Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Методи глибинного навчання та обробки природномовних текстів в задачах психологічного профілювання людей
(тема)

Виконав:
студент 2 курсу, групи СШІм-21-2
Шаталов О.В.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник проф. Рябова Н.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Шаталову Олексію Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи глибинного навчання та обробки природномовних текстів з задачах психологічного профілювання людей

затверджена наказом університету від 31 березня 20 23 р. № 306Ст

2. Термін подання студентом роботи до екзаменаційної комісії 17 травня 20 23 р.

3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел, книг, патентів та відомих наукових проектів щодо розробки та дослідження аналізу текстів з боку розпізнавання особистості людини, Python-документація, TypeScript-документиація, документація для бібліотек аналізу та обробки даних.

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі та постановка задачі

2) Підготовчі дослідження та методи вирішення задачі

3) Проведення експериментів

4) Створення програмного застосунку

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі	04.04.2023	виконано
2	Огляд основних теорій особистості	04.04.2023	виконано
3	Вивчення досліджень аналізу тексту для профілювання	05.04.2023	виконано
4	Аналіз релевантних методів обробки даних	08.04.2023	виконано
5	Проведення експериментів	10.04.2023	виконано
6	Розробка програмного застосунку	28.04.2023	виконано
7	Написання пояснювальної записки	10.05.2023	виконано
8	Попередній захист	12.05.2023	виконано
9	Захист перед ЕК	17.05.2023	

Дата видачі завдання 3 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Рябова Н.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 103 с., 9 рис., 21 табл., 2 дод., 35 джерел.

ВЕЛИКА МОВНА МОДЕЛЬ, ГЛИБОКЕ НАВЧАННЯ, ІНДИКАТОР ТИПІВ МАЄРС-БРІҐС, ОБРОБКА ПРИРОДНЬОЇ МОВИ, ПАТЕРН ПОВЕДІНКИ ЛЮДИНИ, ПСИХОЛОГІЧНЕ ПРОФІЛЮВАННЯ, ПСИХОЛОГІЯ.

Об’єкт дослідження – процеси розпізнавання патернів поведінки людини та її психологічне профілювання на основі написаного нею тексту засобами обробки природньої мови та методів глибокого навчання.

Предмет дослідження – система виявлення рис поведінки людини за зазначеними типологіями за допомогою написаного нею тексту.

Мета роботи – дослідження у сфері спроможності виявлення певних патернів поведінки людини за написаним нею текстом засобами глибокого навчання та обробки природньої мови.

Методи дослідження – аналіз теоретичного матеріалу, предметної області, алгоритмів глибокого навчання, математичний апарат знаходження патернів поведінки людини, програмна реалізація.

Розроблено програмний модуль, здатний проводити аналіз написаного людиною тексту для розпізнавання її патернів поведінки, інтегрований до веб-платформи.

Впровадження поданої розробки підвищить якість рекомендаційних систем, стане у нагоді в сфері маркетингу та продажу товарів різних сфер, допоможе персоналізувати однаковий контент тощо.

Результати дослідження можуть бути використані у створенні універсальної платформи для персоналізації різних типів контенту, а також для знаходження відповідностей уподобання контенту та користувача.

ABSTRACT

Explanatory note: 103 p., 9 fig., 21 tabl., 2 ann., 35 sources.

DEEP LEARNING, HUMAN BEHAVIOR PATTERN, LARGE LANGUAGE MODEL, MYERS-BRIGGS TYPE INDICATOR, NATURAL LANGUAGE PROCESSING, PSYCHOLOGICAL PROFILING, PSYCHOLOGY.

The object of research is the processes of recognizing patterns of human behavior and its psychological profiling based on the text written by a person using natural language processing and deep learning methods.

The subject of the study is a system for identifying human behavioral traits according to the specified typologies using the text written by a person.

The aim of the work is to study the ability to detect certain patterns of human behavior based on the text written by a person using deep learning and natural language processing.

Research methods are analysis of theoretical material, subject area, deep learning algorithms, mathematical apparatus for finding patterns of human behavior, software implementation.

A software module capable of analyzing human-written text to recognize its behavioral patterns, integrated into a web platform, has been developed.

Implementation of the presented development will improve the quality of recommendation systems, be useful in marketing and sales of goods in various fields, help personalize the same content, etc.

The results of the study can be used to create a universal platform for personalizing different types of content, as well as for finding matches between content and user preferences.

ЗМІСТ

ВСТУП

Останніми роками сфера обробки природної мови (NLP) зазнала значного прогресу завдяки появі алгоритмів глибокого навчання [1] та великих мовних моделей [2]. Ці досягнення дозволили краще зрозуміти написаний людиною текст і його основні значення, надаючи можливості для різних застосувань в області психології та профілювання особистості.

На теперішній час існують різні підходи до розуміння патернів поведінки людини. Частіше за все, ці підходи зводяться до використання спеціалістів у сфері психології для консультації або фінального висновку. Існують методи автоматичного розпізнавання рис, але вони не універсальні та використовують інші модальності для роботи, відмінні від тексту [3].

Метою цієї магістерської роботи є дослідження засобів обробки природньої мови і методів глибокого навчання для психологічного профілювання людей на основі їхніх письмових текстів. Ми прагнемо дослідити, як показники типів (наприклад, Маєрс-Бріґс (MBTI) [4] чи Великої п’ятірки (Big Five) [5]) можуть бути виведені з написаного людиною тексту сучасними моделями, такими як Generative Pre-trained Transformer (GPT) [6], Bidirectional Encoder Representations from Transformers (BERT) [7], а також мовними інструментами, такими як Linguistic Inquiry and Word Count (LIWC) [8].

Це може стати безцінним інструментом для фахівців, яким потрібно оцінити особистісні риси та поведінку людини, а також для компаній, яким необхідно персоналізувати контент та створювати рекомендаційні системи.

Актуальність роботи визначається поширенням проблематики розуміння патернів поведінки людини, а також потребами компаній та фахівців у розумінні ставлення аудиторії до різних дій та інформації.

Результати дослідження можуть бути застосовані для створення системи розпізнавання типів поведінки людини, що приймає на вхід текст, написаний людиною, та надає специфіку рис її поведінки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної галузі

Все частіше в останній час з’являються новини щодо використання великою кількістю компаній та платформам систем персоналізації контенту. Це може бути як прості алгоритми рекомендацій та їх модифікації, так і більш складні комплексні системи із використанням різних модальностей.

Системи, що використовують аналіз особистості, також входять до переліку нових типів подібних рішень. До аналогів можна віднести продукти Crystal, Personalyze, TestGorilla, The Predictive Index, Apply Magic Sauce та інші. Частина з них є комерційними та використовуються у бізнес-рішеннях, частина є суто дослідницькими закритими проектами. Поява таких рішень, а також проведення досліджень за темою показує, зокрема, актуальність обраного напряму дослідження.

Для більш детального аналізу ми розглянемо спочатку основи теорій психології особистості, а потім перейдемо до опису обраної типології для дослідження та існуючих попередніх робіт із нею.

1.1.1 Основи психології особистості

Особистість – це структура, яка охоплює поведінкові, когнітивні та емоційні патерни, на які впливають біологічні та середовищні фактори. Ці взаємопов'язані патерни відносно стабільні в часі, але вони можуть розвиватися протягом усього життя людини.

Хоча не існує загальноприйнятого визначення особистості, більшість теорій підкреслюють роль мотивації та психологічної взаємодії з оточенням. Теорії особистості, засновані на рисах, визначають особистість як набір рис, які передбачають поведінку людини. І навпаки, поведінкові підходи визначають особистість через навчання та звички. Незважаючи на ці

відмінності, більшість теорій сходяться на думці, що особистість загалом стабільна.

Для подальшої роботи треба ознайомитись із різними теоріями, на яких будуються різні моделі патернів поведінки людей.

1.1.1.1 Теорії типів

У психології використовується термін «тип особистості» для класифікації різних людей на основі їхніх психологічних особливостей. Цю класифікацію іноді відрізняють від рис особистості, які відносяться до меншого набору поведінкових патернів.

Вважається, що типи мають якісні відмінності між людьми, тоді як риси розглядаються як кількісні відмінності. Наприклад, теорії типів припускають, що інтроверти та екстраверти є принципово різними категоріями людей, тоді як теорії рис припускають, що інтроверсія та екстраверсія лежать на безперервному спектрі, а багато людей знаходяться десь посередині. Однак існування типів особистості є дуже дискусійним і суперечливим у порівнянні з особистісними рисами.

Ефективні типології особистості корисні для виявлення і поглиблення нашого розуміння людей, на відміну від стереотипів, які можуть зменшити таке розуміння. Крім того, ефективні типології можуть допомогти в прогнозуванні клінічно значущої інформації про людей і розробці успішних стратегій лікування.

Існує велика кількість літератури про класифікацію людського темпераменту, а також особистісних рис і сфер, які мають на меті описати нормальний темперамент і особистість, виділяючи при цьому домінуючі характеристики різних типів. Ці системи класифікації в першу чергу належать до сфери психології. На відміну від них, розлади особистості знаходяться в центрі уваги психіатрії, медичної спеціальності, і класифікуються в Діагностичному і статистичному посібнику (DSM),

створеному Американською психіатричною асоціацією [9]. Цей посібник пропонує підхід до розуміння та класифікації розладів особистості, орієнтований на захворювання.

У психології існує значна плутанина щодо використання терміну «тип». Крім того, результати тестування особистості мають тенденцію слідувати за кривою, а не чітко вписуватися в певні категорії, що призвело до значної критики теорій типів особистості з боку дослідників-психометрів. Пряме порівняння інструменту «типу» (MBTI) та інструменту «риси» показало, що вимірювання рис є більш ефективним предиктором особистісних розладів [10]. Через ці проблеми теорії типів особистості втратили популярність у психології. Більшість дослідників зараз вважають, що неможливо пояснити різноманітність людської особистості, використовуючи обмежену кількість дискретних типів. Тому вони виступають за моделі рис, такі як модель Big Five.

Попри це, у наведеному дослідженні будуть використані здебільшого моделі типів для демонстрації спроможності розробленого рішення до якісної роботи. Розпізнавання рис пропонується розглянути як покращення рішення та продовження цієї гілки досліджень. Також, наша система розпізнавання типів буде побудована на основі ймовірностей, що може дещо звести простий дискретний поділ на показники належності тому чи іншому типу, як це втілено у системі рис.

Також представниками подібної теорії є Теорія типів A та B [11], деякі праці Карла Юнга [12], що є початківцем подібних теорій.

1.1.1.2 Психоаналітичні теорії

Психоаналітична теорія пояснює людську поведінку, аналізуючи різні компоненти особистості, такі як ід, его та суперего. Зигмунд Фрейд є засновником цієї школи, який ввів термін «психодинаміка», що базується на ідеї перетворення тепла в механічну енергію. Теорія Фрейда надає

центральне значення динамічним, несвідомим психологічним конфліктам. Теорія Фрейда також включає в себе спрямування і вивільнення сексуальної та агресивної енергій, які є основними компонентами його теорії. Він вважав, що досвід раннього дитинства впливає на особистість і поведінку дорослої людини, і запропонував п'ять психосексуальних стадій розвитку особистості [13].

Альфред Адлер, один із соратників Фрейда, погоджувався, що досвід раннього дитинства важливий для розвитку, і вважав, що порядок народження може впливати на розвиток особистості. Гайнц Кохут і Карен Хорні також є важливими фігурами у світі теорії особистості. Кохут розширив теорію нарцисизму Фрейда, ввівши те, що він назвав «перенесенням на себе», віддзеркаленням та ідеалізацією. Карен Хорні не погоджувалася з Фрейдом у деяких ключових моментах і натомість висунула три основні невротичні потреби:

- базова тривога;
- базова ворожість;
- базове зло.

Вона також надавала великого значення таким поняттям, як переоцінка любові та романтичні партнери.

1.1.1.3 Поведінкові теорії

Біхевіористи розглядають особистість як результат зовнішніх стимулів, що впливають на поведінку. Цей підхід відомий як поведінкова теорія або теорія научання, яка суттєво відрізняється від фрейдистської філософії. Розроблена Б.Ф. Скіннером, ця теорія наголошує на науковому мисленні та експериментуванні і підкреслює взаємну взаємодію між індивідами та їхнім оточенням. Модель Скіннера припускає, що поведінка формується через оперантне обумовлювання, коли певна поведінка підкріплюється певними наслідками. Річард Геррнстайн розширив теорію

Скіннера, включивши до неї установки та риси, які, на його думку, мають генетичну або біологічну складову. Класичні експерименти Івана Павлова з обумовлення за участю собак також відіграли значну роль у розвитку біхевіоризму.

1.1.1.4 Соціальні когнітивні теорії

Когнітивна теорія пояснює поведінку як таку, що керується знаннями про світ, зокрема про інших людей. Теорії особистості, що наголошують на когнітивних процесах, таких як мислення і судження, включають роботи Альберта Бандури, який припустив, що пам'ять і емоції працюють під впливом навколишнього середовища. Його «експеримент з лялькою Бобо» продемонстрував силу навчання через спостереження або моделювання. Ранні підходи до когнітивного стилю перераховані Бароном [14] і включають теорію атрибутивного стилю і теорію стилю досягнення. Перша розглядає різні способи, якими люди пояснюють події у своєму житті, тоді як друга зосереджується на визначенні схильності людини до контролю. Робота Вальтера Мішеля відноситься до «Когнітивних афективних одиниць» і розглядає такі фактори, як створення стимулів, афекти, цілепокладання та саморегуляторні переконання. Когнітивно-експериментальна теорія самосвідомості (CEST) стверджує, що люди діють за допомогою двох незалежних систем обробки інформації: емпіричної та раціональної. Психологія особистісних конструктів (PCP) – теорія особистості, розроблена Джорджем Келлі, яка розглядає людей як наївних вчених, що бачать світ через певну лінзу, засновану на їхніх унікально організованих системах конструювання. Психопатологія може виникати через системи конструювання, які не здатні зрозуміти і передбачити мінливий соціальний світ. На основі цієї теорії Келлі розробив психотерапевтичний підхід і техніку під назвою «Інтерв'ю з репертуарною

сіткою», яка допомагала його пацієнтам розкрити власні «конструкти» з мінімальним втручанням чи інтерпретацією з боку терапевта [15].

1.1.1.5 Гуманістичні теорії

Гуманістична психологія наголошує на свободі волі та активній ролі, яку вона відіграє у визначенні поведінки, зосереджуючись на суб'єктивному досвіді, а не на визначальних факторах. Маслоу і Роджерс були прихильниками цього погляду, вивчаючи самоактуалізованих людей, які прагнуть до зростання, щастя і задоволення. Характерними рисами самоактуалізаторів є:

- постійна насолода життям;
- зосередженість на проблемах;
- прийняття та спонтанність;
- невороже почуття гумору.

Гуманістичні теорії підкреслюють тенденцію людської особистості до зростання і самоактуалізації, розглядаючи людину як активну, творчу, чуттєву істоту, яка живе в сьогоденні і суб'єктивно реагує на поточні сприйняття, відносини і зустрічі. Гуманістична терапія покладається на інформацію від клієнта і передбачає індивідуальний підхід, при цьому Роджерс наголошує на рефлексивній або емпатійній реакції, що сприяє глибшому розумінню висловлених почуттів [16].

1.1.1.6 Біопсихологічні теорії

За версією прихильників подібного роду теорій, на розвиток особистості значною мірою впливає біологія, а вивчення генетичних детермінант є ключовим напрямком психології особистості на біологічному рівні. Одним з перших прикладів можливої біологічної основи особистості був Фінеас Гейдж, який пережив зміну особистості після нещасного випадку

в 1848 році. Однак описи цих змін часто перебільшені. Хоча пацієнтів з ушкодженнями мозку було важко вивчати, у 1990-х роках дослідники почали використовувати передові методи візуалізації, такі як електроенцефалограма та томографія для виявлення особистісних рис у мозку.

1.1.1.7 Теорії генетичної основи особистості

Роль генетики у формуванні особистості є предметом суперечок і постійних досліджень з часів проекту «Геном людини». Якщо попередні дослідження були зосереджені на конкретних генах, пов'язаних з певними рисами, то сучасні дослідження зосереджені на активації та експресії генів, пов'язаних з особистістю, в галузі генетики поведінки. Дослідження близнюків показали, що генетика і середовище взаємодіють, визначаючи особистість людини, причому однояйцеві близнюки демонструють вищі кореляції в особистісних рисах, ніж двояйцеві близнюки. Хоча гени забезпечують різні варіанти клітинної експресії, навколишнє середовище в кінцевому підсумку визначає, які гени будуть активовані. Взаємодія ДНК-середовища є важливим фактором у визначенні особистості людини.

1.1.1.8 Еволюційні теорії

Теорія еволюції, введена Чарльзом Дарвіном, є основою еволюційного підходу до психології особистості. Ця теорія вивчає, як природний відбір впливає на розвиток індивідуальних рис особистості [17]. Природний відбір спричиняє зміни в організмах з плином часу через адаптацію та відбір, при цьому певні риси та гени проявляються залежно від середовища, в якому перебуває організм, а також від того, як ці риси допомагають виживати та розмножуватися.

