НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Автоматизація перевірки тексту з використанням методів обробки природньої мови»

Виконав:
студент IV курсу, групи ІС-93
Горошко Олександр Антонович _____

Керівник:
доцент Валерій Завгородній _____

Рецензент:
завідувач кафедри системного проектування, _____
доктор технічних наук, професор, Мухін В.Є.

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.

Студент 

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 126 «Інформаційні системи та технології»
Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту
Горошку Олександру Антоновичу

1. Тема проєкту «Автоматизація перевірки тексту з використанням методів обробки природньої мови», керівник проєкту Завгородній В.В., доцент, доктор технічних наук, професор.
.затверджені наказом по університету від 31 травня 2023 №2101-с
2. Термін подання студентом проєкту 12 червня 2023 року
3. Вихідні дані до проєкту: Категорія текстового документу

4. Зміст пояснювальної записки

1. Аналіз існуючих рішень: огляд існуючих навігаційних систем, їх роботи, функціоналу та технічного забезпечення

2. Структура системи: опис мови програмування, інструментів та бібліотек, які було використано для виконання проєкту

3. Реалізація системи: постановка задачі, аналіз популярних алгоритмів для вирішення питання, обґрунтування методів розв'язку системи.

4. Технологічний розділ: детальний покроковий опис роботи з програмою та випробування системи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плака-тів, презентацій тощо)
*1. Схема конвеєру NLP*_____
*2. Схема ансамблевого алгоритму*_____
*3. Діаграма компонентів*_____
*4. Діаграма прецендентів*_____

6. Дата видачі завдання 23 лютого 2023 року

Календарний план

	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Визначення теми дипломного проєкту та погодження з дипломним керівником*	*23.02.2023*	
2.	*Опис предметного середовища, процесу діяльності, функціональної моделі*	*2.03.2023*	
3.	*Опис постановки задачі. Визначення задач та цілей розробки*	*7.03.2023*	
4.	*Аналіз наявних аналогів*	*21.03.2023*	
5.	*Опис математичного забезпечення*	*11.04.2023*	
6.	*Опис програмного та технічного забезпечення*	*28.04.2023*	
7.	*Розробка ПЗ*	*21.05.2023*	
8.	*Оформлення пояснювальної записки*	*29.05.2023*	

Студент Олександр ГОРОШКО
Керівник проєкту Валерій ЗАВГОРОДНІЙ

АНОТАЦІЯ

Горошко О. А. Автоматизація перевірки тексту з використанням методів обробки природньої мови. КПІ ім. Ігоря Сікорського, Київ 2023.

Проєкт містить 61 с., 10 рисунків, 12 таблиць, посилання на 15 літературні джерела.

Ключові слова: автоматизація, природна мова, машинне навчання, категоризація, Python.

Об'єктом дослідження є система автоматизації перевірки тексту з використанням методів обробки природньої мови.

Предметом дослідження є методи та алгоритми машинного навчання, які використовуються для задач багатокласової класифікації .

Для реалізації системи було використано мову програмування Python.

У дипломному проєкті було розроблено систему категоризації, яка за допомогою ансамблевого алгоритму машинного навчання здатна відносити попередньо невідомий текст, написаний природною мовою, до однієї з 5 попередньо обраних категорій.

ANNOTATION

Horoshko O. A. Automation of text verification using natural language processing methods. KPI named after Igor Sikorskyi, Kyiv 2023.

The project contains 61 pages, 10 figures, 12 tables, references to 15 literary sources.

Keywords: automation, natural language, machine learning, categorization, Python.

The object of the research is a text verification automation system using natural language processing methods.

The subject of the study is the methods and algorithms of machine learning, which are used for multi-class classification problems.

The Python programming language was used to implement the system.

In the diploma project, a categorization system was developed, which, with the help of an ensemble machine learning algorithm, is able to assign previously unknown text written in natural language to one of 5 pre-selected categories.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІС93.180БАК.004 ПЗ	Пояснювальна записка	61		
6	А3	ІС93.180БАК.004 Д1	Схема конвеєру NLP	1		
7	А3	ІС93.180БАК.004 Д2	Схема роботи ансамблевого	1		
8			алгоритму			
9	А3	ІС93.180БАК.004 Д3	Діаграма компонентів	1		
10	А3	ІС93.180БАК.004 Д4	Діаграма прецендентів	1		
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	ІС93.180БАК.004 ТП		
Розроб.		Горошко О. А.			Автоматизація перевірки тексту з використанням методів обробки природньої мови. Відомість проєкту	Літ.	Аркуш 1 / Аркушів 1
Керівн.		Завгородній В.В				Т	
						КПІ ім. Ігоря Сікорського Група ІС-93	
Затв.							

Пояснювальна записка
до дипломного проєкту
на тему: «Автоматизація перевірки тексту з використанням методів обробки природньої мови»

Київ – 2023 року

ЗМІСТ

ВСТУП

У сучасному інформаційному суспільстві, де величезні обсяги текстових даних генеруються кожну секунду, ефективна обробка та аналіз тексту стають дедалі важливішими завданнями. Категоризація тексту - одне з фундаментальних завдань у галузі обробки природних мов (Natural Language Processing, NLP), яке націлене на автоматичне присвоєння текстів певним категоріям чи темам. Категоризація тексту має широкі застосування у різних сферах, включаючи інформаційний пошук, фільтрацію контенту, аналіз соціальних медіа, машинне навчання та багато інших. Вона дозволяє ефективно структурувати, класифікувати та витягувати інформацію з великих обсягів текстових даних. В останні десятиліття методи обробки природних мов (NLP) досягли значного прогресу, відкриваючи нові можливості автоматичної категоризації тексту. Такі методи включають токенізацію, лематизацію, стемінг, векторне подання тексту, аналіз тональності, вилучення ключових слів та інші.

Метою даної дипломної роботи є дослідження та розробка системи категоризації тексту англійською мовою з використанням методів обробки природних мов. Основним завданням роботи є практичне застосування алгоритмів і моделей машинного навчання, здатних автоматично класифікувати тексти на певні категорії.

Не менш важливим завданням роботи є розгортання системи у реальних умовах та надання до неї доступу кожному охочому у мережі Інтернет. Це можливо завдяки реалізації програми у форматі бота на платформі одного відомого месенджеру.

У сучасному інформаційному світі, де цифрова трансформація проникає у всі сфери нашого життя, важливість ботів стає дедалі помітнішою. Боти, або чат-боти, є програмними агентами, здатними взаємодіяти з людьми через різні канали комунікації, включаючи месенджери, веб-сайти та голосові помічники.

Однією з ключових причин, через які боти стали невід'ємною частиною нашого інформаційного світу, є їхня здатність забезпечувати швидке та ефективне спілкування з користувачами. Це суттєво заощаджує час та зусилля користувачів, дозволяючи їм отримувати потрібну інформацію або виконувати завдання безпосередньо через чат-інтерфейс.

В рамках роботи буде проведено огляд існуючих методів та технологій у галузі категоризації тексту, а також буде розроблено та реалізовано прототип системи, заснований на вибраних методах обробки природних мов. Потім прототип буде протестовано та оцінено на реальних текстових даних для перевірки його ефективності та точності.

Надалі, на основі результатів даної роботи, можна буде проводити подальші дослідження та розробки в галузі NLP, спрямовані на покращення точності та ефективності систем категоризації тексту.

Загалом дана дипломна робота має актуальність і важливість у контексті сучасного інформаційного суспільства, пропонуючи підходи та рішення для завдання категоризації тексту із застосуванням методів обробки природних мов.

Дипломний проєкт складається з наступних розділів: вступ, основна частина, висновки, список використаних джерел із 15 найменувань. Загальний обсяг 61 сторінка.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Огляд існуючих аналогічних рішень.

Загалом, на ринку існує чимало інструментів для категоризації текстових даних. Більшість з них реалізовані у вигляді веб-застосувань та різноманітних API. Також існує велика кількість готових бібліотек для різних мов програмування з попередньо навченими моделями, що дозволяє не витрачати час на конструювання власних моделей, пошук даних для навчання та самого навчання моделей.

Далі детально наведені приклади кількох таких інструментів.

1.1.1 TextRazor.

Даний частково безкоштовний веб-сервіс пропонує широкий перелік функцій щодо аналізу текстової інформації. Насамперед, це знаходження ймовірностей присутності категорій у документі, а також виконання усіх основних задач NLP, таких як токенізація, стемінг, лематизація, тегінг, побудова дерева залежностей тощо. Однак доступна лише англійська мова.

Також пропонується REST API та набір бібліотек для багатьох мов програмування.

На рисунку наведено результат аналізу документу:

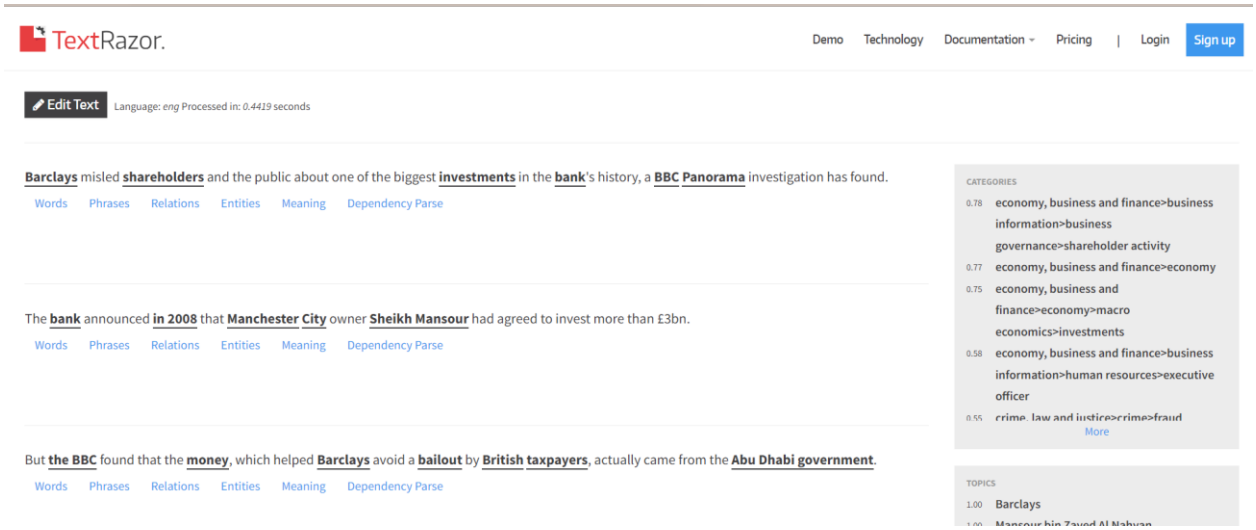

Рисунок 1.1 – Результат аналізу текстового документу сервісом TextRazor

Як можна побачити з рисунку, сервіс надає дуже багато функцій з обробки природньої мови, але нас цікавить саме категоризація. Сервіс надає певну ймовірність того, що документ належить до однієї з дискретних категорій.

1.1.2 uClassify

Цей веб-сервіс має мінімалістичний інтерфейс та здатен на знаходження ймовірністного розподілу 10 категорій. Доступні категорії: мистецтво, комп'ютери, ігри, здоров'я, наука, бізнес, спорт, суспільство.

Сервіс підтримує такі мови: англійська, французька, іспанська, шведська.

Також пропонується REST API у форматі XML. Окрім цього надається функція перекладу.

