачу на обробку великої кількості інформації [17].

Для початку було використано текст деяких вибіркових глав зазначеної вище книги та, таким самим чином, як й з теологічними матеріалами, їх було приведено до потрібного формату набору даних, як показано на рисунку 3.3.

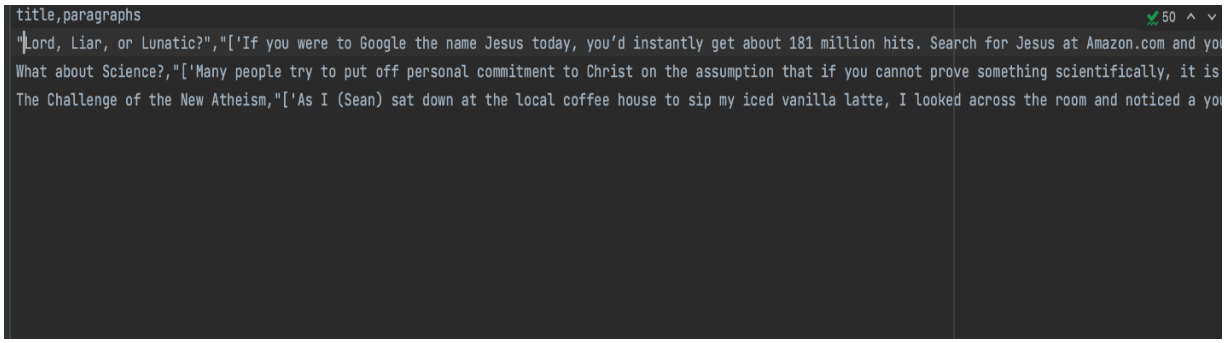


Рисунок 3.3 – Створений .csv файл з правильно відформатованими віршами

У результаті маємо 2 файли з наборами даних теологічного та апологетичного характерів, які будуть використані моделлю штучного інтелекту задля знаходження відповідей на питання користувача застосунку [11, 17].

3.2 Тонке налаштування попередньо навченої моделі DistilBERT

Тонке налаштування (англ. fine-tuning) — це процес адаптації попередньо натренованої моделі до конкретного завдання [27].

DistilBERT — це спрощена та менш важка версія моделі BERT, оптимізована для швидшої обробки з меншими витратами обчислювальних ресурсів, при цьому зберігаючи значну частину ефективності BERT [7]. Для налаштування попередньо навченої моделі BERT спочатку необхідно підготувати SQuAD-подібний .json файл з підготовленими питаннями та заздалегідь правильними відповідями на них у контексті – це надає можливість навчити модель на цих підготовлених питаннях.

За для підготовки згаданого вище набору даних, скористаємося cdQA-annotator, для цього виконаємо наступні кроки:

1) клонуємо cdQA-annotator з офіційного GitHub репозиторію [13];

2) у терміналі заходимо в кореневу директорію проекту та встановлюємо всі необхідні залежності командою «npm install» (має бути встановлений Node на локальній операційній системі);

3) запускаємо проект командою «npm run serve»;

4) переходимо за адресою http://localhost:8080 та завантажуємо наш набір даних, попередньо підготовлений під SQuAD-подібний формат (приклад цього формату показаний на рисунку 3.5);

5) далі вручну створюємо питання та позначаємо відповіді до них з тексту, шляхом виділення фрагментів тексту.

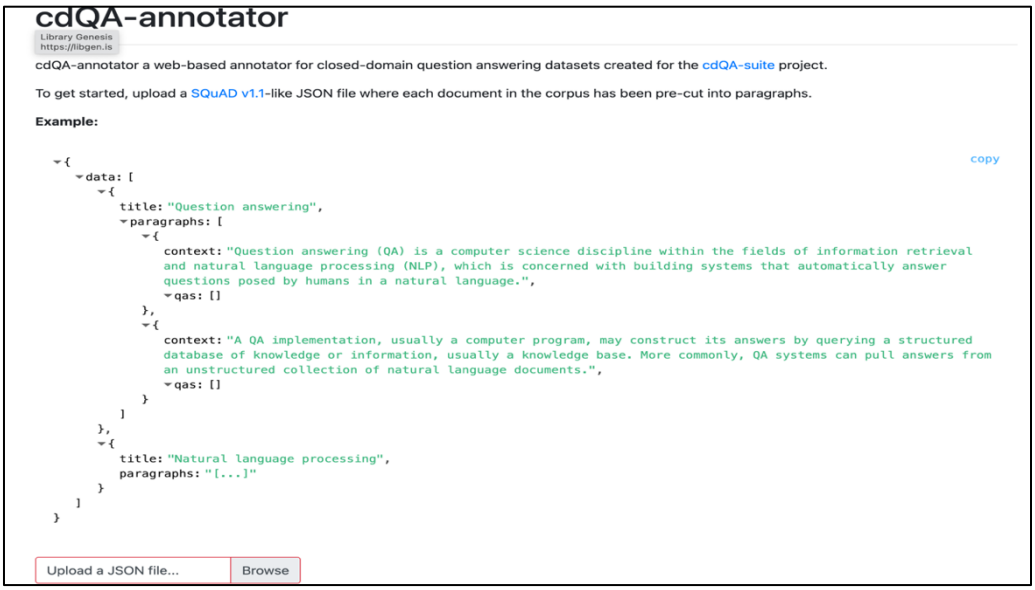


Рисунок 3.4 – Головна сторінка cdQA-annotator

На рисунку 3.5 показаний процес анотування підготовленого набору даних в запущеному cdQA-annotator.

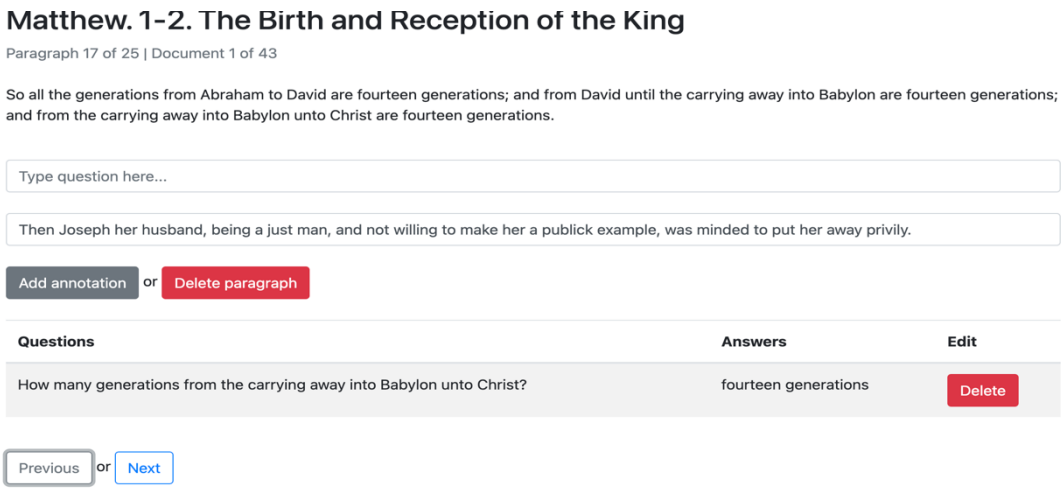


Рисунок 3.5 – Процес анотування підготовленого набору даних

У результаті отримуємо великий (11 144 рядка) .json файл з підготовленими питаннями та відповідями дотичних до набору даних (рис. 3.6). Цей файл буде використано для навчання попередньо навченої моделі DistilBERT для отримання більш точних результатів, підвищення ефективності роботи моделі з кастомним набором даних. Далі необхідно підготувати середовище розробки в проекті розробки інтелектуальної системи.

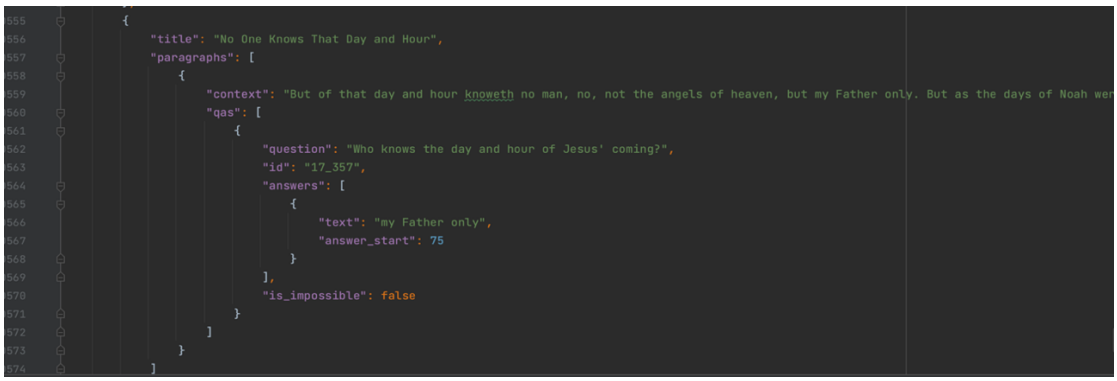


Рисунок 3.6 – Отриманий результат анотування набору даних

Тепер встановлюємо у віртуальне оточення бібліотеки з файлу *requirements.txt,* який має усі необхідні інструменти для налаштування нейронної мережі. Це можливо зробити за допомогою команди *“pip install -e .”*, як показано на малюнку 3.7.

```
(evangelistbot) (venv) user@MacBook-Pro cdQA % pip install -e .
Obtaining file:///Users/user/PycharmProjects/EvangelistBot/cdQA
  Preparing metadata (setup.py) ... done
Requirement already satisfied: Cython in /opt/homebrew/Caskroom/miniforge/base/envs/evangelistbot/lib/python3.8/site-packages (from cdqa==1.3.9) (0.29.23)
Collecting Flask==1.1.1 (from cdqa==1.3.9)
  Using cached Flask-1.1.1-py2.py3-none-any.whl.metadata (4.4 kB)
Collecting flask_cors==3.0.8 (from cdqa==1.3.9)
  Using cached Flask_Cors-3.0.8-py2.py3-none-any.whl.metadata (5.4 kB)
Collecting joblib==0.13.2 (from cdqa==1.3.9)
  Using cached joblib-0.13.2-py2.py3-none-any.whl.metadata (4.5 kB)
Collecting pandas==0.25.0 (from cdqa==1.3.9)
  Using cached pandas-0.25.0.tar.gz (12.6 MB)
```

Рисунок 3.7 – Завантаження необхідних залежних бібліотек для роботи

Таким чином ми маємо підготовлене виртуальне середовище для подальшої роботи з системою. На рисунку 3.8 показана архітектура проекту.