Поліморфізми, такі як стать і група крові, є формами різноманітності, які приносять користь виду в цілому. Особистість, розглянута через призму еволюційної біології, робить акцент на конкретних рисах, які, найімовірніше, допомагають виживати і розмножуватися, таких як сумлінність, товариськість, емоційна стабільність і домінування. Соціальні аспекти особистості можна розглядати з еволюційної точки зору, оскільки певні риси характеру мають вплив на еволюційно важливі речі, включаючи розподіл важливих ресурсів, сімейні та шлюбні взаємодії, а також шкоду або допомогу, яку організми можуть надавати один одному.

1.1.1.9 Теорії потягів

Джон Доллард і Ніл Елгар Міллер розробили теорію особистості в 1930-х роках, яка об'єднала потяги і звички. Вони розглядали особистість як звичні реакції людини, побудовані на набутих, вторинних потягах, які виникають в результаті навчання. Вторинні потяги є більш конкретними проявами первинних потягів, таких як голод, спрага чи потреба в сексуальній активності, і можуть базуватися на кількох первинних потягах, а також на інших вторинних потягах, надаючи їм сили та стійкості [18]. Вони також соціально обумовлені і відрізняються в різних культурах.

Доллард і Міллер вважали, що набуття вторинних потягів є важливим для дитячого розвитку і що психосексуальний розвиток відповідає успішному засвоєнню певних вторинних потягів. Вторинні потяги впливають на наші звичні реакції, включаючи гнів, соціальну конформність, наслідування і тривожність. Люди, які узагальнюють ситуацію, в якій вони відчувають потяг до тривоги, будуть відчувати тривогу набагато більше, ніж повинні, і це може стати частиною їхньої особистості.

1.1.2 Індикатор типів Маєрс-Бріґс

Як вже зазначалось у підрозділі 1.1.1.1, в роботі буде використані представники теорії типів та теорії рис, що є своєрідним нащадком теорії типів. Розглянемо типологію Маєрс-Бріґс.

Індикатор типів Маєрс-Бріґс (MBTI) – це опитувальник, який оцінює психологічні переваги людей у сприйнятті та прийнятті рішень. Він класифікує людей за чотирма критеріями: екстраверсія/інтроверсія, відчуття/інтуїція, мислення/відчуття та судження/сприйняття. Кожна категорія має дві протилежні цінності, а одна цінність з кожної категорії комбінується для створення типу особистості, що складається з чотирьох літер. MBTI був створений Кетрін Кук Бріґс та її дочкою Ізабель Бріґс Маєрс, які перебували під впливом досліджень Карла Юнга.

Характеристика MBTI кожного типу складається з 4 описових рис характеру. Кожна з рис має бінарне значення в залежності від того, яка з них має більш виражений прояв у поведінці людини. Таким чином, бінарні риси, використані в цій системі, разом із кодуванням виглядають так:

- екстраверти/інтроверти (E/I);
- сенсорики/інтуїтиви (S/N);
- мислителі/відчуття (T/F);
- оцінювачі/сприймачі (J/P).

Для опису типу особистості використовується комбінація з 4-х літер (наприклад, ESTJ). Як видно зі схеми складання таких комбінацій, їх може бути всього 16.

Кожен з 16 типів особистості має відмінні риси, які можуть впливати на спосіб життя, вибір професії, стратегії покупок тощо. Список типів особистості з коротким описом можна знайти нижче:

- архітектор (INTJ);
- логік (INTP);
- командир (ENTJ);

- дебатер (ENTP);
- захисник (INFJ);
- посередник (INFP);
- протагоніст (ENFJ);
- агітатор (ENFP);
- логіст (ISTJ);
- захисник (ISFP);
- керівник (ESTJ);
- консул (ESFJ);
- віртуоз (ISTP);
- авантюрист (ISFP);
- підприємець (ESTP);
- конферансьє (ESFP).

Як вже зазначалось, буде вирішуватись задача 4-хкратної бінарної класифікації логістичною регресією, де на виході ми будемо мати подібність ймовірностей належності тому чи іншому класу. Це допоможе зробити систему більш гнучкою.

1.1.3 Загальний огляд попередніх досліджень

В останні роки було проведено достатньо багато досліджень з приводу відношення людини до певного типу за MBTI з залученням підходів машинного та глибокого навчання, а також інших інструментів обробки природньої мови для аналізу написаного людиною тексту. Розглянемо деякі з них.

Дослідження, проведене Планком і Хові у 2015 році [19], було однією з перших спроб класифікувати типи особистості за MBTI на основі відкритого словникового підходу, який не спирався на особистісні лексикони або подібні ресурси. Дослідження було зосереджене на

класифікації особистості в домені Twitter і використовувало логістичну регресію для аналізу n-грам слів, характеристик, пов'язаних з користувачами, таких як стать, і характеристик, пов'язаних з мережею, таких як кількість підписників у соціальних мережах. Результати продемонстрували відносно непогані результати.

У роботі [20] представлено багатомовний набір даних Twitter, позначений інформацією MBTI шістьма мовами (німецькою, голландською, французькою, італійською, португальською та іспанською) під назвою TwiSty corpus. Корпус містить 34 мільйони твітів, авторами яких є понад 18000 користувачів, причому значна частина даних (близько 59%) написана іспанською мовою, що робить інші мови менш представленими (наприклад, 2,2% німецькою та 2,6% італійською). Через природну рідкісність деяких рис особистості, корпус є незбалансованим за класами MBTI (як ми побачимо далі, подібна проблема є одною з ключових у дослідженні). Щоб продемонструвати корисність корпусу, представлено результати лінійного SVM-класифікатора.

Була проведена ще одна спроба [21], присвячена класифікації типів особистості за MBTI на основі даних Twitter індонезійською мовою. Дослідники порівняли різні моделі, включаючи наївну класифікацію Байєса, частотний метод TF-IDF та підрахунок частин мови, щоб визначити найбільш ефективний підхід. Після порівняння моделей було зроблено висновок, що стандартна класифікація тексту за методом наївного Байєса виявилася найкращою.

Дослідження, описане в [22], пропонує ґрунтовне вивчення типології Big Five та класифікації особистості MBTI різними мовами та наборами даних. Запропонований метод використовує декомпозицію одного значення (SVD) для розрізнення контрастних рис особистості, таких як інтроверсія та екстраверсія. Результати показують кращі результати порівняно з тими, що були отримані в роботі [21] для більшості мов, включених до корпусу TwiSty.

У роботі [23] представлено велику колекцію дописів на Reddit під назвою MBTI9K corpus. Корпус містить 354996 дописів, написаних 9872 англомовними користувачами, і є незбалансованим за класами MBTI. На корпусі були проведені різні експерименти, включаючи логістичну регресію, SVM та багатошаровий перцептронний класифікатор (MLP), які використовували різні ознаки тексту, такі як n-грами слів та символів, а також психолінгвістично мотивовані ознаки. Результати показують, що MLP-класифікатори, які використовують весь набір ознак, загалом працюють краще, ніж інші класифікатори.

У дослідницькій роботі Кех і Ченг [24] автори досліджують застосування попередньо навчених мовних моделей для класифікації особистості за шкалою MBTI та генерації мови, залежної від особистості. Вони налаштували попередньо навчену модель BERT на наборі даних онлайн-дискусій, пов'язаних з особистістю, щоб класифікувати текст. Автори повідомляють про точність понад 70% для моделі BERT, що виявилося кращим за результати, отримані з корпусу MBTI9k Reddit [23]. Однак у дослідженні бракує базових результатів з того ж корпусу, що залишає невизначеність щодо переваги моделі. Крім того, незрозуміло, чи є завдання класифікації особистості на основі текстів, пов'язаних з особистістю, простішим, ніж те ж саме завдання, виконане на загальному тексті соціальних мереж.

Дослідження, представлене в [25], вивчає потенційне практичне використання класифікації MBTI для набору проектних команд шляхом аналізу даних соціальних мереж, таких як блоги, Twitter та Stack Overflow. У дослідженні оцінюється модель, яка використовує класифікацію наївного Байєса та підрахунки TF-IDF, і застосовується до вибірки з 40 користувачів з Twitter та Stack Overflow. Результати демонструють, що за текстом можна робити висновки як про особистісні риси, так і про технічні навички, щоб допомогти в процесі рекрутингу.

Дослідження, представлене в [26], зосереджене на класифікації авторського профілю і представляє модель під назвою Author2Vec для вирішення проблеми залежних від автора вбудовувань слів. У статті також обговорюються два подальші застосування для ілюстрації можливого використання цієї моделі, а саме: виявлення депресії та класифікація особистості за MBTI на основі тексту. Для останнього застосування автори використовують логістичний регресійний класифікатор, побудований на основі підмножини вже загаданного корпусу MBTI9k, що містить приблизно половину даних оригінального корпусу. Результати показують, що модель перевершує інші регресійні моделі, засновані на статичних вбудовуваннях Word2vec, підрахунках TF-IDF та моделюванні тем LDA.

Також нещодавні дослідження щодо класифікації особистості за MBTI використовували корпус соціальних мереж, доступний на Kaggle [27], якому бракує детальної інформації про сферу застосування та методи збору даних. Ці дослідження включають порівняння ансамблевих методів, таких як бустінг, пакування та стекінг, з використанням конкатенації підрахунків TF-IDF та вбудовування слів за Дасом та Праджапаті [28], використання випадкового лісу та ознак на основі текстової статистики, таких як довжина речення та розділові знаки Абідіна та ін. [29], а також роботи Хана та ін. [30], Амірхоссейні та Каземіана [31], в обох з яких застосовувалося ансамблеве навчання XGBoost на основі підрахунку кількості слів.

Нарешті, існує робота [32], що підбиває підсумки щодо використання попередніх робіт, перелічених вище, та пропонує рішення на основі BERT, що значно підвищує якість роботи моделі та проводить порівняльний аналіз різних підходів. Автори також зазначають, що більшість даних з попередніх досліджень вже не доступна у відкритому доступі, але нещодавно з’явилися додаткові дані, які можуть бути використані у навчанні моделей. Також, у тексті роботи не звертаються до процедури з приводу вирішення вже зазначеної проблеми незбалансованих даних, але, виходячи з представленої у дослідженні інформації стосовно набору даних, використаного авторами,

дисбаланс є дуже великим. Попри все це, результати у вигляді міри F1 [33] є дуже вражаючими.

1.1.4 Проміжні висновки

Виходячи із представлених вище даних, можна зробити висновки, що тема дослідження та варіативність підходів до робочого матеріалу, а також цілий ланцюг щорічних релізів нових результатів роботи груп авторів наголошує на актуальності роботи, реальності створення достатньо якісної моделі для вирішення задачі класифікації текстів людей за типами особистості, а також додає наукових базис до майбутніх дослідників.

Також слід наголосити на тому, що подібні технології мають дуже великий рівень придатності у різних сферах: від медицини та прогнозування розладів людини до використання її слабкостей у продажах товарів чи послуг та сильних сторін при прийомі на роботу.

1.2 Постановка задачі

Проаналізувавши предметну область, було поставлено завдання: дослідити та реалізувати окремі модулі аналізу тексту засобами штучного інтелекту та інструментів обробки природньої мови для класифікації типів особистості за MBTI та приклад використання отриманого знання на практиці для персоналізації текстових повідомлень, інтегрувати це у тестову веб-платформу, яка виконувала б примітивний стек дій:

– розпізнавання типів особистості за введеним текстом;

– відображення результатів у вигляді псевдо-ймовірностей належності до певних типів за MBTI;

– формулювання текстових повідомлень, орієнтованих на людей певного типу особистості за MBTI, генеративними моделями.

2 ПІДГОТОВЧІ ДОСЛІДЖЕННЯ ТА МЕТОДИ ВИРІШЕННЯ ЗАДАЧІ

У цьому розділі ми обговоримо теоретичні дослідження та методи, що використовуються для вирішення проблеми психологічного профілювання людей за допомогою обробки природної мови. Зокрема, ми зосередимося на моделях і алгоритмах машинного навчання, які використовуються в наших дослідженнях, а також на інших методах, таких як аналіз настроїв і тематичне моделювання, обговоримо необхідні ресурси для роботи та зробимо огляд системи, що розроблюється, з позиції пташиного ока.

2.1 Методи класифікації тексту за MBTI

2.1.1 Моделі та алгоритми машинного навчання

Для проведення нашого дослідження щодо класифікації авторів текстів за MBTI ми використовували різноманітні моделі та алгоритми машинного навчання. Серед них були такі:

– моделі на основі BERT: BERT (Bidirectional Encoder Representations from Transformers) – це попередньо навчена мовна модель, яка продемонструвала найсучаснішу продуктивність на широкому спектрі завдань NLP, включаючи класифікацію текстів. Ми використовували моделі на основі BERT, налаштовані на нашому наборі даних, для класифікації типів MBTI авторів;

– логістична регресія: логістична регресія – це простий, але ефективний алгоритм для задач бінарної класифікації. Ми використали логістичну регресію для класифікації вимірів MBTI (наприклад, екстраверсія проти інтроверсії) авторів;

– лінійна машина опорних векторів (SVM): SVM – це популярний алгоритм для задач класифікації, який працює шляхом пошуку

гіперплощини, що найкраще розділяє точки даних на різні класи. Ми використали лінійну SVM для класифікації типів MBTI авторів;

– мультиноміальний наївний Байєс: наївний Байєс – це імовірнісний алгоритм, який часто використовується для задач класифікації тексту. Ми використали мультиноміальний наївний Байєс для класифікації вимірів MBTI авторів;

– дерево рішень: дерево рішень – це непараметричний алгоритм, який працює шляхом рекурсивного розбиття даних на підмножини на основі ознак. Ми використали дерево рішень для класифікації типів MBTI авторів;

– випадковий ліс: випадковий ліс – це ансамблевий алгоритм, який поєднує декілька дерев рішень для покращення продуктивності та зменшення перенавчання. Ми використали випадковий ліс для класифікації типів MBTI авторів.

Вибір цих моделей та алгоритмів ґрунтувався на їхній ефективності при вирішенні подібних завдань та придатності для нашого набору даних.

2.1.2 Словниковий аналіз

До моделей і алгоритмів машинного навчання, перерахованих вище, ми також можемо додати декілька інших методів, щоб дослідити можливість вирішення проблеми, описаної в цій роботі, за допомогою простих словникових підходів. Однією з таких методик був інструментарій Linguistic Inquiry and Word Count (LIWC-22).

LIWC-22 – це програмний інструмент, який можна використовувати для аналізу текстових даних на основі категорій слів (наприклад, позитивні емоції, негативні емоції, соціальні слова тощо). Ми використовували LIWC-22 для вилучення лінгвістичних особливостей з текстових даних та дослідження їх кореляції з вимірами та типами MBTI. Зокрема, ми використовували LIWC-22 для обчислення відсоткового співвідношення слів у кожній з категорій для кожної вибірки текстів.

2.1.3 Аналіз настрою

Аналіз настрою – це метод, який використовується для визначення емоційного тону тексту. Він застосовується в особистісному профілюванні для виявлення закономірностей у тому, як люди використовують мову для вираження емоцій, що може дати уявлення про їхні особистісні риси [34]. У нашому дослідженні ми використали аналіз настроїв, щоб відстежити зв'язок між емоційним тоном тексту та вимірами і типами MBTI.

Для проведення аналізу сентиментів ми використали алгоритм VADER (Valence Aware Dictionary and sEntiment Reasoner – словник з урахуванням валентності та сентиментного аналізу). VADER – це алгоритм на основі лексики, який присвоює кожному реченню в тексті оцінку позитивного, негативного або нейтрального настрою. Потім ми об'єднали ці оцінки, щоб обчислити загальний настрій кожного зразка тексту.

2.1.4 Тематичне моделювання

Тематичне моделювання – це метод, який використовується для визначення основних тем або тем у тексті. Вона використовується в особистісному профілюванні для виявлення закономірностей у тому, як люди використовують мову для вираження своїх інтересів і проблем, що може дати уявлення про їхні особистісні риси. У нашому дослідженні ми використовували моделювання тем для вивчення зв'язку між темами, що обговорюються в тексті, та вимірами і типами MBTI.

Ми використали алгоритм латентного розподілу Діріхле (Latent Dirichlet Allocation, LDA) для моделювання тем на наших текстових даних. LDA – це генеративний імовірнісний алгоритм, який моделює кожен документ як суміш тем, де тема – це розподіл ймовірностей над словами. Ми використали LDA для визначення тем, які обговорюються в тексті, та розподілу тем для кожного автора.

2.2 Огляд метрік для оцінки якості моделювання

Для роботи з моделями необхідно постановити, як саме буде проходити порівняння їх якості роботи. На роль подібного маркеру підходить метрика F1 [33], що є середнім геометричним інших двох метрик: precision та recall.

Precision оцінює якість класифікації за кількістю помилок 1-го типу. Це можна побачити у формулі 2.1.

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{2.1}$$

де TP – кількість елементів true positive;

FP – кількість елементів false positive.

