ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА
(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем
(повна назва кафедри (предметної, циклової комісії))

Кваліфікаційна робота

на здобуття рівня вищої освіти «магістр»
(рівень вищої освіти)

Інформаційна технологія виділення та класифікації онейрологічних образів в природньомовному тексті.

Information technology for extraction and classification of oneirological images in natural language text

Виконав: студент денної форми навчання
спеціальності 126 – Інформаційні системи та технології.
(шифр і назва напряму підготовки, спеціальності)
Освітня програма «Інформаційні системи та технології»
(назва освітньої програми)

Жар Михайло Юрійович
(прізвище, ім'я, по-батькові)

Керівник д.т.н, проф. Малахов Є.В.
(науковий ступінь, вчене звання, прізвище та ініціали, **підпис**)

Рецензент доц., канд. фіз.-матем. наук Петрушина Т.І.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент доц., канд. техн. наук Глава М. Г.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
№___ від «___»_____ 2024 р.
Завідувач кафедри
_____ Євгеній МАЛАХОВ
(підпис) (ім'я, прізвище)

Захищено на засіданні ЕК № ___
протокол № __ від «_ _» ____ 2024 р.
Оцінка _____/_____/_____
(за національною шкалою, шкалою ECTS, бали)
Голова ЕК
_____ Володимир ВИЧУЖАНІН
(підпис) (ім'я, прізвище)

Одеса – 2024

АНОТАЦІЯ

В роботі досліджено методи та технології для виявлення та класифікації онейрологічних образів у текстах українською мовою. Основною метою було зменшення кількості помилок другого роду під час обробки текстів снів шляхом налаштування моделей машинного навчання.

Розроблено методологію обробки україномовних текстів снів, яка може бути застосована для створення систем автоматичного ведення щоденників сновидінь та адаптована для інших завдань обробки природньомовних текстів.

Розглянуто два основні підходи: розділений, що базується на моделях сімейства BERT, та уніфікований з використанням великих мовних моделей (LLM).

Для розділеного підходу оцінено різні комбінації моделей сімейства BERT. Для уніфікованого – сучасні великі мовні моделі: GPT-4o та Claude 3.5 Sonnet. Проведено порівняльний аналіз між найкращими комбінаціями моделей сімества BERT та великими мовними моделями.

ABSTRACT

The paper investigates methods and technologies for detecting and classifying oneirological images in Ukrainian-language texts. The main goal was to reduce type II errors during dream text processing by fine-tuning machine learning models.

A methodology for processing Ukrainian dream texts has been developed, which can be applied to create automatic dream diary systems and adapted for other natural language processing tasks.

Two main approaches were considered: a separated approach based on BERT family models, and a unified approach using Large Language Models (LLM).

For the separated approach, different combinations of BERT family models were evaluated. For the unified approach, modern large language models were tested: GPT-4o and Claude 3.5 Sonnet. A comparative analysis was conducted between the best combinations of BERT family models and large language models.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

TP – вірно прийнята гіпотеза (true positive)

TN – вірно відкинута гіпотеза (true negative)

FP – невірно прийнята гіпотеза, визначає помилку першого роду (false positive)

FN – невірно відкинута гіпотеза, визначає помилку другого роду (false negative)

NLP – обробка природньої мови

NER – виділення іменованих сутностей

RNN – рекурентні нейронні мережі

LSTM – мережі з довгою короткочасною пам’яттю

LLM – великі мовні моделі

BERT – двоспрямовані кодувальні представлення з трансформерів

BIO – схема розмітки (Beginning, Inside, Outside)

ВСТУП

Сон – це активний періодичний фізіологічний процес у тілі людини, який характеризується неповним припиненням свідомої психічної діяльності та зниженням активної взаємодії з навколишнім середовищем.

Сон потрібен людині для відновлення органів, синтезу клітин, підвищення ефективності імунної системи, поліпшення настрою, пам'яті, концентрації уваги та працездатності, зменшення ризику депресії і тривоги.