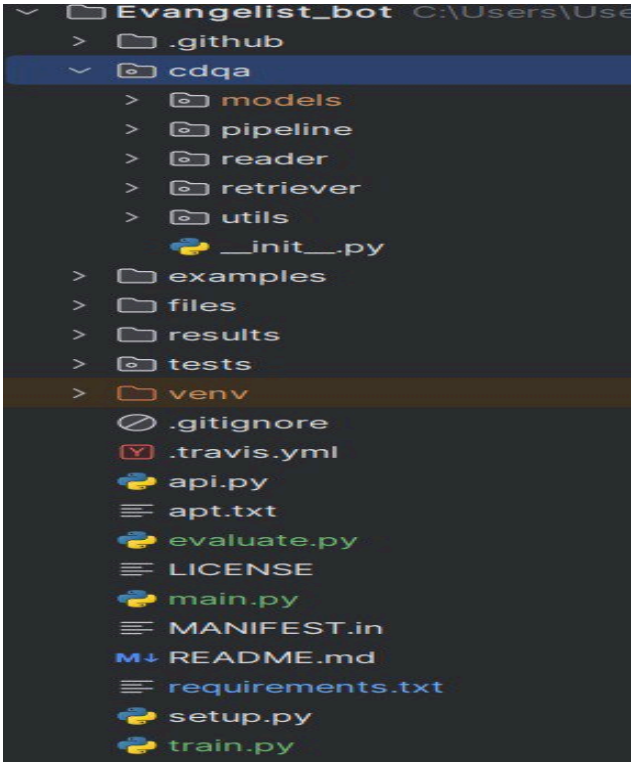


Рисунок 3.8 – Структура проекту

Далі було завантажено модель DistilBERT для її подальшого навчання на підготовленому SQUAD-подібному наборі даних.

Тонке налаштування (fine-tuning) моделі DistilBERT під біблійний набір даних має ряд значних переваг:

– покращена точність і контекстуальність: тонке налаштування дозволяє моделі краще зрозуміти специфіку та контекст біблійних текстів, що є дуже важливим для точного розпізнавання і відповіді на питання. Модель навчиться краще розрізняти нюанси біблійних віршів і відповідати відповідно до них [27];

– адаптація до специфічної мови та стилю: біблійні тексти мають унікальний стиль і словник, який відрізняється від сучасної мови. Тонке налаштування моделі BERT на цьому конкретному корпусі даних допомагає моделі краще розуміти ці тексти і відповідати на питання більш адекватно [27];

– підвищення релевантності відповідей: завдяки тонкому налаштуванню, модель буде краще виявляти найбільш релевантні частини тексту для відповіді на запитання, що забезпечить більш точні та інформативні відповіді [27];

– зменшення помилок: навчання моделі на специфічному наборі даних дозволяє знизити кількість помилок, пов'язаних з неправильною інтерпретацією тексту. модель навчиться краще визначати початок і кінець відповідей в контексті біблійних текстів, що зменшить кількість помилкових відповідей [27];

– висока узгодженість відповідей: тонке налаштування допомагає забезпечити високу узгодженість у відповідях моделі. Це означає, що модель буде давати відповіді, які узгоджуються з загальним контекстом біблійних текстів, а не суперечать їм [27];

– оптимізація для конкретного завдання: налаштування моделі під конкретне завдання дозволяє максимально ефективно використовувати її можливості. У випадку біблійних питань це означає, що модель буде спеціалізована на завданні відповіді на питання з високою точністю [27];

– підвищена здатність до генералізації: хоча модель буде спеціалізована на біблійних текстах, тонке налаштування також підвищить її здатність до генералізації в межах подібних текстів, що робить її більш універсальною для інших релігійних текстів [27].

На рисунку 3.9 відображене табличне порівняння різних моделей NLP за основними характеристиками: розмір (в мілійонах), час на навчання, ефективність, внутрішня база даних, метод.

	BERT	RoBERTa	DistilBERT	XLNet
Size (millions)	**Base**: 110 **Large**: 340	**Base**: 110 **Large**: 340	**Base:** 66	**Base**: ~110 **Large**: ~340
Training Time	**Base**: 8 x V100 x 12 days* **Large:** 64 TPU Chips x 4 days (or 280 x V100 x 1 days*)	**Large:** 1024 x V100 x 1 day; 4-5 times more than BERT.	**Base**: 8 x V100 x 3.5 days; 4 times less than BERT.	**Large:** 512 TPU Chips x 2.5 days; 5 times more than BERT.
Performance	Outperforms state-of-the-art in Oct 2018	2-20% improvement over BERT	3% degradation from BERT	2-15% improvement over BERT
Data	16 GB BERT data (Books Corpus + Wikipedia). 3.3 Billion words.	160 GB (16 GB BERT data + 144 GB additional)	16 GB BERT data. 3.3 Billion words.	**Base**: 16 GB BERT data **Large**: 113 GB (16 GB BERT data + 97 GB additional). 33 Billion words.
Method	BERT (Bidirectional Transformer with MLM and NSP)	BERT without NSP**	BERT Distillation	Bidirectional Transformer with Permutation based modeling

Рисунок 3.9 – Порівняння різних NLP моделей [7]

Таким чинмо, тонке налаштування моделі DistilBERT на специфічному наборі даних біблійних питань зробить її більш ефективною, точною і корисною для використання в системі CDQA. Далі переходимо до файлу «train.py», де задаємо наступний код для навчання нейронної мережі.

```
from cdqa.pipeline import QAPipeline
cdqa_pipeline = QAPipeline(reader='cdqa/models/distilbert_qa.joblib')
cdqa_pipeline.fit_reader('files/squad-like_dataset.json')
cdqa_pipeline.dump_reader('cdqa/models/distilbert_qa_tuned.joblib')
```

Завантажуємо з офіційного сайту переднавчену модель DistilBert, та далі запускаємо навчання з 3 епохами на локальному пристрої.

Модель складається із 6 шарів, які містять у сумі 66 мільйонів параметрів. Архітектура побудована на основі трансформерів, що включає в себе шар уваги, нормалізації та feed-forward шари. Модель має спрощену структуру порівняно з оригінальним BERT, що дозволяє їй зберігати значну частину ефективності, але при цьому бути більш швидкою та легкою. DistilBERT використовує приблизно 40% менше параметрів та виконує обчислення в 60% швидше, ніж BERT, що зазначено у таблиці 2.1 з характеристиками моделей DistilBERT. Це робить модель ідеальною для використання в задачах обробки природної мови, таких як класифікація тексту, запитання-відповіді та інші NLP завдання. Після чого запускаємо код та отримаємо наступне (рис. 3.10).

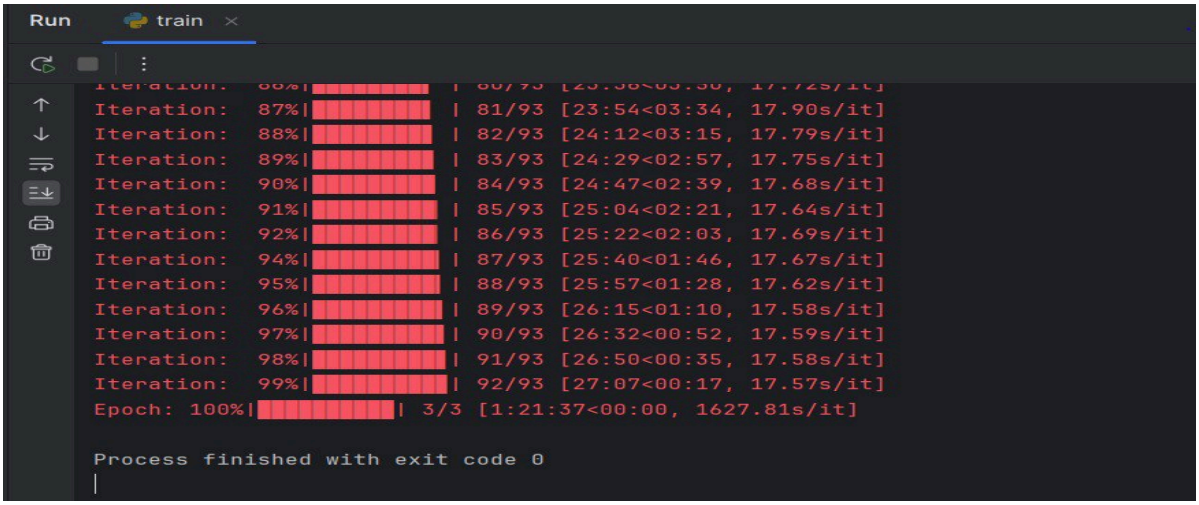


Рисунок 3.10 – Звершений процес навчання NLP мережі

Процес тонкого налаштування (fine-tuning):

– розбивка на епохи та ітерації: модель тренується на всьому наборі даних кілька разів (епохи). Кожна епоха складається з певної кількості ітерацій. На зображенні видно, що процес складається з 93 ітерацій;

– оновлення ваг моделі: на кожній ітерації модель отримує невеликий пакет (batch) даних, розраховує передбачення та порівнює їх з правильними відповідями. Ваги моделі оновлюються на основі різниці між передбаченням та правильною відповіддю (втрата або loss);

– оцінка прогресу: кожна ітерація показує, скільки часу було витрачено та скільки часу залишилось. Це дозволяє слідкувати за ефективністю тренування;

– адаптація до конкретного завдання: в процесі fine-tuning DistilBERT навчається краще відповідати на питання, використовуючи специфічний контекст, що є в наборі даних. Вона використовує вже наявні знання про структуру мови та адаптує їх для завдання запитань-відповідей.