Recall оцінює якість класифікації за кількістю помилок 2-го типу. Це можна побачити у формулі 2.2.

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{2.2}$$

де TP – кількість елементів true positive;

FN – кількість елементів false negative.

Як вже зазначалось, метрика F1 є середнім геометричним двох попередніх метрик. Тому знаходимо середнє геометричне значення в загальному вигляді у формулі 2.3 через формули 2.1 та 2.2.

$$\text{F1 score} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}}, \tag{2.3}$$

де TP – кількість елементів true positive;

FP – кількість елементів false positive;

FN – кількість елементів false negative.

Також слід зазначити, що при класифікації на декілька класів за клас positive приймається тільки розглядаємий на поданий момент, а всі інші об’єднуються у negative.

2.3 Огляд даних

Для вирішення поставленої задачі буде використано набір даних з Kaggle, популярної онлайн-платформи для конкурсів та проектів у галузі Data Science, під назвою «MBTI Personality Types 500 Dataset» [35]. Набір даних містить загалом більше 106607 текстів, зібраних з платформ Reddit та PersonalityCafe. Розмір кожного запису обмежений 500 словами. Набір даних має ліцензію CC0, що дозволяє використовувати його з будь-якою метою, включаючи наукові дослідження.

Набір даних є дуже незбалансованим. Цей дисбаланс створює проблему для завдання класифікації, оскільки моделі можуть мати тенденцію класифікувати документи до більш поширених класів. Детальніше можна побачити на рисунку 2.1.

Ми можемо побачити, що різниця в кількості даних між класами просто колосальна: на клас INTP припадає близько 24% всіх даних, тоді як на INTJ – 21%, що вже складає в сумі трохи менше половини даних, які ніяк не підходять для забезпечення якісного навчання моделі. Тим не менш, цей набір даних буде використано в експериментах, щоб продемонструвати невдачу попередніх експериментів у минулих дослідженнях, де не було проведено балансування даних.

Для вирішення задачі балансування даних за класами будуть використані техніки роботи з текстовими та іншими типами даних. Також слід зазначити, що зміни у наборі даних можуть призвести до несподіваних результатів, що ми побачимо під час проведення експериментів.

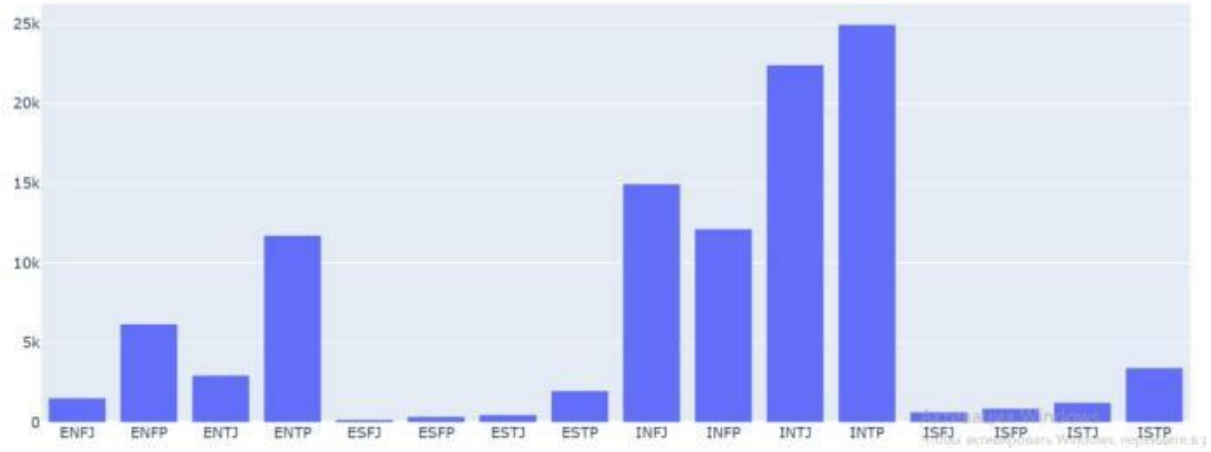

Рисунок 2.1 – Розподіл даних по класах у наборі даних

Комбінування даних по кортежах з обмеженням кількості слів дає дуже гарну основу для використання моделей із фіксованим вікном контексту. Також слід зазначити, що кількість окремих приладів з набору даних не відповідає кількості користувачів, що є авторами даних: на одного автора приходиться кілька прикладів текстів, написаних ним.

2.4 Інженерна складова дослідження

Для проведення експериментів, пов’язаних із машинним навчанням, а також для імплементації основних компонентів системи буде використано мову програмування Python 3.10 та цілий ряд бібліотек, що можна використовувати у сполученні із цією мовою (pandas, PyTorch Lightning, Transformers, openai, Langchain та інші). Ця мова програмування є дуже поширеною, а також має велику спільноту людей, що її використовують, що надає можливості її постійного розвитку та доповнення рішень з відкритим кодом на базі цієї мови.

Для реалізації деяких компонентів системи, що відносяться до відображення візуального інтерфейсу та взаємодії із кінцевим користувачем, буде використано мову програмування TypeScript. Це підвисить надійність розробленого продукту у порівняні із традиційною

мовою програмування JavaScript, що використовується для подібних задач. Обидві мови є стандартами EcmaScript, тому сумісні для цілої низки браузерів.

Для підвищення мобільності платформи та її надійності, а також спрощення циклу зборки, тестування та розгортання на сервері слід використовувати контейнеризацію. Для подібного завдання буде використовуватись Docker. Це сучасний інструмент з низкою важливого функціоналу для роботи із імеджами та контейнерами додатків.

Для зручного користування декількома моделями всередині додатку, а також для надання можливості масштабування, якщо буде велике навантаження на сервіс, буде використано оркестратор Kubernetes.

2.5 Планування циклу дослідження

Загалом, увесь цикл дослідження буде складатись з наступних кроків, що є обов’язковими для вирішення поставленого завдання:

- огляд предметної галузі;
- огляд попередніх досліджень;
- вибір інструментів та методів дослідження;
- знаходження ресурсів для дослідження;
- проведення експериментів та вибір найкращого підходу;
- програмна реалізація для використання розробок експериментів.

Наразі, поданий фрагмент тексту вже знаходиться на третьому та четвертому етапі, тобто огляд предметної галузі та попередніх досліджень вже було проведено.

В ході подальших кроків у дослідженні буде проведена низка експериментів для знаходження найкращого підходу з окреслених вище, а також створена програмна реалізація для кращої ілюстрації працездатності системи.

2.6 Проміжні висновки

У розділі 2 були розглянуті основні моменти планування подальших робіт, а також проведено попередню підготовку до дослідження. Були зазначені основні підходи до аналізу тексту, що проводились із даними під час попередніх досліджень, окреслено ресурс для набору даних та зроблено його загальний огляд, а також вказані інструменти, що будуть використовуватись як протягом проведення експериментів, так і протягом розробки безпосередньо платформи.

Також слід зазначити, що під час проведення експериментів буде використано лише методи обробки, зазначені у пунктах 2.1.1 та 2.1.2, бо використання інших методів потребує значного розширення обсягу досліджень.

3 ПРОВЕДЕННЯ ЕКСПЕРИМЕНТІВ

3.1 Середа та обчислювальні ресурси для проведення експериментів

Для проведення експериментів необхідно мати доступну середу розробки, яка б гарантувала швидкість та відповідала наступним критеріям:

– Jupyter Notebook-подібне середовище;

– доступ до обчислювальних ресурсів графічних процесорів (GPU);

– підтримка мови програмування Python версії 3.6 та вище;

– можливість швидко оперувати даними за рахунок нативної інтеграції сховища;

– достатні ресурси для роботи із великими за розміром моделями.

Якщо не зважати на останній пункт, то в якості середи розробки підійшов би Google Colab, але безкоштовні віртуальні машини, що є основою цієї платформи, не мають необхідного для стабільної роботи набору ресурсів. Тому було прийнято рішення використати платформу Azure Machine Learning. Вона відповідає всім критеріям, що описані вище, а також дозволяє запускати блокноти, подібні до Jupyter Notebook, на створених машинах із кастомізацією набору ресурсів.

Azure Machine Learning є продуктом компанії Microsoft та включений до сервісів хмарної платформи Microsoft Azure. Функціонал дозволяє проводити експерименти із машинним навчанням, реєструвати артефакти експериментів, використовувати вбудований простір для збереження моделей для їх подальшого використання, а також закладено можливість запускати нативне розгортання моделі з API для доступу з інших розроблених додатків. Подібні рішення значно спрощують роботу над дослідженнями та дозволяють зосередитись над плануванням та імплементацією безпосередньо самого дослідження.

На рисунку 3.1 можна побачити інтерфейс платформи.

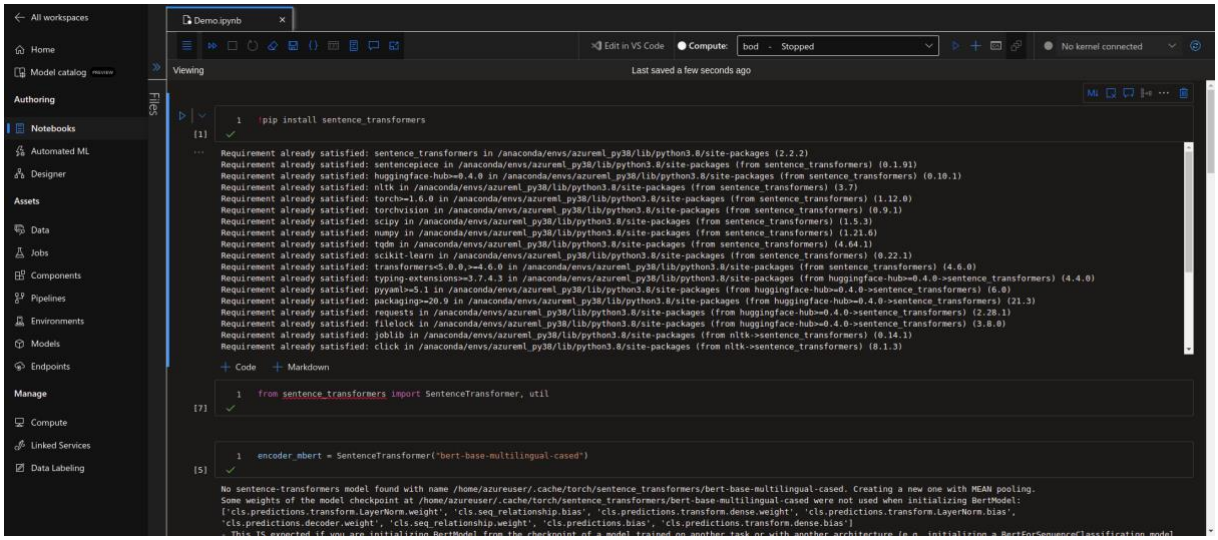

Рисунок 3.1 – Інтерфейс платформи Azure Machine Learning

Для роботи з великими моделями буде використано віртуальну машину під назвою Standard_NC6, у якої 6 ядер центрального процесору (CPU), 56 гігабайт (ГБ) оперативної пам’яті та 380 ГБ простору на твердотільному накопичувачі (SSD). Для прискорення тензорних обчислень включена відеокарта NVIDIA Tesla K80 з 24 ГБ пам’яті та відносно великою потужністю на ринку GPU.

3.2 Робота з набором даних

Як вже зазначалось, буде використаний корпус даних з платформи Kaggle з достатньо великою кількістю семплів для тренування навіть великих моделей (наприклад, BERT).

Перші дослідження набору показали, що дисбаланс даних достатньо великий: 2 класи з 16 займають майже половину всіх прикладів, що при неправильній роботі може спричинити неякісний інференс моделі. Тому треба провести низку операцій з обробки даних перед використанням у навчанні.

Подібний дисбаланс спричинений не стільки неякісною процедурою збору даних, скільки існуючим дисбалансом у соціумі представників тих чи

інших типів MBTI, а також різними показниками прояву активності у мережі Інтернет різними типами. Такий bias доведеться вирішувати програмними засобами.

Структура набору даних достатньо проста: це таблиця даних, де кожен кортеж має лише 2 атрибути: сам текст, за яким будет проводитись передбачення моделі під назвою «posts» та назва типу за MBTI під назвою «type».

3.2.1 Підхід з обрізанням набору даних

Найбільш очевидним здається спосіб зведення до балансу шляхом обрізання «зайвих» семплів даних, бо даних достатньо багато і навіть без певної частини цих даних, але з більшим процентом збалансованості треба провести експерименти щодо якості класифікації.

Для подібної операції було обрано число в 5000 екземплярів кожного класу: таким чином, класи, що мають більше за вказаний обсяг даних будуть зменшені в контексті кількості семплів.

На рисунку 3.2 зображена стовбчаста діаграма кількості екземплярів кожного класу.

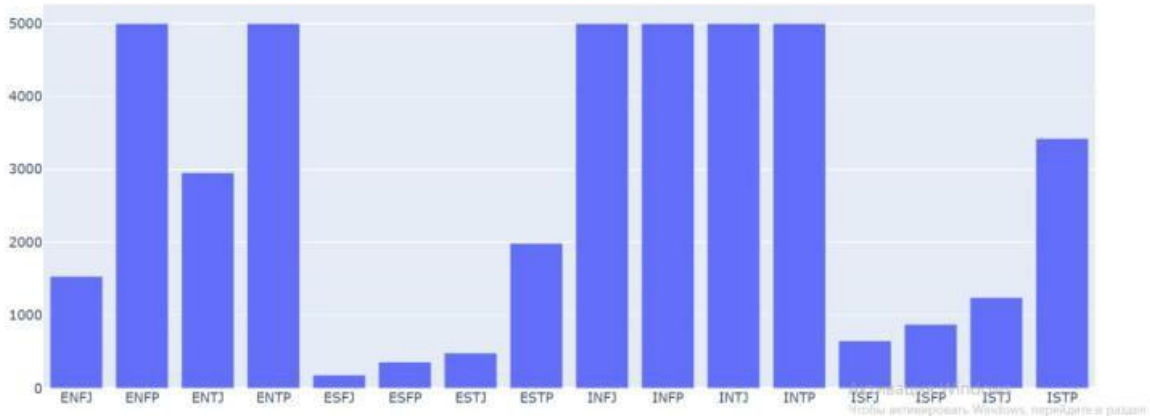

Рисунок 3.2 – Кількість даних у кожному класі всередині обрізаного набору даних

3.2.2 Підхід із синтетичними даними

У цьому дослідженні не буде використано Техніка Передискретизації Синтетичної Меншини (SMOTE), який наразі є досить популярним у задачах NLP, оскільки він вимагає формування векторного представлення для кожного окремого тексту до латентного простору, що суперечить ідеям використання деяких обраних підходів машинного навчання для роботи з нашими даними. Це ж зауваження стосується і підходу доповнення даних або створення синтетичного набору даних.

Ми бачимо, що картина балансування даних стала значно кращою, але все ж таки вона не може претендувати на те, щоб бути якісним набором даних для навчання. Тому ми вирішили використати третьою версію набору даних, де на додаток до операції випадкового вирізання даних з класів, які є занадто великими, ми зробимо кілька копій вже наявних даних у класах, де їх мало. Враховується також подальше навчання: дані, які будуть присутні у валідаційній вибірці, не будуть копіюватися для збільшення вибірки певного класу. Результат копіювання даних можна побачити на рисунку 3.3.

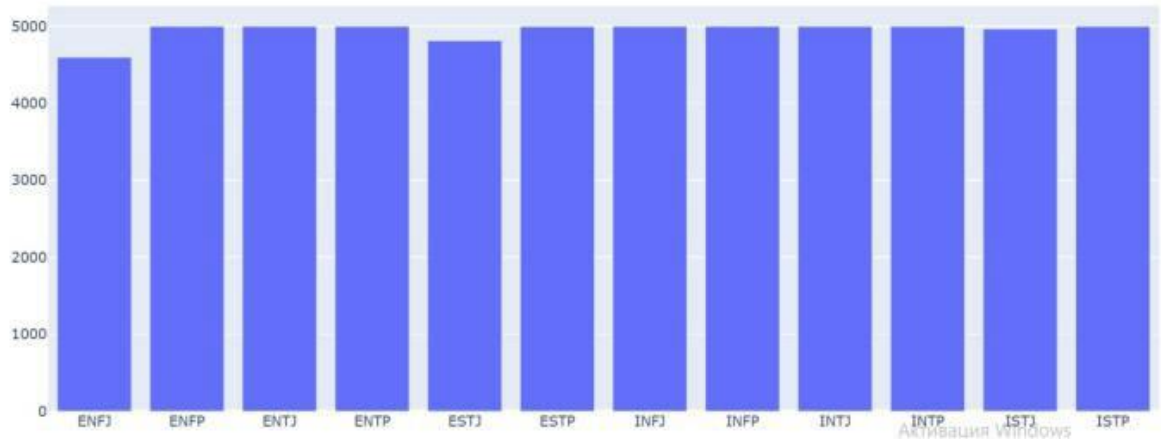

Рисунок 3.3 – Кількість даних у кожному класі всередині скопійованого набору даних

3.2.3 Передобробка текстових даних

Для роботи з текстовими даними, окрім загального балансування набору даних, необхідно виконати деякі стандартні процедури обробки для покращення чутливості моделей до критично важливих параметрів.