Для того, щоб задовольнити потребу у сні середньостатистичній людині треба спати від 7 до 10 годин на добу. Якщо взяти до уваги середню тривалість життя в світі на 2024 рік – 73 роки, то виявиться, що людина втрачає на сон приблизно третину свого життя або приблизно 24 роки.[1][2]

Також важливо, що сон має свої фази: швидкого руху очей (ШРО) і не-швидкого руху очей (Не-ШРО). Саме під час фази ШРО людина і бачить більшу частину снів, але вона займає лише близько чверті часу всього процесу сну. Також, зважаючи на її динаміку росту, більшу частину снів людина бачить під ранок.[3][4]

Саме фазу ШРО використовують треновані люди для усвідомлення себе уві сні.

Свідомий сон – це сон, у якому людина усвідомлює, що спить і за багатьма дослідженнями він покращує як фізичний так і психологічний стан людини. [5]

Зазвичай частка свідомих снів для нетренованої людини становить близько 1% від всіх снів, проте існують способи покращити як їх кількість, якість та довжину такого сну. Вони складають дві основні групи: DILD (досягнення свідомого сну із стану сну) та WILD (досягнення свідомого сну при засинанні). [6]

Обидва класи методів спираються на навички індивіда усвідомлювати власну реальність. І одним із допоміжних методів для розвитку даної навички є ведення щоденнику сну.

У щоденнику сновидець має після прокидання записати все, що він може згадати про свої пережиті сни. Після того як у людини набереться деяка кількість снів людина може почати знаходити ознаки сну у власних записах.[7]

Ознака сну (онейрологічний образ) – це достатньо яскравий образ, на який людина може звернути увагу уві сні.

Для інформативності ці ознаки ще і класифікують по деяким категоріям, наприклад: дії, трансформації, емоції, місця, тощо. Таким чином людина уві сні може виявляти не тільки конкретні ознаки та образи, а цілі категорії цих ознак, з якими вона часто стикається, що може допомогти їй частіше потрапляти у свідомий сон. [8]

Проте робота з щоденником сну на сьогоднішній день проводиться по більшій мірі вручну, що з інженерної точки зору, звісно, має свої недоліки, тому існують телефонні та веб-додатки, що лише частково беруть на себе функції аналізу снів користувачів. [9]

Практична цінність цієї кваліфікаційної роботи є розвиток технологій, які дозволять виділяти ознаки сну і класифікувати їх по певним категоріям. Така технологія може стати корисною не тільки для обробки текстів снів, а і інших природньомовних текстах, наприклад, у системах обробки користувацьких відгуків, каталогізації книг та документів та інших.

Але для означеного розвитку потрібно мати дані, на основі яких можливо проводити цей самий розвиток і в даній сфері існують проблеми з отриманням і обробкою снів.

Серед україномовних датасетів відсутні пов’язані зі снами різних груп людей. Крім того сам процес створення датасету вимагає згоди користувачів на обробку і зберігання записів їх снів. [10]

Також самі тексти снів є достатньо суб’єктивними: одна людина може описати абсурдну ситуацію, що коїться уві сні, як декількома словами так і цілими реченнями та навіть присвятити декілька абзаців. До того ж сам текст може бути перенасичений граматичними помилками, неологізмами, різномаїтними сленговими словами та іншими мовними явищами.

Отак, виявлення ознаки сну, означає пошук в природньомовному тексті словосполучень, фраз, речень, абзаців, які можливо трактувати як ознака сну.

Класифікація ознаки сну – процес призначення декотрих категорій ознаці сну, які можуть описати більшість її аспектів.

Саме тому означений процес виявлення та класифікації повинен мати у собі складові, які мають бути достатньо гнучкими, пильними, креативними та зберігати конфіденційність користувачів.

Мета дослідження – зменшення кількості помилок другого роду під час процесу виділення та класифікації онейрологічних образів із природномовного тексту українською мовою шляхом налаштування моделей машинного навчання.

Об’єкт дослідження є процеси відбору та класифікації змістовних фрагментів тексту.

Предметом дослідження виступають технології, методи та системи, які дозволять обробляти тексти снів, виділяти особливі їх ознаки та робити висновок до яких категорій відноситься та чи інша ознака сну.

Звісно для досягнення мети потрібно обробити достатньо великий масив текстів від різних людей, щоб створити або удосконалити модель, яка зможе з гарною якістю виділяти ознаки снів у текстах нових користувачів.

Технологію вдосконалену за результатами дослідження буде впроваджено в компанії VTM Group (див. додаток М).