Можна побачити, що модель навчалась півтори години, та в результаті було створено .joblib файл з назвою «distilbert_qa_tuned.joblib'», розташовану у папці cdqa/models – це й є натренована модель.

Тепер є можливість запустити наступний код та протестувати роботу навченої моделі.

```
from ast import literal_eval
import pandas as pd from cdqa.pipeline
import QAPipeline
df = pd.read_csv('files/bible_mattheve_converted_dataset', converters={'paragraphs': literal_eval})
cdqa_pipeline = QAPipeline(reader='cdqa/models/distilbert_qa_tuned.joblib') cdqa_pipeline.fit_retriever(df=df)
def ask_question(question):
    prediction = cdqa_pipeline.predict(question)
    answer, title, paragraph, _ = prediction
    return (f"Answer: {answer}\n" f"Title: {title}\n" f"Paragraph: {paragraph}")
if name == 'main':
    while True:
      question = input("Question: ")
       if question.lower() == 'q':
            break
       print(ask_question(question))
```

Висновки до розділу 3

Було детально описано процес створення та підготовки набору даних для навчання нейронної мережі, а також тонке налаштування попередньо навченої моделі DistilBERT для задачі автоматичного надання відповідей на питання з теологічних та апологетичних текстів.

Створення набору даних для навчання моделі BERT включало обробку віршів з Євангелії від Матвія та текстів апологетичного характеру, зокрема інформації з книги Джоша Макдауелла «Не просто тесля». Ці тексти були приведені до необхідного формату, що включає поділ на параграфи та додавання заголовків. В результаті, було отримано два файли з теологічними та апологетичними даними, які будуть використовуватись моделлю CDQA для знаходження відповідей на запитання користувачів.

Тонке налаштування моделі DistilBERT є ключовим етапом для підвищення її ефективності та точності. Завдяки адаптації моделі до специфічного набору даних, було досягнуто покращення контекстуальності відповідей, підвищення релевантності та зменшення кількості помилок. Було описано процес підготовки даних для навчання моделі за допомогою cdQA-annotator, а також кроки з налаштування та навчання моделі DistilBERT [27].

Після завершення процесу тонкого налаштування, було створено натреновану модель, яка може використовуватись у системі CDQA для автоматичного надання відповідей на теологічні та апологетичні питання.

Результати перевірки показали, що модель здатна знаходити правильні відповіді на прямі запитання, що задовольняє потреби нашої системи з урахуванням подальшого поля для зростання.

4 ТЕСТУВАННЯ НАВЧЕНОЇ МОДЕЛІ. ІНТЕГРАЦІЇ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ В ТЕЛЕГРАМ БОТ

4.1 Тестування навченої моделі

Спершу запустимо готовий застосунок для перевірки його працездатності. В результаті отримуємо відповідь про успішний запуск застосунка. На рисунку 4.1 можна бачити результат вивід консолі про запуск.

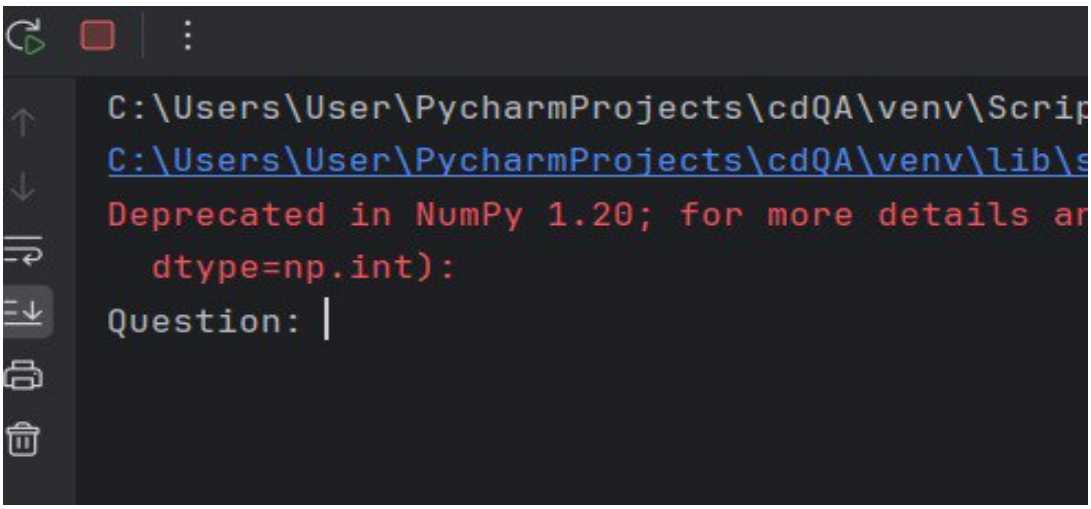


Рисунок 4.1 – Запущений застосунок

Тепер надішлемо системі декілька теологічних питань для перевірки роботи QA системи, як показано на рисунку 4.2.

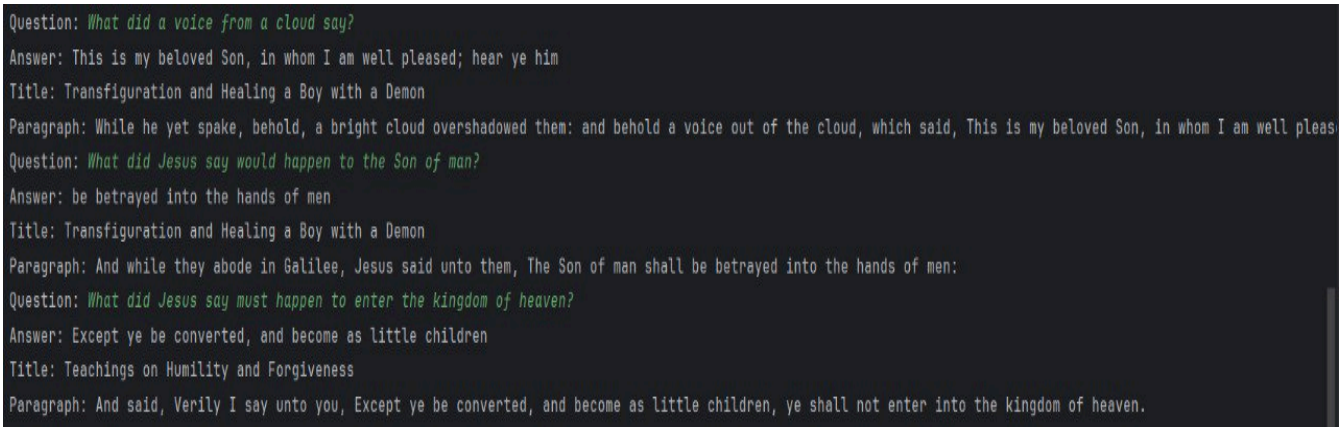


Рисунок 4.2 – Перевірка робот системи питання-відповідь

Як бачимо, система працює й може відповідати на теологічні та апологетичні питання, основючись на наданому моделі набору даних.

Для більш детального аналізу можливостей навченної моделі проведемо ряд ручних тестів, які мають показати здатність моделі відповідати на питання різної складності.

Для вирішення цього питання був складений перелік запитань на основі підготовлених текстів теологічного та апологетичного характеру, розділеного на категорії:

- **прості запитання** – вони характеризуються тим, що за для знаходження відповіді на ці запитання майже не потрібно розуміти контекст частин тексту; відповіді на такі запитання можна надати посилаючись прямо на певні частини тексту й, зазвичай, правильна відповідь на такі запитання одна;
- **запитання середньої складності** – вимагають більшого розуміння контексту, адже ці запитання не складені так прямолінійно до тексту, як перша категорія. Також на цей тип запитань може бути декілька правильних відповідей, з яких саме 1 буде найправільнішою;
- **вакжі запитання** – вимагають високого розуміння контексту речень підготовлених текстів, адже відповідь на них прямо не виходить з тексту. Також на ці запитання може бути багато відповідей, які можуть бути як частково правильні, так зовсім неправильні в контексті.

Цей принцип використовувайся, як для тестування правильності відповідей на запитання теологічного характера, так й апологетичного. Перейдемо до тестування.

Запитання простої складності.

Як вже було сказано, ці питання виходять прямо з тексту, тому на них модель досить легко відповідала. Для прикладу розглянемо декілька прикладів запитань теологічного характеру на рисунках 4.3 та 3.4.

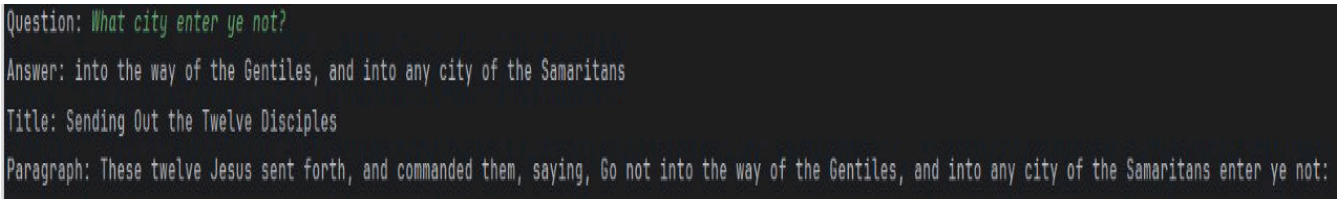


Рисунок 4.3 – Відповідь на питання «В яке місто не входити?»