На рисунку наведено результат аналізу тексту:

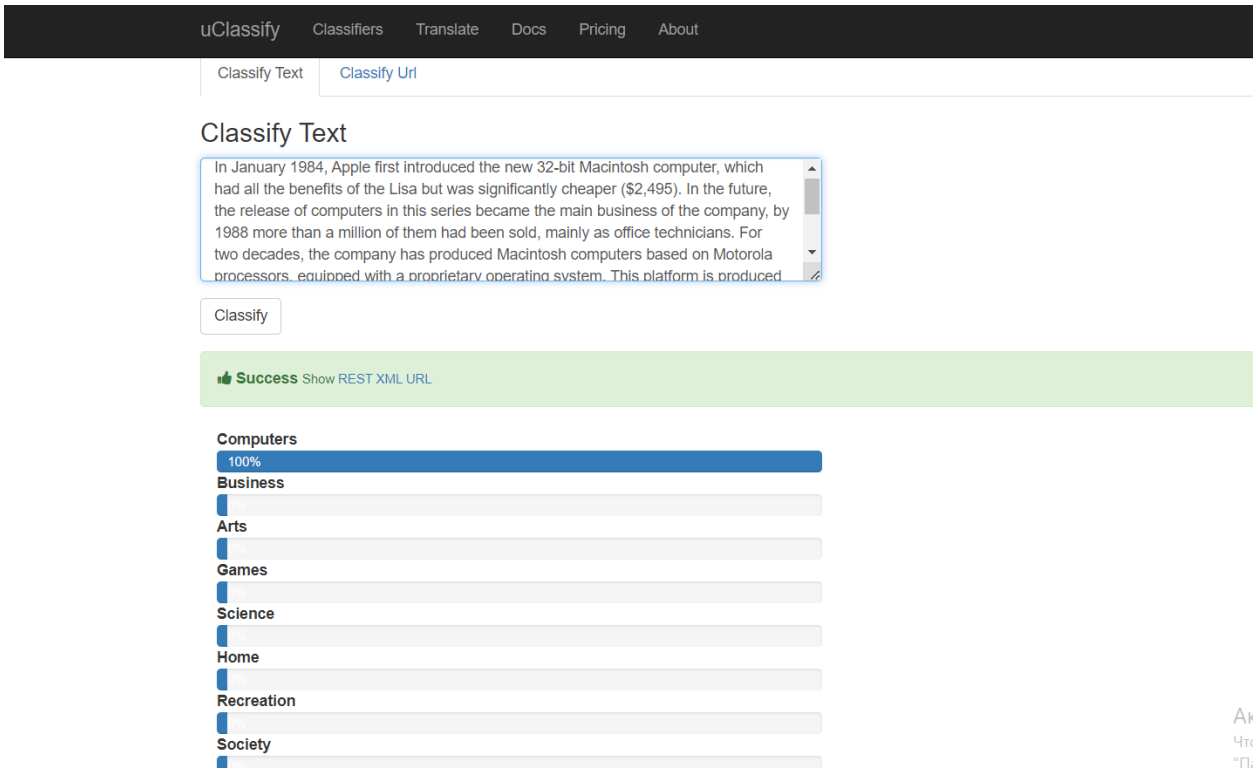

Рисунок 1.2 – Результат аналізу текстового документу сервісом uClassify

1.1.3 Natural Language AI

Дана служба платформи Google Cloud NLP надає широкий спектр різноманітних функцій по обробці текстових даних. Однією з них є якраз класифікація на основі змісту(Content Classification), тобто та сама категоризація.

Для використання сервісу потрібно мати Google акаунт та підписку на хмарній платформі Google Cloud.

Ця функція дозволяє визначити категорію чи клас, до якого належить текстовий документ. Google Cloud NLP надає заздалегідь навчені моделі, які можуть класифікувати текст у широкому діапазоні категорій, таких як новини, спорт, фінанси, здоров'я та інші. Можливо використовувати ці моделі або навчити власну модель, якщо є розмічений набір даних. Підтримується 11 мов.

На рисунку наведено результат аналізу тексту:

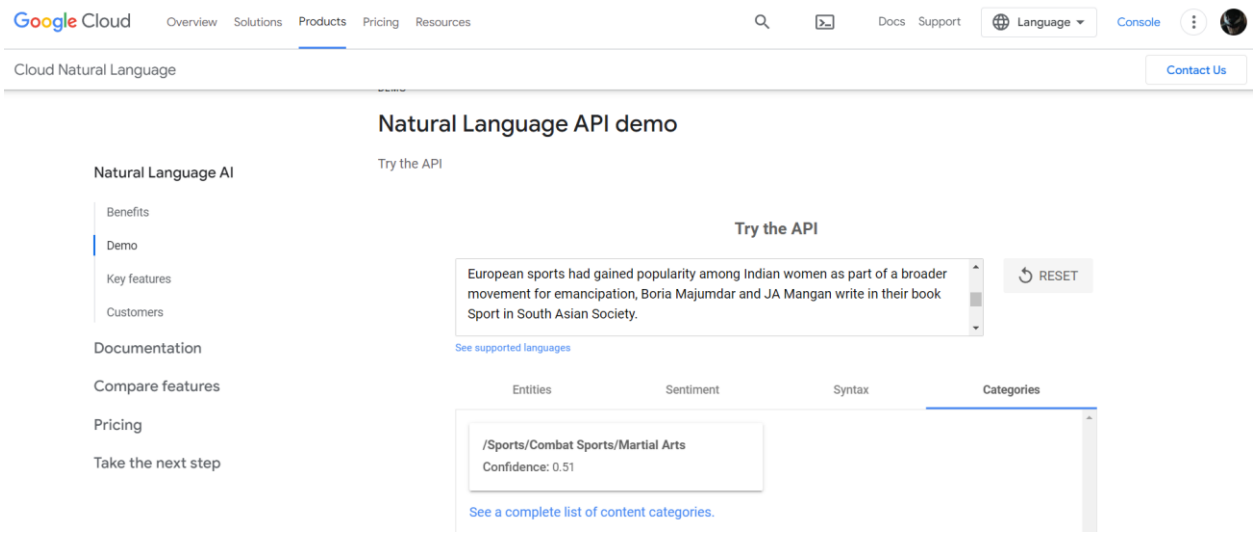

Рисунок 1.3 – Результат аналізу текстового документу сервісом Natural Language AI від Google Cloud NLP

1.1.4 Amazon Comprehend

Ця служба від провайдера хмарних послуг Amazon Web Services надає широкий спектр можливостей щодо класифікації текстових даних. Серед головних особливостей можна виділити наступні:

1. Автоматична класифікація.

Amazon Comprehend дозволяє автоматично класифікувати тексти на основі попередньо вивчених моделей. Він надає набір заздалегідь визначених категорій, які тексти можуть бути класифіковані. Це може бути корисно, наприклад, для класифікації відгуків за тональністю або статей новин за тематиками.

2. Створення власних моделей.

Також можливо створювати власні моделі класифікації тексту, навчаючи їх на власних даних. Amazon Comprehend надає можливість завантажити розмічені дані та навчити модель на основі цих даних. Це дозволяє створювати моделі, специфічні для певної предметної галузі чи завдання класифікації.

3. Оцінка точності моделі.

Amazon Comprehend надає метрики оцінки якості класифікації, такі як точність, повнота та F1 міра. Це допомагає оцінити продуктивність моделі та визначити, наскільки добре вона працює на ваших даних.

4. Багатокласова класифікація.

Amazon Comprehend підтримує багатокласову класифікацію тексту, що означає, що тексти можуть бути віднесені до кількох категорій одночасно. Це корисно, коли вам потрібно класифікувати тексти на кілька різних категорій або тематик.

5. Підтримка багатьох мов.

Amazon Comprehend підтримує класифікацію тексту різними мовами, що робить його корисним інструментом для роботи з багатомовними текстовими даними.

2 СТРУКТУРА СИСТЕМИ

2.1 Структура системи

Система являє собою програму-бота, реалізовану за допомогою інтерфейсу ботів у месенджері Telegram. Система отримує вхідні дані від користувача або у вигляді безпосередньо тексту, або як текстовий файл та відносить зміст тексту до однієї з категорій з певною ймовірністю або виконує ймовірнісний розподіл усіх категорій, виконуючи таким чином задачу багатокласової класифікації.

Система складається з трьох підсистем.

Перша підсистема це інтерфейс бота, що оброблює вхідні дані та виконує основну логіку взаємодії з користувачем.

Друга підсистема це система категоризації текстових даних, що здатна відносити вхідний текст до певної категорії. Система являє собою по суті навчені моделі машинного навчання, що викликаються першою підсистемою.

Третя підсистема являє собою так званий конвеєр NLP.

Конвеєр NLP - це послідовність кроків обробки початкових текстових даних з метою перетворення їх у вигляд, придатний до використання моделлю машинного навчання.

Першим кроком конвеєру NLP у даному проєкті є визначення мови вхідного тексту. Так як це сама по собі складна задача машинного навчання та вона виходить за рамки дипломного проєкту, то для цього застосовується сторонній інструмент.

Схематично конвеєр NLP у даному проєкті можна представити наступним чином:

Рисунок 2.1 – Схема конвеєру NLP

Після того як конвеєр підготує початкові дані, вони подаються на вхід моделі машинного навчання, яка вже і видає кінцеву відповідь.

Виконується система категоризації текстових даних як застосунок-бот на платформі ботів месенджеру Telegram.

Дуже спрощено схему функціонування боту можна представити наступним чином:

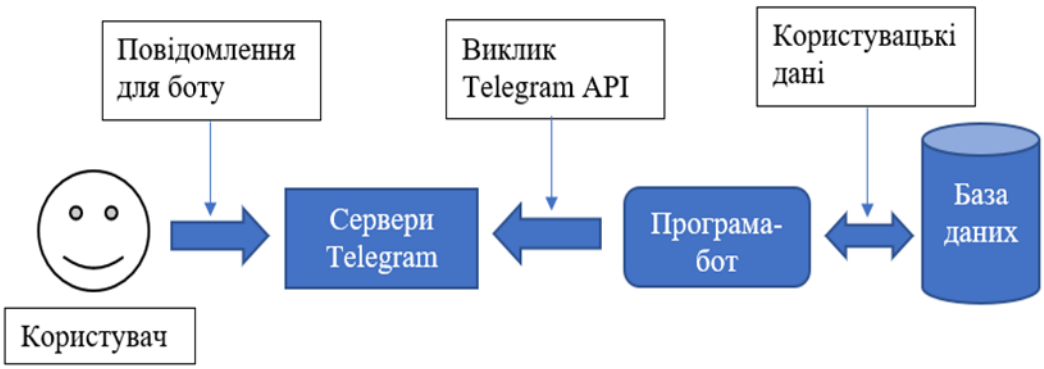

Рисунок 2.2 – Схема функціонування боту

Як можна побачити з рисунку, структура системи є доволі тривіальною та зрозумілою будь-кому. Користувач через месенджер Telegram надсилає дані спочатку на сервери Telegram, що оброблюють їх та зберігають протягом певного часу. Після цього програма-бот надсилає запит. Зберігання користувацьких даними програма-бот здійснює за допомогою СУБД.

Розгортається система із застосуванням технології контейнеризації.

Контейнер програми-бота є ізольованим і автономним середовищем, в якому запускається і функціонує бот. Контейнер забезпечує упаковку бота і всіх його залежностей, включаючи код програми, бібліотеки, файли конфігурації та інші ресурси, в єдину сутність.

Середу виконання контейнерів системи надає провайдер хмарних послуг, що відповідає за управління інфраструктурою.

Для зберігання даних системи використовується документна СУБД, що відмінно підходить для зберігання неструктурованих даних та є швидкою у роботі та використанні.

Створення системи у форматі бота було обрано через те, що Telegram надає зручне API та інструменти, які роблять процес створення бота відносно простим та швидким. Більш того, інтерфейс бота легкий у використанні та доступний з будь-якої платформи.

2.2 Вибір мови програмування для створення системи

Для створення системи було обрано скриптову мову програмування Python. Python є однією з найбільш популярних та широко використовуваних мов програмування для завдань обробки природної мови (NLP) з кількох причин:

1. Багате різноманіття спеціалізованих бібліотек.

Python пропонує безліч потужних бібліотек та інструментів, спеціально розроблених для обробки та аналізу текстових даних. Наприклад, бібліотеки NLTK (Natural Language Toolkit), SpaCy, Gensim, scikit-learn та TensorFlow надають широкий спектр функцій для токенізації, лематизації, стеммінгу, отримання ознак, класифікації, кластеризації та інших задач NLP.

2. Простота використання.

