РЕФЕРАТ

Кваліфікаційна робота на здобуття освітнього ступеню «бакалавр» за спеціальністю 121 – Інженерія програмного забезпечення. Тернопільський національний технічний університет ім. Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СПз-41, 2024 рік. Пояснювальна записка до кваліфікаційної роботи на здобуття освітнього ступеню «бакалавр» містить: 52 с., 12 рис., 1 додаток.

Об'єкт дослідження – процес взаємодії абітурієнтів із чат-ботом у реальних умовах.

Мета роботи – Розробка та впровадження інтелектуального чат-бота, який забезпечить ефективну та зручну взаємодію між абітурієнтами та закладами вищої освіти.

Метод дослідження – опис предметної області шляхом опрацювання та аналізу теоретичних відомостей та наявних лгоритмів; проєктування бази даних із використанням методів машинного навчання; розробка і налагодження програми.

Отримані результати – розроблено систему для ведення діалогу між абітурієнтом та закладом вищої освіти

Ключові слова: машинне навчання, модель, векторний пошук, трансформер, ембеддінг, нормалізація, абітурієнт, код.

ANNOTATION

Qualification work for the degree of bachelor in the specialty 121 - Software Engineering. Ternopil Ivan Puluj National Technical University, Faculty of Computer and Information Systems and Software Engineering, Department of Software Engineering, Group СПз-43, 2023. The explanatory note for the qualification work for the degree of bachelor contains: 52 pages, 12 figures, 1 appendix.

The object of the research is the interaction process between applicants and the chatbot in real conditions.

The purpose of the work is to develop and implement an intelligent chatbot that ensures effective and convenient interaction between applicants and higher education institutions.

The research method is a description of the subject area through the study and analysis of theoretical information and existing algorithms; database design using machine learning methods; development and debugging of the program.

The results obtained are a system for conducting a dialogue between the applicant and the higher education institution has been developed.

Keywords: machine learning, model, vector search, transformer, embedding, normalization, applicant, code.

Перелік умовних скорочень

- **ЗВО** — заклад вищої освіти
- **Чат-бот** — програма для ведення автоматизованого діалогу
- **GPT** — генеративний передтренований трансформатор (Generative Pre-trained Transformer)
- **AI** — штучний інтелект (Artificial Intelligence)
- **ML** — машинне навчання (Machine Learning)
- **NLP** — обробка природної мови (Natural Language Processing)
- **NN** — нейронна мережа (Neural Network)

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: **Розробка чат-бота діалогу абітурієнта та ЗВО**

Виконав: студент IV курсу, групи СПз-41 спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

_____ Драбик О. І.
(підпис) (прізвище та ініціали)

Керівник _____ Пастух О. А.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Петрик М.Р.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2024

Зміст

Вступ

У сучасному світі роль штучного інтелекту (ШІ) зростає з кожним днем, відкриваючи нові можливості для автоматизації та вдосконалення процесів у різних сферах життя. Однією зі сфер, яка вже відчуває на собі вплив інноваційних технологій, є система вищої освіти. В умовах посилення конкуренції між вищими навчальними закладами (ВНЗ) та збільшення кількості абітурієнтів важливо забезпечити швидкий, зручний та інформативний спосіб взаємодії між абітурієнтом та університетом.

У цьому контексті існує нагальна потреба у розробці та впровадженні інтелектуального чат-бота для взаємодії між кандидатами та закладами вищої освіти. Такий чат-бот може стати ефективним інструментом для автоматизації процесу надання інформації, відповідей на запитання абітурієнтів і навіть попереднього відбору. При цьому він може працювати цілодобово, забезпечуючи доступність інформації для користувачів у будь-який час доби.

Дана дипломна робота присвячена розробці та впровадженню чат-бота для діалогу між абітурієнтом та вищим навчальним закладом. За допомогою методів data science ми прагнемо створити ефективний інструмент, який полегшить взаємодію між абітурієнтами та університетами, спростить процес вступу та зробить його більш доступним і зручним.

Завдання дослідження:

1. Збір і аналіз потреб: визначення потреб кандидатів і функціональних вимог чат-бота.
2. Розробка та навчання моделей машинного навчання: створення моделей для розпізнавання та обробки запитів кандидатів.
3. Інтеграція та тестування: Інтеграція розроблених моделей в чат-бот і тестування його працездатності та ефективності.
4. Оптимізація та вдосконалення: Аналіз результатів тестування та вдосконалення чат-бота для покращення взаємодії з користувачем.

Предмет дослідження: розробка та впровадження інтелектуального чат-бота для взаємодії абітурієнтів та закладів вищої освіти.

Об'єкт дослідження: процес взаємодії абітурієнтів із чат-ботом у реальних умовах.

Стек використаних технологій:

Python, Azure Search, Google Cloud, LLM, і т д.

Ця робота сприятиме застосуванню сучасних методів науки про дані у сфері вищої освіти, полегшить процес вступу абітурієнтів та забезпечить ефективну взаємодію між ними та університетами. Крім того, розроблений чат-бот може стати важливим інструментом підвищення привабливості та конкурентоспроможності закладів вищої освіти.

Основні поняття

Машинне навчання: це галузь штучного інтелекту, яка вивчає алгоритми та моделі, які дозволяють комп'ютерам «навчатися» на даних, розпізнавати шаблони та робити прогнози без явного програмування.

Нейронні мережі: це клас моделей машинного навчання, які намагаються імітувати функціонування людського мозку. Вони складаються з нейронів і шарів нейронів, які разом вирішують проблеми.

Аналіз тексту: це процес аналізу текстових даних для визначення шаблонів, тем, настроїв тощо. Включає такі завдання, як класифікація тексту, виявлення об'єктів, аналіз тону тощо.

Обробка природної мови (NLP): це галузь інформатики, яка вивчає, як комп'ютери можуть розуміти та створювати людську мову. Включає такі завдання, як розпізнавання мовлення, створення тексту, машинний переклад тощо.

Згорточні нейронні мережі (CNN): це тип нейронної мережі, яка спеціалізується на обробці зображень. Вони використовуються для таких завдань, як розпізнавання об'єктів, класифікація зображень, розпізнавання облич тощо.

Послідовності: у обробці природної мови це відповіді або запити, що обробляються як послідовності токенів, таких як слова чи символи.

LLM (Large Laguage Model): це тип моделі штучного інтелекту, який має великий обсяг знань про мову та може вирішувати різноманітні завдання, такі як створення тексту, відповіді на запитання та переклад. Наприклад, GPT-3 від OpenAI є одним із найвідоміших LLM.

Azure: це хмарна платформа від Microsoft, яка надає широкий спектр послуг, таких як обчислення, зберігання даних, штучний інтелект, аналітика та інші.

Azure Search: це служба Azure, яка надає можливість створювати потужні рішення для індексованого пошуку для вашої програми або веб-сайту. Це дозволяє швидко й ефективно шукати, оновлювати та аналізувати дані.

Ембедінги (Embeddings): це представлення об’єктів у векторному просторі, де подібні об’єкти мають близькі вектори. У контексті обробки природної мови вбудовування можна використовувати для представлення слів, речень або документів у векторній формі.

Векторна база даних: це база даних, яка зберігає дані у векторному форматі та підтримує ефективний векторний пошук. Це особливо корисно для вирішення завдань, пов'язаних з пошуком і аналізом схожих об'єктів.

Векторний пошук: це тип пошуку, який використовує векторне представлення об’єктів для пошуку схожих або відповідних об’єктів. Векторний пошук часто використовується в областях, де важливо знайти схожість між об’єктами, наприклад, обробка природної мови, пошук зображень тощо.

Трансформери – це тип нейронної мережі, яка використовує механізми уваги для ефективного моделювання взаємодії між елементами в послідовності, наприклад реченнях або документах. Вони широко використовуються в обробці природної мови та добре виконують такі завдання, як машинний переклад, генерація тексту та аналіз настроїв.

Метрики продуктивності моделі: це числові показники, які використовуються для оцінки якості моделей, наприклад точність, відповідь, точність тощо.

1. Теоретичні основи та технології

1.1 Основи машинного навчання та його роль в NLP

Машинне навчання (ML) — це підгалузь штучного інтелекту (AI), яка займається розробкою алгоритмів і моделей, які дозволяють комп'ютерам навчатися та робити висновки з даних. Основною метою машинного навчання є виявлення шаблонів у даних і використання цих шаблонів для прогнозування чи прийняття рішень.