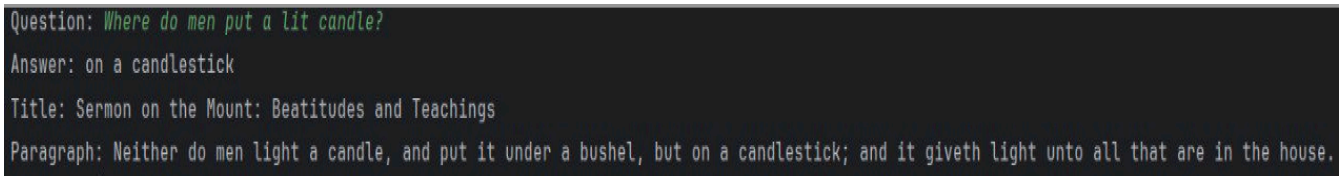


Рисунок 4.4 – Відповідь на питання «Куди чоловік ставить палаючи свічку?»

Тепер перейдемо до запитань апологетичного характеру, як проілюстровано на рисунках 4.5 та 4.6.

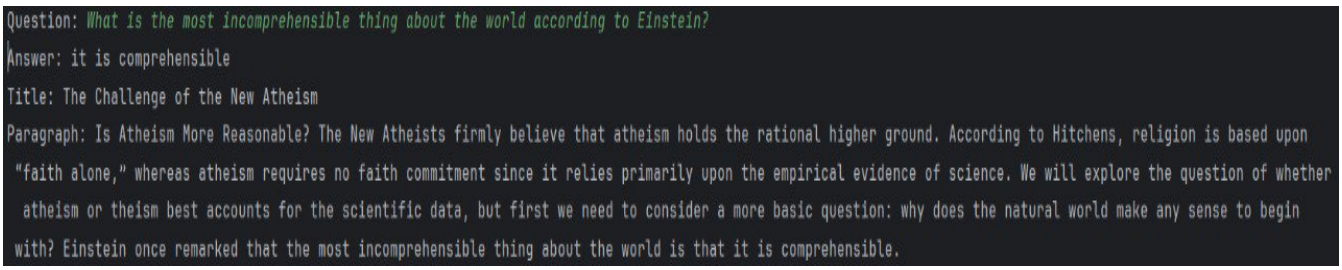


Рисунок 4.5 – Відповідь на питання «Що є найбільш незрозумілим у світі за Ейнштейном?»

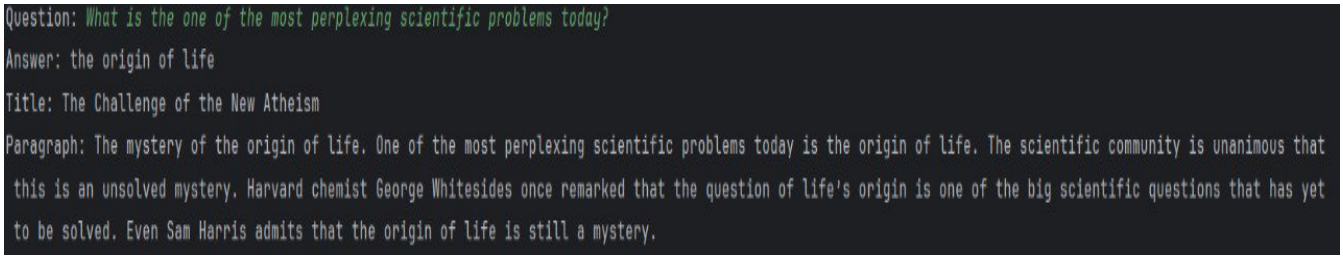


Рисунок 4.6 – Відповідь на питання «Що є однією з найбільш заплутаних наукових проблем сучасності?»

Як можна побачити, на всіх наведених прикладах простих запитань теологічного та апологетичного характерів модель надає правильні та прямі відповіді, посилаючись на відповідні параграфи підготовленого теологічного тексту.

Було проаналізовано 10 відповідей на питання цієї категорії, в результаті було визначено, що точність становить близько 95%, а середній час на відповідь ~2 секунди.

Запитання середньої складності.

Було зазначено, що питання цієї категорії вимагають більшого розуміння контексту за для надання правильної відповіді. Для прикладу розглянемо декілька прикладів на рисунках 4.7, 4.8 та 4.9.

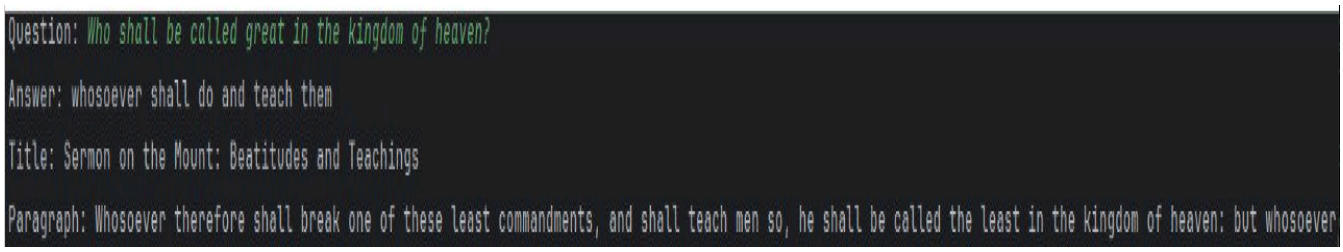


Рисунок 4.7 – Відповідь на питання «Хто буде названий великий в Царстві Небесному?»

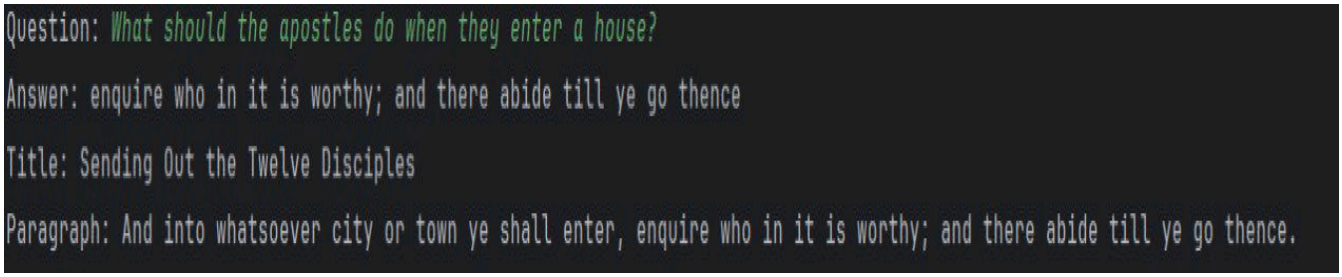


Рисунок 4.8 – Відповідь на питання «Що мають робити апостоли, коли вони входять в чиїсь дім?»

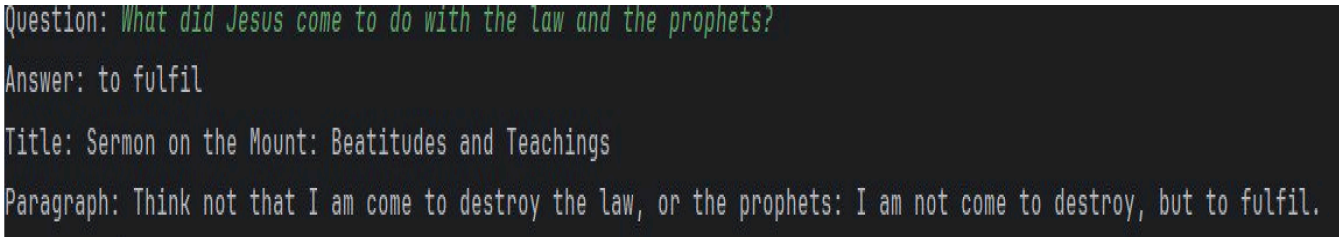


Рисунок 4.9 – Відповідь на питання «Що Ісус прийшов зробити з законом та пророками?»

Як можна побачити в 2 з 3 наведених прикладах модель надає стовідсотково правильні відповіді на запитання середньої складності.

У прикладі, що зображений на рисунку 3.16, модель надає частково правильну відповідь, адже надана відповідь є дотичною до питання, але її не можна назвати повністю правильною, адже в ній модель посилається на слова Ісуса про місто, в яке зайдуть апостоли, а питання було про дома, в яких вони будуть зупинятися.

Тепер перейдемо до запитань апологетичного характеру, як показано на рисунках 4.10, 4.11 та 4.12.

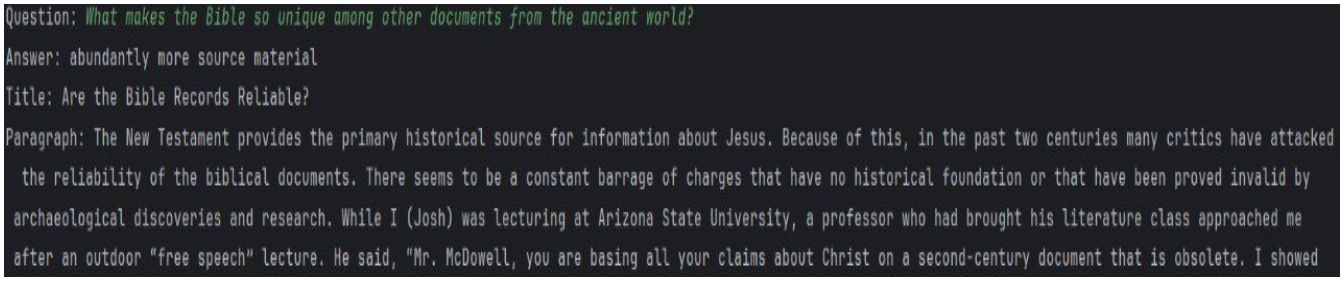


Рисунок 4.10 – Згенерована відповідь на питання «Що робить Біблію такою унікальною серед документів давнього світу?»