Python має простий і зрозумілий синтаксис, який робить його дуже доступним для швидкого створення прототипів та розробки рішень для завдань NLP без зайвої складності.

3. Велика спільнота.

Python має величезну та активну спільноту розробників, яка створює та підтримує безліч бібліотек та інструментів для NLP. Це означає, що можна легко знайти відповіді на свої питання, обмінятися досвідом з іншими розробниками і отримати підтримку у вирішенні проблем, що виникають.

4. Інтеграція з іншими інструментами.

Python легко інтегрується з іншими інструментами та технологіями, такими як бази даних, веб-фреймворки та системи машинного навчання. Це дозволяє створювати повноцінні програми для обробки та аналізу текстових даних з використанням широкого спектру інструментів.

5. Машинне навчання та глибоке навчання.

Python є основною мовою програмування для машинного навчання та глибокого навчання. Бібліотеки, такі як TensorFlow, Keras, PyTorch та scikit-learn, надають найсучасніші інструменти для побудови та навчання моделей NLP, включаючи рекурентні нейронні мережі, згорткові нейронні мережі та трансформери.

2.3 Обгрунтування вибору бібліотек та інструментів розробки

Для побудови даної системи було обрано певну кількість бібліотек мови програмування Python:

1. Бібліотека для взаємодії з API Telegram pyTelegramBotAPI.

Дана бібліотека надає зручний інтерфейс для розробки ботів з використанням Telegram Bot API. Ось деякі переваги використання zyTelegramBotAPI для розробки ботів:

а) Простота використання.

PyTelegramBotAPI надає простий та інтуїтивно зрозумілий інтерфейс, що робить розробку телеграм-ботів більш доступною для розробників.

б) Багатий функціонал.

Бібліотека надає широкий спектр функцій та можливостей Telegram Bot API, включаючи відправлення та отримання повідомлень, обробку команд, управління клавіатурами, роботу з медіафайлами, адміністрування груп та багато іншого. Це дозволяє розробникам створювати потужні та функціональні телеграм-боти.

в) Гнучкість та розширюваність.

PyTelegramBotAPI дозволяє розробникам гнучко налаштовувати та розширювати функціональність своїх роботів. Бібліотека забезпечує можливість обробки різних типів повідомлень, використання inline-режиму, створення кастомних обробників та додаткових функцій для взаємодії з користувачами.

г) Підтримка асинхронності.

PyTelegramBotAPI підтримує асинхронну обробку запитів, що дозволяє створювати ефективні та надійні програми, здатні обробляти безліч запитів одночасно.

2. Бібліотека машинного навчання Scikit-Learn.

Дана бібліотека надає широкий спектр алгоритмів та інструментів для обробки даних та побудови моделей машинного навчання. У цій бібліотеці містяться реалізації усіх алгоритмів машинного навчання, потрібних для вирішення поставленої задачі.

Серед інших причин вибору бібліотеки можна виділити наступні: простота використання, вбудовані інструменти попередньої обробки тексту, розширюваність.

3. Бібліотека машинного навчання Keras.

Дана бібліотека пропонує високорівневі абстракції та зручні функції, які спрощують створення та навчання нейронних мереж. Вона розроблена з урахуванням модульності, що дозволяє легко створювати та комбінувати різні шари, активаційні функції, оптимізатори та функції втрат. Це дає можливість гнучкого визначення архітектури нейронної мережі та її налаштування під конкретне завдання. Вона також включає в себе різні інструменти для попередньої обробки даних, візуалізації результатів і оцінки моделей.

- Бібліотека машинного навчання NLTK.

Дана бібліотека надає широкий спектр функцій попередньої обробки тексту. Зокрема, у цьому проєкті використовується список стоп-слів англійської мови та функція токенізації тексту з цієї бібліотеки.

4. Бібліотека машинного навчання Gensim.

Дана бібліотека надає потужні інструменти для роботи з текстовими даними, включаючи методи векторизації тексту. У даному проєкті використовується модель Word2Vec з цієї бібліотеки. Ця модель дозволяє перетворити слова на вектори фіксованої довжини для застосування їх у нейромережі.

5. Бібліотека драйверу документної СУБД MongoDB.

Дана бібліотека потрібна для встановлення зв’язку з документною СУБД.

Для розробки програмного коду були використані наступні інструменти:

1. Середовище розробки Visual Studio Code.

Є популярним інтегрованим середовищем розробки (IDE), яке чудово підходить для створення різноманітних проєктів завдяки по перше легкості та швидкості. Воно швидко запускається, швидко відкриває та обробляє файли, а також має низьке споживання ресурсів комп'ютера.VS Code має потужний редактор коду з підсвічуванням синтаксису для різних мов програмування, автозаповнення, відладчика та інших корисних функцій. Підтримує безліч розширень, дозволяючи настроїти середовище розробки під свої потреби.

VS Code має нативну підтримку інтеграції із системою контролю версій Git.

VS Code пропонує широкий вибір розширень, які дають змогу розширити функціональність IDE під різні потреби. Існують розширення для розробки різними мовами, інструменти для роботи з базами даних, засоби налагодження та багато іншого.

2. Система контролю версій коду GIT.

В цілому, Git є невід'ємною частиною сучасної розробки програмного забезпечення, забезпечуючи ефективне керування версіями програмного коду. Вона дозволяє відстежувати зміни у коді та файловій структурі проєкту. Завдяки цьому можливо зберігати різні версії файлів та повертатися до попередніх версій за потреби. Це значно спрощує та пришвидшує процес розробки та усунення помилок.

3. Документоорієнтована СУБД MongoDB.

Ця СУБД була обрана для проєкту завдяки наступним перевагам: гнучкі схеми даних, масштабованість, висока доступність, підтримка запитів та індексів, розширені можливості та легкість використання.

4. Платформа для управління ізольованими контейнерами Docker.

Docker для даного проєкту було обрано як промисловий стандарт середи виконання контейнерів.

5. Провайдер хмарних послуг Microsoft Azure.

Даний провайдер використовується у проєкті для розміщення контейнеру із застосунком на віртуальній інфраструктурі. Це має наступні переваги:

- Azure дозволяє гнучко масштабувати інфраструктуру під потреби. Можлиов масштабувати контейнери горизонтально або вертикально, щоб забезпечити потрібну продуктивність та ресурси.

- Azure забезпечує високу доступність та відмовостійкість для контейнерів. Можливо використовувати функції автомасштабування та групи доступності для забезпечення безперервної роботи контейнеру.

Висновок по розділу

У даному розділі було розглянуто систему зі структурної точки зору та надано аргументи щодо вибору мови програмування та інструментів розробки програмного коду.

3 РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Змістовна постановка задачі

Суть задачі полягає у створенні системи класифікації текстових документів на заздалегідь визначені категорії та її розгортанні у реальному середовищі виконання з вільним доступом до нього з мережі Інтернет.

На початку потрібно створити підсистеми обробки текстових даних та їх категоризації. Процес створення включає наступні кроки:

1. Визначення категорій.

Спочатку потрібно визначити список категорій або тем, на які потрібно класифікувати тексти. Наприклад, це можуть бути категорії, пов'язані з новинами (політика, спорт, наука і т.д.), областю бізнесу (фінанси, маркетинг, виробництво і т.д.) або будь-які інші тематики. У даній дипломній роботі було обрано 5 наступних категорій: бізнес, розваги, політика, спорт, технології.

2. Збір та попередня обробка даних.

Наступним кроком є збір достатньої кількості текстових документів, що належать до кожної категорії. Важливо забезпечити різноманітність текстів у кожній категорії для більш точної класифікації.

Для навчання моделей було обрано загальнодоступний датасет BBC News Archive. Датасет BBC News Archive є набором новостних статей, зібраних з різних розділів веб-сайту BBC News. Він містить новини, що охоплюють різні теми, такі як політика, економіка, наука, спорт і т.д. Датасет складається з текстових документів, кожен із яких містить заголовок та зміст статті. Датасет містить 2225 статей.

Після збору даних проводиться їх попередня обробка, що являє собою конвеєр NLP, що був розглянутий раніше на рисунку 2.1. Більш детально кожен етап конвеєру описаний далі.

Етапи конвеєру NLP для задачі попередньої обробки:

а) Визначення мови тексту.

Визначення мови тексту виконується за допомогою за допомогою Cognitive Services API, зокрема сервісу Text Analytics провайдера хмарних послуг Microsoft Azure. Цей сервіс надає можливість автоматичного визначення мови тексту на основі його вмісту.

Процес визначення мови тексту включає такі кроки:

1. Створення ресурсу та отримання ключа API.

Для використання сервісу Text Analytics необхідно створити ресурс у Microsoft Azure і отримати ключ API, який буде використовуватися для аутентифікації в сервісі.

2. Формування запиту.

HTTP-запит до API сервісу Text Analytics виконується за допомогою API бібліотеки для мови програмування Python, вказуючи текст запиту, для якого потрібно визначити мову.

3. Надсилання запиту та отримання відповіді.

Запит надсилається до API сервісу, використовуючи ключ API для аутентифікації. У відповідь отримується відповідь JSON, що містить результат визначення мови тексту.

4. Обробка результату.

У JSON-відповіді вказується можлива мову тексту, а також оцінка достовірності цього визначення.

б) Токенізація тексту.

Токенізація тексту - це процес поділу тексту на окремі частини, які називають токенами. Токени можуть бути словами, символами, фразами або будь-якими іншими елементами тексту, які ми хочемо розглядати як одиниці обробки. У даному конвеєрі виконується токенізація на рівні окремих слів. Виконується у даному проекті за допомогою бібліотеки NLTK.

в) Видалення стоп-слів та пунктуації.

Хоча видалення стоп-слів та пунктуації не завжди є обов'язковим або необхідним для кожного завдання класифікації тексту, в більшості випадків це

корисний крок, який допомагає покращити якість моделі та спростити аналіз текстових даних.

Стоп-слова - це поширені слова, які зазвичай не несуть смислового навантаження і часто зустрічаються в текстах. Ці слова можуть бути виключені з аналізу при обробці тексту, щоб зменшити розмірність даних та зосередитись на більш інформативних словах. Далі наведений список найбільш поширених стоп-слів в англійській мові: a, and, the, in, on, is, of, to, for, with, that, it, as, at, by, this, from, or, but.

Для даної роботи був обраний список стоп-слів, що надається бібліотекою NLTK та містить наступні слова:

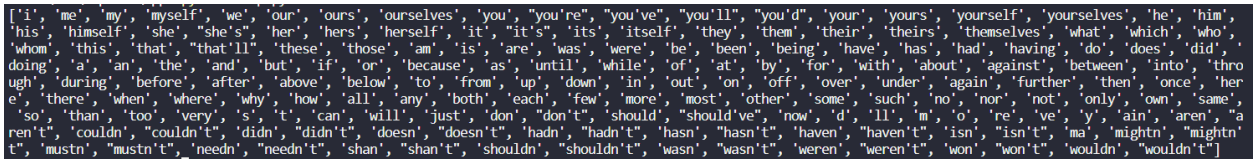

Рисунок 3.1 – Стоп-слова англійської мови у бібліотеці nltk

г) Векторизація текстових даних.

Далі текстові документи необхідно перетворити на числові вектори, щоб їх можна було використовувати моделями машинного навчання. Це може бути виконано за допомогою методів, таких як мішок слів (Bag of Words), TF-IDF (Term Frequency-Inverse Document Frequency) або складніших підходів, таких як Word2Vec або BERT для отримання більш семантичних векторних уявлень.

У даній роботі було застосовано реалізацію TF-IDF із бібліотеки scikit-learn. Окрім самої векторизації текстових даних, дана реалізація виконує ще низку етапів попередньої обробки даних.

3. Навчання моделі класифікації.

Після векторизації текстових даних модель машинного навчання навчається на розміченому навчальному наборі даних. Можливими моделями можуть бути такі алгоритми, як метод опорних векторів (SVM), випадковий ліс (Random Forest), метод k найближчих сусідів(KNN), нейронні мережі та інші.

4. Оцінка та налаштування моделі.