Основні види машинного навчання

- Кероване навчання (Supervised Learning):

Алгоритми навчання на основі даних з мітками, де кожен вхідний приклад має відповідну мітку (вихід). Завдання полягає в тому, щоб навчитися передбачати мітки нових даних.[16]

Приклади: класифікація тексту, регресія.

- Навчання без контролю (Unsupervised Learning):

Алгоритми навчання на основі даних без міток, де немає відповідних міток. Завдання полягає в тому, щоб визначити приховані структури або закономірності в даних.

Приклади: кластеризація, зменшення розмірності.

- Навчання з підкріпленням (Reinforcement Learning):

Алгоритми, які навчаються, взаємодіючи з оточенням і отримуючи винагороди або покарання за виконані дії. Мета — навчитися оптимізувати послідовність дій, щоб максимізувати винагороду. [2]

Приклади: керування роботами, алгоритми гри.

Роль машинного навчання в обробці природної мови (NLP)

Машинне навчання відіграє ключову роль у розвитку NLP, дозволяючи створювати моделі, здатні розуміти, інтерпретувати та генерувати природну мову. Ось основні завдання NLP, де використовуються методи машинного навчання:

- Класифікація тексту:

Завдання: класифікація документів або текстів за певними категоріями (наприклад, аналіз настроїв, визначення теми).[20]

Алгоритми: логістична регресія, дерева рішень, нейронні мережі.

- Виділення сутностей (NER):

Завдання: визначити та класифікувати названі сутності в тексті (наприклад, імена, дати, місця).

Алгоритми: CRF (Conditional Random Fields), BERT.

- Автоматичний переклад:

Завдання: автоматичний переклад тексту з однієї мови на іншу.

Алгоритми: послідовні моделі RNN (Recurrent Neural Networks), трансформатори (Transformer).

- Аналіз настрою:

Завдання: визначити емоційну тональність тексту (позитивна, негативна, нейтральна).

Алгоритми: SVM (Support Vector Machines), LSTM (Long Short-Term Memory), BERT.

- Генерація тексту:

Завдання: Автоматична генерація тексту на основі заданих параметрів або контексту.

Алгоритми: RNN, трансформери, GPT-3.[13]

Приклади моделей і алгоритмів машинного навчання NLP

1. Word2Vec:

Метод вбудовування слів, який перетворює слова на вектори, зберігаючи їх семантичну подібність.

Алгоритми: CBOW (Continuous Bag of Words), Skip-gram.

2. BERT (Подання двонаправленого кодера від трансформаторів):

Двонаправлена модель для різних завдань NLP, яка враховує контекст лівих і правих слів.

Застосування: виділення сутностей, класифікація текстів, відповіді на питання. [3]

3. GPT (Генеративний трансформатор із попереднім приводом):

Потужний шаблон генерації тексту, здатний виконувати широкий спектр завдань НЛП, включаючи написання тексту, відповіді на запитання, переклад тощо.

Застосування: чат-боти, генерація контенту, автоматизація завдань.

Машинне навчання та його роль у NLP є основою для розробки сучасних систем обробки природної мови, таких як чат-боти, системи рекомендацій, машинні перекладачі та інші інтелектуальні програми. Завдяки використанню передових моделей і алгоритмів можна створювати більш точні та ефективні рішення, які відповідають потребам користувачів.

1.2 Історія та еволюція моделей обробки природної мови

Обробка природної мови (NLP) має довгу історію, яка почалася задовго до появи сучасних комп'ютерів і алгоритмів машинного навчання. Протягом десятиліть техніки НЛП еволюціонували від простих правил і моделей до складних моделей з використанням глибокого навчання та нейронних мереж.[8]

- Початкові підходи: правила та шаблони

У перші роки досліджень у галузі обробки природної мови основна увага була зосереджена на розробці систем на основі правил. Ці системи використовували створені вручну правила для аналізу та створення тексту. Такі підходи мали обмежену гнучкість і могли працювати лише з чітко визначеними сценаріями. Однією з перших систем такого типу була ELIZA, розроблена в 1960-х роках Джозефом Вайзенбаумом. ELIZA моделювала діалог із психотерапевтом, використовуючи моделі для формулювання відповідей.[17]

- Статистичні методи

У 1980-х і 1990-х роках відбувся зсув у бік статистичних методів обробки природної мови. Ці методи використовували ймовірнісні моделі для аналізу тексту, що дозволяло враховувати невизначеність і варіативність мови. Одним із головних досягнень цього періоду було впровадження моделей n-грам, які використовували ймовірності послідовності слів для передбачення наступного слова в тексті. Наприклад, у моделі біграми ймовірність слова визначається на основі попереднього слова.[6]

- Машинне навчання

З розвитком машинного навчання в 2000-х роках з'явилися нові методи, які дозволили значно поліпшити результати НЛП. Зокрема, методи, засновані на деревах рішень, опорних векторах (SVM) і наївному Байєсі, стали популярними для класифікації тексту, аналізу настроїв та інших завдань. Ці методи використовувалися для автоматичного навчання моделей на основі великих обсягів

текстових даних, що призвело до кращої продуктивності порівняно з ручним створенням правил.

- Рекурентні нейронні мережі (RNN)

Одним із ключових досягнень NLP було впровадження рекурентних нейронних мереж (RNN). RNN були розроблені для роботи з послідовностями даних і могли враховувати залежності між елементами послідовності. Вони використовували зворотний зв'язок для передачі інформації з часом, що робить їх ідеальними для завдань обробки текстів, таких як машинний переклад і розпізнавання мови.[4]

Однак RNN мали свої обмеження, включаючи проблему зникнення градієнтів, що ускладнювало вивчення довготривалих залежностей у послідовностях. Щоб вирішити цю проблему, були розроблені вдосконалені версії RNN, такі як LSTM (довгокороткочасна пам'ять) і GRU (Gated Recurrent Unit). Ці моделі використовували спеціальні механізми для збереження довготривалих залежностей і значно покращували результати в багатьох завданнях NLP.

- Трансформери

У 2017 році в статті Google «Увага — це все, що вам потрібно» було показано трансформер, який став революційним кроком у розвитку НЛП. Трансформери відмовилися від повторюваної архітектури на користь механізмів самоуважності, які дозволяли паралельну обробку всіх елементів послідовності. Це значно прискорило навчання та покращило продуктивність моделі.[18]

Трансформери швидко стали основою багатьох просунутих моделей НЛП, таких як BERT від Google (Bidirectional Encoder Representations from Transformers) і GPT (Generative Pre-trained Transformer) від OpenAI. Ці моделі досягли вражаючих результатів у багатьох завданнях обробки мови, включаючи машинний переклад, генерацію тексту, аналіз настроїв тощо.[5]

- Поточна ситуація та перспективи

Сучасні LLM продовжують удосконалюватися, використовуючи ще більші обсяги даних і складнішу архітектуру. Вони вже широко використовуються в

різних галузях, від автоматизованої підтримки клієнтів до створення та аналізу контенту.

Однак потужність LLM пов’язана з проблемами, пов’язаними з етикою, конфіденційністю та упередженістю моделі. У майбутньому дослідникам та інженерам потрібно буде знайти шляхи вирішення цих проблем, одночасно забезпечуючи безпечне та етичне використання технологій НЛП.

Таким чином, еволюція моделей обробки природної мови демонструє значний прогрес від простих правил і моделей до складних нейронних мереж і трансформаторів, що дає змогу досягти нових висот у розумінні та створенні природної мови.

1.3 Основи LLM

Великі мовні моделі (LLM) — це клас моделей штучного інтелекту, які використовують глибоке навчання та нейронні мережі для розуміння та створення природної мови. Основні архітектурні компоненти LLM включають блоки повторюваного кодування та декодування, які містять шари самоконтролю, повністю пов’язані шари та нормалізацію. Ця архітектура дозволяє моделі ефективно імітувати взаємодію між словами чи символами в тексті та створювати природне мовлення.[1]

Навчальний процес LLM передбачає попереднє навчання великих обсягів текстових даних, таких як корпуси Інтернету або тексти книг, з використанням таких методів, як маскування та передбачення наступного слова. Після цього модель можна налаштувати під конкретне завдання, підвищивши її продуктивність і точність.

Однією з головних характеристик LLM є їх здатність автоматично вивчати складні мовленнєві моделі та використовувати ці знання для розуміння нових текстів або створення нового вмісту. Використовуючи механізми уваги, моделі можуть навчитися враховувати контекст і структуру тексту під час вирішення завдань обробки мовлення.