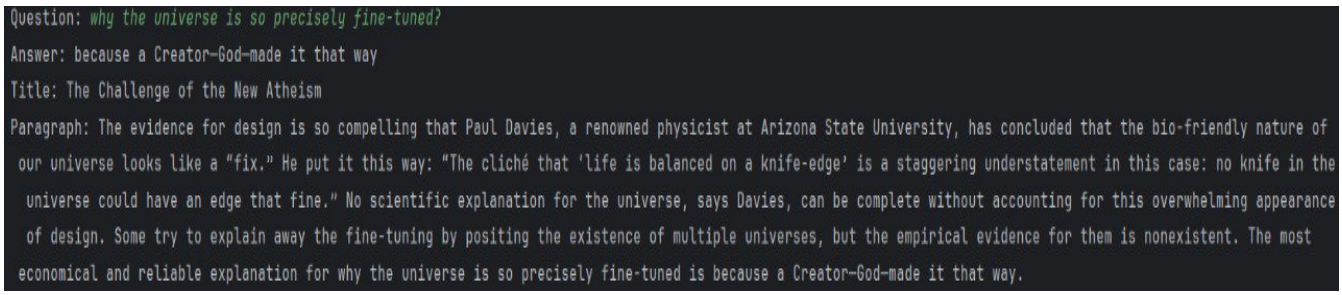


Рисунок 4.11 – Згенерована відповідь на питання «Чому Всесвіт так тонко налаштований?»

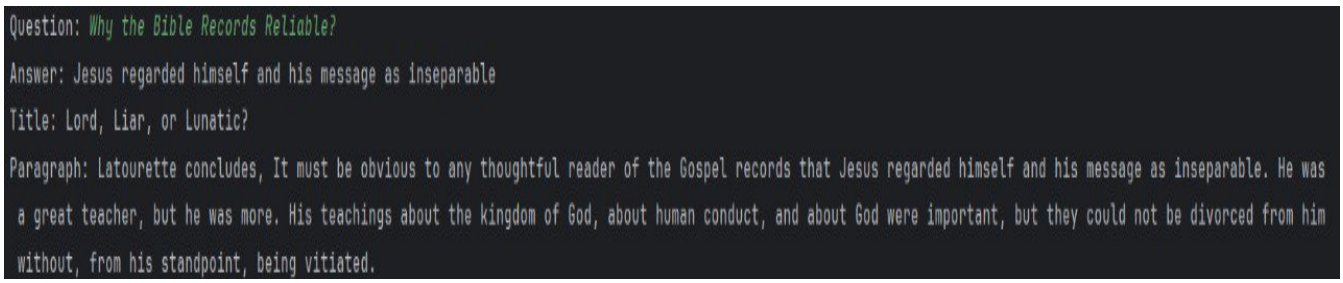


Рисунок 4.12 – Згенерована відповідь на питання «Чому Біблії можна довіряти?»

На прикладах відповідей на апологетичні питання можемо спостерігати приблизну ту саму картину – на два з трьох показаних прикладів модель надала правильну відповідь, але з останнім неконтекстуальним питанням (рис. 4.12) модель не справилась й надала некоректну відповідь. Таку поведінку моделі можна пояснити тим, що вона має складнощі з розуміння контексту текстів й може помилятися при відповіді на нестандартні питання.

Було проаналізовано 10 відповідей на питання цієї категорії, в результаті було визначено, що точність становить близько 80%, а середній час на відповідь ~3 секунди.

Запитання високої складності.

Запитання цієї категорії що питання цієї категорії вимагають високого розуміння контексту речень підготовлених текстів, адже відповідь на них прямо не виходить з тексту. Розглянемо декілька прикладів на рисунках 4.13, 4.14 та 4.15.

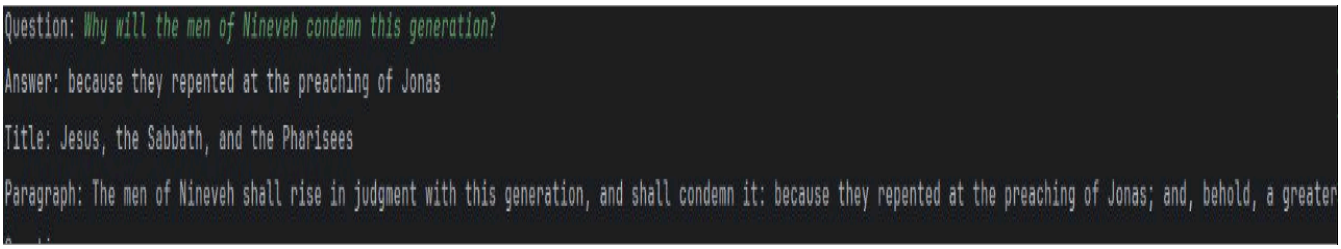


Рисунок 4.13 – Згенерована відповідь на питання «Чому чоловіки Ніневії засудять це покоління?»

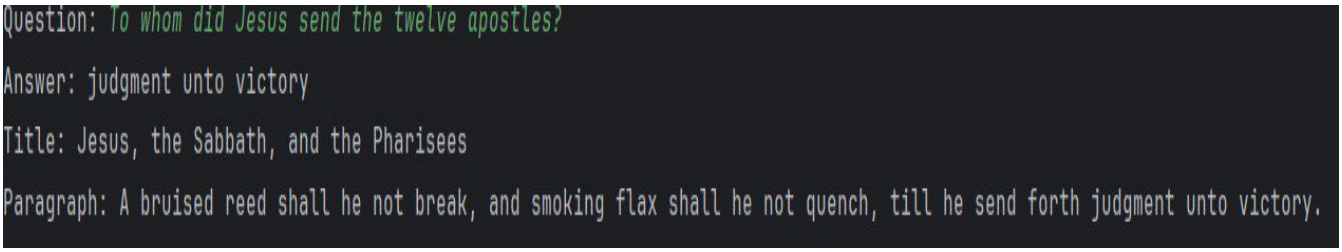


Рисунок 4.14 – Відповідь на питання «До кого Ісус послав 12 апостолів?»

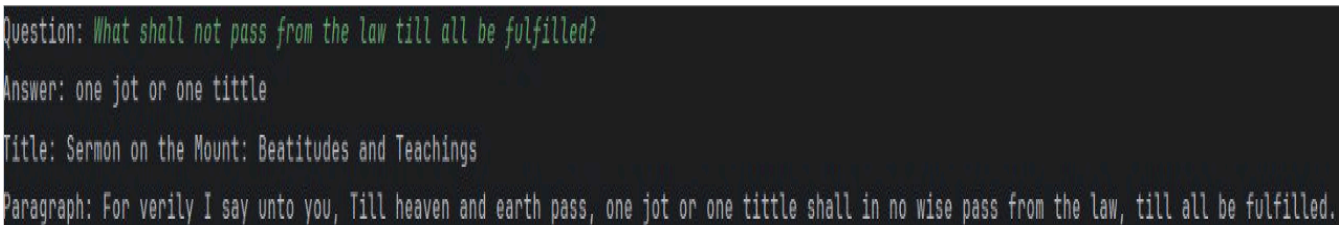


Рисунок 4.15 – Відповідь на питання «Що не мине з закону, поки все не виповниться?»

Як можна побачити в перших двох наведених прикладах модель надає цілком правильні відповіді на запитання високої складності.

У прикладі, що зображений на рисунку 4.14, модель надає повністю неправильну відповідь, яка взагалі не є дотичною до контексту запитання й в ньому модель посилається на помилковий параграф тексту. Це ще раз доводить, що модель не зовсім підходить для відповідей на складні теологічні запитання через її технічні обмеження в контекстуальному аналізі складної для сприйняття літератури. Цю проблему, авжеж, частково можна вирішити аугментацією даних, гіперпараметричним налаштуванням, пост-обробкою відповідей.

Тепер перейдемо до запитань апологетичного характеру цієї категорії, як показано на рисунках 4.16, 4.17.

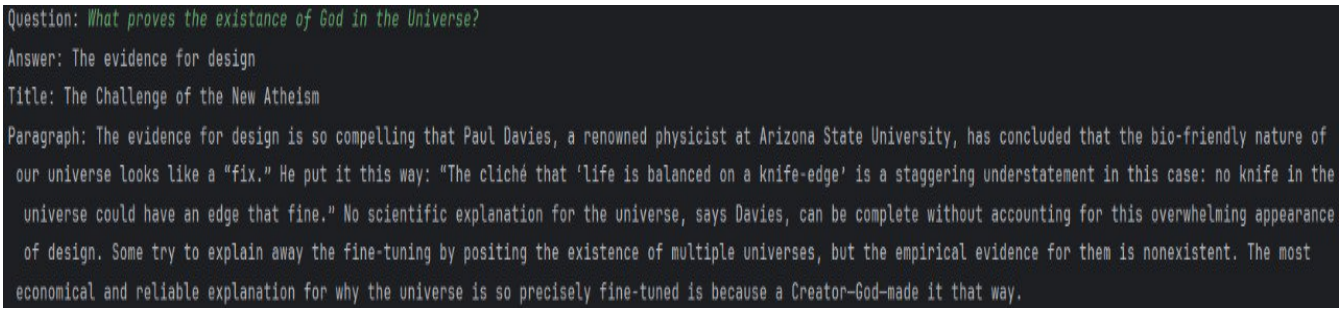


Рисунок 4.16 – Згенерована відповідь на питання «Що у Всесвіті доводить існування Бога?»

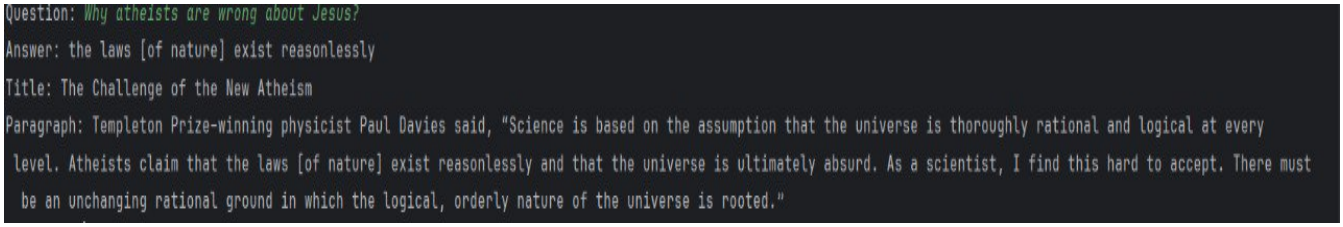


Рисунок 4.17 – Згенерована відповідь на питання «Чому атеїсти помиляються стосовно Ісуса?»