Після навчання моделі необхідно оцінити її продуктивність з використанням відкладеного тестового набору даних або з використанням крос-валідації. Якщо потрібно, можна налаштувати параметри моделі або спробувати різні архітектури для досягнення кращої продуктивності. Для того щоб краще зрозуміти отриману модель

5. Застосування моделі у реальних умовах.

Після успішного навчання та оцінки моделі її можна використовувати для класифікації нових, невідомих раніше текстових документів, прогнозуючи їхню приналежність до певних категорій.

Так як система надається користувачеві через інтерфейс бота на платформі месенджеру Telegram, потрібно створити цей інтерфейс. Цей процес складається з наступних кроків:

1. Реєстрація бота.

Спочатку необхідно зареєструвати бота в Telegram. Це робиться шляхом створення нового бота через спеціальний бот-помічник в Telegram, званий BotFather. В результаті реєстрації отримується унікальний токен, який використовуватиметься для взаємодії з API Telegram.

2. Налаштування обробників.

Після реєстрації бота необхідно налаштувати обробники для різних команд чи дій, які має виконувати бот. Обробники визначаються на основі вхідних повідомлень від користувачів та певних шаблонів чи умов. Наприклад, можна налаштувати обробник для команди /start або для певного текстового повідомлення.

3. Взаємодія з API Telegram.

Для взаємодії з API Telegram використовується відповідна бібліотека, яка підтримує Telegram Bot API. Бібліотека, така як pyTelegramBotAPI для Python надає зручні методи та функції для надсилання та отримання

повідомлень, управління клавіатурами, завантаження медіафайлів та інших операцій.

4. Обробка вхідних повідомлень.

Коли бот отримує повідомлення, він обробляє його за допомогою налаштованих обробників. Залежно від вмісту повідомлення та налаштувань обробників, бот може виконати певні дії, відправити повідомлення, запитати додаткову інформацію або виконати інші завдання.

Отже, принцип роботи телеграм-бота полягає в отриманні вхідних повідомлень від користувачів, їх обробці за допомогою налаштованих обробників та надсилання відповідних відповідей або виконання завдань.

Завдяки цьому принципу боти можуть виконувати різні функції, взаємодіяти з користувачами і автоматизувати процеси в Telegram.

Якщо боту потрібно зберігати або обробляти дані, наприклад, для створення профілів або збереження налаштувань, він може використовувати базу даних або інші методи зберігання даних. Сервіс бази даних повинен надаватись провайдером хмарних послуг, що бере на себе увесь контроль над інфраструктурою та надає розробнику доступ лише

Далі виконується процес упаковки програми та всіх її залежностей у контейнер.

Контейнери надають середовище виконання, в якому програма може працювати незалежно від своєї хостової операційної системи та інших додатків.

Основна ідея контейнеризації полягає в тому, щоб упакувати додаток та всі його залежності разом, створивши ізольоване та повторюване оточення, яке можна легко перенести та розгорнути на різних платформах та системах.

Контейнери забезпечують середовище, що містить всі необхідні компоненти для роботи програми, включаючи код, бібліотеки, конфігураційні файли та інші залежності. Контейнери можуть бути запущені на будь-якій

підтримуваній платформі, будь то фізичний сервер, віртуальна машина або хмарна інфраструктура.

Після цього готовий контейнер має бути розгорнутий на сервері або хостингу, щоб бути доступним для користувачів. Контейнер з ботом може працювати постійно, відповідаючи на вхідні повідомлення в реальному часі або періодично перевіряти нові повідомлення за допомогою webhook-сервера або інших методів.

3.2 Математична постановка задачі

З математичної точки зору сутність підсистеми категоризації текстових документів можна сформулювати наступним чином:

Нехай ми маємо набір текстових документів, що складаються з колекції текстових даних. Припустимо, що кожен документ має мітку чи категорію, до якої він належить. Наша мета полягає в тому, щоб розробити модель машинного навчання, яка може класифікувати нові документи на основі їхнього вмісту та присвоїти їм відповідну мітку чи категорію.

Формально, нехай у нас є такі позначення:

- Множина документів: D = {d1, d2, ..., dn}, де кожен документ di представлений як текстового представлення.

- Множина категорій: C = {c1, c2, ..., ck}, де кожна категорія ci є міткою або класом, до якого може бути віднесений документ.

- Множина навчальних прикладів: {(d1, c1), (d2, c2), ..., (dn, cn)}, де кожен приклад є парою документа та відповідної йому категорії.

Завдання класифікації текстових документів полягає в тому, щоб навчити модель машинного навчання на основі навчального набору даних, щоб вона могла приймати нові документи та надавати їм найбільш відповідну категорію. Математично це може бути сформульовано як завдання навчання з учителем, де ми прагнемо знайти функцію класифікації f: D → C, яка

відображає документи на їх відповідні категорії. Модель машинного навчання навчається на основі навчального набору даних, мінімізуючи помилку класифікації та знаходячи оптимальні параметри чи ваги моделі.

Перед навчанням моделі вхідні текстові дані проходять процес векторизації. Цей процес полягає у перетворенні текстових даних на числові вектори, які математична модель здатна зрозуміти та обробити.

Важливим етапом створення моделей машинного навчання є оптимізація гіперпараметрів моделі. Гіперпараметри - це параметри моделі, які оптимізуються у процесі навчання, а вибираються перед початом навчання. Оптимальний вибір гіперпараметрів може значно вплинути на якість та продуктивність моделі.

Ось кілька причин, чому оптимізація гіперпараметрів важлива:

1. Поліпшення продуктивності.

Підбір оптимальних значень гіперпараметрів дозволяє досягти кращої продуктивності моделі. Це може виявлятися у покращенні точності передбачень, зниженні перенавчання, прискоренні навчання моделі та інших аспектах продуктивності.

2. Підвищення узагальнюючої здатності.

Оптимізація гіперпараметрів допомагає знайти баланс між простотою моделі та її здатністю узагальнювати нові дані. Надмірна складність моделі може призвести до перенавчання, тоді як недостатньо складна модель може мати низьку точність. Оптимізація гіперпараметрів дозволяє налаштувати модель таким чином, щоб вона була здатна узагальнювати нові дані.

3. Економія часу та ресурсів.

Оптимізація гіперпараметрів дозволяє знайти оптимальні значення з мінімальними витратами часу та обчислювальних ресурсів. Це дозволяє скоротити час навчання моделі та зменшити необхідність повторних експериментів з різними значеннями гіперпараметрів.

4. Врахування особливостей даних.

Різні набори даних можуть вимагати різних значень гіперпараметрів для досягнення оптимальних результатів. Оптимізація гіперпараметрів дозволяє врахувати особливості конкретного набору даних та налаштувати модель відповідним чином.

5. Поліпшення інтерпретації.

Оптимізація гіперпараметрів може допомогти знайти такі значення, які роблять модель більш інтерпретованою. Наприклад, вибір правильного значення коефіцієнта регуляризації або числа прихованих вузлів у нейронній мережі може зробити модель більш зрозумілою та інтерпретованою.

Загалом оптимізація гіперпараметрів є важливим кроком у побудові моделі машинного навчання, який допомагає досягти кращої продуктивності, покращити узагальнюючу здатність, економити ресурси та враховувати особливості даних.

Для того, щоб оцінити навчені моделі машинного навчання, застосовуються різноманітні метрики. Для випадку багатокласової класифікації це наступні:

1. Матриця помилок (Confusion Matrix).

Матриця помилок є основою оцінки моделі багатокласової класифікації. Вона являє собою квадратну матрицю, де кожна клітинка є кількістю зразків, які були класифіковані у відповідний клас. Вона допомагає візуалізувати результати класифікації та обчислити інші метрики.

2. Влучність (Precision).

Влучність вимірює, наскільки влучно модель передбачає конкретний клас. Вона обчислюється як відношення числа правильно класифікованих зразків даного класу до загальної кількості зразків, передбачених для цього класу.

Формула:

$$Accuracy(a) = \frac{\sum_{i=1}^{N} \mathbb{I}[a(x_i) = y_i]}{N} = \frac{TP + TN}{TP + TN + FP + FN}$$

3. Повнота (Recall).

Повнота вимірює, як добре модель виявляє всі зразки даного класу. Вона обчислюється як відношення числа правильно класифікованих зразків даного класу до загального числа зразків цього у вихідному наборі даних.

Формула:

$$Recall(a) = \frac{TP}{TP + FN}$$

4. F-міра (F1-Score).

F-міра є збалансованим показником точності та повноти. Вона обчислюється як гармонійне середнє між точністю та повнотою. F-міра дозволяє врахувати і точність, і повноту одночасно.

Формула:

$$Fmeasure = \frac{2Precision \cdot Recall}{Precision + Recall}$$

5. Макро-усереднення та мікро-усереднення.

При оцінці моделей багатокласової класифікації часто використовуються усереднені значення метрик за всіма класами. Макро-усереднення обчислює середнє значення метрик кожного класу незалежно і потім усереднює їх. Мікро-усереднення обчислює загальну точність, повноту та F-міру для всіх класів.

3.3 Опис існуючих методів розв’язання

Для векторизації текстових даних найчастіше використовуються наступні методи:

1. Мішок слів (Bag-of-Words).

Нехай є колекція текстових документів D = {d1, d2, ..., dn}. На її основі створюється словник V, що містить унікальні слова із усіх документів. Кожен документ di подається у вигляді вектора, де кожен елемент представляє лічильник слова зі словника V документі di. Результатом є матриця T, де кожен рядок відповідає документу, а кожен стовпець відповідає слову зі словника V.

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

TF (Частота терміна) відображає частоту слова в документі. Вона обчислюється як відношення числа входження слова до загальної кількості слів у документі.

Формула:

$$\mathrm{tf}(t,d) = \frac{n_t}{\sum_k n_k}$$

IDF(зворотня частота документа) відображає важливість слова в колекції документів. Вона обчислюється як логарифм відношення загального числа документів до документів, що містять це слово.

Формула:

$$\mathrm{idf}(t,D) = \log \frac{|D|}{|\{\, d_i \in D \mid t \in d_i \,\}|}$$

де |D| - загальна кількість документів у колекції,

Кожен документ di представляється вектором, де кожен елемент представляє TF-IDF-вага слова зі словника V документі di. Результатом є матриця T, де кожен рядок відповідає документу, а кожен стовпець відповідає слову зі словника V.

Порівняння методів Bag-of-Words(BoW) та TF-IDF:

Таблиця. 3.1 – Порівняння методів векторизації тексту

Критерій порівняння	BoW	TF-IDF
1. Врахування важливості слів	Не враховує важливість слів, вважаючи їх	Враховує важливість слів у документі, враховуючи їх частоту в

	рівноцінними. Метод просто відображає наявність чи відсутність слів у тексті.	документі і зворотну частоту слова в колекції документів. Це дозволяє виділити ключові слова, що характеризують документ.
2. Обробка стоп-слів та рідкісних слів	Зазвичай не видаляє стоп-слова і може включати їх у векторне уявлення. Також він не враховує рідкісних слів, які можуть нести важливу інформацію.	Часто видаляє стоп-слова, оскільки вони мало інформативні, та їх включення може спотворити важливість інших слів. Він також враховує рідкісні слова, оскільки вони можуть бути більш інформативними для класифікації.
3. Розмірність векторів	Розмірність векторів у BoW залежить від кількості унікальних слів у документі.	Розмірність векторів у TF-IDF також залежить від кількості унікальних слів, але кожен елемент вектора має числове значення, що відображає важливість слова.

Отже, можна зробити висновок, що метод TF-IDF є більш доречним для використання у задачі класифікації, так як більше враховує семантику тексту.

3. Word2Vec.

Word2Vec використовує нейронні мережі для створення векторних уявлень слів. Модель навчається на великому корпусі текстових даних, передбачаючи контекстні слова цього слова. У результаті кожному слову надається щільний вектор фіксованої довжини, де семантично близькі слова мають векторні уявлення. Вектори слів можуть бути використані для подання документів шляхом агрегації векторів слів, наприклад шляхом усереднення або підсумовування.