Одними з найбільш вражаючих досягнень у галузі LLM є створення таких моделей, як серія GPT (Generative Pre-trained Transformer) від OpenAI або BERT (Bidirectional Encoder Representations from Transformers) від Google. Ці моделі навчаються на великих обсягах текстових даних і здатні генерувати природне мовлення, вирішувати завдання класифікації та аналізу тексту, а також виконувати багато інших завдань, пов’язаних з обробкою природної мови.[15]

Проте використання LLM тягне за собою певні виклики та проблеми, такі як можливість відтворення стереотипних або неправильних результатів, а також проблеми етики та конфіденційності, пов’язані з використанням таких моделей. У майбутньому LLM можуть стати невід’ємною частиною багатьох технологічних рішень і додатків, надаючи нові можливості для розуміння мови та взаємодії з нею.

Однак їх успішне використання вимагатиме ретельного розгляду етичних, соціальних і правових аспектів їх застосування.

1.4 Трансформери

Transformers — це одна з найбільш інноваційних архітектур нейронних мереж, яка змінила підхід до обробки послідовностей, особливо в обробці природної мови. Трансформери, розроблені Google у 2017 році, були описані в статті «Увага — це все, що вам потрібно» і швидко стали ключовим інструментом у багатьох голосових завданнях.

Основна ідея трансформерів полягає у використанні механізму уваги для ефективної обробки послідовності. Вони складаються з двох основних компонентів: блоків кодування та декодування. Блок кодування відповідає за перетворення вхідної послідовності, тоді як блок декодування використовується для генерації вихідної послідовності (наприклад, у випадку машинного перекладу).[19]

Ключовими компонентами трансформерів є механізм уваги та механізм самоуважності. Механізм уваги дозволяє моделі звертати увагу на різні частини вхідної послідовності під час генерації вихідної послідовності. Механізм самоуважності дозволяє моделі взаємодіяти між різними елементами вхідної послідовності, визначаючи важливість кожного елемента відносно всіх інших елементів.

Однією з переваг трансформерів є їх здатність ефективно обробляти довгі послідовності. Вони можуть обробляти послідовності будь-якої довжини, оскільки вони не мають фіксованого розміру вхідних або вихідних послідовностей. Незважаючи на свої переваги, трансформери також мають свої обмеження. Навчання та їх використання вимагає значних обчислювальних ресурсів, що може бути проблемою для великих моделей або великих наборів даних. Крім того, їх ефективність може залежати від якості та кількості навчальних даних.

Трансформери використовуються в різних задачах обробки природної мови, таких як машинний переклад, генерація тексту, тональний аналіз тощо. Вони стали одними з найпопулярніших інструментів у сучасних системах обробки мови та

продовжують знаходити нові застосування в різних наукових і технологічних галузях.

Принцип дії трансформерів заснований на використанні механізму уваги для ефективного моделювання залежностей між різними елементами вхідної та вихідної послідовностей. Основна ідея полягає в тому, щоб зробити кожен елемент вихідної послідовності залежним від кожного елемента вхідної послідовності, а не лише від його безпосередніх сусідів.

Механізм уваги: вхідна послідовність подається на вхід трансформера у формі вбудованих векторів (див рисунок 1.4.1). Кожен вектор представляє слово або символ у послідовності. Для кожного елемента вхідної послідовності обчислюється ваговий коефіцієнт (оцінка уваги) для всіх інших елементів у послідовності, що визначає важливість цих елементів для цього елемента. Ці ваги нормалізуються (зазвичай за допомогою функції softmax) і використовуються для обчислення зважених сум гнізд елементів. Це дозволяє моделі приділяти більше уваги важливим елементам і менше – менш важливим.

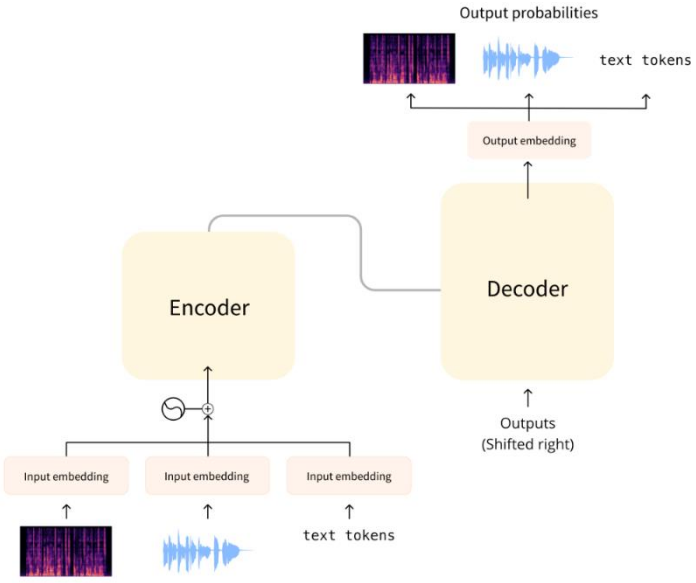

рис. 1.4.1 Будова трансформерів

Кілька рівнів уваги: трансформери використовують багаторівневий механізм уваги, який називається «блоками повторення». Кожен блок містить рівні уваги, які

взаємодіють один з одним. Це дозволяє моделі фіксувати різні рівні залежностей і абстракцій від вхідної послідовності.

Зміщення та нормалізація: після обчислення зважених сум вбудовань до кожного результату застосовують звичайний рівень нейронної мережі (наприклад, з одним або декількома шарами ReLU) і операцію нормалізації, таку як нормалізація шару або пакетна нормалізація. Це допомагає зменшити вибух градієнта та покращити конвергенцію моделі.

Блок кодування та блок декодування: Трансформери зазвичай мають як блок кодування, так і блок декодування. Блок кодування використовується для обробки вхідної послідовності, тоді як блок декодування використовується для генерації вихідної послідовності. Блок декодування має додатковий механізм уваги, який дозволяє моделі звертати увагу на вихідні послідовності під час генерації. [14]

Маскування: під час навчання трансформерів використовуються маски, щоб модель не могла «бачити» майбутні елементи вхідної послідовності. Це допомагає моделі генерувати вихідні дані послідовність за послідовністю.[12]

Ці принципи дозволяють трансформерам ефективно моделювати взаємодію між елементами вхідних і вихідних послідовностей, що робить їх дуже ефективними для різноманітних завдань обробки природної мови.

1.5 Ембеддінги – принципи створення та їх застосування

Ембеддінги - вбудовані цифрові векторні представлення слів або інших одиниць тексту, що використовуються для обробки природної мови (NLP). Основна ідея ембеддінгів полягає в тому, щоб перетворити неструктуровані текстові дані, які важко обробити комп'ютером, у векторні представлення, які зберігають семантичну інформацію та можуть використовуватися для машинного навчання.[9]

Ембеддінги створюються шляхом навчання нейронних мереж або інших алгоритмів на великих масивах текстових даних. До основних методів створення інтеграцій відносяться:

Word2Vec:

Безперервний мішок слів (CBOW): Ця модель передбачає, що дане слово визначається на основі його контексту, тобто слів навколо нього. Наприклад, для пропозиції «кіт сидить на вікні» модель спробує передбачити слово «сидить», використовуючи контекст «кіт...на вікні».

Skip-gram: ця модель працює у зворотному порядку: вона передбачає контекстні слова на основі певного слова. Наприклад, для одного і того ж речення «кіт сидить на вікні» модель спробує передбачити слова «кіт» і «на вікні» від слова «сидіти».

GloVe (Глобальні вектори для представлення слів):

Ця модель поєднує глобальну та локальну інформацію з тексту. GloVe використовує матрицю співчастот слів у контекстах для створення векторів. Мета полягає в тому, щоб навчити вектори таким чином, щоб скалярний добуток між ними відповідав логарифмічній ймовірності їх спільного появи в контекстах.

FastText:

FastText розширює Word2Vec, беручи до уваги одиниці інформації про підслова, такі як символи та їх послідовності. Це дозволяє моделі ефективно

працювати з рідкісними або навіть новими словами, яких не було в навчальному корпусі.

BERT (Подання двонаправленого кодера від трансформаторів):

BERT використовує двосторонній підхід до моделювання тексту. На відміну від Word2Vec, BERT враховує контекст обох сторін (лівої та правої) даного слова. Це дозволяє створювати контекстні векторні представлення, які краще представляють значення слова в різних контекстах.