Можемо спостерігати, що модель відповідала стовідсотково правильно на перше апологетичне питання (рис. 4.16), а на друге відповіла не зовсім очікувано, адже в ньому питалося про помилковість думки атеїстів стосовно особистості Ісуса, на що модель відповіла: закони природи існують безпричинно. Ця відповідь може вважатися й правильною, й неправильною, адже, спираючись на текст, це не зовсім

та відповідь, що була очікувана, але, в той самий час, ця відповідь ілюструє більш глибоке розуміння контексту моделлю, адже ця відповіді вказує на помилкове судження атеїстів стосовно безпричинного (створеного само по собі) існування законів природи, всесвіту, тому можна логічна заключити, що в цій відповіді їдеться про особистість Ісуса, яка й в цьому уривку прирівнюється до Бога-Творця, шо цілком можна вважати за правильну відповідь моделі на складне запитання.

За результатом аналізу можна зазначити, що модель справляється краще зі складними запитаннями саме апологетичного характеру, що може пояснюватися тим, що наданий моделі апологетичний набір даних є більш простим для розуміння, адже в ньому менше староанглійських слів та він є більш структурованим за змістом.

Загалом, було проаналізовано 10 відповідей на питання теологічного характеру на питання цієї категорії, в результаті було визначено, що точність становить 50%. Так само було проаналізовано 10 відповідей на питання апологетичного характеру цієї категорії, за результатом маємо таку картину: точність – ~70%. Середній час на відповідь в обох випадках склав близько 5 секунд.

В результаті перевірки, система працює та, в більшості випадків здатна знайти правильну відповідь з тексту на прямі запитання, але має складнощі з непрямими питаннями (тобто, недотичні прямо до тексту набору даних). Цей результат задовольняє потреби нашої системи з урахуванням подальшого поля для вдосконалення системи.

4.2 Розробка телеграм-боту та інтеграція cdQA системи

Для надання користувачеві зручного доступу до створеної інтелектуальної системи питання/відповідь на базі ШІ, створимо телеграм-бот за допомогою асинхронного фреймворку Aiogram. Архітектура телеграм-боту виглядає наступним чином, як показано на рисунку 4.18.

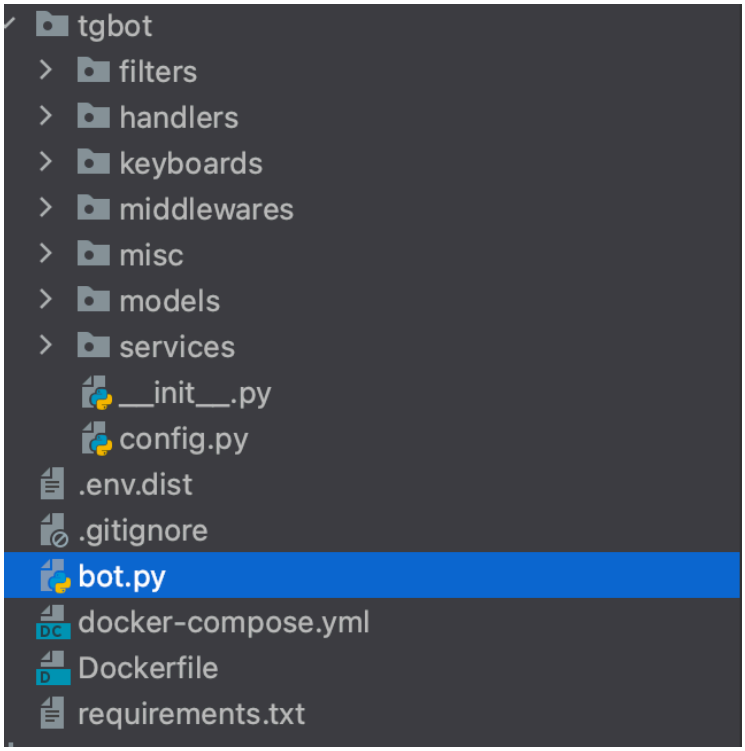


Рисунок 4.18 – Архітектура телеграм-боту

Загальний опис роботи шаблону.

1. Запуск Бота: точка входу для бота – «bot.py»; ініціалізує бот і диспетчер за допомогою Aiogram; завантажує конфігурації з файлу «config.py».

2. Конфігурація («config.py»): завантажує і управляє налаштуваннями бота (наприклад, токени, налаштування webhook); використовує змінні середовища або конфігураційний файл.

3. Dispatcher та Event loop (Диспетчер і Цикл Подій): dispatcher - центральний компонент, який маршрутизує оновлення (повідомлення) до відповідних обробників; цикл подій - бот працює в асинхронному циклі, постійно перевіряючи нові оновлення від серверів Telegram.

4. Папка «/middlewares» (Проміжні Програми): проміжні компоненти дозволяють здійснювати кастомну обробку до того, як оновлення досягнуть обробників (рис. 4.19); можуть змінювати або логувати оновлення, обробляти аутентифікацію тощо.

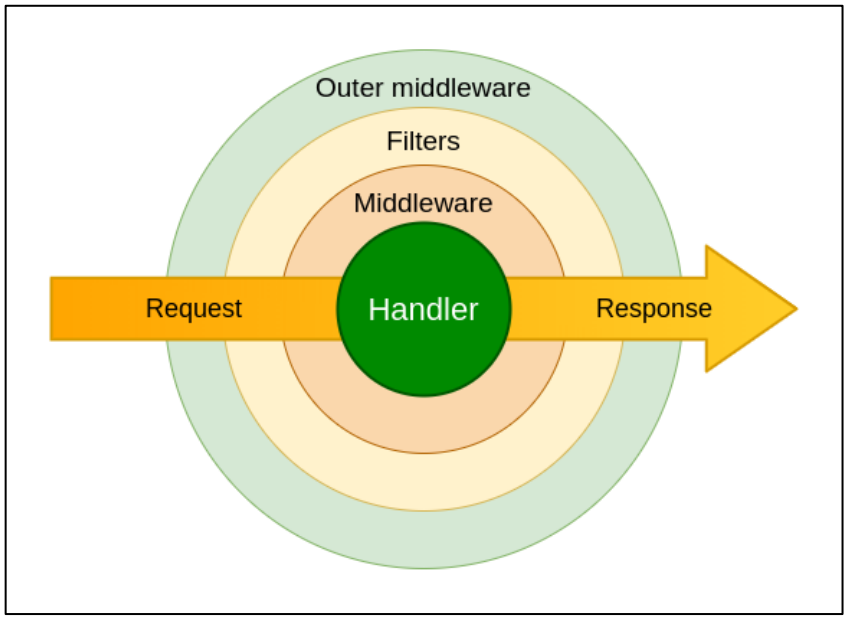


Рисунок 4.19 – Схема роботи middlewares в Aiogram [4]

5. Папка «/filters» (фільтри): кастомні фільтри для попередньої обробки оновлень; допомагає уточнити, які оновлення повинні викликати які обробники (наприклад, обробка повідомлень з певними ключовими словами).

6. Папка «/handlers» (обробники): містить модулі, які визначають функції для відповіді на різні типи оновлен; поділяються на підмодулі (наприклад, обробники повідомлень, обробники запитів).

7. Папка «/services»: містить логіку, яка не належить до обробників чи мідлварів, але потрібна для роботи бота; сервіси зазвичай обслуговують різні аспекти, такі як взаємодія з базою даних, API-запити, допоміжні функції та інші утиліти.

8. Папка «/keyboards»: містить визначення клавіатур (інлайн або звичайних) для Telegram, які використовуються для взаємодії з користувачем; клавіатури допомагають створювати зручний інтерфейс для користувачів бота, надаючи їм можливість вибору різних опцій; ці клавіатури зазвичай імпортуються в обробники для відправки разом із повідомленнями.

Для початку роботи з проектом встановимо необхідні залежності, а саме aiogram~=2.18, environs~=9.0 та aioredis<2.0 [рис 4.20].

```
Using cached aiogram-2.25.2-py3-none-any.whl (203 kB)
Using cached aioredis-1.3.1-py3-none-any.whl (65 kB)
Downloading environs-9.5.0-py2.py3-none-any.whl (12 kB)
Installing collected packages: environs, aioredis, aiogram
Successfully installed aiogram-2.25.2 aioredis-1.3.1 environs-9.5.0
```

Рисунок 4.20 – Встановлення залежностей

Тепер створимо «handlers/user.py» файл та напишемо наступний код для привітання користувача на команду «/start».

```
from aiogram import Dispatcher
from aiogram.types import Message
async def user_start(message: Message):
    await message.reply("Welcome to the Theological and Apologetical QA Bot! Ask me any question related to theology or apologetics, and I'll do my best to provide you with an answer.")


def register_user(dp: Dispatcher):
    dp.register_message_handler(user_start, commands=["start"], state="*")
```

Далі створимо файл «handlers/qa.py», в якому буде реалізована логіка системи питання-відповіді в інтеграції з попередньо підготовленим набором даних та навченою моделлю.

```
import pandas as pd
from ast import literal_eval
from aiogram import Dispatcher
from aiogram.types import Message
from cdqa.pipeline import QAPipeline
# Load the dataset and model
df = pd.read_csv('files/bible_mattheve_converted_dataset', converters={'paragraphs': literal_eval})
cdqa_pipeline = QAPipeline(reader='cdqa/models/distilbert_m_tuned.joblib')
cdqa_pipeline.fit_retriever(df=df)
```

```
# Function to get the answer from the model
def ask_question(question):
    prediction = cdqa_pipeline.predict(question)
    answer, title, paragraph, _ = prediction
    return (f"Answer: {answer}\n"
            f"Title: {title}\n"
            f"Paragraph: {paragraph}")
# Handler for processing questions
async def process_question(message: Message):
    question = message.text
    answer = ask_question(question)
    await message.reply(answer)
def register_qa(dp: Dispatcher):
    dp.register_message_handler(process_question, state="*")
```

Запускаємо файл ініціалізації «bot.py» та отримуємо повідомлення про успішний старт застосунку (рис. 4.21).