Так, під час попередньої обробки тексту ми відмовилися від ідеї фільтрації за стоп-словами, віддавши перевагу лише стандартній лематизації та фільтрації непотрібного сміття з тексту (посилання, спеціальні символи і т.д.). Це було здебільшого зроблено для адаптації моделі до реальних умов аналізу тексту, а також для підвищення рівня її надійності, монолітності та скорочення часу і процесу передобробки.

3.2.4 Проміжні висновки стосовно набору даних

Таким чином, ми маємо 3 версії набору даних для подальшого дослідження, а саме

- набір даних за замовчуванням;
- обрізаний набір даних;
- скопійований набір даних.

Всі 3 версії набору даних будуть використані для ілюстрації важливості правильного балансування даних перед навчанням моделі та ілюстрації впливу якості даних на якість навчання та використання моделі.

3.3 Навчання моделей

Через те, що у минулому розділі було розібрано роботи, пов’язані із набором даних, а також презентовано 3 типи набору, кількість результатів для порівняння один з одним буде дорівнювати кількості моделей, що ми збираємось протестувати, помножене на кількість набору даних. У нашому випадку це 18. Також слід зазначити, що для більш точних показників для

кожної з 18 отриманих моделей проводились 10 замірів якості навчання з різним поділом тестової та навчальної вибірки, а також з різними значеннями ініціалізації моделей (стосується більше моделей, заснованих на засадах штучних нейронних мереж). Результатом якості буде середнє значення за 10 експериментів.

На рисунку 3.4 схематично зображено проведення експериментів під час навчання моделей.

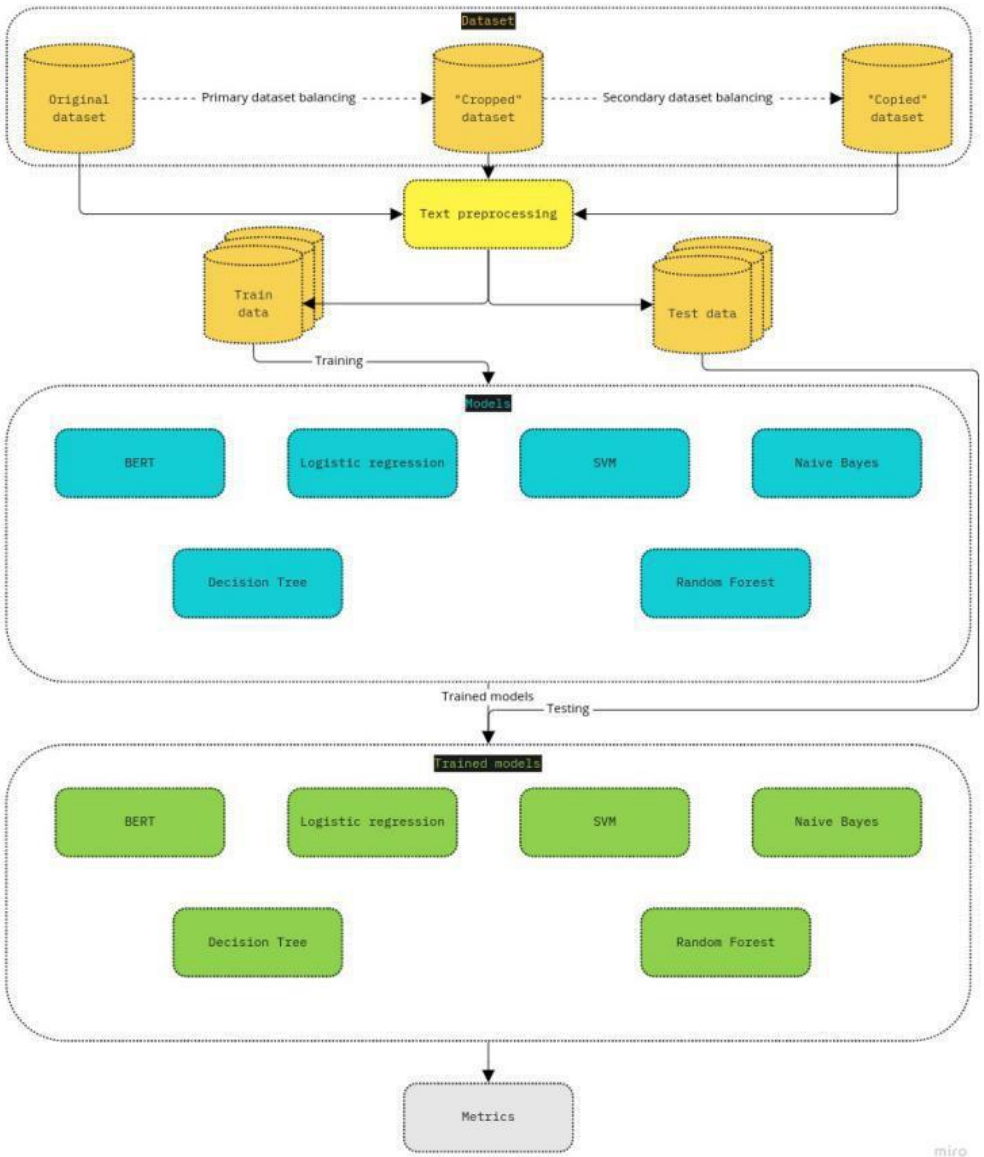

Рисунок 3.4 – Схема проведення експериментів з навчанням моделей

3.3.1 Експерименти з логістичною регресією

Для проведення експериментів із логістичною регресією використовувався класичний пакет модулю Scikit-learn. Невелику частину коду для виконання навчання та тестування можна побачити у лістингу 3.1.

Лістинг 3.1 – Використання логістичної регресії для класифікації текстів за MBTI

```
model_log=LogisticRegression(max_iter=3000,C=0.5,n_jobs=-1)
model_log.fit(train_post,train_target)
LogisticRegression(C=0.5, max_iter=3000, n_jobs=-1)
```

Тепер розберемо результати, отримані в результаті навчання моделі. Почнемо з початкового набору даних. Результати навчання моделі за допомогою нього можна побачити у таблиці 3.1 (показники за тестовими даними).

Таблиця 3.1 – Результати класифікації текстів за допомогою логістичної регресії та набору даних за замовчуванням

Назва типу	Значення F1-score	Кількість екземплярів
1	2	3
ENFJ	0.56	307
ENFP	0.79	1233
ENTJ	0.75	591
ENTP	0.81	2345
ESFJ	0.24	36
ESFP	0.36	72
ESTJ	0.75	96
ESTP	0.89	397
INFJ	0.81	2993

Продовження таблиці 3.1

1	2	3
INFP	0.80	2427
INTJ	0.83	4486
INTP	0.84	4992
ISFJ	0.45	130
ISFP	0.47	175
ISTJ	0.50	249
ISTP	0.78	685
Усього	0.8	21214

У таблиці 3.2 можна побачити результати класифікації текстів за допомогою логістичної регресії та скороченого («обрізаного») набору даних.

Таблиця 3.2 – Результати класифікації текстів за допомогою логістичної регресії та «обрізаного» набору даних

Назва типу	Значення F1-score	Кількість екземплярів
1	2	3
ENFJ	0.73	307
ENFP	0.84	1000
ENTJ	0.85	591
ENTP	0.81	1000
ESFJ	0.24	36
ESFP	0.49	72
ESTJ	0.78	96
ESTP	0.94	397
INFJ	0.79	1000
INFP	0.79	1000

Продовження таблиці 3.2

1	2	3
INTJ	0.78	1000
INTP	0.79	1000
ISFJ	0.64	130
ISFP	0.67	175
ISTJ	0.71	249
ISTP	0.86	685
Усього	0.80	8738

Як ми бачимо, загальний показник F1-score не змінився, це може свідчити про 2 речі:

– дані такої ж якості;

– модель не надто чутлива до кількості даних.

Детальний аналіз буде проведено після отримання результатів за всіма моделями.

Тепер перейдемо до тестування «скопійованого» набору даних та моделі, що на ньому навчалась. Результати можна побачити у таблиці 3.3.

Таблиця 3.3 – Результати класифікації текстів за допомогою логістичної регресії та «скопійованого» набору даних

Назва типу	Значення F1-score	Кількість екземплярів
1	2	3
ENFJ	0.88	718
ENFP	0.84	805
ENTJ	0.89	953
ENTP	0.81	800
ESFJ	0.99	790
ESFP	0.97	801

Продовження таблиці 3.3

1	2	3
ESTJ	0.99	808
ESTP	0.96	925
INFJ	0.80	807
INFP	0.79	794
INTJ	0.77	804
INTP	0.76	804
ISFJ	0.96	806
ISFP	0.92	825
ISTJ	0.92	794
ISTP	0.92	1094
Усього	0.89	13328

Як ми бачимо, найкращі результати серед всіх за показниками загального значення F1-score саме з останнім набором даних. Гіпотеза щодо покращення результатів за рахунок балансу даних частково підтверджується.

3.3.2 Експерименти з деревом рішень

Для проведення експериментів із деревом рішень використовувався класичний пакет модулю Scikit-learn. Невелику частину коду для виконання навчання та тестування можна побачити у лістингу 3.2.

Лістинг 3.2 – Використання дерева рішень для класифікації текстів за MBTI

```
model_tree=DecisionTreeClassifier(max_depth=14)
model_tree.fit(train_post,train_target)
DecisionTreeClassifier(max_depth=14)
```

Тепер розберемо результати, отримані в результаті навчання моделі. Почнемо з початкового набору даних. Результати навчання моделі за допомогою нього можна побачити у таблиці 3.4 (показники за тестовими даними).

Таблиця 3.4 – Результати класифікації текстів за допомогою дерева рішень та набору даних за замовчуванням

Назва типу	Значення F1-score	Кількість екземплярів
ENFJ	0.08	307
ENFP	0.58	1233
ENTJ	0.33	591
ENTP	0.59	2345
ESFJ	0.00	36
ESFP	0.00	72
ESTJ	0.04	96
ESTP	0.60	397
INFJ	0.65	2993
INFP	0.64	2427
INTJ	0.73	4486
INTP	0.69	4992
ISFJ	0.04	130
ISFP	0.12	175
ISTJ	0.04	249
ISTP	0.53	685
Усього	0.62	21214

У таблиці 3.5 можна побачити результати класифікації текстів за допомогою дерева рішень та скороченого («обрізаного») набору даних.

Таблиця 3.5 – Результати класифікації текстів за допомогою дерева рішень та «обрізаного» набору даних

Назва типу	Значення F1-score	Кількість екземплярів
ENFJ	0.04	307
ENFP	0.46	1000
ENTJ	0.23	591
ENTP	0.39	1000
ESFJ	0.05	36
ESFP	0.00	72
ESTJ	0.00	96
ESTP	0.41	397
INFJ	0.42	1000
INFP	0.01	1000
INTJ	0.28	1000
INTP	0.29	1000
ISFJ	0.00	130
ISFP	0.00	175
ISTJ	0.00	249
ISTP	0.02	685
Усього	0.25	8738

Як ми бачимо, загальний показник F1-score змінився у гірший бік, це може свідчити про 2 речі:

– дані гіршої якості;

– модель надто чутлива до кількості даних.

Детальний аналіз буде проведено після отримання результатів за всіма моделями.

Тепер перейдемо до тестування «скопійованого» набору даних та моделі, що на ньому навчалась. Результати можна побачити у таблиці 3.6.

Таблиця 3.6 – Результати класифікації текстів за допомогою дерева рішень та «скопійованого» набору даних

Назва типу	Значення F1-score	Кількість екземплярів
ENFJ	0.67	718
ENFP	0.64	805
ENTJ	0.71	953
ENTP	0.49	800
ESFJ	0.80	790
ESFP	0.83	801
ESTJ	0.91	808
ESTP	0.78	925
INFJ	0.54	807
INFP	0.49	794
INTJ	0.01	804
INTP	0.27	804
ISFJ	0.80	806
ISFP	0.78	825
ISTJ	0.67	794
ISTP	0.70	1094
Усього	0.63	13328

Як ми бачимо, найкращі результати серед всіх за показниками загального значення F1-score саме з останнім набором даних. Гіпотеза щодо покращення результатів за рахунок балансу даних частково підтверджується. Слід також зазначити, що покращення зовсім невелике. Під час формування фінальних висновків це буде взято до уваги.

3.3.3 Експерименти з випадковим лісом

Для проведення експериментів із випадковим лісом використовувався класичний пакет модулю Scikit-learn. Невелику частину коду для виконання навчання та тестування можна побачити у лістингу 3.3.

Лістинг 3.3 – Використання випадкового лісу для класифікації текстів за MBTI

```
model_forest=RandomForestClassifier(max_depth=10)
model_forest.fit(train_post,train_target)
RandomForestClassifier(max_depth=10)
```

Тепер розберемо результати, отримані після навчання моделі. Почнемо з початкового набору даних. Результати навчання моделі за допомогою нього можна побачити у таблиці 3.7 (показники за тестовими даними).

Таблиця 3.7 – Результати класифікації текстів за допомогою випадкового ліса та набору даних за замовчуванням

Назва типу	Значення F1-score	Кількість екземплярів
1	2	3
ENFJ	0.00	307
ENFP	0.18	1233
ENTJ	0.00	591
ENTP	0.35	2345
ESFJ	0.00	36
ESFP	0.00	72
ESTJ	0.34	96
ESTP	0.45	397
INFJ	0.60	2993

Продовження таблиці 3.7

1	2	3
INFP	0.55	2427
INTJ	0.67	4486
INTP	0.58	4992
ISFJ	0.00	130
ISFP	0.00	175
ISTJ	0.00	249
ISTP	0.00	685
Усього	0.49	21214

У таблиці 3.8 можна побачити результати класифікації текстів за допомогою випадкового лісу та скороченого («обрізаного») набору даних.

Таблиця 3.8 – Результати класифікації текстів за допомогою випадкового лісу та «обрізаного» набору даних

Назва типу	Значення F1-score	Кількість екземплярів
1	2	3
ENFJ	0.00	307
ENFP	0.00	1000
ENTJ	0.00	591
ENTP	0.00	1000
ESFJ	0.00	36
ESFP	0.00	72
ESTJ	0.00	96
ESTP	0.05	397
INFJ	0.00	1000
INFP	0.00	1000
INTJ	0.22	1000

Продовження таблиці 3.8

1	2	3
INTP	0.24	1000
ISFJ	0.00	130
ISFP	0.00	175
ISTJ	0.00	249
ISTP	0.00	685
Усього	0.06	8738

Як ми бачимо, загальний показник F1-score змінився у сильно гірший бік, це може свідчити про 2 речі:

– дані гіршої якості;

– модель надто чутлива до кількості даних.

Детальний аналіз буде проведено після отримання результатів за всіма моделями.

Тепер перейдемо до тестування «скопійованого» набору даних та моделі, що на ньому навчалась. Результати можна побачити у таблиці 3.9.

Таблиця 3.9 – Результати класифікації текстів за допомогою випадкового лісу та «скопійованого» набору даних

Назва типу	Значення F1-score	Кількість екземплярів
1	2	3
ENFJ	0.70	718
ENFP	0.66	805
ENTJ	0.73	953
ENTP	0.48	800
ESFJ	0.92	790
ESFP	0.86	801
ESTJ	0.94	808

Продовження таблиці 3.9

1	2	3
ESTP	0.82	925
INFJ	0.61	807
INFP	0.47	794
INTJ	0.48	804
INTP	0.52	804
ISFJ	0.78	806
ISFP	0.77	825
ISTJ	0.71	794
ISTP	0.53	1094
Усього	0.68	13328

Як ми бачимо, найкращі результати серед всіх за показниками загального значення F1-score саме з останнім набором даних. Гіпотеза щодо покращення результатів за рахунок балансу даних частково підтверджується. Слід також зазначити, що покращення велике. Під час формування фінальних висновків це буде взято до уваги.

3.3.4 Експерименти з мультиноміальним наївним Байєсом

Для проведення експериментів із наївним Байєсом використовувався класичний пакет модулю Scikit-learn. Невелику частину коду для виконання навчання та тестування можна побачити у лістингу 3.4.

Лістинг 3.4 – Використання наївного Байєсу для класифікації текстів за MBTI

```
model_multinomial_nb=MultinomialNB()
model_multinomial_nb.fit(train_post,train_target)
MultinomialNB()
```

Тепер розберемо результати, отримані в результаті навчання моделі. Почнемо з початкового набору даних. Результати навчання моделі за допомогою нього можна побачити у таблиці 3.10 (показники за тестовими даними).