Рейтинг якості:

Визначте якість інтеграції за допомогою таких завдань, як аналогії (наприклад, «король – чоловік + жінка = королева»), кластеризація або інші завдання НЛП.

Використання інтеграцій

Ембеддінги мають широкий спектр застосувань у задачах обробки природної мови:

- Класифікація тексту:

Векторні представлення слів або документів використовуються як вхідні дані для таких алгоритмів класифікації, як логістична регресія, SVM або нейронні мережі.

- Аналіз настрою:

Ембеддінги допомагають визначити емоційний тон тексту (позитивний, негативний, нейтральний).

- Автоматичний переклад:

Використання ембеддінги для представлення слів у багатомовних системах перекладу.

- Відповіді на запитання:

Системи, які використовують інтеграцію для пошуку відповідей на запити користувачів на основі контексту.

- Системи рекомендацій:

Векторні зображення користувачів і продуктів допомагають створювати персоналізовані рекомендації.

- Пошук інформації:

Використовуйте ембеддінги для вдосконалення алгоритмів пошуку, враховуючи семантичну схожість запитів і документів.

Ембеддінги значно підвищило ефективність багатьох завдань NLP, надаючи більш точні та контекстно значущі представлення тексту. Вони дозволяють моделям краще розуміти та обробляти природну мову, що відкриває нові можливості для створення потужних та інтелектуальних систем.

1.6 Векторний пошук: принципи та застосування

Векторний пошук — це метод пошуку інформації, який використовує векторне представлення даних для пошуку об’єктів, які найкраще відповідають запиту. На відміну від традиційних методів пошуку, які зазвичай покладаються на точне збіг ключових слів, векторний пошук оцінює семантичну подібність між векторними представленнями запиту та об’єктами бази даних.

Основний принцип векторного пошуку полягає в тому, що об’єкти (наприклад, документи, зображення, аудіо) і запити представлені як вектори в багатовимірному просторі. Потім визначається показник подібності між цими векторами, щоб знайти найбільш відповідні результати. Ось як це працює докладніше:

1. Векторизація даних:

Кожен об’єкт перетворюється на вектор за допомогою відповідної техніки векторизації (наприклад, вставлення тексту, функції зображення для комп’ютерного зору). Запит також перетворюється на вектор у тому самому векторному просторі.

2. Розрахунок ступеня подібності:

Для вимірювання подібності між векторами використовуються різні показники. Найпоширенішими мірами є косинус подібності, евклідова відстань і манхеттенська відстань.

Косинусна подібність визначає кут між векторами та добре підходить для даних великої розмірності:

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

Евклідова відстань вимірює прямолінійну відстань між векторами у просторі:

$$\text{Euclidean Distance}(A, B) = \sqrt{\sum_{i=1}^{n}(A_i - B_i)^2}$$

3. Індексація та пошук:

Спеціалізовані структури даних, такі як k-d дерева, LSH (Locality-Sensitive Hashing) або бібліотеки, такі як FAISS (Facebook AI Similarity Search), використовуються для ефективного пошуку у великих базах даних.

Ці структури дозволяють швидко знаходити вектори, найближчі до вектора запиту.

Векторний пошук використовується в різних областях і для різних завдань:

- Пошук документів та інформації:

Використовується для пошуку текстових документів, статей, книг та інших текстових документів за їх семантичним змістом. Системи векторного пошуку можуть знайти документи, що відповідають запиту, навіть якщо вони не містять точних ключових слів для запиту.

- Рекомендаційні системи:

Векторні представлення користувачів і продуктів використовуються для створення персоналізованих рекомендацій. Наприклад, музичні та відеоплатформи можуть рекомендувати контент на основі схожості векторів уподобань користувачів.

- Комп'ютерний зір:

Векторний пошук використовується для розпізнавання об'єктів на зображеннях, пошуку схожих зображень і відео. Векторні представлення зображень можна отримати за допомогою глибоких нейронних мереж, таких як CNN (конволюційні нейронні мережі).

- Пошук по аудіо:

Векторний пошук використовується для ідентифікації та порівняння аудіозаписів. Аудіофайли перетворюються на векторні за допомогою шаблонів, як-от вбудованих аудіофайлів, що дозволяє знаходити схожі музичні доріжки, шаблони мовлення тощо.

- Біоінформатика:

Він використовується для порівняння геномних послідовностей та інших біологічних даних. Векторні представлення можуть допомогти знайти подібні генні або білкові послідовності, що важливо для здоров'я та медичних досліджень.

- Процес виробництва та якість:

Він використовується для моніторингу якості продукції та виявлення дефектів на основі аналізу даних сенсора та зображення. Векторне представлення даних від різних датчиків може допомогти швидко виявити аномалії.

Технології реалізації векторного пошуку

Існує кілька інструментів і бібліотек, які спрощують реалізацію векторного пошуку:

- FAISS (Пошук подібності через ШІ у Facebook):

Дуже ефективна бібліотека для пошуку схожих векторів, розроблена Facebook.

Підтримує широкий набір алгоритмів і структур даних для ефективного пошуку в просторах великої розмірності.

- Annoy (Approximate Nearest Neighbors Oh Yeah):

Бібліотека для швидкого пошуку схожих векторів, розроблена Spotify.

Використовує деревоподібні структури даних для швидкого пошуку найближчих сусідів.

- HNSW (Hierarchical Navigable Small World):

Алгоритм пошуку найближчих сусідів, який забезпечує високу швидкість і точність.

Він використовується в багатьох сучасних програмах векторного пошуку.

- NMSLIB (Non-Metric Space Library):

Бібліотека векторного просторового пошуку, що підтримує різні показники подібності. Підходить для широкого кола завдань, від тексту до зображень.

Векторний пошук стає все більш важливим інструментом у сучасному світі даних, що постійно змінюється. Це дозволяє ефективно знаходити релевантну інформацію, покращувати якість рекомендацій і аналізувати великі масиви даних у різних галузях.

2. Технологічні інструменти та платформи

2.1 Microsoft Azure

Microsoft Azure — це потужна хмарна платформа, яка надає широкий спектр послуг для обчислень, зберігання даних, машинного навчання та аналізу даних. Ваша дисертація, яка зосереджена на розробці чат-бота для діалогу між кандидатами та вищими навчальними закладами, може отримати користь від кількох ключових послуг від Azure.

1. Azure Cognitive Search (векторний пошук):

Azure Cognitive Search — це повністю керована служба пошуку, яка надає потужні можливості пошуку та індексування для програм. Він підтримує як традиційний пошук за ключовими словами, так і векторний пошук, який використовує вбудовування для підвищення точності результатів.[11]

- Функціональність:

Інтеграція з іншими когнітивними сервісами для аналізу та обробки тексту.

Підтримка векторних індексів для високоефективного пошуку подібних векторів.

Можливість створення індексів на основі різних джерел даних (SQL, NoSQL, Blob Storage тощо).

- Застосування:

Пошук інформації для запитів користувачів на основі семантичної подібності.

Фільтруйте та сортуйте результати за релевантністю.

2. Azure Storage Account (сховище даних):

Обліковий запис зберігання Azure — це комплексне рішення для зберігання різноманітних даних у хмарі, включно з файлами, таблицями, чергами та великими об’єктами.

- Функціональність:

1. Зберігання Blob: зберігання великих обсягів неструктурованих даних, таких як документи, зображення та відео.
2. Зберігання таблиць: легке сховище NoSQL для структурованих даних.
3. Зберігання файлів: сховище файлів з доступом через протокол SMB.
4. Зберігання черги: організація черг для обробки повідомлень.

- Застосування:

1. Зберігання текстових даних та інших документів для індексації та пошуку.
2. Резервне копіювання та архівування даних.
3. Сервіс Azure OpenAI:

Сервіс Azure OpenAI надає доступ до моделей OpenAI, наприклад GPT-3, на платформі Azure. Це дозволяє використовувати потужні моделі для створення тексту, обробки природної мови та інших завдань.

- Функціональність:

Створення тексту та відповідей на основі вхідних запитів.

Аналіз тексту, включаючи розпізнавання емоцій, класифікацію тощо.

Підтримка інтеграції з іншими службами Azure для комплексного аналізу даних.

- Застосування:

Створення діалогів для чат-бота.

Автоматизація відповідей на запитання кандидатів.