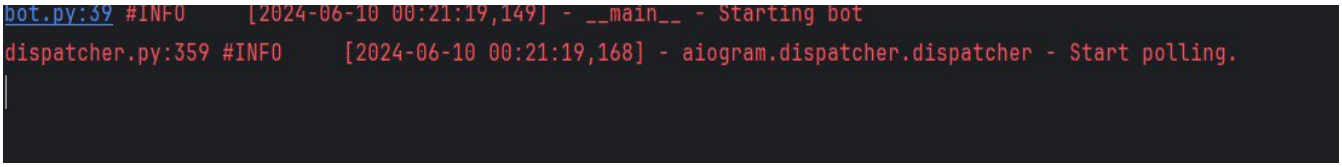


Рисунок 4.21 – Запуск застосунку

Заходимо в телеграм-бот та запускаємо його, надсилаємо перші тестові запитання, як показано на рисунку 4.22.

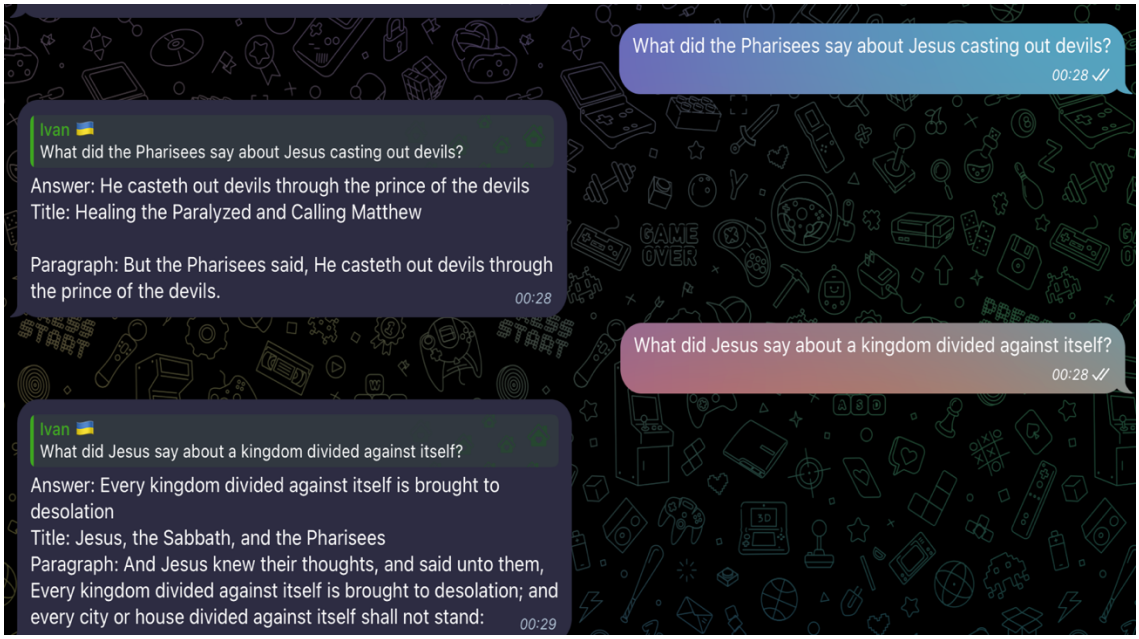


Рисунок 4.22 – Запуск та успішна робота застосунку

У результаті маємо маємо цілком працюючи інтелектуально систему питання-відповідь на базі штучного інтелекту представлену в Телеграм.

Висновки до розділу 4

У четвертому розділі було проведено ряд ручних тестів, які показали здатність моделі відповідати на питання різної складності. Питання були поділені на три категорії: прості, середньої складності та високої складності. На питання простої складності модель відповідала з точністю 95%, на питання середньої складності – 80%, а на питання високої складності – 50% для теологічних питань та 65% для апологетичних. Це свідчить про високу ефективність моделі на простих питаннях та необхідність подальшого вдосконалення для складних питань.

Також було розглянуто процес створення користувацького інтерфейсу (UI) на базі Telegram-боту для інтеграції інтелектуальної системи питання-відповідь (QA) з використанням бібліотеки Aiogram. У результаті, створена інтелектуальна QA система дозволяє користувачам отримувати відповіді на запитання з теологічних та апологетичних тем у зручному форматі месенджера Telegram. Інтеграція з Telegram-ботом забезпечила простий та ефективний інтерфейс для

взаємодії з системою, підвищуючи доступність та зручність використання створеного рішення. Аналіз відповідей показав, що модель ефективно справляється з простими та середньої складності питаннями, проте потребує додаткової оптимізації для більш складних запитів, особливо в сфері теології, що є завданням для подальших досліджень та вдосконалень системи.

ВИСНОВКИ

З кожним роком актуальність надання консультацій з теологічних та апологетичних питань зростає, адже люди все більше шукають відповіді на питання сенсу життя, шукають надію та підтримку.

Актуальність створеного проєкту полягає у тому, що розумна система питання-відповідь на платформі Telegram дасть доступ до теологічної та апологетичної інформації та допоможе користувачам отримувати доцільні відповіді на свої духовні питання на їхньому духовному шляху.

Метою даної кваліфікаційної роботи було створення інтелектуальної системи, що надає відповіді на теологічні та апологетичні питання через платформу Telegram.

Для досягнення поставленої мети було вирішено такі завдання:

– проаналізовано сучасний стан задачі надання консультацій з теологічних та апологетичних питань;

– створено спеціалізований набір даних на основі біблійних текстів та матеріалів з апологетики;

– проведено навчання моделі DistilBERT для підвищення точності та контекстуальності відповідей;

– проведено тестування системи питання-відповідь для оцінки її ефективності та здатності відповідати на питання різної складності;

– інтегровано модель з Telegram-ботом за допомогою фреймворку Aiogram.

Інтеграція з Telegram-ботом забезпечує простий та ефективний інтерфейс для взаємодії з системою, підвищуючи її доступність та зручність для користувача.

Розроблена система успішно справляється з відповіддю на теологічні та апологетичні питання різної складності, а саме:

– прості питання – 95% правильності відповідей;

– середньої складності – 80% правильності відповідей;

– високої складності – 50% та 70% правильності відповідей для теологічних питань та апологетичних питань відповідно.

Середній час відповіді становить

Було проаналізовано і досліджено методи безпеки при роботі з середою розробки програмного забезпечення, включаючи вимоги до приміщень з використанням комп’ютерного обладнання та порядок дій у разі пожежі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. AI and Christianity: Navigating the Intersection of Technology and Faith in Ministry Work. URL: https://www.delmethod.com/blog/ai-and-christianity. (дата звернення 27.05.2024).
2. AI in Religion, AI for Religion, AI and Religion: Towards a Theory of Religious Studies and Artificial Intelligence. URL: https://doi.org/10.3390/rel12060401. (дата звернення 30.05.2024).
3. AI Is Changing Religious Practices: Exploring the Benefits and Challenges. URL: https://www.linkedin.com/pulse/ai-changing-religious-practices-exploring-benefits-challenges-al-. (дата звернення 27.05.2024).
4. Aiogram Middleware Basics. URL: https://docs.aiogram.dev/en/dev-3.x/_images/basics_middleware.png (дата звернення 05.06.2024).
5. Artificial Intelligence. What Is Artificial Intelligence (AI)? How Does AI Work? URL: https://builtin.com/artificial-intelligence. (дата звернення 27.05.2024).
6. Barrat, James. Our Final Invention: Artificial Intelligence and the End of the Human Era. St. Martin's Press, 2013. 322 с.
7. BERT, RoBERTa, DistilBERT, XLNet: Which one to use? URL: https://towardsdatascience.com/bert-roberta-distilbert-xlnet-which-one-to-use-3d5ab82ba5f8 (дата звернення 04.06.2024).
8. cdQA architecture. URL: https://miro.medium.com/v2/resize:fit:1400/format:webp/1*v7s0WvOj-Z-ZwVzWFuzR7Q.png. (дата звернення 05.06.2024).
9. cdQA structure dataframe. URL: https://miro.medium.com/v2/resize:fit:1112/format:webp/1*fyMjrb6c7VhIE5ui2yYBBA.png. (дата звернення 05.06.2024).

10. cdQA Suite Training Models. URL: https://github.com/cdqa-suite/cdQA?tab=readme-ov-file#training-models (дата звернення 03.06.2024).
11. cdQA Suite. URL: https://camo.githubusercontent.com/86eb63a162e77b4496edadcd13b9216ca518bfdfa506574e755632fa21278dfe/68747470733a2f2f636471612d73756974652e6769746875622e696f2f636451412d7765627369746652f696d672f73756974652d332e35633834653532342e706e67 (дата звернення 02.06.2024).
12. Deep Learning vs Machine Learning. URL: https://images.prismic.io/turing/652ebc26fbd9a45bcec81819_Deep_Learning_vs_Machine_Learning_3033723be2.webp?auto=format%2Ccompress&fit=max&w=3840 (дата звернення 27.05.2024).
13. Document indexing and annotation. URL: https://www.researchgate.net/profile/Fabio-Ciravegna/publication/221466956/figure/fig2/AS:667633099747338@1536187543674/Document-indexing-and-annotation-traditional-keyword-indexing-and-document-ranking-top.ppm (дата звернення 01.06.2024).
14. Ethics of Artificial Intelligence and Robotics. URL: https://plato.stanford.edu/entries/ethics-ai/. (дата звернення 01.06.2024).
15. Kenneally, Christine. The First Word: The Search for the Origins of Language. Penguin Books, 2007. 368 с.
16. King James Bible. URL: https://www.kaggle.com/datasets/kk99807/the-king-james-bible (дата звернення 25.05.2024).
17. McDowell, Josh. Not Just a Carpenter. Tyndale House Publishers, 1977. 128 с.
18. Nolen, John. Bible Verses for You: A Comprehensive Guide to Understanding the Bible. Zondervan, 2018. 412 с.
19. Russell, Stuart J., and Peter Norvig. Artificial Intelligence: A Modern Approach. Pearson, 2016. 1132 с.