Алгоритм Word2Vec має дві основні архітектури: Continuous Bag-of-Words (CBOW) та Skip-gram. Обидві архітектури працюють на основі прогнозу контексту слова в заданому вікні. Ось математичний опис кожної архітектури:

Continuous Bag-of-Words (CBOW):

Припустимо, що ми маємо речення, і наше завдання - передбачити цільове слово на основі його контексту.

Нехай w(t) буде цільовим словом і C - контекстні слова у вікні навколо w(t). Цільове слово w(t) передбачається на основі суми векторних уявлень контекстних слів: w(t) = sum(v(w(c))), де v(w(c)) - векторне уявлення контекстного слова w(c).

Мета полягає у мінімізації функції втрат між передбачуваним та реальним цільовим словом, щоб оновити векторні уявлення всіх слів у процесі навчання.

Skip-gram:

На відміну від CBOW, Skip-gram використовує цільове слово для прогнозування контекстних слів.

Нехай w(t) буде цільовим словом і C - контекстні слова у вікні навколо w(t). Контекстні слова C передбачаються з урахуванням векторного уявлення цільового слова: w(c) = v(w(t)), де v(w(t)) - векторне уявлення цільового слова w(t).

Мета полягає в мінімізації функції втрат між передбачуваними контекстними словами та реальними контекстними словами, щоб оновити векторні уявлення всіх слів у процесі навчання.

Обидва підходи, CBOW і Skip-gram використовують нейронні мережі з одним прихованим шаром, де векторні уявлення слів оновлюються за допомогою зворотного поширення помилки і методу градієнтного спуску.

Для вирішення задачі категоризації або мультикласової класифікації існує чимала кількість алгоритмів машинного навчання.

Далі наведені найбільш популярні та ефективні алгоритми, які часто застосовуються для категоризації текстових даних:

1. Наївний Байєсовський класифікатор (Naive Bayes Classifier).

Цей алгоритм ґрунтується на теоремі Байєса та на припущенні про незалежність ознак, що дозволяє ефективно оцінювати ймовірності та класифікувати текстові дані. Він є простим у реалізації та вимагає менше обчислювальних ресурсів у порівнянні з деякими іншими алгоритмами класифікації. Однак, припущення про незалежність ознак може бути надто сильним і не завжди відповідати реальним залежностям у даних.

Опис роботи:

Нехай у нас є навчальна вибірка, що складається з текстових документів із відповідними мітками класів. Позначимо текстовий документ як $x = (x_1, x_2, ..., x_n)$, де $x_1, x_2, ..., x_n$ - ознаки (слова, терміни) у тексті. Мітка класу позначається як y.

Підрахунок апріорної ймовірності класів P(y) - ймовірність появи кожного класу в навчальній вибірці. Підрахунок умовних ймовірностей $P(x|y)$ - ймовірність появи кожної ознаки x_i за умови класу y. Для цього підраховується кількість документів у кожному класі, в яких зустрічається ця ознака, та ділиться на загальну кількість документів у даному класі.

Для нового текстового документа x_t обчислюємо апостеріорну ймовірність $P(y|x_t)$ для кожного класу y. Для кожного класу y, використовуючи наївне

припущення про незалежність ознак, обчислюємо $P(x_t|y)$ як добуток умовних ймовірностей $P(x_i|y)$ для кожної ознаки x_i у документі x_t. Вибираємо клас y з найбільшою апостеріорною ймовірністю $P(y|x_t)$ як передбачений клас для текстового документа x_t.

2. Метод k-найближчих сусідів(KNN).

Алгоритм KNN класифікує новий об'єкт на основі його найближчих сусідів у навчальному наборі даних. Переваги методу KNN включають простоту реалізації, відсутність вимог до попереднього навчання моделі та здатність обробляти нелінійні залежності.

Принцип роботи моделі машинного навчання KNN заснований на тому, що об'єкти зі схожими ознаками схильні до належності одного класу або мають схожі значення цільової змінної.

Гіперпараметри моделі:

а) k (кількість сусідів).

Гіперпараметр K визначає кількість найближчих сусідів, які використовуватимуться для класифікації чи регресії. Значення k має бути вибрано заздалегідь і може впливати на точність та стабільність моделі.

б) Метрика відстані.

KNN використовує метрику відстані для визначення найближчих сусідів. Найбільш поширеними метриками є евклідова відстань, манхеттенська відстань та відстань Чебишева. Вибір правильної метрики залежить від характеристик даних та необхідної задачі.

в) Вага сусідів.

При класифікації з використанням KNN можна надати вагу кожному сусідові на основі їх відстані до досліджуваного об'єкта. Це дозволяє враховувати найближчих сусідів із великою вагою. Зазвичай ваги обчислюються з допомогою зворотної величини відстані.

г) Алгоритм пошуку найближчих сусідів.

Для ефективної роботи KNN потрібен ефективний алгоритм пошуку найближчих сусідів. Одним із найпоширеніших алгоритмів є KD-дерево.

г) Нормалізація ознак.

KNN може бути чутливим до різних масштабів ознак, тому нормалізація ознак може бути корисною для забезпечення однакової важливості кожної ознаки.

Основні кроки алгоритму KNN:

1. Нормалізація даних.

Якщо потрібно, проводиться нормалізація ознак об'єктів, щоб зробити їх порівнянними та уникнути спотворень ваг.

2. Вибір параметра k.

Визначається кількість найближчих сусідів, які використовуватимуться для класифікації нового об'єкта. Значення k може бути вибрано емпірично або з використанням крос-валідації.

3. Обчислення відстаней.

Розраховується відстань між новим об'єктом та всіма об'єктами навчального набору даних. Відстань може бути виміряна з використанням різних метрик, таких як евклідова відстань або манхеттенська відстань.

Формула евклідової відстані:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2}$$

де p – перша точка з координатами (p1, … , pn), q – друга точка з координатам (q1, …, qn).

Формула манхеттенської відстані:

$$d_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|,$$

де p – перша точка з координатами (p1, … , pn), q – друга з координатами (q1, …, qn).

4. Визначення найближчих сусідів.

Вибираються k об'єктів із навчального набору даних, найближчих до нового об'єкта, на основі розрахованих відстаней.

5. Ухвалення рішення.

Для завдання класифікації новий об'єкт відноситься до класу, який є найбільш представленим серед найближчих сусідів. Для завдання регресії новому об'єкту надається середнє значення цільової змінної серед найближчих сусідів.

3. Логістична регресія (Logistic Regression).

Вона заснована на моделі лінійної регресії, але із застосуванням логістичної функції для отримання ймовірностей приналежності до класів.

Опис роботи:

У логістичній регресії ми припускаємо, що ймовірність приналежності до класу 1 можна описати за допомогою логістичної функції. Формально для вхідних даних xi, ми визначаємо гіпотезу hθ(xi) наступним чином: hθ(xi) = g(θ^T*xi) де g(z) – логістична функція, θ – вектор параметрів моделі. Для навчання моделі ми визначаємо функцію вартості, яка вимірює різницю між передбаченими значеннями та дійсними мітками. Для багатокласової класифікації часто використовується категоріальна крос-ентропія. Метою навчання є знаходження оптимальних значень параметрів моделі, які мінімізують функцію вартості. Це може бути досягнуто за допомогою методу градієнтного спуску або його варіантів.

У даній роботі використовується різновид логістичної регресії - мультиноміальна логістична регресія(Multinomial Logistic Regression). Вона є узагальненням бінарної логістичної регресії на випадок, коли ми маємо більше двох класів.

У мультиноміальній логістичній регресії ми прагнемо передбачити можливість приналежності до кожного класу для заданого спостереження. Для цього модель використовує логістичну функцію для перетворення лінійної комбінації ознак у ймовірності. Зазвичай використовується функція Softmax,

яка виконує нормалізацію та перетворює лінійні оцінки у ймовірності, сума яких дорівнює 1.

Математичний опис мультиноміальної логістичної регресії ґрунтується на принципі максимальної правдоподібності.

Припустимо, є набір даних з безліччю спостережень та кількома класами. Нехай X є матрицею ознак розмірності (n x m), де n - кількість спостережень, а m - кількість ознак. Нехай y є вектором позначок класів розмірності (n x 1), де кожен елемент yi відноситься до класу, до якого належить i-е спостереження.

Мультиноміальна логістична регресія моделює можливість належності кожного класу для заданого спостереження. Нехай P(y=j|X) означає можливість приналежності спостереження до класу j при заданих ознаках X. Тоді модель мультиноміальної логістичної регресії можна описати так:

1. Лінійна комбінація

Обчислюємо лінійну комбінацію ознак та параметрів моделі для кожного класу j:

zj = X * Wj + bj,

де Wj - матриця ваги розмірності (m x k), де k - кількість класів, bj - вектор зміщення розмірності (1 x k).

2. Softmax

Застосовуємо функцію Softmax для перетворення лінійних комбінацій у ймовірності: P(y=j|X) = softmax(zj) = exp(zj) / sum(exp(zk)), де softmax(zj) – ймовірність приналежності до класу j, exp(zj) – експонента лінійної комбінації, sum(exp(zk)) – сума експонент усіх лінійних комбінацій.

3. Оцінка параметрів.

Ціль полягає в тому, щоб оцінити параметри Wj і bj шляхом максимізації правдоподібності даних. Максимізується функція правдоподібності, яка виражає ймовірність спостережень відповідно до моделі.

4. Навчання моделі.

Для навчання моделі використовують різні оптимізаційні алгоритми, такі як стохастичний градієнтний спуск (SGD) або його варіації, для знаходження оптимальних значень параметрів моделі, які максимізують правдоподібність даних.

Таким чином, математичний опис мультиноміальної логістичної регресії пов'язує лінійні комбінації ознак із ймовірностями приналежності до різних класів за допомогою функції Softmax. Оцінка параметрів моделі здійснюється шляхом максимізації правдоподібності даних.

Гіперпараметри мультиноміальної логістичної регресії:

1. Сила регуляризації (C).

Гіперпараметр C обернено пропорційний силі регуляризації. Більше значення C зменшує силу регуляризації, дозволяючи моделі краще підлаштовуватися під навчальні дані, але може призвести до перенавчання. Найменше значення C підвищує силу регуляризації, сприяючи узагальненій моделі.

2. Метод регуляризації.

Мультиноміальна логістична регресія підтримує різні методи регуляризації, такі як L1 (Лассо) та L2 (гребенева) регуляризації. Ці методи допомагають контролювати перенавчання та покращити узагальнюючу здатність моделі.

3. Максимальна кількість ітерацій.

Гіперпараметр max_iter визначає максимальну кількість ітерацій, яку модель виконуватиме під час навчання. Якщо модель не відповідає оптимальному рішенню, можна збільшити значення max_iter.

4. Схема оптимізації.

Мультиноміальна логістична регресія підтримує різні методи оптимізації, такі як алгоритм стохастичного спуску градієнта (SGD) або метод L-BFGS. Вибір оптимізаційного методу може впливати на швидкість збіжності та якість моделі.

4. Випадковий ліс (Random Forest).

Він ґрунтується на ансамблевому підході, комбінуючи передбачення кількох дерев рішень для отримання більш точних та стійких результатів. Здатен обробляти великі обсяги даних і має хорошу здатність до узагальнення. Також варто відмітити, що не потребується попередня обробка даних.

Опис роботи:

Опис роботи випадкового лісу варто почати з опису роботи його структурних одиниць – окремих дерев прийняття рішень.

Основна ідея дерева прийняття рішень у тому, щоб розбити дані різні групи, з значимості різних ознак. Це робиться шляхом побудови деревоподібної структури, де кожен вузол є тестом на певну ознаку, а кожне ребро виражає можливе значення цієї ознаки. Кожен лист дерева відповідає класифікації чи прогнозу.

Процес побудови дерева прийняття рішень включає наступні кроки:

- Вибір кореневого вузла.

На цьому етапі вибирається ознака, яка найкраще поділяє дані на підгрупи з найбільшою чистотою або інформаційним приростом. Чистота означає однорідність класів у підгрупі, а інформаційний приріст вимірює, наскільки поділ зменшує невизначеність.