Створення контенту для сайтів та інших інформаційних ресурсів.[10]

2.2 LangChain: Інтеграція ланцюжків мовної обробки

LangChain — це платформа, призначена для об’єднання різних мовних моделей і методів обробки природної мови (NLP) в єдиний ланцюжок. Такий підхід дозволяє створювати комплексні рішення обробки текстових даних, які можна адаптувати до різних завдань і контекстів. У контексті адаптивних чат-ботів для освітніх цілей LangChain пропонує гнучкі та точні інструменти для покращення взаємодії зі студентами, персоналізації навчання та автоматизації рутинних процесів. [7]

- Композиція моделей:

LangChain дозволяє об’єднувати кілька мовних моделей у послідовний ланцюжок, де кожна модель виконує своє конкретне завдання. Наприклад, перший крок може включати розпізнавання та виправлення помилок у тексті, друга модель може аналізувати тон, а третя може генерувати відповіді на запити. Така структура підвищує точність і гнучкість обробки даних.

- Модульність і масштабованість:

Платформа пропонує модульний підхід, який дозволяє легко додавати нові моделі або змінювати існуючі компоненти без капітального ремонту системи. Це важливо для чат-ботів, яким потрібно швидко оновлюватися та адаптуватися до нових програм або методологій навчання.

- Підтримка різних мовних завдань:

LangChain підтримує широкий спектр завдань NLP, включаючи класифікацію тексту, видобуток інформації, генерацію тексту, тональний аналіз, машинний переклад тощо. Це дозволяє створювати багатофункціональні системи, здатні вирішувати різні завдання одночасно, тим самим підвищуючи якість та ефективність обслуговування студентів.

Використання LangChain в адаптивних чат-ботах

- Персоналізація відповідей:

LangChain дозволяє налаштовувати мовні моделі для надання персоналізованих відповідей, враховуючи контекст попередніх взаємодій та індивідуальні потреби кожного студента. Наприклад, бот може запам’ятати теми, з якими у студента виникли труднощі, і надати додаткові пояснення чи ресурси.

- Контекстуальна обізнаність:

Ланцюжки моделей можуть зберігати контекст попередніх діалогів, дозволяючи чат-ботам надавати більш актуальні та точні відповіді. Це особливо корисно для освітніх чат-ботів, де важливо розуміти прогрес студента та адаптувати рекомендації на основі його прогресу та потреб.

- Аналіз навчальних матеріалів:

LangChain може аналізувати навчальні матеріали та автоматично створювати навчальні плани або тести на основі текстів підручників. Це значно спрощує процес підготовки матеріалів для викладачів і надає студентам персоналізовані завдання.

Технічний аспект LangChain

- Інтеграція з іншими системами:

LangChain легко інтегрується з іншими системами та інструментами, такими як бази даних, інструменти для великих даних та API машинного навчання. Це дозволяє створювати комплексні рішення, поєднуючи різні джерела даних і методи обробки, забезпечуючи ефективне використання ресурсів.

- Впровадження та оптимізація моделей:

Платформа надає інструменти для вдосконалення та оптимізації мовних моделей, що забезпечує високу точність і ефективність обробки. Викладачі та розробники можуть налаштовувати шаблони відповідно до конкретних потреб курсів або окремих студентів.

- Масштабованість і продуктивність:

LangChain пропонує високу продуктивність і масштабованість, що дозволяє обробляти великі обсяги даних у реальному часі. Це важливо для чат-ботів, які

повинні взаємодіяти одночасно з великою кількістю користувачів, зберігаючи якість обслуговування.

Приклади використання LangChain

- Інтерактивні освітні майданчики:

Інтерактивні навчальні платформи можуть використовувати LangChain для створення чуйних чат-ботів, які допомагають студентам у режимі реального часу, надають персоналізовані рекомендації та аналізують продуктивність. Наприклад, платформа може автоматично створювати додаткові завдання для учнів на основі їхніх відповідей на попередні тести.

- Автоматизація навчального процесу:

LangChain можна використовувати для автоматизації багатьох аспектів навчального процесу, таких як оцінювання завдань, створення навчальних матеріалів і відстеження прогресу учнів. Це сприяє зниженню навантаження на вчителів і підвищенню ефективності навчання. Автоматичне оцінювання та миттєвий зворотній зв'язок значно покращують процес навчання.

- Дослідницькі проекти:

Університети та дослідницькі інститути можуть використовувати LangChain для проведення досліджень обробки природної мови, розробки нових адаптивних методів навчання та перевірки їх ефективності. Наприклад, дослідники можуть експериментувати з різними комбінаціями моделей для досягнення оптимальних результатів у конкретних навчальних завданнях.

LangChain — потужний інструмент обробки природної мови, який відкриває нові можливості для створення адаптивних систем навчання. Використання цієї платформи дозволяє розробляти багатофункціональні чат-боти, здатні запропонувати індивідуальний підхід до навчання, підвищити ефективність навчального процесу та забезпечити якісну підтримку студентів. Інтеграція LangChain з іншими технологіями та інструментами робить його важливою частиною сучасних освітніх рішень, спрямованих на покращення взаємодії та

результатів навчання. LangChain дозволяє використовувати переваги різних мовних моделей і методів, забезпечуючи гнучкість, точність і масштабованість, необхідні для успішного впровадження адаптивного навчання в навчальних закладах.

3. Практична реалізація та аналіз

3.1 Аналіз проблеми

У сучасному університетському середовищі виникає низка проблем, пов’язаних із процесом вступу абітурієнтів. Одним із головних аспектів цих проблем є брак часу та ресурсів для надання вичерпних відповідей на запитання потенційних абітурієнтів. Часто буває, що співробітники університету не встигають відповісти на всі запитання через велику кількість вхідних повідомлень.

Крім того, існує проблема недостатнього інформаційного забезпечення абітурієнтів. Студенти часто не можуть знайти відповіді на свої запитання на веб-сайті університету чи інших ресурсах, що призводить до стресу та невпевненості у виборі університету та спеціальності.

Значною проблемою є також недостатній рівень персоналізації та індивідуального підходу до кожного кандидата. Через обмежені ресурси університети не завжди можуть приділяти індивідуальну увагу кожному абітурієнту та відповідати на всі його запитання та потреби.

Зокрема, аналіз показав, що більшість запитань абітурієнтів стосуються умов вступу, порядку подання документів, умов вступної кампанії та іншої важливої інформації. Більшість цих питань можна автоматизувати та оптимізувати за допомогою чат-бота, що спростить роботу співробітників університету та покращить якість обслуговування абітурієнтів.

Підсумовуючи, проблема академічної підтримки абітурієнтів потребує ефективного та інноваційного підходу. Розробка чат-бота на базі трансформерів GPT може стати одним із рішень, що спростить та оптимізує процес взаємодії університету та абітурієнтів, забезпечить швидкий доступ до інформації та допомогу в режимі реального часу.

3.2 Аналіз потреб

Аналіз вимог користувачів можна розглядати з кількох ключових аспектів:

- Інформаційні потреби кандидатів:

Процес вступу: Абітурієнтам потрібна чітка та доступна інформація про вимоги до вступу, процедури подання заяв, дати вступних іспитів тощо.

Навчальні програми та спеціальності: Інформація про навчальні програми, спеціальності, характеристики навчання та кар'єрні можливості є важливою для абітурієнтів при виборі університету.

Фінансові питання: багато абітурієнтів хочуть отримати інформацію про вартість навчання, наявність стипендій та інші фінансові можливості.

- Функціональні вимоги до кандидатів:

Отримувати відповіді: кандидати повинні мати можливість швидко й ефективно отримувати відповіді на свої запитання без зайвих затримок.

Персоналізація: кожен кандидат унікальний, тому важливо забезпечити індивідуальний підхід і відповіді, враховуючи їхні індивідуальні потреби та попередні взаємодії з системою.

- Технічні потреби:

Простота взаємодії: Кандидати очікують практичний та інтуїтивно зрозумілий інтерфейс, який дозволить їм легко та швидко отримувати необхідну інформацію.

Стабільність і доступність: важливо, щоб система була стабільною та доступною в будь-який час, оскільки у кандидатів можуть виникнути запитання в будь-який час.

Аналіз цих потреб допоможе створити ефективну та корисну систему чат-ботів, яка відповідатиме потребам користувачів та сприятиме успішній взаємодії між абітурієнтами та університетом.

3.3 Реалізація RAG для покращення роботи чатбота

Методологія RAG (Retrieval-Augmented Generation) поєднує в собі два ключові аспекти: пошук інформації та генерацію тексту. Розробляючи чат-бот на основі RAG, важливо розуміти теоретичні принципи, що лежать в основі обох аспектів.