Таблиця 3.10 – Результати класифікації текстів за допомогою наївного Байєсу та набору даних за замовчуванням

Назва типу	Значення F1-score	Кількість екземплярів
ENFJ	0.05	307
ENFP	0.16	1233
ENTJ	0.20	591
ENTP	0.37	2345
ESFJ	0.00	36
ESFP	0.00	72
ESTJ	0.66	96
ESTP	0.79	397
INFJ	0.63	2993
INFP	0.54	2427
INTJ	0.66	4486
INTP	0.65	4992
ISFJ	0.00	130
ISFP	0.00	175
ISTJ	0.00	249
ISTP	0.21	685
Усього	0.52	21214

У таблиці 3.11 можна побачити результати класифікації текстів за допомогою наївного Байєсу та скороченого («обрізаного») набору даних.

Таблиця 3.11 – Результати класифікації текстів за допомогою наївного Байєсу та «обрізаного» набору даних

Назва типу	Значення F1-score	Кількість екземплярів
ENFJ	0.04	307
ENFP	0.00	1000
ENTJ	0.03	591
ENTP	0.07	1000
ESFJ	0.02	36
ESFP	0.02	72
ESTJ	0.05	96
ESTP	0.43	397
INFJ	0.00	1000
INFP	0.01	1000
INTJ	0.15	1000
INTP	0.24	1000
ISFJ	0.39	130
ISFP	0.34	175
ISTJ	0.13	249
ISTP	0.03	685
Усього	0.10	8738

Як ми бачимо, загальний показник F1-score змінився у гірший бік, це може свідчити про 2 речі:

– дані гіршої якості;

– модель надто чутлива до кількості даних.

Детальний аналіз буде проведено після отримання результатів за всіма моделями.

Тепер перейдемо до тестування «скопійованого» набору даних та моделі, що на ньому навчалась. Результати можна побачити у таблиці 3.12.

Таблиця 3.12 – Результати класифікації текстів за допомогою наївного Байєсу та «скопійованого» набору даних

Назва типу	Значення F1-score	Кількість екземплярів
ENFJ	0.81	718
ENFP	0.70	805
ENTJ	0.83	953
ENTP	0.67	800
ESFJ	0.98	790
ESFP	0.94	801
ESTJ	0.90	808
ESTP	0.92	925
INFJ	0.65	807
INFP	0.62	794
INTJ	0.61	804
INTP	0.61	804
ISFJ	0.88	806
ISFP	0.73	825
ISTJ	0.73	794
ISTP	0.77	1094
Усього	0.79	13328

Як ми бачимо, найкращі результати серед всіх за показниками загального значення F1-score саме з останнім набором даних. Гіпотеза щодо покращення результатів за рахунок балансу даних частково підтверджується.

Також слід зазначити, що контраст з даними другого набору даних та третього дуже великий. Це буде відображено у висновках.

3.3.5 Експерименти з лінійним класифікатором опорних векторів

Для проведення експериментів із лінійним класифікатором опорних векторів (SVM) використовувався класичний пакет модулю Scikit-learn. Невелику частину коду для виконання навчання та тестування можна побачити у лістингу 3.5.

Лістинг 3.5 – Використання SVM для класифікації текстів за MBTI

```
model_linear_svc=LinearSVC(C=0.1)
model_linear_svc.fit(train_post,train_target)
LinearSVC(C=0.1)
```

Тепер розберемо результати, отримані в результаті навчання моделі. Почнемо з початкового набору даних. Результати навчання моделі за допомогою нього можна побачити у таблиці 3.13 (показники за тестовими даними).

Таблиця 3.13 – Результати класифікації текстів за допомогою SVM та набору даних за замовчуванням

Назва типу	Значення F1-score	Кількість екземплярів
1	2	3
ENFJ	0.61	307
ENFP	0.79	1233
ENTJ	0.78	591
ENTP	0.81	2345
ESFJ	0.43	36
ESFP	0.48	72
ESTJ	0.79	96
ESTP	0.91	397
INFJ	0.82	2993

Продовження таблиці 3.13

1	2	3
INFP	0.81	2427
INTJ	0.84	4486
INTP	0.85	4992
ISFJ	0.57	130
ISFP	0.48	175
ISTJ	0.63	249
ISTP	0.81	685
Усього	0.81	21214

У таблиці 3.14 можна побачити результати класифікації текстів за допомогою SVM та скороченого («обрізаного») набору даних.

Таблиця 3.14 – Результати класифікації текстів за допомогою SVM та «обрізаного» набору даних

Назва типу	Значення F1-score	Кількість екземплярів
1	2	3
ENFJ	0.24	307
ENFP	0.31	1000
ENTJ	0.29	591
ENTP	0.35	1000
ESFJ	0.00	36
ESFP	0.03	72
ESTJ	0.00	96
ESTP	0.55	397
INFJ	0.31	1000
INFP	0.40	1000
INTJ	0.31	1000

Продовження таблиці 3.14

1	2	3
INTP	0.36	1000
ISFJ	0.42	130
ISFP	0.42	175
ISTJ	0.17	249
ISTP	0.26	685
Усього	0.33	8738

Як ми бачимо, загальний показник F1-score змінився у гірший бік, це може свідчити про 2 речі:

– дані гіршої якості;

– модель надто чутлива до кількості даних.

Детальний аналіз буде проведено після отримання результатів за всіма моделями.

Тепер перейдемо до тестування «скопійованого» набору даних та моделі, що на ньому навчалась. Результати можна побачити у таблиці 3.15.

Таблиця 3.15 – Результати класифікації текстів за допомогою SVM та «скопійованого» набору даних

Назва типу	Значення F1-score	Кількість екземплярів
1	2	3
ENFJ	0.90	718
ENFP	0.84	805
ENTJ	0.90	953
ENTP	0.81	800
ESFJ	0.99	790
ESFP	0.98	801
ESTJ	0.99	808

Продовження таблиці 3.15

1	2	3
ESTP	0.97	925
INFJ	0.81	807
INFP	0.80	794
INTJ	0.77	804
INTP	0.77	804
ISFJ	0.96	806
ISFP	0.95	825
ISTJ	0.92	794
ISTP	0.92	1094
Усього	0.89	13328

Як ми бачимо, найкращі результати серед всіх за показниками загального значення F1-score саме з останнім набором даних. Гіпотеза щодо покращення результатів за рахунок балансу даних частково підтверджується. Слід також зазначити, що покращення велике. Під час формування фінальних висновків це буде взято до уваги.

3.3.6 Експерименти з рішенням на основі моделі BERT

Для проведення експериментів із BERT використовувався класичний пакет модулю Transformers. Невелику частину коду для виконання навчання та тестування можна побачити у лістингу 3.6.

Лістинг 3.6 – Використання BERT для класифікації текстів за MBTI

```
tokenizer = transformers.AutoTokenizer.from_pretrained('bert-large-uncased')
maxlen = 1500
```

Продовження лістингу 3.6

```
train_input_ids = [tokenizer.encode(str(i), max_length = maxlen
, pad_to_max_length = True) for i in train.cleaned_text.values]
val_input_ids = [tokenizer.encode(str(i), max_length = maxlen ,
pad_to_max_length = True) for i in val.cleaned_text.values]
def create_model():
   input_word_ids = tf.keras.layers.Input(shape=(maxlen,),
dtype=tf.int32, name="input_word_ids")
    bert_layer =
transformers.TFBertModel.from_pretrained('bert-large-uncased')
    bert_outputs = bert_layer(input_word_ids)[0]
    pred = tf.keras.layers.Dense(16,
activation='softmax')(bert_outputs[:,0,:])
    model = tf.keras.models.Model(inputs=input_word_ids,
outputs=pred)
    loss =
tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
    model.compile(loss='categorical_crossentropy',
optimizer=tf.keras.optimizers.Adam(
    learning_rate=0.00001), metrics=['accuracy'])
    return model
use_gpu = True
tpu = tf.distribute.cluster_resolver.GPUClusterResolver()
tf.config.experimental_connect_to_cluster(gpu)
tf.gpu.experimental.initialize_gpu_system(gpu)
strategy = tf.distribute.experimental.GPUStrategy(gpu)
with strategy.scope():
   model = create_model()
model.summary()
batch_size = 16
model.fit(np.array(train_input_ids),
one_hot_labels,validation_data = (np.array(val_input_ids),
val_labels), verbose = 1, epochs = 20, batch_size = batch_size,
callbacks = [tf.keras.callbacks.EarlyStopping(patience = 5)])
```

Тепер розберемо результати, отримані в результаті навчання моделі. Почнемо з початкового набору даних. Результати навчання моделі за допомогою нього можна побачити у таблиці 3.16 (показники за тестовими даними).

Таблиця 3.16 – Результати класифікації текстів за допомогою BERT та набору даних за замовчуванням

Назва типу	Значення F1-score	Кількість екземплярів
ENFJ	0.69	307
ENFP	0.80	1233
ENTJ	0.84	591
ENTP	0.84	2345
ESFJ	0.59	36
ESFP	0.62	72
ESTJ	0.87	96
ESTP	0.92	397
INFJ	0.83	2993
INFP	0.81	2427
INTJ	0.85	4486
INTP	0.86	4992
ISFJ	0.69	130
ISFP	0.69	175
ISTJ	0.67	249
ISTP	0.84	685
Усього	0.83	21214

У таблиці 3.17 можна побачити результати класифікації текстів за допомогою BERT та скороченого («обрізаного») набору даних. Також слід

зазначити, що дані всіх наборів даних з використанням BERT токенізуються за допомогою нативного токенізатору BERT.

Таблиця 3.17 – Результати класифікації текстів за допомогою BERT та «обрізаного» набору даних

Назва типу	Значення F1-score	Кількість екземплярів
ENFJ	0.69	307
ENFP	0.80	1000
ENTJ	0.84	591
ENTP	0.84	1000
ESFJ	0.59	36
ESFP	0.62	72
ESTJ	0.87	96
ESTP	0.92	397
INFJ	0.83	1000
INFP	0.81	1000
INTJ	0.85	1000
INTP	0.86	1000
ISFJ	0.69	130
ISFP	0.69	175
ISTJ	0.76	249
ISTP	0.84	685
Усього	0.83	8738

Як ми бачимо, загальний показник F1-score не змінився, це може свідчити про 2 речі:

- дані високої якості;
- модель нечутлива до кількості даних.

Детальний аналіз буде проведено після отримання результатів за всіма моделями.

Тепер перейдемо до тестування «скопійованого» набору даних та моделі, що на ньому навчалась. Результати можна побачити у таблиці 3.18.

Таблиця 3.18 – Результати класифікації текстів за допомогою BERT та «скопійованого» набору даних

Назва типу	Значення F1-score	Кількість екземплярів
ENFJ	0.97	718
ENFP	0.86	805
ENTJ	0.96	953
ENTP	0.85	800
ESFJ	1.00	790
ESFP	0.99	801
ESTJ	1.00	808
ESTP	0.99	925
INFJ	0.82	807
INFP	0.82	794
INTJ	0.80	804
INTP	0.78	804
ISFJ	0.99	806
ISFP	0.99	825
ISTJ	0.98	794
ISTP	0.96	1094
Усього	0.92	13328

Як ми бачимо, найкращі результати серед всіх за показниками загального значення F1-score саме з останнім набором даних. Гіпотеза щодо

покращення результатів за рахунок балансу даних частково підтверджується.

Також слід приділити окрему увагу навченій моделі на основі BERT. Модель представляє собою комбінацію безпосередньо самої моделі BERT як основи для аналізу тексту та повнозв'язної нейронної мережі наприкінці, яка приймає на вхід вихід зі слою BERT, що відповідає за формування ембедингів, а на виході – 4 значення, що за допомогою функції активації зможуть формувати псевдо значення ймовірностей.

3.3.7 Проміжні висновки стосовно навчання моделей

Для того, щоб порівняти якість всіх моделей, що розглядались вище, треба зробити зведення даних у таблицю значень індикаторів якості, тобто F1-score. Приймаючи до уваги, що ми маємо 3 різних набори даних та проводити порівняння між моделями, що тренувались на різних наборах, у нашому випадку некоректно, ми маємо сформувати 3 різні таблиці із отриманими значеннями.

Першу таку таблицю 3.19, що зводить показники за набором даних за замовчуванням, можна побачити нижче.

Таблиця 3.19 – Зведення значень F1-score моделей, що навчались на наборі даних за замовчуванням

Назва типу	Моделі					
	BERT	Логістична регресія	SVM	Наївний Байєс	Дерево рішень	Випадковий ліс
1	2	3	4	5	6	7
ENFJ	0.69	0.56	0.61	0.05	0.08	0.00

Продовження таблиці 3.19

1	2	3	4	5	6	7
ENFP	0.80	0.79	0.79	0.16	0.58	0.18
ENTJ	0.84	0.75	0.78	0.20	0.33	0.00
ENTP	0.84	0.81	0.81	0.37	0.59	0.35
ESFJ	0.59	0.24	0.43	0.00	0.00	0.00
ESFP	0.62	0.36	0.48	0.00	0.00	0.00
ESTJ	0.87	0.75	0.79	0.66	0.04	0.34
ESTP	0.92	0.89	0.91	0.79	0.60	0.45
INFJ	0.83	0.81	0.82	0.63	0.65	0.60
INFP	0.81	0.80	0.81	0.54	0.64	0.55
INTJ	0.85	0.83	0.84	0.66	0.73	0.67
INTP	0.86	0.84	0.85	0.65	0.69	0.58
ISFJ	0.69	0.45	0.57	0.00	0.04	0.00
ISFP	0.69	0.47	0.48	0.00	0.12	0.00
ISTJ	0.76	0.50	0.63	0.00	0.04	0.00
ISTP	0.84	0.78	0.81	0.21	0.53	0.00
Усього	0.84	0.80	0.81	0.52	0.62	0.49

Ми можемо побачити, що найкращі показники якості має модель на основі BERT, друга за якістю – SVM, а замикає трійку лідерів логістична регресія. Як можна побачити, різниця між показниками міри F1 дуже велика між 3 та 4 за першістю показниками, а трійка лідерів – навпроти, має дуже близькі значення.

Також слід зазначити варіативність значення показнику якості серед першої трійки лідерів: у моделі на основі BERT цей показник найнижчий, що є також однією з явних ознак якості цього класифікатору.

Перейдемо до перегляду результатів дослідження із обрізаним набором даних. Зведення результатів за ним серед моделей можна побачити у таблиці 3.20.

Таблиця 3.20 – Зведення значень F1-score моделей, що навчались на обрізаному наборі даних

Назва типу	Моделі					
	BERT	Логістична регресія	SVM	Наївний Байєс	Дерево рішень	Випадковий ліс
ENFJ	0.69	0.73	0.24	0.04	0.04	0.00
ENFP	0.80	0.84	0.31	0.00	0.46	0.00
ENTJ	0.84	0.85	0.29	0.03	0.23	0.00
ENTP	0.84	0.81	0.35	0.07	0.39	0.00
ESFJ	0.59	0.24	0.00	0.02	0.05	0.00
ESFP	0.62	0.49	0.03	0.02	0.00	0.00
ESTJ	0.87	0.78	0.00	0.05	0.00	0.00
ESTP	0.92	0.94	0.51	0.43	0.41	0.05
INFJ	0.83	0.79	0.31	0.00	0.42	0.00
INFP	0.81	0.79	0.40	0.01	0.01	0.00
INTJ	0.85	0.78	0.31	0.15	0.28	0.22
INTP	0.86	0.79	0.36	0.24	0.29	0.24
ISFJ	0.69	0.64	0.42	0.39	0.00	0.00
ISFP	0.69	0.67	0.42	0.34	0.00	0.00
ISTJ	0.76	0.71	0.17	0.13	0.00	0.00
ISTP	0.84	0.86	0.26	0.03	0.02	0.00
Усього	0.83	0.80	0.33	0.10	0.25	0.06

Як можна побачити, попри припущення щодо покращення якості роботи моделей, що навчались на цьому наборі даних, він має навпаки,

зниження показнику F1-score. Також слід зазначити, що зниження цього показнику в середньому не пропорційне серед різних моделей. Це, скоріше за все, свідчить про тип роботи тексту та чутливість моделі до кількості даних. Це також може бути корисним при майбутніх дослідженнях.

Переходимо до зведення даних за скопійованим набором даних. Таблицю 3.21 із ними можна побачити нижче.

Таблиця 3.21 – Зведення значень F1-score моделей, що навчались на обрізаному наборі даних

Назва типу	Моделі					
	BERT	Логістична регресія	SVM	Наївний Байєс	Дерево рішень	Випадковий ліс
ENFJ	0.97	0.88	0.90	0.81	0.67	0.70
ENFP	0.86	0.84	0.84	0.70	0.64	0.66
ENTJ	0.96	0.89	0.90	0.83	0.71	0.73
ENTP	0.85	0.81	0.81	0.67	0.49	0.48
ESFJ	1.00	0.99	0.99	0.98	0.80	0.92
ESFP	0.99	0.97	0.98	0.94	0.83	0.86
ESTJ	1.00	0.99	0.99	0.90	0.91	0.94
ESTP	0.99	0.96	0.97	0.92	0.78	0.82
INFJ	0.82	0.80	0.81	0.65	0.54	0.61
INFP	0.82	0.79	0.80	0.62	0.49	0.47
INTJ	0.80	0.77	0.77	0.61	0.01	0.48
INTP	0.78	0.76	0.77	0.61	0.27	0.52
ISFJ	0.99	0.96	0.96	0.88	0.80	0.78
ISFP	0.99	0.92	0.95	0.83	0.78	0.77
ISTJ	0.98	0.92	0.92	0.83	0.67	0.71
ISTP	0.96	0.92	0.92	0.77	0.70	0.53
Усього	0.92	0.89	0.89	0.79	0.63	0.68

Тут можна побачити значне покращення якості моделей не тільки у порівнянні із обрізаною версією, але і в порівнянні з набором даних за замовчуванням. Лідери серед дослідження ті ж самі: модель на основі BERT, логістична регресія та SVM. При чому, значення показнику якості для BERT є навіть вищим за 0.9, що є доволі гарним результатом, приймаючи до уваги предметну галузь, а також подібне значення показника ще раз підтверджує працездатність MBTI та спроможність аналізу тексту та вплив патерну поведінки людини на стиль написання тексту.