- Поділ даних.

Після вибору кореневого вузла дані поділяються на підгрупи відповідно до вибраної ознаки. Кожна підгрупа стає дочірнім вузлом кореневого вузла.

- Рекурсивний поділ.

Процес поділу продовжується для кожного дочірнього вузла, доки не будуть виконані певні умови зупинки. Ці умови можуть включати досягнення певної глибини дерева, досягнення мінімальної кількості об'єктів у вузлі або відсутність подальшого поліпшення при розподілі.

- Присвоєння класу.

Коли поділ закінчується, кожному листу дерева надається конкретний клас чи значення, що відповідає прогнозу.

Переваги дерева прийняття рішень включають простоту інтерпретації, можливість обробки як числових, так і категоріальних ознак, а також можливість автоматичного оброблення пропущених значень та викидів у даних. Дерева прийняття рішень також можуть бути ефективними для роботи з великими обсягами даних і здатні виявляти нелінійні залежності між ознаками та цільовою змінною.

Однак, недоліками дерев прийняття рішень є їх схильність до перенавчання, особливо при використанні глибоких дерев, а також їх обмежена здатність обробляти складні залежності даних. Для зниження перенавчання та поліпшення узагальнюючої здатності часто використовуються методи обрізання дерева, ансамблювання або застосування алгоритмів побудови випадкового лісу.

Гіперпаметри дерева прийнятя рішень:

1. Глибина дерева (max_depth).

Гіперпараметр визначає максимальну глибину дерева, тобто кількість рівнів розбиття. Більш глибоке дерево може краще відповідати навчальним даним, але також може призвести до перенавчання. Оптимальне значення цього параметра можна вибрати з використанням крос-валідації або інших методів оцінки моделі.

2. Критерій поділу вузлів (criterion).

Дерево прийняття рішень використовує різні критерії вимірювання якості поділу вузлів. Найбільш поширені критерії включають "gini impurity" та "information gain" (ентропія). Вибір критерію залежить від конкретної задачі та необхідних властивостей моделі.

3. Кількість ознак для розбиття (max_features).

Цей гіперпараметр визначає кількість ознак, які алгоритм розглядатиме при кожному розбиття вузла. Найменша кількість ознак може зменшити обчислювальну складність та запобігти перенавченню.

4. Мінімальна кількість об'єктів у листі.

Цей параметр визначає мінімальну кількість об'єктів, які мають знаходитись у листі дерева. Якщо кількість об'єктів менша, ніж це значення, то розбиття буде зупинено. Це може допомогти запобігти перенавченню та покращити узагальнюючу здатність моделі.

5. Мінімальна кількість об'єктів для розбиття (min_samples_split).

Гіперпараметр визначає мінімальну кількість об'єктів, необхідних для розбиття вузла. Якщо кількість об'єктів менша від цього значення, то розбиття не буде виконуватися. Це також може допомогти запобігти перенавчанню та зменшити складність моделі.

6. Метод вимірювання якості розбиття (splitter).

Дерево прийняття рішень може використовувати різні методи вибору ознаки і значення поділу. Найбільш поширені методи включають "best" (кращий роздільник) та "random" (випадковий вибір роздільника).

Складність побудови дерева рішень.

Складність побудови дерева прийняття рішень залежить від кількох факторів, включаючи кількість ознак, кількість зразків у навчальному наборі даних та критерії зупинки, що використовуються під час побудови дерева.

Тимчасова складність та просторова складність можуть бути описані як:

Часова складність.

Часова складність описує кількість операцій, необхідних для побудови дерева прийняття рішень.

У гіршому випадку, коли кожен вузол дерева поділяє дані на дві рівні частини, час побудови дерева може бути пропорційно O(n * m * log (n)), де n - кількість зразків у навчальному наборі даних, m - кількість ознак .

Середня складність побудови дерева залежить від використаного алгоритму побудови та даних. Деякі алгоритми, такі як CART (Classification and Regression Trees), мають середню складність O(n * m * log (n)), але можуть бути ефективнішими на практиці.

У разі, коли дерево може ідеально розділити дані кожному вузлі, складність може бути O(n * m), де n - кількість зразків у навчальному наборі даних, m - кількість ознак.

Просторова складність.

Просторова складність описує кількість пам'яті, необхідне зберігання дерева прийняття рішень. Просторова складність зазвичай становить O(k * n), де k - кількість вузлів у дереві, n - кількість зразків у навчальному наборі даних.

Для кожного дерева у випадковому лісі проводиться випадковий вибір підмножини даних із поверненням (bootstrap-вибірка). Це означає, що кожне дерево будується на основі набору даних, отриманого шляхом вибору випадкових зразків вихідного набору даних. Для кожного обраної підмножини даних будується окреме рішення дерева.

Після побудови всіх дерев у випадковому лісі передбачення кожного дерева комбінуються для отримання остаточного передбачення. Остаточне передбачення визначається шляхом голосування усіх дерев, зазвичай обирається найчастіший серед усіх варіант. Випадковий ліс дозволяє оцінити важливість кожної ознаки. Це може бути зроблено шляхом вимірювання вкладу кожної ознаки зменшення помилки або збільшення інформаційного виграшу при розділенні даних.

5. Нейронні мережі.

Існує кілька типів нейронних мереж, які широко застосовуються для даної задачі:

а) Нейронна мережа прямого поширення (Feedforward Neural Network).

Є одним із найпоширеніших типів нейронних мереж, що використовуються для завдань класифікації даних.

б) Рекурентні нейронні мережі (RNN).

RNN є основним вибором для аналізу послідовностей, включаючи тексти. Одна з найпоширеніших архітектур RNN - LSTM (Long Short-Term Memory) здатна вловлювати довгострокові залежності в тексті. RNN має можливість зберігати інформацію про попередні стани, що дозволяє враховувати контекст та послідовність слів під час класифікації тексту.

Для цієї роботи була обрана якраз мережа LSTM та використана її реалізація із бібліотеки Keras. LSTM як раз особливо ефективна при роботі з довгостроковими залежностями у послідовностях, таких як природна мова або тимчасові ряди. Загалом, LSTM є потужним інструментом класифікації текстових даних з кількох причин:

1. Врахування контексту.

LSTM здатна враховувати контекст та послідовну структуру текстових даних. Вона може аналізувати залежності між словами та фразами в реченнях, враховуючи їхнє розташування та порядок. Це дозволяє LSTM краще розуміти зміст тексту та виявляти складні патерни та залежності, що важливо для завдань класифікації тексту.

2. Робота з послідовностями різної довжини.

LSTM може працювати з послідовностями різної довжини, що дуже корисно для класифікації текстових даних. Тексти можуть мати різну кількість слів або символів, і LSTM здатна ефективно обробляти їх, не потребуючи фіксованого розміру вхідних даних. Це забезпечує гнучкість під час роботи з різноманітними текстовими даними.

3.Обробка довгострокових залежностей.

LSTM оснащена механізмом, що дозволяє зберігати довгострокові залежності даних. Це особливо важливо для класифікації тексту, де розуміння зв'язків та залежностей між різними частинами тексту може відігравати

важливу роль у прийнятті правильного рішення. LSTM здатна запам'ятовувати інформацію на тривалі часи та використовувати її для прогнозування класів.

4. Уникнення проблеми зникаючого градієнта.

LSTM використовує механізми вентилів (вентиль забування, вентиль входу і вентиль виведення), які допомагають уникнути проблеми зникаючого або вибухового градієнта при навчанні нейронної мережі. Це дозволяє ефективно навчати LSTM та отримувати стабільні результати під час класифікації тексту.

5. Гнучкість та налаштовуваність.

LSTM надає гнучкі налаштування, які дозволяють налаштовувати її параметри відповідно до вимог конкретної задачі класифікації тексту. Різні архітектури LSTM, такі як односпрямовані, двонаправлені або багатошарові LSTM, можуть бути використані залежно від специфіки завдання.

Далі розглянута архітектура мережі. LSTM включає наступні компоненти:

1. Вхідний шар.

Вхідна послідовність представляється як послідовності векторів чи матриць. Кожен елемент вхідної послідовності представляється як x(t), де t – поточний часовий крок.

2. Вентилі.

LSTM використовує спеціальні вентилі, що регулюють потік інформації всередині мережі. Три основних типи вентилів у LSTM: вентиль забування (forget gate), вентиль входу (input gate) та вентиль виведення (output gate).

3. Внутрішній стан (cell state).

LSTM зберігає внутрішній стан, що допомагає запам'ятовувати довгострокові залежності. Внутрішній стан оновлюється на кожному тимчасовому кроці з урахуванням входу та попереднього стану.

4. Вентилі забування (Forget gate).

Вентиль забування визначає, яку інформація з попереднього стану слід зберегти та яку слід забути. Вентиль забування має сигмоїдальну функцію активації, яка видає значення між 0 та 1 для кожного елементу внутрішнього стану. Одне значення вентиля забування відповідає кожному елементу внутрішнього стану.

5. Вентиль входу (Input gate).

Вентиль входу визначає, яка нова інформація має бути додана у внутрішній стан. Вентиль входу також має сигмоїдальну функцію активації, яка визначає, які значення слід оновити.

6. Оновлення внутрішнього стану.

Оновлення внутрішнього стану включає об'єднання інформації з вентилями забування і входу. Потім використовується гіперболічний тангенс для визначення нового стану, який буде передано наступному тимчасовому кроку.

7. Вентиль виведення (Output gate).

Вентиль виведення визначає, яку частину внутрішнього стану слід використовувати для виходу. Вентиль виведення застосовує сигмоїдальну функцію активації до входу та внутрішнього стану, і результат передається через гіперболічний тангенс.

8. Вихідний шар.

Вихідний шар LSTM може бути доданий після виведення вентиля для отримання вихідних значень на кожному тимчасовому кроці.

в) Згорткові нейронні мережі (CNN).

В основному застосовуються для завдань комп'ютерного зору, але також ефективні для обробки текстових даних. CNN здатні витягувати локальні ознаки з тексту за допомогою операції згортки та об'єднання. Вони зазвичай використовуються для аналізу окремих фраз або речень, а потім поєднуються для прийняття остаточного рішення про класифікацію тексту.

Гіперпараметри нейронних мереж:

1. Кількість прихованих шарів (hidden layers).

Нейронна мережа може мати один або кілька шарів. Глибока мережа може вивчати складніші залежності в даних, але потребує більше обчислювальних ресурсів і може призвести до перенавчання. Оптимальна кількість прихованих шарів залежить від конкретного завдання та доступних даних.

2. Кількість нейронів у кожному прихованому шарі.

Кількість нейронів визначає ємність моделі та її здатність до узагальнення. Занадто низька кількість нейронів може призвести до недонавчання, а надто висока – до перенавчання. Оптимальна кількість нейронів також залежить від розміру даних та складності завдання.

3. Розмір пакета (batch size).

Розмір пакета визначає, скільки прикладів даних оброблюється одночасно перед оновленням ваг моделі. Великі розміри пакета можуть прискорити навчання, але потребують більше пам'яті. Маленькі розміри пакета можуть призвести до стабільнішого навчання, але сповільнити процес.

4. Кількість епох (epochs).

Епоха – це один повний прохід по всіх зразках навчального набору даних. Кількість епох визначає, скільки разів модель бачитиме весь набір даних у процесі навчання. Занадто невелика кількість епох може призвести до недонавчання, а занадто велика - до перенавчання.

5. Швидкість навчання(learning rate).

Швидкість навчання визначає величину зміни ваги моделі на кожному кроці оновлення. Оптимальна швидкість навчання може змінюватись в залежності від завдання. Занадто велика швидкість навчання може призвести до нестабільності та розходження моделі, а надто маленька – до повільної збіжності.

3.4 Порівняння існуючих методів розв’язання

Для порівняння різних методів машинного навчання потрібно ввести наступні позначення:

d - кількість ознак, k - кількість класів та n - розмір навчальної вибірки, t – кількість дерев у випадковому лісі, T – кількість епох навчання LSTM, s – розмірність прихованого стану LSTM

У таблиці 3.2 порівнюються вищенаведені алгоритми машинного навчання за часовою та просторовою складністю навчання.