По-перше, важливу роль у RAG відіграє пошук інформації. Цей аспект передбачає створення документальної бази, яка використовуватиметься для отримання контекстуально обґрунтованих відповідей. Для пошуку інформації використовуються різні алгоритми та моделі, наприклад TF-IDF, BM25 або більш складні моделі, такі як BERT. Ці алгоритми допомагають знайти найбільш релевантні документи на запит користувача, що є запорукою якісної відповіді чат-бота.

По-друге, генерування тексту на основі знайденої інформації є ще одним важливим аспектом RAG. Після отримання відповідних документів модель генерації тексту використовує цю інформацію для генерації відповідей на запити користувачів. Ці моделі зазвичай базуються на архітектурах трансформаторів, таких як Generative Pre-trained Transformer (GPT), які навчені створювати текст на основі контексту та структури мовлення. Важливо, щоб генеративна модель адаптувалася до інформації, яку вона отримує від моделі пошуку, оскільки це дозволяє їй надавати змістовні та зрозумілі відповіді.

Основи RAG:

- Пошук інформації: RAG поєднує методи генерації тексту з пошуком інформації серед документів. Це дозволяє чат-боту отримувати контекстно обґрунтовані відповіді на запити користувачів.
- Інформаційна релевантність: модель інформаційного пошуку визначає релевантність документів запиту користувача. Це допомагає гарантувати, що згенеровані відповіді базуватимуться на поточних і релевантних даних.
- Генерація тексту на основі знайденої інформації: після отримання відповідних документів модель генерації тексту використовує цю інформацію для створення

відповідей на запити користувачів. Це дозволяє чат-боту генерувати більш обґрунтовані та змістовні відповіді.

Щоб краще реалізувати методологію Retrieval-Augmented Generation (RAG), важливо враховувати наступні ключові деталі:

- Якість документальної бази: Ефективне впровадження КРП вимагає наявності репрезентативної та якісної документальної бази. Це можуть бути наукові статті, книги, рекомендації, правила або будь-яка інша інформація, що стосується домену вашого чат-бота. Якість документальної бази безпосередньо впливає на якість відповідей.
- Ефективність моделі пошуку: важливо використовувати ефективні алгоритми пошуку інформації, які забезпечують швидкий і точний пошук відповідних документів. Використання продуктивних і точних методів дослідження допоможе підвищити якість взаємодії з чат-ботом.
- Адаптивність генеративної моделі: модель генерації тексту має бути адаптованою до інформації, яку вона отримує від моделі пошуку. Це означає, що генеративна модель повинна мати можливість ефективно використовувати контекст знайдених документів для створення відповідей.
- Керування різноманітністю відповідей: важливо гарантувати різноманітність та адекватність отриманих відповідей. Цього можна досягти за допомогою методів генерації різних варіантів відповідей або шляхом реалізації механізмів ранжування варіантів.

Тестування та оцінка: систематичне тестування та оцінка системи є важливими для забезпечення ефективності та якості впровадження RAG. Це допоможе виявити проблеми та з часом покращити роботу чат-бота.

Ці основні деталі допоможуть забезпечити успішне впровадження методології RAG у вашому чат-боті. Таким чином, впровадження RAG у чат-бот передбачає впровадження ефективних алгоритмів пошуку інформації, які забезпечують точний і швидкий пошук відповідних документів, а також використання генеративних моделей, здатних ефективно використовувати цю інформацію для створення значущих відповідей.

3.4 Prompt engineering

Prompt engineering— це процес створення та оптимізації вхідних запитів або «підказок» для моделей штучного інтелекту, особливо моделей, створених текстом. Цей підхід передбачає визначення точних і ефективних моделей, які можна використовувати для отримання бажаних відгуків від моделей.

Промпти є відправною точкою для моделей ШІ, які генерують текстові відповіді. Їхні формулювання відіграють важливу роль у визначенні того, які відповіді будуть отримані. Якщо промпт сформульований чітко та точно, модель може бути більш здатною вирішити завдання та надати більш адекватні відповіді.

1. Формулювання підказок: ключовим кроком є правильне формулювання підказок. Підказки мають бути чіткими, конкретними та легкими для розуміння моделі. Вони повинні відображати тип очікуваних відповідей і спрямовані на досягнення конкретної мети.
2. Унікальність і різноманітність: важливо, щоб підказки були унікальними та різноманітними, щоб забезпечити широкий спектр можливих відповідей. Використання різних варіантів підказок дозволяє моделі вивчати різноманітні контексти та завдання.
3. Контроль контексту: під час розробки підказок важливо враховувати, що шаблон може використовувати лише контекст, який міститься в підказці. Тому важливо включити в підказку необхідну контекстну інформацію, але уникати надмірностей, які можуть призвести до семантичної нечіткості.
4. Експериментування та аналіз: Часте експериментування з різними підказками допомагає зрозуміти, які підказки найкраще працюють для певного завдання чи контексту. Аналіз результатів дозволяє покращити підказки та покращити взаємодію з моделлю.
5. Оптимізація та налаштування: після експериментів і аналізу важливо оптимізувати та налаштувати підказки для досягнення найкращих результатів. Це може включати зміну формату, додавання або видалення ключових слів і уточнення мети запиту.

Отже, щоб досягти кращих результатів у швидкому проектуванні, ось деякі інші найкращі практики:

6. Застосування принципу подібності-відмінності: створюючи підказки, використовуйте принцип подібності-відмінності, який передбачає вказівку на шаблони подібності та відмінності між сценаріями. Наприклад, для підказки «Порівняти X і Y» одна підказка може починатися з «X і Y», а інша — з «Різниця між X і Y».
7. Використання ключових слів: додавання ключових слів у підказки може допомогти отримати більш точні та релевантні відповіді. Це можуть бути ключові слова, що вказують на тип запиту або очікуваної інформації. Наприклад, підказка для запиту рецепта може містити слова «рецепт», «інгредієнти», «процес приготування» та інші.
8. Експериментуйте з довжиною підказки. Експериментуйте з довжиною підказки, щоб покращити результати. У деяких випадках можуть знадобитися короткі прості підказки, тоді як для складніших запитів можуть знадобитися довші та детальніші підказки.
9. Постійне вдосконалення та оновлення: як і в будь-якому іншому аспекті розробки ШІ, постійне вдосконалення та оновлення підказок є важливим. Підказки слід час від часу переглядати та оновлювати, щоб відобразити мінливий контекст, нові потреби користувачів або нові шаблони взаємодії з моделлю.
10. Уникайте двозначності: створюючи підказки, важливо уникати двозначності, тобто вони мають бути однозначними та зрозумілими для моделі. Якщо підказка може викликати розуміння різними способами, це може призвести до неправильних відповідей.
11. Адаптуйтеся до аудиторії: створюючи підказки, важливо враховувати аудиторію, з якою ви працюєте. Термінологія та стиль підказок можуть відрізнятися залежно від цільової аудиторії, її мови та рівня знань.
12. Підтримка діалогічного характеру: підказки можуть бути сформульовані таким чином, щоб сприяти діалогічному характеру взаємодії з моделлю. Наприклад,

використовуючи підказки з такими фразами, як «Про що ти думаєш...?» може привести модель до більш докладних відповідей.

Загалом prompt engineering є важливим кроком у розвитку систем штучного інтелекту, особливо для моделей з генерацією тексту, таких як GPT. Ефективне використання підказок дозволяє підвищити якість взаємодії з моделлю та отримати більш точні та актуальні результати.

3.6 Можливі рекомендації щодо подальшого розвитку системи

Розширення функціональних можливостей: враховуючи динамічний характер вимог абітурієнта та різноманітність процесів вступу, систему можна постійно розвивати, додаючи нові функціональні можливості, такі як онлайн-консультування, допуск до подання заяв та відстеження статусу іспиту, перегляд відгуків випускників тощо. .

Оптимізація алгоритмів: постійне вдосконалення алгоритмів взаємодії та обробки запитів може покращити якість відповідей і швидкість реагування системи. Використання передових методів машинного навчання та обробки природної мови може допомогти краще зрозуміти контекст і наміри користувача.

Персоналізація та адаптація: розвиток системи в напрямку персоналізованої взаємодії з користувачами, враховуючи їхні індивідуальні потреби та попередні взаємодії, може зробити взаємодію більш ефективною та зручною для користувачів.