Вважаю подібний підхід до аналізу тексту для поставленого завдання ефективним та спроможним виконувати покладену функцію з високою якістю.

3.4 Робота зі словниковим підходом

Linguistic Inquiry and Word Count (LIWC) – це широко використовуваний інструмент аналізу тексту, який надає комплексний набір мовних характеристик для аналізу тексту. Інтерфейс програмного застосунку можна побачити на рисунку 3.5.

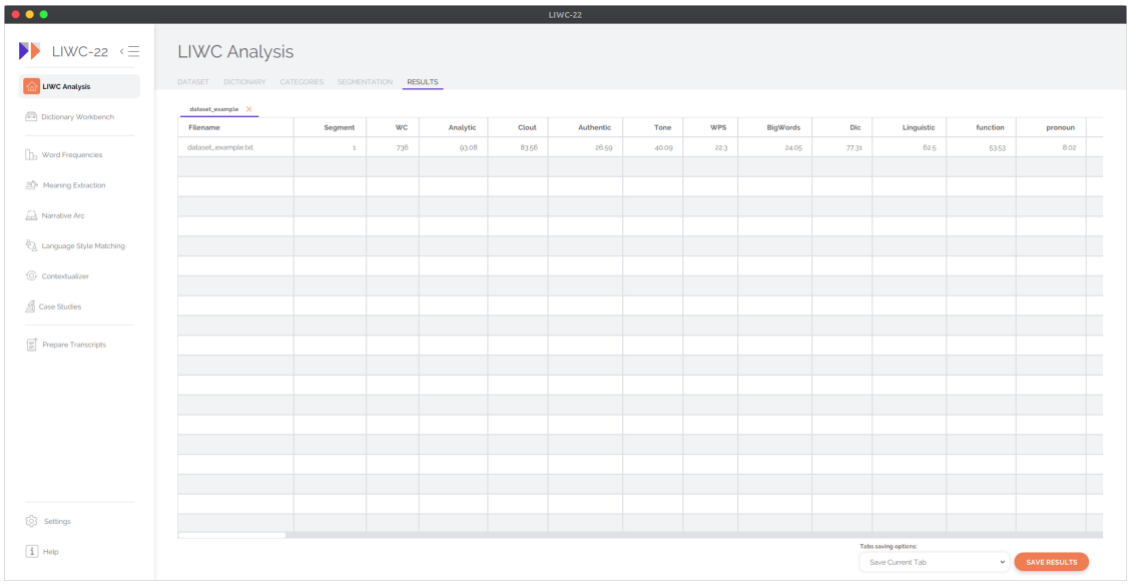

Рисунок 3.5 – Програмний інтерфейс мовного інструменту LIWC

LIWC включає понад 90 мовних вимірів, таких як емоційний тон, когнітивні процеси та соціальні тенденції, які можна використовувати для аналізу змісту тексту.

Ми використовували для роботи набір даних за замовчуванням, оскільки при роботі зі словником баланс набору даних за класами не має сенсу. Для кожного тексту ми знаходили ознаки LIWC за допомогою програмного пакету LIWC, версія 2022 року. Ми використовували стандартні категорії LIWC, які включають емоційний, когнітивний, соціальний та лінгвістичний виміри.

Потім ми вручну проаналізували характеристики LIWC, щоб виявити закономірності та тенденції, пов’язані з типами MBTI. Ми дослідили частоту та розподіл категорій LIWC між різними типами MBTI, а також визначили спільні мовні особливості, пов’язані з кожним типом.

Наші експерименти показали, що функції LIWC можуть надати корисну інформацію про мовні патерни та характеристики, пов’язані з різними типами MBTI. Ми виявили, що люди з типами INFP і ENTP, як правило, використовують більш емоційну та когнітивну мову, тоді як люди з типами ESTJ і ESFP, як правило, використовують більш соціальну та лінгвістичну мову.

Ми також помітили, що певні категорії LIWC були більш тісно пов’язані з певними типами MBTI. Наприклад, категорія «афект», яка вимірює емоційний тон, мала вискоку кореляцію з типами INFP і ENTP, тоді як категорія «заперечення» – ESTJ і ESFP.

В ході експериментів не вдалося добитись значних результатів класифікації тексту за MBTI в контексті міри точності: після знаходження параметрів, що мають кореляцію із типами особистості, було створено модуль класифікації за частотним аналізом та використанням певних слів, спираючись на значення категорій зі словнику, де точність не перевищувала показника 0.44.

Наші експерименти показують, що LIWC може бути цінним інструментом для аналізу та розуміння мовних патернів і характеристик, пов’язаних з різними типами MBTI. Однак, для повного вивчення потенціалу LIWC для класифікації текстів за типом MBTI необхідні подальші дослідження. Майбутня робота може включати розробку більш досконалих категорій LIWC або використання інших інструментів аналізу тексту в поєднанні з LIWC для глибшого розуміння мови, якою користуються люди з різними типами MBTI. На поточний час подібний інструмент може бути використаний тільки в якості допоміжного сервісу валідації чи одним з голосувальників при створенні ансамблевої системи предикторів.

3.5 Проміжні висновки

Отже, наші експерименти з текстовими даними та класифікацією MBTI дозволили зробити кілька висновків щодо ефективності різних підходів до вирішення цієї проблеми. Наші результати показують, що моделі машинного навчання, зокрема моделі на основі BERT, логістична регресія та SVM, є дуже ефективними для класифікації авторів текстів за типом MBTI. Ці моделі перевершили всі інші підходи, які ми тестували, включаючи прості підходи на основі словників і використання інструментарію LIWC.

Хоча наші експерименти з LIWC не дали такого ж рівня точності, як моделі машинного навчання, ми вважаємо, що LIWC залишається цінним інструментом для розуміння мовних патернів і характеристик, пов’язаних з різними типами MBTI. Однак для повного вивчення потенціалу LIWC для класифікації текстів за типом MBTI потрібні подальші дослідження.

4 СТВОРЕННЯ ПРОГРАМНОГО ЗАСТОСУНКУ

У цьому розділі ми описуємо процес створення веб-додатку, який демонструє можливості наших досліджень у сфері психологічного профілювання з використанням методів обробки природної мови. Веб-додаток буде побудовано на хмарній платформі Azure, яка надає масштабовану та гнучку інфраструктуру для розміщення та розгортання веб-додатків.

4.1 Архітектура програмного забезпечення

Для створення веб-додатку ми будемо використовувати архітектуру на основі мікросервісів, яка дозволяє розбити додаток на невеликі незалежні компоненти, які можна розробляти, тестувати та розгортати окремо. Такий підхід має низку переваг, включаючи покращену масштабованість, відмовостійкість, а також легше обслуговування та оновлення.

Основні компоненти нашого веб-додатку включатимуть користувацький інтерфейс для введення текстових даних та відображення результатів класифікації MBTI, внутрішній сервер для обробки текстових даних та запуску класифікаційних моделей, а також сервіс для демонстрації застосовність розробки на прикладі написання таргетованого тексту для окремого типу особистості. Для останнього ми будемо використовувати сучасну розробку компанії OpenAI під назвою GPT-4. Ця модель є загальною та спроможна переписувати текст з різним стилем, спираючись на попередній досвід. Для правильного користування моделлю буде використаний підхід prompt engineering’у.

Для реалізації архітектури мікросервісів ми використаємо кілька сервісів Azure, зокрема Azure API Management Service для конфігурації Application Programming Interface (API), Azure App Service для розміщення веб-додатку та сервісів користування функціональністю платформи та

Azure Kubernetes для оркестрації контейнерами для масштабування та сепарації сервісів один від одного. Ми також використовуватимемо Azure DevOps для управління процесом розробки та розгортання, включаючи конвеєри безперервної інтеграції та розгортання (CI/CD).

В цілому, наша мета при створенні цього веб-додатку – продемонструвати практичне застосування наших досліджень і забезпечити зручний і доступний інтерфейс для користувачів, щоб вони могли вивчати і розуміти ідеї, які можна отримати за допомогою методів обробки природної мови в області психологічного профілювання.

На рисунку 4.1 можна побачити просту схематизацію архітектури платформи.

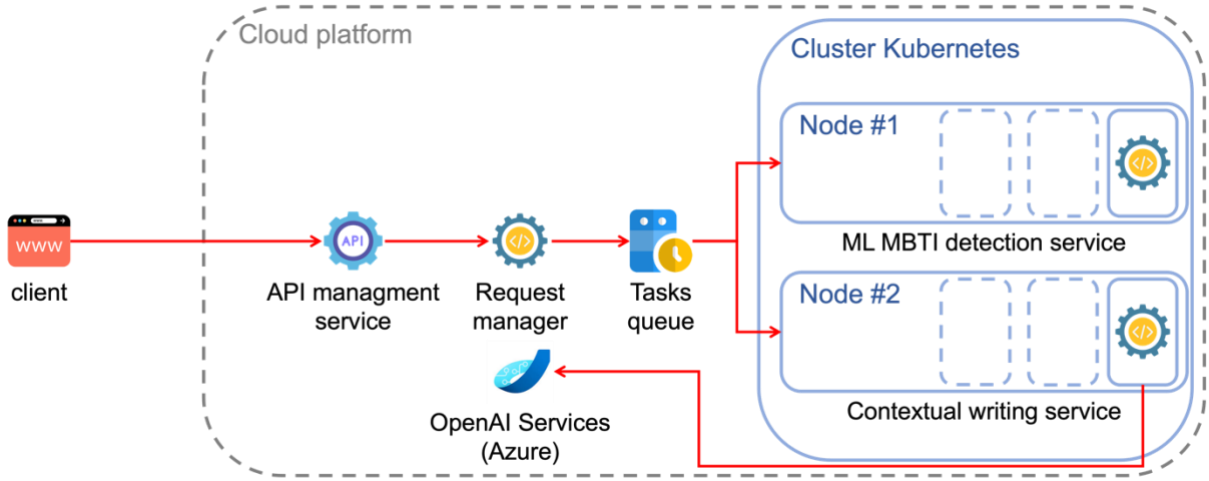

Рисунок 4.1 – Схематичне зображення архітектури платформи

Також слід зазначити, що не зважаючи на те, що додаток створено за прийнятими у галузі стандартами, патерн Model-View-Controller (MVC) тут використовується без зберігання даних, тобто моделі. Єдине місце, де дані тимчасово зберігаються, це черга задач, що постійно моніториться сервісами.

4.2 Особливості програмної реалізації

Деякі подробиці створення застосунку з інженерної точки зору вже розглядались у розділі 2, тут буде більше розкрито подробиці та використання технологій та концепцій, а не інструментів.

Розробка буде проводитись засобами середи розробки JetBrains PyCharm 2023.1 Professional Edition, в якій зосереджено величезний інструментарій, що значно полегшує розробку, а отже знижує кількість часу, необхідного на розробку додатку. Серед корисних інструментів це маніпуляція системним оточенням та системними змінними, прямий доступ до Kubernetes та Docker, а також доступ до сервісів платформи Microsoft Azure за допомогою плагінів. Ще однією з переваг цього застосунку є його можливість використання навіть для створення фронтенду із застосуванням мови TypeScript, хоча середа позиціонується зробленою під потреби на можливості мови програмування Python. Інтерфейс користувача можна побачити на рисунку 4.2.

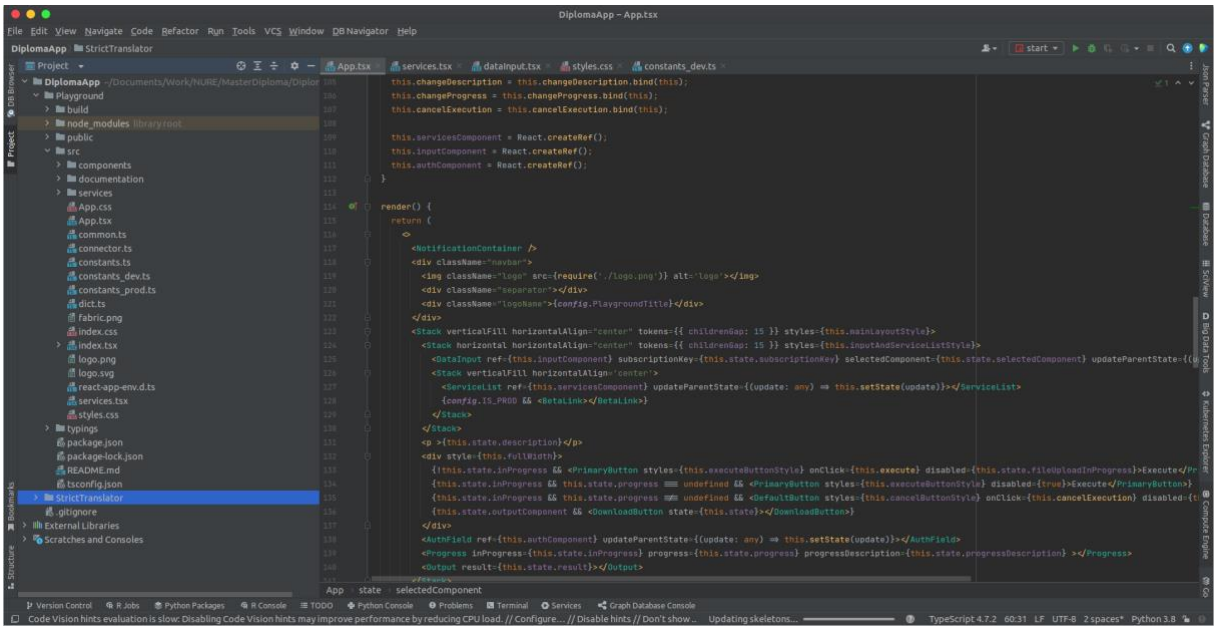

Рисунок 4.2 – Візуальний інтерфейс середи розробки

Для реалізації архітектури мікросервісів ми будемо використовувати контейнерний підхід з використанням Docker та Kubernetes, що дозволить нам більш ефективно розгортати, масштабувати та керувати додатком.

Інтерфейс користувача буде побудований з використанням сучасних технологій веб-розробки, таких як React, що забезпечує динамічний та інтерактивний користувальницький інтерфейс.

Внутрішній сервер буде побудований з використанням Python та фреймворку Flask, який забезпечує легкий та гнучкий фреймворк веб-додатків для побудови передачі репрезентативного стану (REST) API. Ми також будемо використовувати бібліотеку TensorFlow для реалізації моделей машинного навчання для класифікації MBTI.

Для забезпечення ефективного зв'язку між фронтенд- і бекенд-компонентами ми будемо використовувати REST API, який надає гнучкий і ефективний спосіб отримання та маніпулювання даними з сервера.

В цілому, наш веб-додаток продемонструє потужність і гнучкість хмарної платформи Azure і продемонструє потенціал методів обробки природної мови в області психологічного профілювання.

Повний програмний код, що розбито за відповідними файлами, можна знайти у додатку А.

4.3 Візуальний інтерфейс

Візуальний інтерфейс додатку повинен бути простим для розробки та користування, а також бути зрозумілим кінцевому користувачеві.

На рисунку 4.3 зображено візуальний інтерфейс розробленого додатку. За номером 1 можна побачити область, яка буде використана для вводу тексту користувачем. За номером 2 можна побачити кнопку для використання демонстраційного екземпляру тексту за замовчуванням для перевірки роботи сервісу та ознайомлення з його роботою.

За номером 3 можна побачити область для вихідних даних роботи сервісів. Тут буде відображатись відповідь системи в залежності від контенту, введеного у поле 1 та обраного сервісу.