Для досягнення означеної мети потрібно:

- визначити технології та методи, які частково, або повністю вирішують задачу та виявити претендента або претендентів на вдосконалення;
- описати можливі рішення із використанням визначених технологій, які можуть вирішити задачу;
- створити датасет на основі записів у щоденнику сну;
- перевірити отриманні рішення на датасеті, визначити перспективні рішення та вдосконалити їх;
- протестувати вдосконалену технологію виділення та класифікації;
- оцінити зазначені рішення.

1 МЕТОДИ І ТЕХНОЛОГІЇ РОЗВ'ЯЗАННЯ ЗАДАЧ КЛАСИФІКАЦІЇ ТЕКСТУ

Так як за означеною ціллю необхідно визначати та класифікувати онейрологічні образи, очевидно, що це стосується класу задач класифікації тексту напрямку обробки природньомовного тексту (NLP).

1.1 Задачі класифікації тексту

Серед задач класифікації тексту існують дві для досягнення зазначеної мети. [11]

1.1.1 Класифікація послідовностей

Задача NLP для аналізу текстових включає процес автоматичного внесення текстових даних до однієї або декількох категорій. Також має свої основні завдання:

- аналіз емоційного забарвлення тексту (Sentiment analysis) ;
- тематична класифікація (Topic classification);
- фільтрація спаму;
- визначення наміру користувача.

Так, за цим завданням текст перетворюється на набір токенів і модель на основі цього набору обирає класи згідно із поточним типом класифікації.

Токен – частина слова, які використовує модель для розбивки вхідного тексту для подальшої обробки. [12]

Проте методи цього класу не виявляють конкретні онейрологічні образи, а скоріше трактують увесь текст як єдиний образ, що може бути корисним для класифікації знайдених в тексті образів.[13]

1.1.2 Класифікація токенів

Задача розуміння природної мови, в якій для кожного токену у фрагменті тексту передбачається мітка. Це відрізняється від класифікації послідовностей, оскільки кожна токен в тексті отримує передбачення. Деякі поширені завдання класифікації токенів включають:

− виділення іменованих сутностей (NER), таких як імена людей, організації, місця розташування та інше;

− позначення частин мови (POS), наприклад: іменників, прикметників, дієслів, тощо.

Отже, серед двох задач найбільше підходить завдання визначення іменованих сутностей задачі класифікації токенів тим, що працює на рівні слів і дає змогу визначати та класифікувати конкретні ознаки з тексту.

Проте, саме те, що модель, що користується цим завданням обробляє текст на рівні малих словосполучень (два – чотири слова) може не дати їй класифікувати увесь онейрологічний образ, особливо при multilabel класифікації. Тому результуюча модель скоріше за все буде використовувати обидва завдання для пошуку і класифікації ознак сну. Однак, саме це завдання найбільше підходить для тематики цієї роботи. [14][15]

1.2 Класи методів NER

Завдання іменованих сутностей можливо вирішити трьома групами методів, що відрізняються рівням складності виконання і потреби у генералізації вхідних даних.

1.2.1 NER на основі правил

Ця група методів дуже гарно працює для предметних областей, що мають чітко визначені правила використання іменованих образів.

У цій групі методів частіше всього використовуються регулярні вирази (виявляення email, URL, телефонів, тощо) (лістінг 1.1), словники (наприклад, для визначення імен людей, міст) (лістінг 1.2), лінгвістичні правила (виявлення місць за їх положенням в реченні та контекстом) та їх комбінації.

```
/^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$/g
```

Лістінг 1.1 – Приклад регулярного виразу для визначення електронної пошти.

```
"authors": [
    {
        "entity":  "Jane Austen",
    },
    {
        "entity":  "Ernest Hemingway",
    },
...
    {
        "entity":  "George Orwell",
    },
]
```

Лістінг 1.2 – Приклад словника для визначення авторів у форматі JSON

Але зазначені методи скоріше за все не спрацюють на предметній області з великою кількістю помилок, неологізмів, неможливістю визначити всі можливі образи та окреслити їх чіткі межі. [16]

1.2.2 Машинне навчання

Використання методів машинного навчання може спрацювати для визначення деяких, чітко визначених сутностей, яким можна навчити модель.

Серед цієї групи методів вирізняють дерева рішень, машини опорних векторів (SVM), умовні випадкові поля (CRF).