Багатомовний підхід: розширення можливостей системи іншими мовами може розширити аудиторію та надати доступ до інформації кандидатам з різних країн і культур.

Забезпечення безпеки даних: посилення заходів щодо забезпечення конфіденційності та захисту персональних даних користувачів є важливим аспектом майбутнього розвитку системи, зокрема щодо зберігання та обробки конфіденційних даних або особливо чутливої інформації.

Постійне тестування та звітність: регулярне тестування та моніторинг роботи системи, а також збір звітів про ефективність та задоволеність користувачів допоможуть своєчасно виявити та виправити можливі проблеми та недоліки, що забезпечить стабільну та ефективну роботу системи в майбутньому.

Реалізація цих рекомендацій допоможе зробити систему чат-ботів, покликану підтримувати діалог між абітурієнтами та університетом, ефективнішою, функціональнішою та чуйнішою на потреби користувачів.

4. Деталі практичної реалізації чат-бота

Нижче наведено імпорт необхідних бібліотек для реалізації чат-бота, що використовує сервіси Microsoft Azure для векторного пошуку та векторизації текстів, а також інтеграцію з моделями обробки природної мови від Vertex AI.

```python
import os

from langchain_community.vectorstores.azuresearch import AzureSearch
from langchain_community.embeddings import VertexAIEmbeddings

from langchain.text_splitter import CharacterTextSplitter
from langchain_community.document_loaders import TextLoader

from langchain_core.prompts import PromptTemplate
from langchain_community.llms import VertexAI
```

рис. 4.1

Нижче наведено функцію для завантаження текстових документів, розділення їх на менші частини (чанки), та подальше додавання у векторний пошук.

```python
# NOTE: For loading documents into vector search

def load_documents(files_path, chunk_size=500):
    all_docs = []

    for file_path in files_path:
        loader = TextLoader(file_path, encoding="utf-8")

        documents = loader.load()
        text_splitter = CharacterTextSplitter(chunk_size=chunk_size, chunk_overlap=150)
        docs = text_splitter.split_documents(documents)

        all_docs.extend(docs)

    return all_docs

docs = load_documents()

vector_store.add_documents(documents=docs)
```

рис. 4.2

Шаблон запиту для задачі запитання-відповідь, який використовується для формування відповіді на основі наданого контексту. Цей шаблон забезпечує, що модель використовує тільки наданий контекст для формування відповіді, що підвищує точність та релевантність відповіді.

```
# prompt template for the Q&A task
final_template = """Answer the question based on the context below. If the
question cannot be answered using the information provided answer
with "I don't know".

Context: {chunks}

Question: {question}

Answer: """
```

рис. 4.3

Нижче наведено код для ініціалізації моделі обробки природної мови від Vertex AI, яка використовується для генерації тексту на основі наданих запитів.

```
model=VertexAI(model_name="text-bison")
```

рис. 4.4

Нижче наведено шаблон запиту, який використовується для переформулювання запитання з урахуванням історії чату. Шаблон складається з двох кроків: створення автономного запитання на основі історії чату та запитання, а також виокремлення головної теми з цього запитання.

```
template_prompt_rephrase_chat_query = """
Your task is:

Step 1. Generate a standalone question which is based on the new question plus the chat history. Just create the standalone question without commentary.

Step 2. Extract from results of Step 1 only main topic.
Make your response as short as possible.

Chat History:
{chat_history}

Follow Up Input: {question}

Answer:
"""
prompt_rephrase_chat_query = PromptTemplate(template=template_prompt_rephrase_chat_query, input_variables=["chat_history", "question"])
```

рис. 4.5

Код, який об'єднує шаблон запиту для переформулювання запитань з мовною моделлю Vertex AI, створюючи таким чином ланцюжок обробки запитів.

```
chain = prompt_rephrase_chat_query | model
```

рис. 4.6

Нижче наведено код, який виконує виклик ланцюжка обробки запитів (chain) із заданою історією чату та новим запитанням для отримання відповіді, та результат роботи на прикладі

```
res = chain.invoke({
    'chat_history':
        [
            SystemMessage(content='You are an AI assistant at University, that helps applicants find information about University'),
            HumanMessage(content='Hi, what are the majors at your university?'),
            AIMessage(content=f'our university offers a wide range of specialties: Computer Science, software engineering, system analysis and others.')
        ],
    'question':'What can you tell about the first?'
})
```

```
res
✓ 0.0s
Комп\'ютерні науки - це галузь знань, що вивчає теорію та практику обробки інформації, що передбачає використання комп\'ютерів. Ось деякі ключові аспекти ко
```

рис. 4.7

```
'Комп'ютерні науки - це галузь знань, що вивчає теорію та практику обробки інформації, що передбачає використання комп'ютерів. Ось деякі ключові аспекти комп'

1. **Програмування**: Вивчення різних мов програмування, таких як Python, Java, C++, і багатьох інших, які дозволяють створювати програми та веб-сайти.

2. **Алгоритми та структури даних**: Дослідження ефективних способів обробки даних та виконання операцій з ними, таких як пошук, сортування, та інші.

3. **Штучний інтелект та машинне навчання**: Розвиток алгоритмів, які дозволяють комп'ютерам вчитися на основі даних та приймати рішення на основі цих даних.

4. **Бази даних**: Вивчення методів збереження та організації великих обсягів даних, а також розробка систем управління базами даних.

5. **Комп'ютерна графіка та візуалізація**: Розробка програм, які дозволяють створювати візуальні ефекти, анімацію та графічні інтерфейси.
```

рис. 4.8

Код, що визначає параметри для підключення до векторного сховища, ініціалізує об'єкт для створення ембедінгів за допомогою Vertex AI, а також задає ім'я індексу для векторного пошуку.

```
# Constants
vector_store_address: str = "VECTOR-STORE-ADDRESS"
vector_store_password: str = "VECTOR-STORE-PASSWORD"

embeddings = VertexAIEmbeddings(model_name='textembedding-gecko@001', chunk_size=1)
index_name = "clindex"
```

рис. 4.9

Код, який підключається до сервісу пошуку Azure, виконує пошук за заданим текстом та отримує результати.

```python
from azure.core.credentials import AzureKeyCredential
from azure.search.documents import SearchClient

service_endpoint = os.environ["AZURE_SEARCH_SERVICE_ENDPOINT"]
index_name = os.environ["AZURE_SEARCH_INDEX_NAME"]
key = os.environ["AZURE_SEARCH_API_KEY"]

search_client = SearchClient(service_endpoint, index_name, AzureKeyCredential(key))

results = search_client.search(search_text="What are the majors at your university?")
```

рис. 4.10

Функція, що виконує основну функціональність, що пов'язана з обробкою запитів та генерацією відповіді.

```python
def main_function(input_request: dict):
    input_message = input_request.get('input_message')

    if input_message is None or input_message == "":
        raise ValueError("Input message is empty")

    model = VertexAI(model_name="text-bison")

    # init vector store
    vector_store = AzureSearch(
        azure_search_endpoint=vector_store_address,
        azure_search_key=vector_store_password,
        index_name=index_name,
        embedding_function=embeddings.embed_query,
    )

    chunks = search_related_chunks(input_message, vector_store)

    final_template_prompt = PromptTemplate.from_template(final_template)

    chain_final_message = final_template_prompt | model

    answer = chain_final_message.invoke({'chunks':"\n\n\n".join(chunks), 'question':input_message})

    return answer
```

рис. 4.11

5. Безпека життєдіяльності та основи охорони праці

5.1 Забезпеченя безпеки працівників

Забезпечення безпеки працівників є важливим аспектом у роботі над розробкою чат-бота для діалогу між кандидатами та ВНЗ. Основні види діяльності включають:

- Фізична безпека працівників: Забезпечити комфортні та безпечні умови праці для працівників, які працюють над проектом. Це включає створення ергономічного робочого середовища, запобігання фізичній напрузі та захист від потенційних небезпек, таких як неправильне розміщення обладнання або відсутність належного освітлення.
- Психологічна безпека: Створити сприятливі умови для збереження психологічного здоров'я працівників. Це передбачає надання можливостей для регулярного відпочинку, підтримку в стресових ситуаціях і запобігання професійному вигоранню через організацію здорового режиму праці.
- Медична допомога: Наявність першої допомоги та доступ до медичних послуг у разі необхідності. Організація регулярних медичних оглядів для попередження та виявлення професійних захворювань.
- Навчання з безпеки життєдіяльності: регулярні інструктажі та навчання з безпеки життєдіяльності, включаючи навички надання першої допомоги, дії в надзвичайних ситуаціях і заходи запобігання нещасним випадкам.