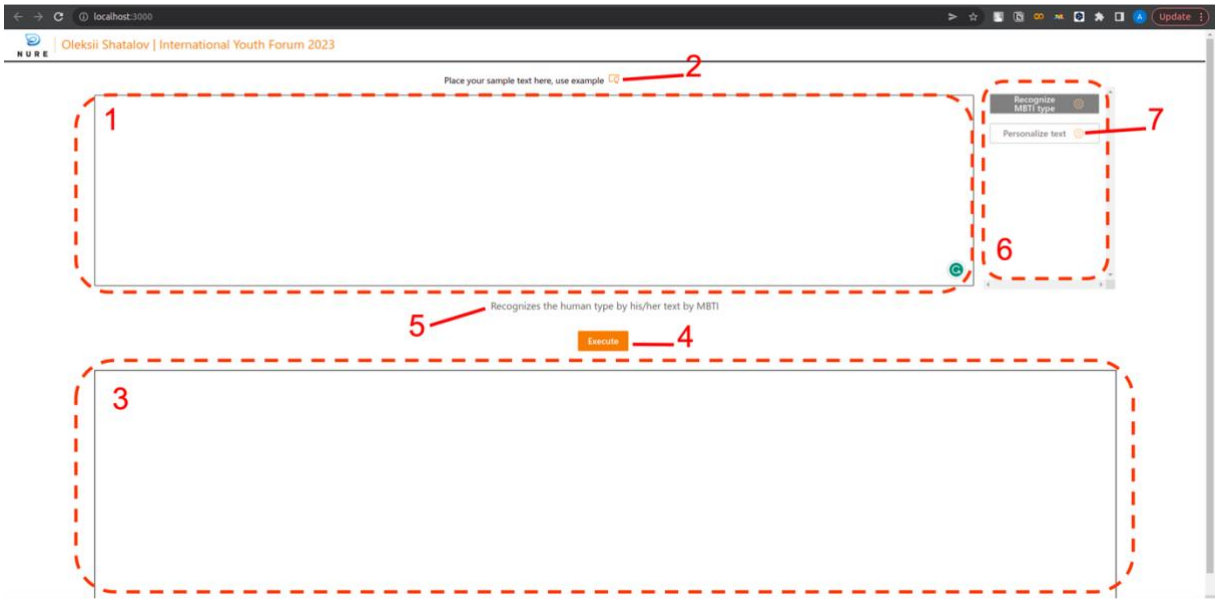

Рисунок 4.3 – Візуальний інтерфейс розробленого застосунку

Під номером 4 знаходиться кнопка для запуску сервісу. Під час натискання створюється запит до сервісу через API та чергу задач, створюється нова задача. Під номером 5 знаходиться короткий опис сервісу для полегшення взаємодії користувача з системою.

Під номером 6 знаходиться область доступних сервісів. Серед них можна обрати необхідний, ввести текст для обробки до поля 1 та натиснути кнопку 5 для користування. Під номером 7 знаходиться кнопка кастомізації сервісу. Подібна кнопка знаходиться у кожного сервісу та дозволяє передавати до інформацію, відмінну від тексту.

4.4 Проміжні висновки

Архітектура нашого веб-додатку для психологічного профілювання з використанням методів глибокого навчання та NLP буде розроблена з

урахуванням масштабованості, гнучкості та стабільності. Ми реалізуємо архітектуру мікросервісів, яка дозволяє легко обслуговувати та масштабувати окремі сервіси. Фронтенд буде побудований з використанням сучасних веб-технологій і буде взаємодіяти з бекенд-сервісами через REST API.

Ми включимо різні моделі та алгоритми машинного навчання, такі як BERT, а також додаткові функції, щоб надати користувачам всебічне розуміння їхньої письмової комунікації та особистості.

Загалом, наш веб-додаток стане потужним і безпечним інструментом для психологічного профілювання, який допоможе людям і організаціям отримати цінну інформацію про власну особистість та особистість інших людей за допомогою письмового спілкування.

ВИСНОВКИ

Результатом кваліфікаційної роботи є створена система розпізнавання типу особистості людини за написаним нею текстом. Наші експерименти показують, що використання методів глибокого навчання та обробки природної мови може дати цінну інформацію про психологічний портрет людини на основі її письмових текстів.

Другим результатом роботи є інтеграція програмних модулів до веб-додатку для демонстрації їх роботи з залучанням хмарних провайдерів.

Протягом дослідження були оброблені значні об’єми теоретичних матеріалів. Висновки щодо їх дослідження можна побачити у роботі.

В ході реалізації проекту була задіяна мова програмування Python та спеціальні бібліотеки для роботи із текстом та машинним навчанням.

Робота відповідає сучасному рівню знань та розвитку технологій.

Робота є актуальною, що розкрито у відповідному пункті про аналіз предметної області та сучасний стан аналогів.

Результати дослідження можуть бути впроваджені у широкий спектр предметних галузей, від маркетингу та реклами до допомоги виявлення розладів.

Вважаю доцільним продовження дослідницьких робіт в цій галузі у подальшому.

Ймовірним удосконаленням може бути продовження роботи з лінгвістичними словниками, а також розроблення застосунків, які можуть виконувати завдання, важливі для індустрії різних предметних галузей, на основі представленого дослідження.

В ході обробки матеріалів, дослідження теми та створення програмного забезпечення були з’ясовані деякі специфічні моменти, пов’язані з аналізом текстової інформації, машинного навчання та створення веб-додатку з застосуванням хмарних технологій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Schulz H., Behnke S. Deep Learning. KI - Künstliche Intelligenz. 2012. Vol. 26, no. 4. P. 357–363. URL: https://doi.org/10.1007/s13218-012-0198-z (date of access: 12.05.2023).

2. Contreras Kallens P., Kristensen-McLachlan R. D., Christiansen M. H. Large Language Models Demonstrate the Potential of Statistical Learning in Language. Cognitive Science. 2023. Vol. 47, no. 3. URL: https://doi.org/10.1111/cogs.13256 (date of access: 12.05.2023).

3. Determining user personality characteristics from social networking system communications and characteristics: patent US9740752B2 United States. Applied on 03.06.2016; published on 22.08.2017. URL: https://patents.google.com/patent/US9740752B2/en (date of access: 11.05.2023).

4. Myers I. B., Myers P. B. Gifts Differing: Understanding Personality Type. 1995. Mountain View, CA : Davies-Black Publishing.

5. Digman J. M. Personality Structure: Emergence of the Five-Factor Model. Annual Review of Psychology. 1990. Vol. 41, no. 1. P. 417–440. URL: https://doi.org/10.1146/annurev.ps.41.020190.002221 (date of access: 12.05.2023).

6. Radford A., Narasimhan K. Improving Language Understanding by Generative Pre-Training. Radford 2018 Improving LU. 2018. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf (date of access: 11.05.2023).

7. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / J. Devlin et al. Devlin 2019. 2019. URL: https://doi.org/10.48550/arXiv.1810.04805 (date of access: 11.05.2023).

8. Kane A. A., van Swol L. M. Using linguistic inquiry and word count software to analyze group interaction language data. Group Dynamics: Theory, Research, and Practice. 2022. URL: https://doi.org/10.1037/gdn0000195 (date of access: 12.05.2023).

9. Flaskerud, J. H. Issues in Mental Health Nursing, Vol 33(9), Sep, 2012. pp. 631-634.

10. Bess T. L., Harvey R. J. Bimodal Score Distributions and the MBTI: Fact or Artifact?. Annual Conference of the Society for Industrial and Organizational Psychology, San Diego. 2001.

11. Friedman M. ASSOCIATION OF SPECIFIC OVERT BEHAVIOR PATTERN WITH BLOOD AND CARDIOVASCULAR FINDINGS. Journal of the American Medical Association. 1959. Vol. 169, no. 12. P. 1286. URL: https://doi.org/10.1001/jama.1959.03000290012005 (date of access: 12.05.2023).

12. Jung, C. G. Psychological Types, Collected Works of C.G. Jung, vol. 6. Princeton, NJ: Princeton University Press. [1921] 1971. ISBN 0-691-01813-8.

13. Kahn, M. Basic Freud: psychoanalytic thought for the twenty first century (1. paperback ed.). New York: Basic Books. 2002. ISBN 9780465037162.

14. Baron, J. Intelligence and Personality In R. Sternberg (Ed.). Handbook of Intelligence. Cambridge: Cambridge University Press. 1982.

15. Kelly, G A. Theory of Personality: the psychology of personal constructs. New York: Norton. 1980. ISBN 978-0393001525.

16. Maslow, A. H. Toward a Psychology of Being (3. ed.). New York: Wiley. 1999. ISBN 978-0-471-29309-5.

17. Buss D. M. Evolutionary Personality Psychology. Annual Review of Psychology. 1991. Vol. 42, no. 1. P. 459–491. URL: https://doi.org/10.1146/annurev.ps.42.020191.002331 (date of access: 12.05.2023).

18. Dollard, J and Miller, N. Social Learning and Imitation (Tenth ed.). New Haven, London: Yale University Press. 1941.

19. Plank, B. and Hovy, D. Personality traits on Twitter—or—How to get 1,500 personality tests in a week. 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pages 92–98, Lisbon. Association for Computational Linguistics. 2015.

20. ben Verhoeven, D. W. and Plank, B. TwiSty: A multilingual twitter stylometry corpus for gender and personality profiling. Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16). 2016. P. 1632–1637. Portorož, Slovenia.

21. Lukito L. C., Purnama J. Social media user personality classification using computational linguistic. 2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE). 2016. P. 1–6. URL: https://doi.org/10.1109/ICITEED.2016.7863313 (date of access: 11.05.2023).

22. Alsadhan N., Skillicorn D. Estimating personality from social media posts. 2017 IEEE International Conference on Data Mining Workshops (ICDMW). 2017. P. 350-356.

23. Gjurković M., Šnajder J. Reddit: A gold mine for personality prediction. Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media. 2018. P. 87–97.

24. Keh S. S., Cheng I. Myers-Briggs personality classifica- tion and personality-specific language generation using pre-trained language models. CoRR. 2019.

25. Katiyar S., Kumar S., and Walia H. Personality prediction from stack overflow by using naive bayes theorem in data mining. International Journal of Innovative Technology and Exploring Engineering (IJITEE). 2020.

26. Author2Vec: A Framework for Generating User Embedding / X. Wu et al. arXiv preprint arXiv:2003.11627. 2020.

27. (MBTI) Myers-Briggs Personality Type Dataset. Kaggle: Your Machine Learning and Data Science Community. URL: https://www.kaggle.com/datasets/datasnaek/mbti-type (date of access: 12.05.2023).

28. Das K., Prajapati H. Personality identification based on MBTI dimensions using natural language processing. International Journal of Creative research Thoughts. 2020. P. 1653–1657.

29. Improving intelligent personality prediction using myers-briggs type indicator and random forest classifier / N. H. Z. Abidin et al. International Journal of Advanced Computer Science and Applications. 2020.

30. Personality classification from online text using machine learning approach / A. S. Khan et al. International Journal of Advanced Computer Science and Applications. 2020. P. 460–476.

31. Amirhosseini M. H., Kazemian H. Machine learning approach to personality type prediction based on the Myers-Briggs type indicator. Multimodal Technologies and Interaction. 2020.

32. dos Santos V. G., Paraboni I. Myers-Briggs personality classification from social media text using pre-trained language models. Journal of Universal Computer Science. 2022. Vol. 28, no. 4.

33. Powers, D. M. W. Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. Journal of Machine Learning Technologies. 2011. P. 37–63.

34. Pennebaker, J.W., Boyd, R.L., Jordan, K., Blackburn, K. The development and psychometric properties of LIWC2015. Austin, TX: University of Texas at Austin. DOI:10.15781/T29G6Z

35. MBTI Personality Types 500 Dataset. Kaggle: Your Machine Learning and Data Science Community. URL: https://www.kaggle.com/datasets/zeyadkhalid/mbti-personality-types-500-dataset (date of access: 12.05.2023).

ДОДАТОК А

Програмний код модулів

Лістинг А.1 – Програмний код файлу «App.tsx»

```
import React from 'react';
import { Stack, PrimaryButton, DefaultButton, initializeIcons,
TooltipHost, Link } from '@fluentui/react';
import { services } from './services'
import { config } from './constants'
import { loadTheme } from '@fluentui/react';
// @ts-ignore
import { NotificationContainer } from 'react-notifications';
import './styles.css';
import { EventEmitter } from 'events';
import { DataInput } from './components/dataInput';
import { FeedbackButton } from './components/feedbackButton';
import { DownloadButton } from './components/downloadButton';
import { ServiceList } from './components/serviceList';
import { Output } from './components/output';
import { Progress } from './components/progress';
import { AuthField } from './components/authField';
import { BetaLink } from './components/betaLink';

loadTheme({
  palette: {
    themePrimary: '#eb8027',
    themeLighterAlt: '#fefaf6',
    themeLighter: '#fce9da',
    themeLight: '#f9d7bb',
    themeTertiary: '#f3b07a',
    themeSecondary: '#ed8e40',
    themeDarkAlt: '#d37324',
    themeDark: '#b2611e',
```

Продовження лістингу А.1

```
    themeDarker: '#834716',
     neutralLighterAlt: '#f8f8f8',
     neutralLighter: '#f4f4f4',
     neutralLight: '#eaeaea',
     neutralQuaternaryAlt: '#dadada',
     neutralQuaternary: '#d0d0d0',
     neutralTertiaryAlt: '#c8c8c8',
     neutralTertiary: '#595959',
     neutralSecondary: '#373737',
     neutralPrimaryAlt: '#2f2f2f',
     neutralPrimary: '#000000',
     neutralDark: '#151515',
     black: '#0b0b0b',
     white: '#ffffff',
   }
});

initializeIcons();
export class App extends React.Component {
  defaultProgressDescription: string = "It takes some time to
evaluate the model, please be patient";
  gapSize = 15;
  intervalDelay = 1000;
  intervalIncrement = 0.1;
  currectPercent = 0;
  serviceNames = Object.keys(services);
  serviceDropDown = this.serviceNames.map(name => { return {
key: name, text: (services as any)[name].name } });
  wasUserNotified = false;
  emitter = new EventEmitter();
  cancellationToken = { isCancelled: false };
  state = {
    selectedComponent: "QnAbot",
```

Продовження лістингу А.1

```
 subscriptionKey: "",
    description: "",
    lastResult: null,
    outputComponent: "",
    percentComplete: 0,
    text: "",
    result: "",
    inProgress: false,
    progress: undefined,
    progressDescription: this.defaultProgressDescription,
    fileUploadInProgress: false,
    cancelButtonClicked: false
  }

  mainLayoutStyle = {
    root: {
      minWidth: '960px',
      width: '85%',
      margin: '0 auto',
      textAlign: 'center',
      color: '#605e5c',
      marginTop: '10px',
    }
  }

  inputAndServiceListStyle = {
    root: {
      width: '100%',
      height: '350px'
    }
  }
```

Продовження лістингу А.1

```
      executeButtonStyle = { root: { width: "30px", margin: "15px
5px 15px 0px" } };
       cancelButtonStyle = { root: { width: "30px", margin: "15px
5px 15px 0px" } };
       fullWidth = { width: "100%" };

       servicesComponent: React.RefObject<ServiceList>;
       inputComponent: React.RefObject<DataInput>;
       authComponent: React.RefObject<AuthField>;
       constructor(props: any) {
         super(props);

         this.execute = this.execute.bind(this);
         this.increasePercent = this.increasePercent.bind(this);
         this.changeDescription = this.changeDescription.bind(this);
         this.changeProgress = this.changeProgress.bind(this);
         this.cancelExecution = this.cancelExecution.bind(this);

         this.servicesComponent = React.createRef();
         this.inputComponent = React.createRef();
         this.authComponent = React.createRef();
       }

       render() {
         return (
           <>
             <NotificationContainer />
             <div className="navbar">
               <img className="logo" src={require('./logo.png')}
alt='logo'></img>
               <div className="separator"></div>
               <div
className="logoName">{config.PlaygroundTitle}</div>
```

Продовження лістингу А.1

```
        </div>
              <Stack verticalFill horizontalAlign="center" tokens={{
childrenGap: 15 }} styles={this.mainLayoutStyle}>
                <Stack horizontal horizontalAlign="center" tokens={{
childrenGap: 15 }} styles={this.inputAndServiceListStyle}>
                  <DataInput ref={this.inputComponent}
subscriptionKey={this.state.subscriptionKey}
selectedComponent={this.state.selectedComponent}
updateParentState={(update: any) =>
this.setState(update)}></DataInput>
                  <Stack verticalFill horizontalAlign='center'>
                    <ServiceList ref={this.servicesComponent}
updateParentState={(update: any) =>
this.setState(update)}></ServiceList>
                    {config.IS_PROD && <BetaLink></BetaLink>}
                  </Stack>
                </Stack>
                <p >{this.state.description}</p>
                <div style={this.fullWidth}>
                  {!this.state.inProgress && <PrimaryButton
styles={this.executeButtonStyle} onClick={this.execute}
disabled={this.state.fileUploadInProgress}>Execute</PrimaryButton>}
                  {this.state.inProgress && this.state.progress ===
undefined && <PrimaryButton styles={this.executeButtonStyle}
disabled={true}>Execute</PrimaryButton>}
                  {this.state.inProgress && this.state.progress !==
undefined && <DefaultButton styles={this.cancelButtonStyle}
onClick={this.cancelExecution}
disabled={this.state.cancelButtonClicked}>Cancel</DefaultButton>}
                  {this.state.outputComponent && <DownloadButton
state={this.state}></DownloadButton>}
                </div>
```