Таблиця 3.2 – Порівняння алгоритмів класифікації за часовою та просторовою складністю навчання

Алгоритм	Часова складність алгоритму	Просторова складність алгоритму
Мультиноміальний Наївний Байєс	O (n * d + k * d)	O (k * n)
KNN	O (n * d + n * log(n) + k)	O (d * n)
Мультиноміальна логістична регресія	O (m * n * k + n * k)	O (n * k)
Випадковий ліс	O (t * n * d * log(n * d))	O (n * k * t + k * t)
LSTM	O (T * d * n * s^2)	O (d * s^2)

Кожний алгоритм відрізняється за своєю складністю. Не має якогось універсального для вирішення даної задачі.

3.5 Обґрунтування методу розв’язання

У даній роботі було вирішено обрати ансамблевий алгоритм машинного навчання, який поєднує кілька моделей класифікації для прийняття рішень шляхом голосування. Обрані моделі для ансамблевого алгоритму наступні: наївний Байєсовський класифікатор, мультиноміальна логістична регресія, k-найближчих сусідів(KNN), випадковий ліс та нейронна мережа LSTM. Цей алгоритм також відомий під назвою Voting Classifier. Він має кілька переваг, які роблять його корисним у різних сценаріях:

1. Збільшення точності.

Voting Classifier може покращити загальну точність класифікації шляхом комбінування передбачень кількох моделей. За наявності різноманітності у моделях, кожна з яких може мати свої сильні та слабкі сторони, ансамбль дозволяє знизити помилку та підвищити загальну точність класифікації.

2. Зниження варіативності.

Використання кількох моделей дозволяє зменшити варіативність результатів. Якщо одна модель схильна до перенавчання або дає нестабільні передбачення, інші моделі можуть згладити ці проблеми, підвищивши загальну надійність та стабільність.

3. Поліпшення стійкості.

Voting Classifier більш стійкий до викидів або шуму даних. Якщо одна модель помиляється через аномалії даних, інші моделі можуть компенсувати цю помилку і надати більш надійні передбачення.

3.6 Навчання моделей.

На початку навчається кожна з обраних вище моделей за допомогою класу GridSearchCV, завдяки якому виконується пошук найкращих

гіперпараметрів моделі. Для нейронної мережі LSTM пошук гіперпараметрів здійснюється емпіричним шляхом.

Для цього для кожної з моделей обирається множина можливих гіперпараметрів:

1. Мультиноміальний наївний Байєсовський класифікатор: alpha.

2. Мультиноміальна логістична регресія: C.

3. KNN: k.

4. Випадковий ліс: критерій розбиття підмножини спостережень у дереві, максимальна глибина дерева.

5. LSTM: швидкість навчання, кількість епох навчання, кількість прихованих шарів та внутрішніх станів, кількість нейронів у кожному шарі, функції активації.

Для оцінки кожної з моделей використовуються наступні метрики:

1. Влучність, повнота та F-міра для кожного класу.

2. Точність моделі.

3. Макро усереднення моделі.

4. Зважене макро усереднення моделі.

Далі наведені гіперпараметри кожної з моделей, що дали найкращі результати на навчальних даних:

1. Мультиноміальний наївний Байєсовський класифікатор.

alpha = 0,01

2. Мультиноміальна логістична регресія.

C = 1

3. KNN.

k=5

4. Випадковий ліс.

Критерій = gini, максимальна глибина дерева = 8.

Далі наведені таблиці метрик класів та загалом кожної моделі машинного навчання.

У таблицях 3.3 та 3.4 наведені метрики для оцінки якості моделі мультиноміальний наївний Байєсовський класифікатор:

Таблиця 3.3 – метрики моделі мультиноміальний наївний Байєсовський класифікатор по класам

Клас	Влучність	Повнота	F-міра	Кількість спостережень
1	0,98	0,95	0,97	134
2	1,00	0,96	0,98	94
3	0,95	0,99	0,97	107
4	1,00	0,99	0,99	134
5	0,94	1,00	0,97	88

Таблиця 3.4 – метрики моделі мультиноміальний Байєсовський класифікатор

	Влучність	Повнота	F-міра	Кількість спостережень
Точність моделі			0,97	557
Макро усереднення моделі	0,97	0,98	0,97	557
Зважене макро усереднення моделі	0,98	0,97	0,97	557

У таблицях 3.5 та 3.6 наведені метрики для оцінки якості моделі мультиноміальна логістична регресія:

Таблиця 3.5 – метрики моделі мультиноміальна логістична регресія по класам

Клас	Влучність	Повнота	F-міра	Кількість спостережень

1	0,92	0,96	0,94	134
2	1,00	0,97	0,98	94
3	0,99	0,99	0,99	107
4	1,00	0,99	0,99	134
5	0,94	0,95	0,95	88

Таблиця 3.6 – метрики моделі мультиноміальна логістична регресія

	Влучність	Повнота	F-міра	Кількість спостережень
Точність моделі			0,96	557
Макро усереднення моделі	0,97	0,96	0,96	557
Зважене макро усереднення моделі	0,96	0,96	0,96	557

У таблицях 3.7 та 3.8 наведені метрики для оцінки якості моделі k-найближчих сусідів(KNN):

Таблиця 3.7 – метрики моделі KNN по класам

Клас	Влучність	Повнота	F-міра	Кількість спостережень
1	0,97	0,81	0,88	134
2	0,98	0,94	0,96	94
3	0,81	0,98	0,89	107
4	0,98	0,98	0,98	134
5	0,91	0,97	0,94	88

Таблиця 3.8 – метрики моделі KNN

	Влучність	Повнота	F-міра	Кількість спостережень
Точність моделі			0,93	557
Макро усереднення моделі	0,93	0,93	0,93	557
Зважене макро усереднення моделі	0,94	0,93	0,93	557

У таблицях 3.9 та 3.10 наведені метрики для оцінки якості моделі випадковий ліс:

Таблиця 3.9 – метрики моделі випадковий ліс по класам

Клас	Влучність	Повнота	F-міра	Кількість спостережень
1	0,81	0,98	0,89	134
2	1,00	0,82	0,90	94
3	0,98	0,85	0,91	107
4	0,92	0,99	0,95	134
5	0,96	0,89	0,92	88

Таблиця 3.10 – метрики моделі випадковий ліс

	Влучність	Повнота	F-міра	Кількість спостережень
Точність моделі			0,92	557
Макро усереднення	0,93	0,91	0,92	557

моделі				
Зважене макро усереднення моделі	0,93	0,92	0,92	557

У таблицях 3.11 та 3.12 наведені метрики для оцінки якості моделі Voting Classifier:

Таблиця 3.11 – метрики моделі VotingClassifier по класам

Клас	Влучність	Повнота	F-міра	Кількість спостережень
1	0,98	0,94	0,96	134
2	1,00	0,97	0,98	94
3	0,94	0,99	0,96	107
4	1,00	0,99	1,00	134
5	0,96	1,00	0,98	88

Таблиця 3.12 – метрики моделі VotingClassifier

	Влучність	Повнота	F-міра	Кількість спостережень
Точність моделі			0,98	557
Макро усереднення моделі	0,98	0,98	0,98	557
Зважене макро усереднення моделі	0,98	0,98	0,98	557

Отже, проаналізувавши якісні метрики моделей, можна зробити висновок, що серед окремих моделей найгірші результати дає випадковий ліс, а найкращі - наївний Байєсовський класифікатор. Завдяки ансамблевому алгоритму голосування моделей досягається найкращий результат.

3.7 Розгортання системи

Розгортання системи здійснюється за допомогою технології CI/CD.

Процес розгортання системи складається з наступних кроків:

1. На початку розробки системи був створений репозиторій для зберігання та керування кодом телеграм-бота. Для цього була використана система контролю версій Git та платформа GitLab для віддаленого розташування коду.

2. Після цього був налаштований процес безперервної інтеграції та доставки (CI/CD). Платформа GitLab надає інструменти для налаштування CI/CD, такі як конвеєр CI/CD, який автоматично збирає і доставляє оновлення телеграм-бота після кожної зміни програмного коду, відправленої на віддалений репозиторій GitLab. Коли вносяться зміни до репозиторію, CI/CD-конвеєр автоматично збирає новий Docker-образ і доставляє його до Azure для оновлення контейнера з ботом. У контейнері знаходяться всі необхідні залежності і компоненти для роботи бота.

3. Був створений ресурс групи Azure, в якому розгортується контейнер. В рамках ресурсної групи був створений сервіс Azure Container Instances (ACI) У ньому були встановлені відповідні установки та параметри для контейнера.

4. Після розгортання телеграм-бота в контейнері Docker на Azure потрібно переконатися, що він працює правильно. Для цього проводиться тестування функціональності та продуктивності бота, також налаштувується моніторинг Azure для відстеження роботи бота, збирання метрик та виявлення проблем.

Висновок до розділу

У цьому розділі було спочатку розглянуто систему з концептуальної точки зору. Було описані її основні компоненти та процес створення системи.

Далі було наведено математичні засади системи, описані основні методи розв’язання, їх порівняння, а також кінцеве обгрунтування обраних методів для реалізації системи.

У кінці розділу було поетапно описані процеси навчання моделей та розгортання системи на хмарній інфраструктурі, завдяки чому програма є завжди доступною користувачеві

4 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

4.1 Вхідні та вихідні дані

Вхідні дані системи повинні обов’язково відповідати наступним вимогам:

1. Бути англійською мовою.
2. Не містити ненормативну лексику.
3. Мати не менш ніж 10 слів з урахуванням стоп-слів.
4. Мати не більш ніж 1000 слів з урахуванням стоп-слів.
5. У разі передачі файлу, файл повинен бути лише текстового формату

Для коректного функціонування підсистеми категоризації вхідні дані повинні відповідати певним вимогам:

1. Якомога менше містити власні назви.
2. Не містити неформальну мову.

Вихідні дані системи:

1. Категорія, до якої відноситься текстовий документ.
2. Мова вхідного текстового документу при невдалій валідації даних(коли текстовий документ не англійською мовою).
3. Наявність ненормативної лексики.
2. Статистика використання системи користувачем.
3. Деякі характеристики використання системи всіма користувачами.

4.2 Керівництво користувача

Для того, щоб ідентифікувати основні функціональні можливості системи, було створено діаграму прецендентів. Вона дозволяє виділити різні дії, які актори можуть виконувати взаємодіючи із системою.