Хоча моделі цього класу методів і мають достатню спроможність узагальнювати отримані дані, але цей підхід потребує багатьох зусиль для

створення міток на самих даних та їх можливої модифікації при навчанні моделі. [17]

1.2.3 Глибоке машинне навчання

Група алгоритмів машинного навчання, що використовуються для вирішення задач, які потребують від моделі високого рівня узагальнення та виявлення закономірностей високого рівня складності. Навчання таких моделей потребує великої кількості даних та високопродуктивного обладнання, з-за їх більш комплексної структури порівняно з традиційними моделями групи машинного навчання. Проте навчання цього класу моделей не потребують втручання в процес навчання моделі.

Серед моделей цього класу відносять рекурентні нейронні мережі (RNN), мережі з довгою короткостроковою пам’яттю (LSTM) та трансформери.

Таблиці порівняння класів методі приведено в таблиці А.1 додатку А.

Отже, серед цих класів методів, незважаючи на свою дороговизну у навчанні, проте з великим потенціалом вирішення задач, використовуючи дані з помилками, нечіткостями та різноманітними мовними явищами обрано моделі глибокого машинного навчання. [18]

1.3 Методи глибокого машинного навчання

1.3.1 Рекурентні нейронні мережі

RNN – це тип нейронної мережі, що використовує послідовні дані для власного навчання. На відміну від традиційних нейронних мереж RNN вважають, що елементи у послідовності залежать один від одного та оновлюють власні ваги відповідно (рис. 1.1).

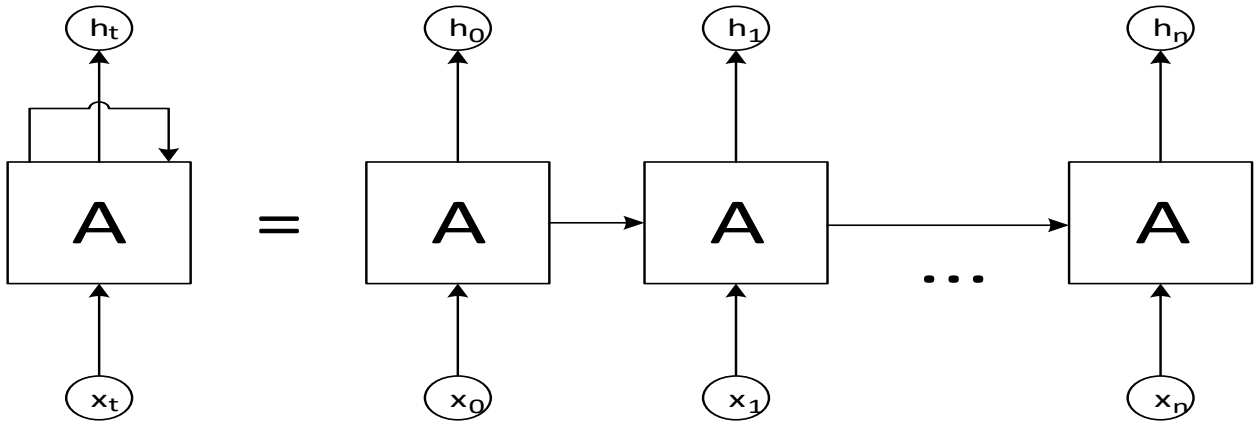


Рисунок 1.1 – Структура рекурентної нейронної мережі

І тим не менш, цей тип глибоких нейронних мереж, по-перше, навчається дуже довго, що ускладнює використання датасетів з великою кількістю даних.

По-друге, RNN не можуть добре запам’ятовувати інформацію під час багатьох ітерацій і мають схильність переписувати свою пам’ять на користь нових даних.

По-третє, існує так звана градієнтна проблема, що призводить до неспроможності виявляти довгострокові залежності. [19]

1.3.2 Мережі з довгою короткостроковою пам’яттю

Мережі LSTM є логічним розвитком RNN для рішення проблеми зникаючого градієнту, тим самим роблячи цей тип нейронних мереж більш підходящим для завдання NER та обробки довгих послідовностей (рис 1.2.).