Продовження лістингу А.1

```
          <AuthField ref={this.authComponent}
updateParentState={(update: any) =>
this.setState(update)}></AuthField>
                <Progress inProgress={this.state.inProgress}
progress={this.state.progress}
progressDescription={this.state.progressDescription} ></Progress>
                <Output result={this.state.result}></Output>
              </Stack>
            </>
          );
        }

        execute() {
          if (!this.servicesComponent.current ||
!this.inputComponent.current) {
            return;
          }

          const service =
this.servicesComponent.current.state.selectedComponent;
          let lines: any = this.inputComponent.current.state.lines;
          let text: any = this.inputComponent.current.state.text;
          let blobName: string =
this.inputComponent.current.state.blobName;
          this.setState({ result: " ", outputComponent: null,
cancelButtonClicked: false });

          if (!text) {
            this.setState({ result: "Input text field is empty" });
            return;
          }
```

Продовження лістингу А.1

```
      let subscriptionKey =
this.authComponent.current?.state.subscriptionKey;
          if (config.AUTH && !subscriptionKey) {
            this.setState({ result: "Subscription key is required"
});
            return;
          }

          lines = (!lines || lines.length === 0)
            ? text.split('\n').map((l: any) => [l])
            : lines;

          this.cancellationToken.isCancelled = false;
          const params = (this.servicesComponent.current.state as
any)[service];
          const connector = (services as any)[service].connector;
          this.changeDescription(this.defaultProgressDescription);
          connector
            .execute(lines, params, this.changeDescription,
this.changeProgress, subscriptionKey, blobName,
this.cancellationToken)
            .then((data: any) => {
              try {
                if (data.error) {
                  this.setState({ result:
<><p>{data.error}</p><p>{data.trace}</p></>, inProgress: false,
progress: undefined });
                } else {
                  const output = (services as
any)[service].output(lines, data.result, params, this.emitter);
                  this.setState({
                    lastResult: [lines, data],
                    result: output,
```

Продовження лістингу А.1

```
       inProgress: false,
                    progress: undefined,
                    outputComponent: service,
                    report: data.report
                  });
                  if (!this.wasUserNotified) {
                    setTimeout(() =>
this.createFeedbackNotification(), 1000);
                    this.wasUserNotified = true;
                  }
                }
              }
              catch (e) {
                this.setState({ result: <p>{"Error occurred: " +
e}</p>, inProgress: false, progress: undefined });
              }
            })
            .catch((e: any) => {
              this.setState({ inProgress: false, progress: undefined,
result: <p>{"Error occurred: " + e}</p> })
            });
          this.setState({ inProgress: true });
        }

        async cancelExecution() {
          if (!this.servicesComponent.current ||
!this.inputComponent.current) {
            return;
          }

          this.setState({cancelButtonClicked: true});
          const service =
this.servicesComponent.current.state.selectedComponent;
```

Продовження лістингу А.1

```
      const connector = (services as any)[service].connector;
          await connector.cancel();
          this.cancellationToken.isCancelled = true;
        }

        increasePercent() {
          this.currectPercent += this.intervalIncrement;
          this.setState({ percentComplete: this.currectPercent });
          if (this.currectPercent < 1) {
            setTimeout(this.increasePercent, this.intervalDelay);
          }
        }

        private changeDescription(description: string) {
          this.setState({ progressDescription: description })
        }

        private changeProgress(newProgress: number) {
          this.setState({ progress: newProgress });
        }

        createFeedbackNotification() {
          return; // We don't want to show notification
          // NotificationManager.warning("Click here to help us
improve", "Let's innovate together! ", 5000, () => {
          //   this.openDialog();
          // });
        }
      }
```

Лістинг А.2 – Програмний код файлу «mbti-front.tsx»

```
import React, {useMemo} from "react";
import {Checkbox, ChoiceGroup, List, Stack} from
"@fluentui/react";
import {OutputProperties} from "../common";
import {BaseDetectorsConnector} from "../connector";
import {AgGridReact} from "@ag-grid-community/react";
import {AllCommunityModules} from "@ag-grid-community/all-
modules";
import {isThisTypeNode} from "typescript";

export class StrictTranslatorState {
  static defaultLanguagesList: string[] = ["Spanish",
"German"];
  static defaultLanguage: string = "Spanish";

  constructor (public languagesList: string[], public language:
string) { }

  static default() {
    return new StrictTranslatorState(
        StrictTranslatorState.defaultLanguagesList,
        StrictTranslatorState.defaultLanguage
        );
  }
}

class StrictTranslatorProperties {
  constructor(public getState: () => StrictTranslatorState,
public onChange: (state: StrictTranslatorState) => void) { }
}

export class StrictTranslator extends
React.Component<StrictTranslatorProperties> {
```

Продовження лістингу А.2

```
constructor(props: StrictTranslatorProperties) {
    super(props);
    let initState = props.getState();
    if (!initState) {
      return;
    }

    this.state.languagesList = initState.languagesList;
    this.state.language = initState.language;

    this.props.onChange(this.state);
  }

  state = {
    languagesList: StrictTranslatorState.defaultLanguagesList,
    language: StrictTranslatorState.defaultLanguage,
  }

  componentDidUpdate() {
    if (this.props.onChange) {
      this.props.onChange(this.state);
    }
  }

  typeChange = (e: any, option: any) => {
      this.setState({ language: option.key });
  };

    render() {
      return (
          <>
            <Stack verticalFill tokens={{childrenGap: 10}}>
              <ChoiceGroup
```

Продовження лістингу А.2

```
      name="type"
                        selectedKey={this.state.language}
                        options={
                            [
                              {
                                key: 'Spanish',
                                text: 'Spanish'
                              },
                              {
                                key: 'German',
                                text: 'German'
                              }
                            ]
                        }

onChange={this.typeChange.bind(isThisTypeNode)}
                        label="Choose the language for translation"
                    />
                  </Stack>
                </>
          );
          }
      }

      export class StrictTranslatorConnector extends
BaseDetectorsConnector {

        constructor() {
          super("StrictTranslator", "Strict Translator");
        }

        preprocessParams(params: StrictTranslatorState): void {
          return {
```

Продовження лістингу А.2

```
 "language": params.language
    } as any;
  }
}

class StrictTranslatorText {
  constructor(public returned_text: any) {  }
}

export class StrictTranslatorOutput extends
React.Component<OutputProperties> {

  data: string[];
  lines: string[];

  paragraphStyle = {
    lineHeight: "1.5",
    margin: "5px 10px 0px 10px",
    textAlign: "left"
  } as React.CSSProperties

  itemCell = {
    minHeight: 54,
    padding: 10,
    boxSizing: 'border-box',
    borderBottom: `1px solid`,
    display: 'flex',
  } as React.CSSProperties

  constructor(props: OutputProperties) {
    super(props);
    this.data = props.data;
    this.lines = props.lines;
```

Продовження лістингу А.2

```
 }

  render() {
    return (
        <div className="ag-theme-alpine" style={{ height:
"100%", width: "100%" }}>
            {this.data.map((result) =>
                (
                    <div style={this.itemCell} data-is-
focusable={true}>
                        <div style={this.paragraphStyle}>
                            <p>{result}</p>
                        </div>
                    </div>
                ))}
        </div>
    )
    }
}
```

Лістинг А.3 – Програмний код файлу «main.py»

```
import tiktoken_monkey_patch

from typing import List
from constants import *
from langchain_ops import StrictTranslatorLangChain
from detector_worker import Worker, Segment

if __name__ == "__main__":
    worker = Worker(WORKER_NAME, segment_limit=50000,
connection_string=CONNECTION_STRING, return_only_source=False)
```

Продовження лістингу А.3

```
strict_translator_langchain =
StrictTranslatorLangChain(DOCUMENTS_DIR, 'txt', LLM_DEPLOYMENT_NAME,
EMBEDDING_DEPLOYMENT_NAME)

        def perform_contextual_answer(texts: List[Segment],
params_json):
            language = params_json['language']
            result = [
                strict_translator_langchain(s.source, language)
                for s in texts
            ]
            return result, []


worker.process_messages_from_queue(perform_contextual_answer)
```

Лістинг А.4 – Програмний код файлу «langchain_ops.py»

```
from langchain.chat_models import AzureChatOpenAI
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.text_splitter import TokenTextSplitter
from langchain.vectorstores import FAISS
from langchain.document_loaders import DirectoryLoader,
TextLoader
from langchain.chains.question_answering import load_qa_chain

class StrictTranslatorLangChain:
    _chain_type = "stuff"
    _prompt = ""
    ""

    def __init__(self,
                 docs_dir,
                 file_format,
```

Продовження лістингу А.4

```
            llm_deployment_name,
                        embedder_deployment_name):
                self._docs_dir = docs_dir
                self._llm =
AzureChatOpenAI(deployment_name=llm_deployment_name)
                self._embeddings =
OpenAIEmbeddings(model=embedder_deployment_name, chunk_size=1)
                self._texts =
self._create_wiki_context_index(file_format)

            def __call__(self,
                         query,
                         language):
                chain = load_qa_chain(self._llm,
chain_type=StrictTranslatorLangChain._chain_type)
                return chain(
                    {
                        "input_documents": self._search(query),
                        "question": StrictTranslatorLangChain._prompt %
(language, query)
                    },
                    return_only_outputs=True
                )["output_text"]

            def _create_wiki_context_index(self,
                                           file_format):
                loader = DirectoryLoader(self._docs_dir,
glob=f"*.{file_format}", loader_cls=TextLoader)
                documents = loader.load()
                text_splitter = TokenTextSplitter(chunk_size=1000,
chunk_overlap=0)
                return text_splitter.split_documents(documents)
```

Продовження лістингу А.4

```
 def _search(self,
                query):
        docsearch = FAISS.from_documents(self._texts,
self._embeddings)
        return docsearch.similarity_search(query)
```

Лістинг А.5 – Програмний код файлу «bert_experiment.ipynb»

```
# Importing libraries
"""

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline
from sklearn import metrics
from sklearn.metrics import classification_report, f1_score
import pickle
import os.path
import plotly.offline as pyo
import plotly.graph_objs as go
import spacy
from nltk.stem.snowball import SnowballStemmer
from sklearn.feature_extraction.text import CountVectorizer

#

"""# Context

MBTI (Myers-Briggs Type Indicator) is an introspective self-
report questionnaire indicating differing psychological preferences
```

Продовження лістингу А.5

```
(cognitive functions) in how people perceive the world and make
decisions

This study was made based on the kaggle dataset
https://www.kaggle.com/zeyadkhalid/mbti-personality-types-500-
dataset

# Content

~106K records of preprocessed posts and their authors'
personality types.

    Posts are equal-sized: 500 words per sample

# About the dataset

Posts are preprocessed texts:

    - No punctuations, stopwords, URLs

    - Lemmatization

    - Reconstruct samples to be equal-sized chunks (500 words
per sample)

Personality types are 16 unique values

# Stop words

As said, this dataset doesn't has any stop words

"Stop words" are words that appears so frequently that don't
require tagging as thoroughly as nouns, verbs and modifiers
```

Продовження лістингу А.5

```
    Let's use the library Spacy to see examples of english stop
words

    Spacy is the Industrial Strength Natural Language Processing:
https://spacy.io/
    """

    # Load one of the availables trained pipelines for English
    # English pipeline optimized for CPU. Components: tok2vec,
tagger, parser, senter, ner, attribute_ruler, lemmatizer.
    nlp = spacy.load('en_core_web_sm')

    # stop words built in spacy (english)
    print(nlp.Defaults.stop_words)

    print(f"Number of default stop words :
{len(nlp.Defaults.stop_words)}")

    # Checking if a word is a stop word
    nlp.vocab['is'].is_stop

    nlp.vocab['below'].is_stop

    nlp.vocab['btw'].is_stop

    """# Lemmatization

    As said before, this dataset also has Lemmatization preprocess
feature

    In order to understand lemmatization, first we'll look at the
concept of Stemming
```

Продовження лістингу А.5

```
### Stemming

Stemming is used to return similarities words on the search
process.

 - Example: search=boat, also returns "boats" and "boating"

Let's use a sophisticated stemmer, the SnowballStemmer from
NLTK (natural language toolkit)
"""

s_stemmer = SnowballStemmer(language='english')

words = ['run', 'runner', 'ran', 'runs', 'easily', 'fairly',
'fairness','boats','boating']

for word in words:
    print(word+ ' ------> ' + s_stemmer.stem(word))

"""Now, let's look about Lemmatization

### Lemmatization example

In constrast with stemming, Lemmatization looks beyond word
reduction, and considers a language's full vocabulary to apply a
morphological analysis to words.
"""

# Function to display lemmas
def show_lemmas(text):
    for token in text:
        print(f'{token.text:{12}} {token.pos_:{6}}
{token.lemma:<{22}} {token.lemma_}')
```

Продовження лістингу А.5

```
doc = nlp(u"I saw eighteen mice today!")
show_lemmas(doc)

doc = nlp(u"I am meeting him tomorrow at the meeting.")
show_lemmas(doc)

"""Now it's time to talk about Text Feature Extraction

# Text Feature Extraction

After preprocess data it's time to extract features from the text in order to prepare the machine learning model

### Count Vectorization

1. Treats each word of a text individually as a feature

2. After that, counts each occurrence of each word in the document

3. Than, makes a matrix DTM (Document Term Matrix)
"""

count_vect = CountVectorizer()

phrase = ["I'd like to have a glass of water please"]

# Fit Vectorizer to the Data (build a vocab, count the number of words...)
# Learn a vocabulary dictionary of all tokens in the raw documents
count_vect.fit(phrase)
```

Продовження лістингу А.5

```
# Show features
count_vect.get_feature_names()

# Learn the vocabulary dictionary and return document-term
matrix
count_vect.fit_transform(phrase)

# shows a mapping of terms to feature indices.
count_vect.vocabulary_

"""## TfidVectorizer

An alternative to CountVectorizer is the TfidVectorizer

TfidVectorizer calculates an inverse frequency for each word

It converts a collection of raw documents to a matrix of TF-IDF
features.

TfidVectorizer will be used to create the machine learning
model for this study

# Read the dataset into a pandas dataframe

Now it's time to read the dataset and make a simple exploratory
analysis
"""

df = pd.read_csv('/kaggle/input/mbti-personality-types-500-
dataset/MBTI 500.csv')

df.head()
```

Продовження лістингу А.5

```
df['posts'][0]

df['type'][0]

df['type'].unique()

print(f"Total of {len(df['type'].unique())} types of classified
MBTI posts")

"""# Checking null values"""

df.isnull().sum()

"""# Checking the number of posts per type"""

df_bar_chart=df.groupby('type').count()

trace1 = go.Bar(x=df_bar_chart.index, y=df_bar_chart['posts'])

data = [trace1]
layout = go.Layout(title='MBTI # Classified Posts per Type')

fig = go.Figure(data=data, layout=layout)

fig.show()

"""# Recreate the model?

This machine learning model takes it's time to train data

To avoid waiting every time, We're going to use the feature
dump/load from pickle
"""
```

Продовження лістингу А.5

```
# Flag to re-create or not the machine learning model
recreate_model=False

# We'll save the model into a file:
filename = 'mbti_svm_v2.sav'

# If the model file doesn't exists
if not os.path.isfile(filename):
    recreate_model=True

"""# Model

The machine learning supervised model that we'll use here is a Classification kind, named Support Vector Machine

References:

 - https://en.wikipedia.org/wiki/Support-vector_machine

- https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC

### Model Pipeline

We're going to need a pipelined model ir order to facilitate the entire process of CountVectorizer (TfidfVectorizer) and svm.LinearSVC model

To do that, we're going to use the Pipeline feature from sklearn.pipeline
```

Продовження лістингу А.5

```
    References: https://scikit-
learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html

    # Split the data into train and test
    """

    X = df['posts'] # features
    y = df['type']  # labels
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

    """# Training the model, save it to disk and open to make
predictions"""

    # Check if need to recreate the model
    if recreate_model:

        # Creating an instance to vectorizer:
        vectorizer = TfidfVectorizer()

        # Training the vectorizer:
        X_train_tfidf = vectorizer.fit_transform(X_train)

        # Training the classifier:
        clf = LinearSVC()
        clf.fit(X_train_tfidf, y_train)

        # Pipelining the vectorizer and the classifier
        text_clf =
Pipeline([('tfidf',TfidfVectorizer()),('clf',LinearSVC())])
        text_clf.fit(X_train, y_train)

        # saving the model to disk
```

Продовження лістингу А.5

```
        pickle.dump(text_clf, open(filename, 'wb'))

    # If there is no need to recreate the model, just open the file
from the disk
    else:
        # loading the model from disk
        text_clf = pickle.load(open(filename, 'rb'))

    """# Using the test data to make predictions and analyze the
accurace of the model"""

    predictions = text_clf.predict(X_test)

    print(classification_report(y_test, predictions))

    print(f"Overall accuracy of the model:
{round(metrics.accuracy_score(y_test, predictions),2)}")

    """# End"""
```

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	103 с.
	Інші документи	
2.	Презентаційні матеріали	плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Шаталов О.В.			Розробка та застосування підсистеми контролю доступу як частини корпоративного розумного будинку	Шифр групи	Код напр./спец.
Перевірив		Рябова Н.В.				СШІм-21-2	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Філатов В.О.					