5.2 Охорона праці та забезпечення здоров’я працівників

Розробка програмного забезпечення, в тому числі чат-ботів, має враховувати основні принципи охорони праці з метою забезпечення безпечних і здорових умов праці працівників. До основних аспектів охорони праці відносяться:

- Ергономіка робочого місця: Створення комфортних умов для працівників, які займаються розробкою та тестуванням програмного забезпечення. Це і правильне розташування комп’ютерів, зручні крісла, достатнє освітлення та вентиляція приміщень.
- Психологічне здоров'я: запобігання стресу та виснаження у співробітників шляхом забезпечення балансу між роботою та відпочинком, організацією психологічної підтримки та заходів щодо зниження навантаження.
- Навчання та інструктаж: регулярне навчання з техніки безпеки для всіх працівників, включаючи навчання використанню обладнання, основним правилам безпеки та діям у надзвичайних ситуаціях.
- Моніторинг умов праці: Регулярний моніторинг умов праці для виявлення потенційних небезпек і вжиття заходів щодо їх усунення. Атестація робочих місць відповідно до нормативних вимог.
- Забезпечення засобами індивідуального захисту: забезпечте працівників необхідним захисним обладнанням, таким як екрани проти відблисків, спеціальні килимки для миші з підставкою для зап’ястя, а також організовуйте регулярні перерви для фізичних вправ.

Висновки

У цій дипломній роботі проведено комплексне дослідження та розробка чат-бота для діалогу між абітурієнтами та вищими навчальними закладами (ВНЗ) з використанням сучасних технологій машинного навчання та обробки природної мови. Основною метою роботи було створення ефективної системи, яка б забезпечувала швидку та якісну взаємодію абітурієнтів з університетом, надаючи відповіді на різноманітні запитання щодо спеціальностей, процесу вступу та інших важливих аспектів.

Отримані результати:

Огляд технологій і методів:

- Проведено аналіз основ машинного навчання та його ролі в обробці природної мови (NLP).
- Вивчаються основи великих мовних моделей (LLM) та їх еволюція.
- Детально розглянуто архітектуру трансформаторів, які складають основу сучасних моделей НЛП.

Розробка чат-бота:

- Технологія Microsoft Azure використовується для створення векторного пошуку, зберігання даних та інтеграції з моделями штучного інтелекту.
- Було розроблено методи для створення та використання вбудованих елементів, які забезпечують точний пошук і аналіз текстових даних.
- Методи RAG (Retrieval-Augmented Generation) реалізовано для покращення якості відповідей чат-бота.

Аналіз та оптимізація:

- Проведено аналіз проблем та потреб кандидатів, які взаємодіють із чат-ботом.

- Запропоновано рекомендації щодо подальшого розвитку системи, у тому числі щодо вдосконалення алгоритмів обробки питань та вдосконалення інтерфейсу користувача.

Практичне значення роботи:

- Створений чат-бот надає автоматизовану допомогу абітурієнтам, тим самим зменшуючи навантаження на приймальні комісії вишів.
- Впровадження цієї системи сприяє покращенню комунікації між університетом та абітурієнтами, підвищенню їх обізнаності та задоволеності процесом вступу.

Перспективи розвитку:

- Подальший розвиток системи може включати інтеграцію з іншими університетськими інформаційними системами, впровадження адаптивного навчання для персоналізації відповідей і розширення функціональності чат-бота для підтримки різних мов і діалектів.

Таким чином, проведена робота підтверджує високу ефективність використання сучасних технологій машинного навчання та обробки природної мови для створення інтелектуальних систем підтримки користувачів у сфері освіти. Розроблений чат-бот не лише покращує процес взаємодії абітурієнтів та університету, а й демонструє перспективи розвитку та вдосконалення технологій у цій сфері.

Список використаних джерел

1. Brown, T. et al. (2020). Language Models are Few-Shot Learners. arXiv preprint arXiv:2005.14165.
2. Chollet, F. (2018). Learning with Python. Manning Publications.
3. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805.
4. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
5. Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing. Pearson.
6. Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980.
7. LangChain. (2023). LangChain Documentation. Retrieved from https://docs.langchain.com/
8. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. Nature, 521(7553), 436-444.
9. Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.
10. Microsoft. (2023). Azure Bot Service Documentation. Retrieved from https://docs.microsoft.com/en-us/azure/bot-service/
11. Microsoft Azure. (2023). Azure Cognitive Services Documentation. Retrieved from https://docs.microsoft.com/en-us/azure/cognitive-services/
12. Mikolov, T. et al. (2013). Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural Information Processing Systems, 26, 3111-3119.
13. OpenAI. (2023). OpenAI API Documentation. Retrieved from https://beta.openai.com/docs/
14. Peters, M. E. et al. (2018). Deep Contextualized Word Representations. arXiv preprint arXiv:1802.05365.
15. Radford, A. et al. (2019). Language Models are Unsupervised Multitask Learners. OpenAI Blog.

16. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning Representations by Back-Propagating Errors. Nature, 323(6088), 533-536.
17. Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. arXiv preprint arXiv:1409.3215.
18. Vaswani, A. et al. (2017). Attention is All you Need. Advances in Neural Information Processing Systems, 30, 5998-6008.
19. Vaswani, A., Shazeer, N., & Parmar, N. (2018). Tensor2Tensor for Neural Machine Translation. arXiv preprint arXiv:1803.07416.
20. Zhang, Y., & Yang, Q. (2017). A Survey on Multi-Task Learning. arXiv preprint arXiv:1707.08114.

Додаток А Код програми

```
import os
from langchain_community.vectorstores.azuresearch import AzureSearch
from langchain_community.embeddings import VertexAIEmbeddings
from langchain.text_splitter import CharacterTextSplitter
from langchain_community.document_loaders import TextLoader
from langchain_core.prompts import PromptTemplate
from langchain_community.llms import VertexAI
# NOTE: For loading documents into vector search
def load_documents(files_path, chunk_size=500):
    all_docs = []
    for file_path in files_path:
        loader = TextLoader(file_path, encoding="utf-8")
        documents = loader.load()
        text_splitter = CharacterTextSplitter(chunk_size=chunk_size, chunk_overlap=150)
        docs = text_splitter.split_documents(documents)
        all_docs.extend(docs)
    return all_docs
docs = load_documents()
vector_store.add_documents(documents=docs)
# prompt template for the Q&A task
final_template = """Answer the question based on the context below. If the
question cannot be answered using the information provided answer
```

```
with "I don't know".

Context: {chunks}

Question: {question}

Answer: """

def search_related_chunks(input_message, vector_store, metadata_filter=('category', 'bleach')):
    docs = vector_store.similarity_search(
        query=input_message,
        k=30
    )
    chunks = filter_docs_by_metadata(docs, metadata_filter[0], metadata_filter[1])
    chunks = [doc.page_content for doc in docs]
    return chunks
# Constants
vector_store_address: str = "VECTOR-STORE-ADDRESS"
vector_store_password: str = "VECTOR-STORE-PASSWORD"
embeddings = VertexAIEmbeddings(model_name='textembedding-gecko@001', chunk_size=1)
index_name = "clindex"
def main_function(input_request: dict):
    input_message = input_request.get('input_message')
    if input_message is None or input_message == "":
        raise ValueError("Input message is empty")
```

```
    model = VertexAI(model_name="text-bison")
    # init vector store
    vector_store = AzureSearch(
        azure_search_endpoint=vector_store_address,
        azure_search_key=vector_store_password,
        index_name=index_name,
        embedding_function=embeddings.embed_query,
    )
    chunks = search_related_chunks(input_message, vector_store)
    final_template_prompt = PromptTemplate.from_template(final_template)
    chain_final_message = final_template_prompt | model
    answer = chain_final_message.invoke({'chunks':"\n\n\n".join(chunks),
'question':input_message})
    return answer
from azure.core.credentials import AzureKeyCredential
from azure.search.documents import SearchClient
service_endpoint = os.environ["AZURE_SEARCH_SERVICE_ENDPOINT"]
index_name = os.environ["AZURE_SEARCH_INDEX_NAME"]
key = os.environ["AZURE_SEARCH_API_KEY"]
search_client = SearchClient(service_endpoint, index_name, AzureKeyCredential(key))
results = search_client.search(search_text="What are the majors at your university?")
```