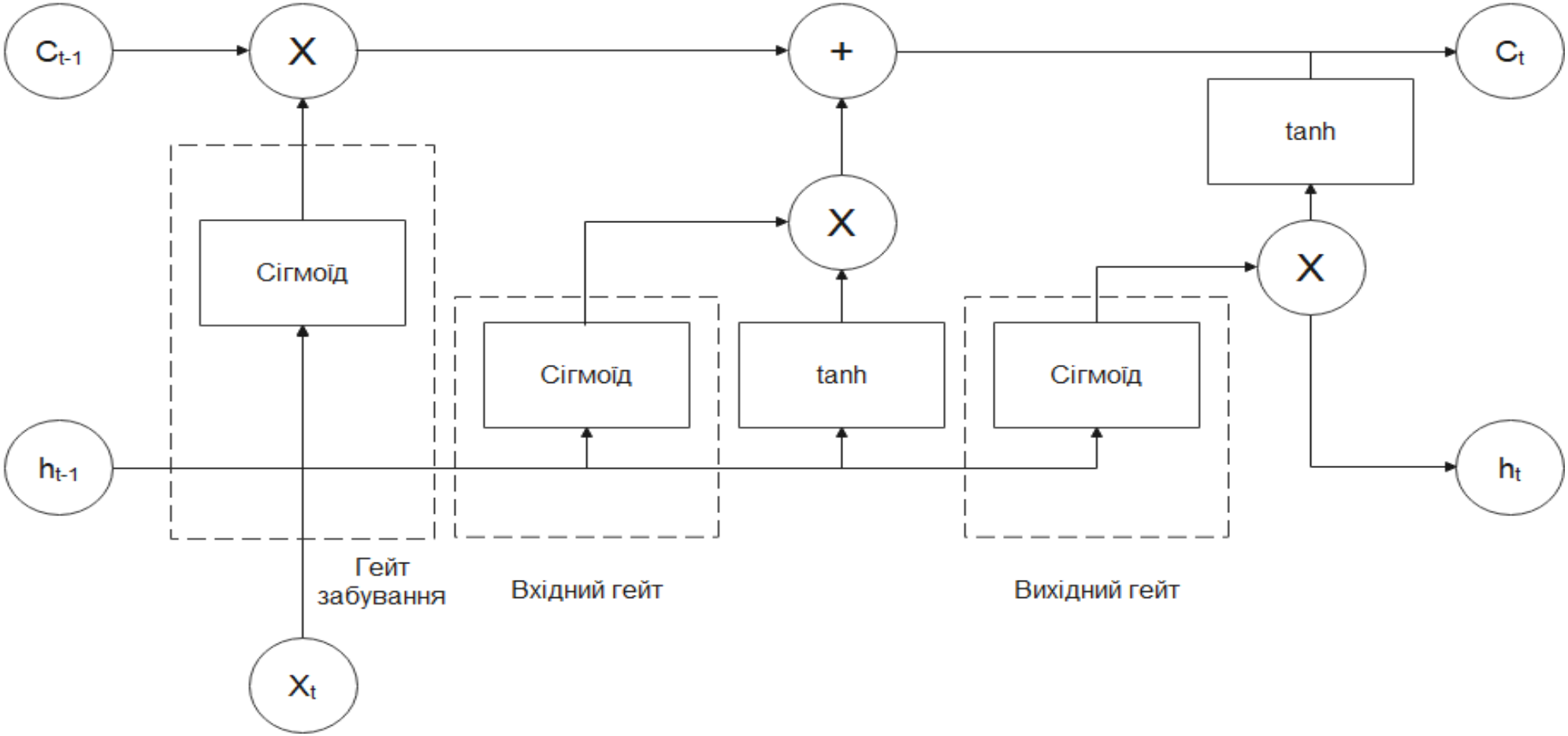


Рисунок 1.2 – Структура LSTM мережі

Незважаючи на це LSTM навчаються ще повільніше ніж RNN, потребують більше високопродуктивного обладнання та схильні до перенавчання при недостатній кількості даних. Крім цього, для налаштування моделі під конкретну предметну область потрібно підібрати правильну комбінацію таких параметрів як швидкість навчання, довжина послідовності та інших. [20]

1.3.3 Трансформерні моделі

Трансформери – це тип нейронної мережі з інноваційним механізмом само-уваги, що було представлено у 2017 році компанією Google. Цей тип моделей особливий тим, що вона навчається послідовними даними і з них генерує нові дані. [21]

Трансформери також розвивають архітектуру encoder-decoder, що використовується у RNN, проте замість неї використовується механізм самоуваги (self-attention), за допомогою якого у послідовності даних визначаються інформаційно важливі її елементи. Структуру трансформерної моделі наведено на рис. 1.3.