На рисунку 4.1 продемонстровано діаграму прецендентів для даної системи:

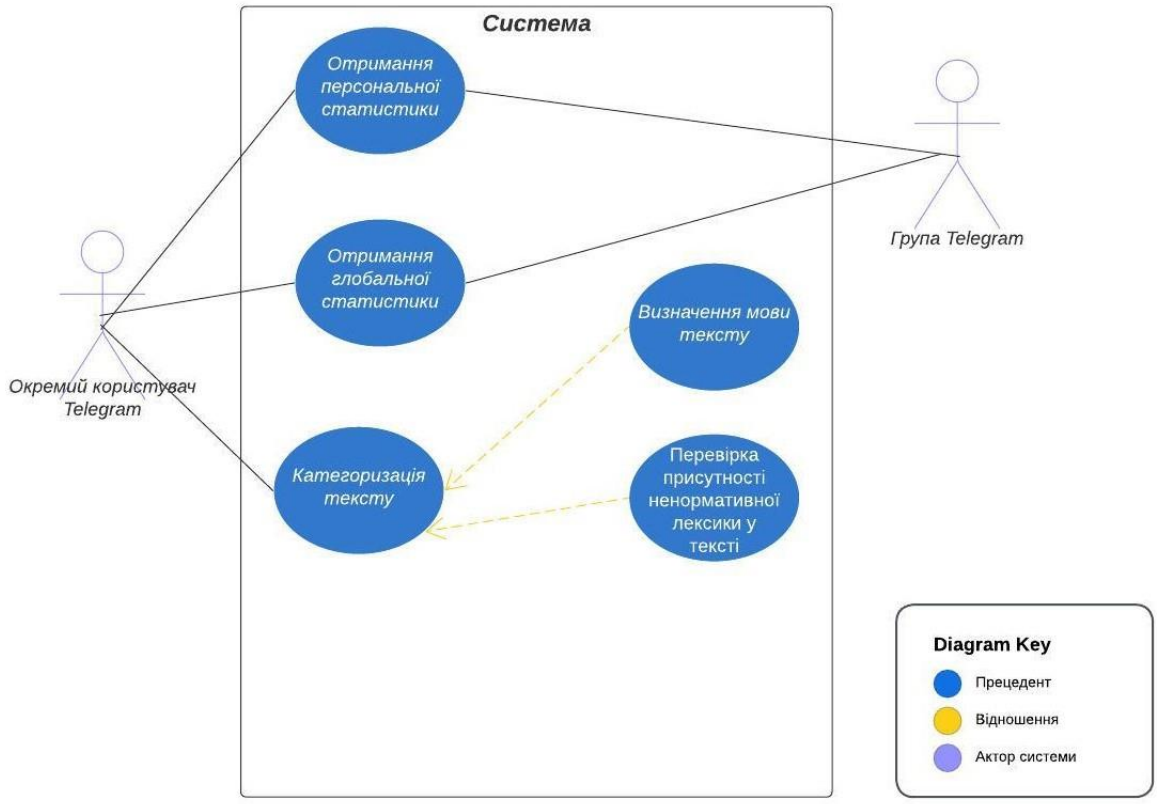

Рисунок 4.1 – Діаграма прецендентів

Далі детально описано взаємодію з програмою.

Для початку роботи з розробленою системою достатньо мати акаунт у відомому месенджері Telegram, клієнта Telegram та відповідно з мережею Internet.

Після цього потрібно за допомогою пошуку Telegram знайти бота за його нікнеймом. Взаємодія з ботом виконується як звичайний обмін повідомленнями у чаті. Також інтерфейс ботів на платформі Telegram дозволяє інтегрувати їх у інші чати та спільноти, що робить програму-бота ще більш гнучкою у використанні.

Щоб почати взаємодію з ботом, потрібно вивести список підтримуваних ним команд:

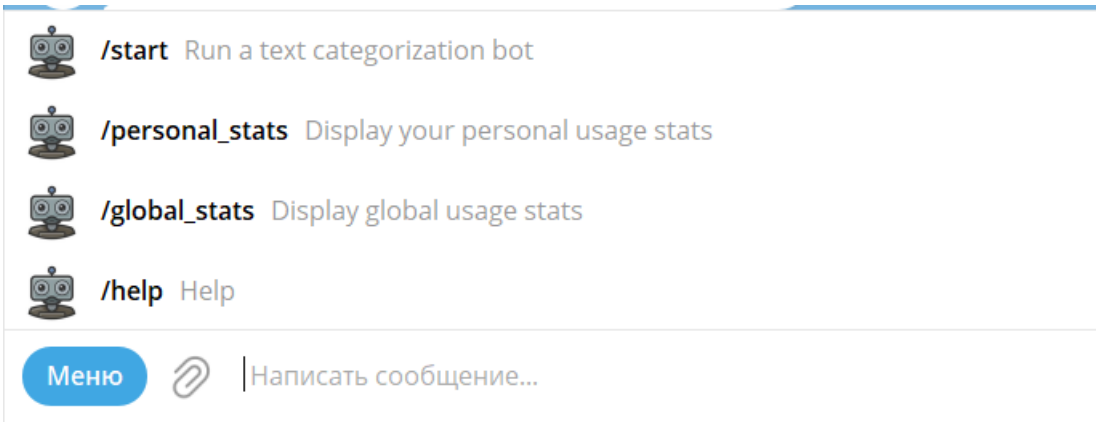

Рисунок 4.2 – Відображення списку підтримуваних ботом команд

Далі, в залежності від обраної команди, можно ввести вхідні дані. Це може бути безпосередньо текст або файл текстового формату.

Якщо введені дані не будуть відповідати вимогам, бот вкаже, що саме не так.

Демонстрація визначення мови вхідного тексту ботом:

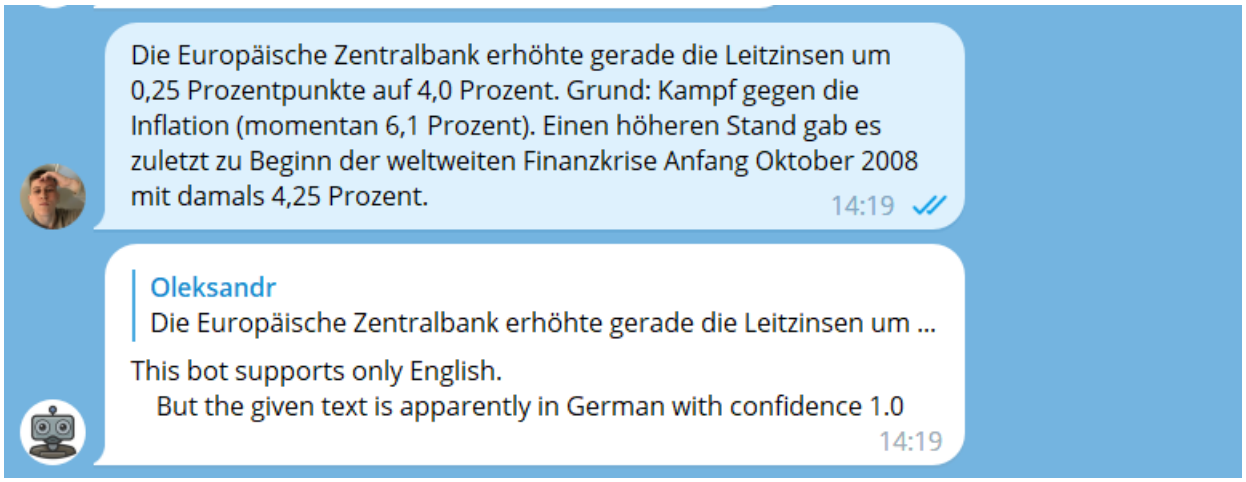

Рисунок 4.3 – Визначення мови текстових даних ботом

Після цього система якийсь час витратить на обчислення результату.

Після відображення результату можна обрати опцію перегляду або власної статистики використання, або деякі характеристики загальної статистики.

4.3 Випробування програмного продукту

Щоб впевнитись у коректній роботі програми, було вирішено провести серію випробувань над нею.

Серія складається з наступних етапів:

1. Перевірка стабільності роботи.

Для цього у різні дні та різні частини доби виконувались запити до бота. У всіх випадках бот продемонстрував надійність, так як виконується у ізольованому середовищі на інфраструктурі провайдера хмарних послуг.

2. Перевірка з’єднання з базою даних.

Так як база даних розташована віддалено на сервері провайдера хмарних послуг, треба було перевірити коректність з’єднання з нею.

3. Перевірка валідації даних.

Для цього були використані вхідні текстові дані на різних природних мовах. Загалом, система вірно розпізнає мову вхідного тексту за вийнятком незначних похибок.

Також частиною валідації вхідних є перевірка на вміст ненормативної лексики. У всіх випадках система бездоганно визначила, чи містять дані ненормативну лексику.

3. Відносна коректність категоризації випадкових вхідних даних.

Для більшості вхідних текстових документів було правильно визначено категорію, до якої вони відносяться.

4. Коректність відображення персональної статистики.

Була також перевірена функція надання персональної статистики користувачеві. У всіх випадках надана статистика відповідала фактичному використанню бота користувачем.

Висновок до розділу

На початку даного розділу були вказані вхідні та вихідні дані системи.

Після цього була надана діаграма прецендентів системи та детально описано покрокову взаємодію користувача з нею.

Після цього був описаний процес проведення випробувань над системою для перевірки її надійності та працездатності. Таким чином, виконані випробування підтвердили коректне функціонування системи.

ВИСНОВКИ

У цьому дипломному проєкті була розроблена система багатокласової класифікації текстових даних, що надає користувачеві інтерфейс у вигляді телеграм-бота, та розгорнута як контейнерний додаток. Мета роботи полягала у створенні ефективної системи, здатної класифікувати текстові дані на кілька категорій, та надання можливості використання системи через інтерфейс телеграм-бота.

Для досягнення цієї мети було виконано наступні кроки:

1. Збір та попередня обробка даних.

Було складено та підготовлено датасет з текстовими даними, призначеними для класифікації. Дані були проаналізовані та очищені від шуму та небажаних символів.

2. Навчання моделей класифікації.

Були обрані відповідні моделі машинного навчання для багатокласової класифікації, а саме модель мультиноміальної логістичної регресії. Моделі була навчена на попередньо підготовлених даних, щоб навчитися розпізнавати та класифікувати тексти відповідно до заданих категорій.

3. Розробка програми-бота.

Було розроблено програму-бот з використанням відповідного API. Бот був налаштований на прийом текстових даних від користувачів і передачу їх у підсистему класифікації з допомогою навченої моделі. Отримані результати класифікації відправлялися користувачеві через інтерфейс бота.

4. Розгортання бота як контейнерної програми.

Була використана технологія Docker для упаковки та розгортання телеграм-бота у вигляді контейнера. Це дозволило спростити процес розгортання та забезпечити портативність та масштабованість системи.

Результати роботи показують, що розроблена система достатньо вдало виконує завдання багатокласової класифікації текстових даних певного формату та надає зручний інтерфейс через телеграм-бот. Розгортання системи

у вигляді контейнерного додатка забезпечує гнучкість та можливість легкого масштабування для роботи з великим обсягом даних та високою продуктивністю.

Перевірка працездатності системи довела, що система працює безперебійно завдяки тому, що контроль інфраструктури виконується провайдером хмарних послуг та таким чином звільняє розробника від зайвих труднощів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Онлайн служба TextRazor URL: https://www.textrazor.com/(дата звернення 16.04.2023)

2. Онлайн служба uClassify URL: https://www.uclassify.com/browse/uclassify/topics/(дата звернення 18.04.2023)

3. Документація Docker URL: https://docs.docker.com/(дата звернення 11.05.2023)

4. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/

5.Офіційний сайт бібліотеки scikit-learn URL, документація по класу NaiveBayes: https://scikit-learn.org/stable/modules/naive_bayes.html/(дата звернення 11.05.2023)

6. Офіційний сайт бібліотеки scikit-learn, документація по класу RandomForestClassifier URL: https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html/(дата звернення 7.05.2023)

7. Онлайн-стаття про ансамблевий алгоритм, 2022. URL: https://towardsdatascience.com/creating-an-ensemble-voting-classifier-with-scikit-learn-ab13159662d

8. Документація бібліотеки pyTelegramBotApi URL: https://pytba.readthedocs.io/en/latest/index.html (дата звернення 7.05.2023)

9. Документація СУБД MongoDB URL: https://www.mongodb.com /docs/(дата звернення 10.05.2023)

10. Sebastiani F., 2010. Machine learning in automated text categorization, ACM Comput. Surv. Vol. 34, No. 1. ст. 1-47.

11. Yang Y. 2012. A re-examination of text categorization methods / Y. Yang, SIGIR’2012, 22nd ACM International Conference on Research and Development in Information Retrieval, ст. 42-49

10. Simeon Kostadinov. Recurrent Neural Networks with Python Quick Start Guide: Sequential learning and language modeling with TensorFlow URL: https://www.amazon.com/Recurrent-Neural-Networks-Python-Quick/dp/1789132339

11. Sathyajith Bha. Practical Docker with Python. 2nd Ed.

12. Офіційний сайт бібліотеки nltk URL: https://www.nltk.org/(дата звернення 8.05.2023)

13. Документація CI на платформі GitLab URL: https://docs.gitlab.com/ee/ci/(дата звернення 12.05.2023)

14. Онлайн-стаття про випадковий ліс URL: https://www.datacamp.com/tutorial/random-forests-classifier-python

15. Офіційний сайт бібліотеки nltk URL: https://keras.io/(дата звернення 7.05.2023)