20. Term Frequency-Inverse Document Frequency (TF-IDF). URL: https://www.researchgate.net/profile/Haider-Al-Khateeb/publication/291950178/figure/fig1/AS:330186932408324@1455734107458/Term-Frequency-Inverse-Document-Frequency-TF-IDF.png (дата звернення 03.06.2024).
21. The AI and data-driven Christian Theology. URL: https://www.researchgate.net/publication/347595406_AI_and_data-driven_Christian_Theology. (дата звернення 05.06.2024).
22. The Artificial Intelligence's Understanding of Religion: Investigating the Moralistic Approaches Presented by Generative Artificial Intelligence Tools (2024). URL: https://www.mdpi.com/2077-1444/15/3/375. (дата звернення 07.06.2024).
23. The ChatGPT's Significance for Theology. URL: https://www.tandfonline.com/doi/full/10.1080/14746700.2023.2188366. (дата звернення 05.06.2024).
24. The Ethics of Artificial Intelligence in Autonomous Religion and Spirituality. URL: https://ts2.space/en/the-ethics-of-artificial-intelligence-in-autonomous-religion-and-spirituality/. (дата звернення 29.05.2024).
25. The Impact of AI-Powered Technology on Religious Practices and Ethics: The Road Ahead. URL: https://www.researchgate.net/publication/378375652_Impact_of_AI-Powered_Technology_on_Religious_Practices_and_Ethics_The_Road_Ahead. (дата звернення 09.06.2024).
26. The Impact of Artificial Intelligence on Religious Belief Systems**.** URL: https://www.tandfonline.com/doi/full/10.1080/09546545.2021.1907341 (дата звернення 05.06.2024).

27. The Fine-tuning Pre-Trained Language Models Effectively by Optimizing Subnetworks Adaptively. URL: https://ar5iv.org/abs/2211.01642. (дата звернення 26.05.2024).

28. What Is Artificial Intelligence (AI)? URL: https://cloud.google.com/learn/what-is-artificial-intelligence. (дата звернення 28.05.2024).

ДОДАТОК А

Програмний код для навчання нейронної мережі

Файл конфігурації .env:

```
BOT_NAME=mybotname
BOT_TOKEN=123456:Your-TokEn_ExaMple
ADMINS=123456,654321
USE_REDIS=False
```

Скрипт preparing_data.py:

```
import pandas as pd
# Titles dictionary as provided
titles = {…}
def split_paragraphs(text):
    return [paragraph.strip() for paragraph in text.strip().split('\n') if paragraph.strip()]
# Function to create the new dataframe with combined paragraphs and titles
def create_combined_dataframe(df):
    combined_data = []
    # Group by book and chapter
    grouped = df.groupby(['book', 'chapter'])
    for (book, chapter), group in grouped:
        paragraphs = group['text'].apply(split_paragraphs).sum()
        # Construct the title based on book and chapter
        title = titles[book][chapter]
        combined_data.append({
            'title': title,
            'paragraphs': paragraphs
```

```
        })
    # Create a new dataframe with the combined data
    combined_df = pd.DataFrame(combined_data)
    return combined_df
# Read the CSV file with semicolon delimiter
df = pd.read_csv('files/gospels.csv', delimiter=';')
# Create the combined dataframe
combined_df = create_combined_dataframe(df)
# Save the new dataframe to a CSV file
combined_df.to_csv('files/combined_gospels.csv', index=False)
```

Скрипт preparing_data_squad_like.py:

```
import pandas as pd
import json # Load the original CSV file
df = pd.read_csv('files/gospels.csv', delimiter=';') # Dictionary to hold the converted data
data = {"data": []}
titles = {…}
# Process each Gospel
for gospel, title_list in titles.items():
    for i, title in enumerate(title_list, 1):
        chapter_paragraphs = df[(df['book'] == gospel) & (df['chapter'] == i)]
        paragraphs = [{"context": row["text"].strip(), "qas": []} for _, row in chapter_paragraphs.iterrows()]
        if paragraphs:
            data["data"].append({"title": f"{gospel}. {title}", "paragraphs": paragraphs})
# Save the JSON file
with open('gospels_squad.json', 'w') as f:
```

```
    json.dump(data, f, indent=4)
```

Скрипт train.py:

```
from cdqa.pipeline import QAPipeline
cdqa_pipeline = QAPipeline(reader='cdqa/models/distilbert_qa-trained.joblib')
cdqa_pipeline.fit_reader('files/squad-like_dataset.json')
cdqa_pipeline.dump_reader('cdqa/models/distilbert_qa_tuned.joblib')
```

Скрипт test.py:

```
from ast import literal_eval
import pandas as pd
from cdqa.pipeline import QAPipeline
df = pd.read_csv('files/bible_mattheve_converted_dataset', converters={'paragraphs': literal_eval})
cdqa_pipeline = QAPipeline(reader='cdqa/models/distilbert_m_tuned.joblib')
cdqa_pipeline.fit_retriever(df=df)
def ask_question(question):
    prediction = cdqa_pipeline.predict(question)
    answer, title, paragraph, _ = prediction
    return (f"Answer: {answer}\n"
            f"Title: {title}\n"
            f"Paragraph: {paragraph}")
if name == 'main':
    while True:
        question = input("Question: ")
        if question.lower() == 'q':
            break
        print(ask_question(question))
```

Скрипт evaluate.py:

```
from ast import literal_eval

import pandas as pd

from cdqa.pipeline import QAPipeline

from cdqa.utils.evaluation import evaluate_pipeline

df = pd.read_csv('files/qa_converted_dataset', converters={'paragraphs': literal_eval})

cdqa_pipeline = QAPipeline(reader='cdqa/models/distilbert_qa-trained.joblib')

cdqa_pipeline.fit_retriever(df=df)

evaluate_pipeline(cdqa_pipeline, 'files/squad-like_dataset.json')
```

ДОДАТОК Б

Програмний код Телеграм-боту

Скрипт-ініціалізації bot.py:

```
import asyncio
import logging
from aiogram import Bot, Dispatcher
from aiogram.contrib.fsm_storage.memory import MemoryStorage
from aiogram.contrib.fsm_storage.redis import RedisStorage2
from tgbot.config import load_config
from tgbot.filters.admin import AdminFilter
from tgbot.handlers.admin import register_admin
from tgbot.handlers.echo import register_echo
from tgbot.handlers.user import register_user
from tgbot.middlewares.environment import EnvironmentMiddleware
logger = logging.getLogger(__name__)
def register_all_middlewares(dp, config):
    dp.setup_middleware(EnvironmentMiddleware(config=config))
def register_all_filters(dp):
    dp.filters_factory.bind(AdminFilter)
def register_all_handlers(dp):
    register_admin(dp)
    register_user(dp)
    register_qa(dp)
    register_echo(dp)
async def main():
```

```
    logging.basicConfig(
        level=logging.INFO,
        format=u'%(filename)s:%(lineno)d #%(levelname)-8s [%(asctime)s] - %(name)s - %(message)s',
    )
    logger.info("Starting bot")
    config = load_config(".env")
    storage = RedisStorage2() if config.tg_bot.use_redis else MemoryStorage()
    bot = Bot(token=config.tg_bot.token, parse_mode='HTML')
    dp = Dispatcher(bot, storage=storage)
    bot['config'] = config
    register_all_middlewares(dp, config)
    register_all_filters(dp)
    register_all_handlers(dp)
    # start
    try:
        await dp.start_polling()
    finally:
        await dp.storage.close()
        await dp.storage.wait_closed()
        await bot.session.close()
if __name__ == '__main__':
    try:
        asyncio.run(main())
    except (KeyboardInterrupt, SystemExit):
        logger.error("Bot stopped!")
```

Файл конфігурації config.py:

```
from dataclasses import dataclass
from environs import Env
@dataclass
class TgBot:
    token: str
    admin_ids: list[int]
    use_redis: bool
@dataclass
class Config:
    tg_bot: TgBot
def load_config(path: str = None):
    env = Env()
    env.read_env(path)
    return Config(
        tg_bot=TgBot(
            token=env.str("BOT_TOKEN"),
            admin_ids=list(map(int, env.list("ADMINS"))),
            use_redis=env.bool("USE_REDIS"),
        )
    )
```

Скрипт-обробник user.py:

```
from aiogram import Dispatcher
from aiogram.types import Message
async def user_start(message: Message):
```

await message.reply("Welcome to the Theological and Apologetical QA Bot! Ask me any question related to theology or apologetics, and I'll do my best to provide you with an answer.")

```
def register_user(dp: Dispatcher):
    dp.register_message_handler(user_start, commands=["start"], state="*")
```

Скрипт-обробник qa.py:

```
import pandas as pd
from ast import literal_eval
from aiogram import Dispatcher
from aiogram.types import Message
from cdqa.pipeline import QAPipeline
# Load the dataset and model
df = pd.read_csv('files/bible_qa_converted_dataset', converters={'paragraphs': literal_eval})
cdqa_pipeline = QAPipeline(reader='cdqa/models/distilbert_qa_tuned.joblib')
cdqa_pipeline.fit_retriever(df=df)


# Function to get the answer from the model
def ask_question(question):
    prediction = cdqa_pipeline.predict(question)
    answer, title, paragraph, _ = prediction
    return (f"Answer: {answer}\n"
            f"Title: {title}\n"
            f"Paragraph: {paragraph}")
# Handler for processing questions
async def process_question(message: Message):
    question = message.text
    answer = ask_question(question)
```

```
    await message.reply(answer)

def register_qa(dp: Dispatcher):

    dp.register_message_handler(process_question, state="*")
```