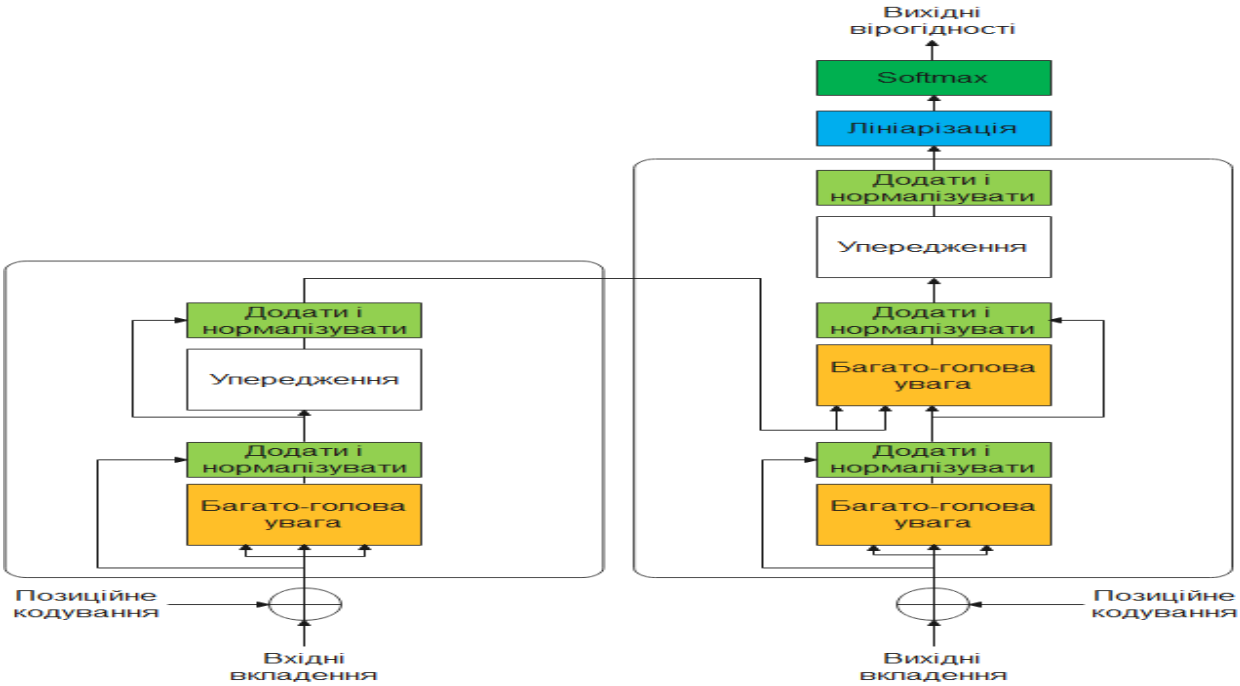


Рисунок 1.3 – Приклад трансформерної моделі

Моделі цього типу переважно навчаються швидше, краще розуміють контекст, підходять для різних предметних областей з-за своєї гнучкої архітектури.

Однак така зміна в структурі йде не без недоліків: для навчання подібної моделі необхідна величезна кількість різноманітних даних, які повинні оброблятись коштовним та високопродуктивним обладнанням. Крім того з-за її складної архітектури та кількості слоїв майже неможливо зрозуміти яким чином модель прийшла до того чи іншого висновку.

Порівняння цих методів глибокого машинного навчання приведено в таблиці А.2 додатку А.

Отже, всі три методи глибокого машинного навчання важко навчити з нуля, за допомогою одного чи декількох персональних комп'ютерів. Для досягнення подібних результатів потрібні як гігабайти текстових даних та датацентри з високопродуктивним обладнанням. Хоча, це можливо було б нівелювати донавчанням моделі на власному датасеті, але лише трансформерні моделі здатні забезпечити потрібну точність та швидкість навчання, чого не можна сказати про RNN та LSTM моделі. [22-24]

1.4 Моделі трансформерів

Трансформерні моделі залежно від їх призначення мають різні частини архітектури encoder-decoder. Дерево трансформер моделей приведено на рис. 1.4.

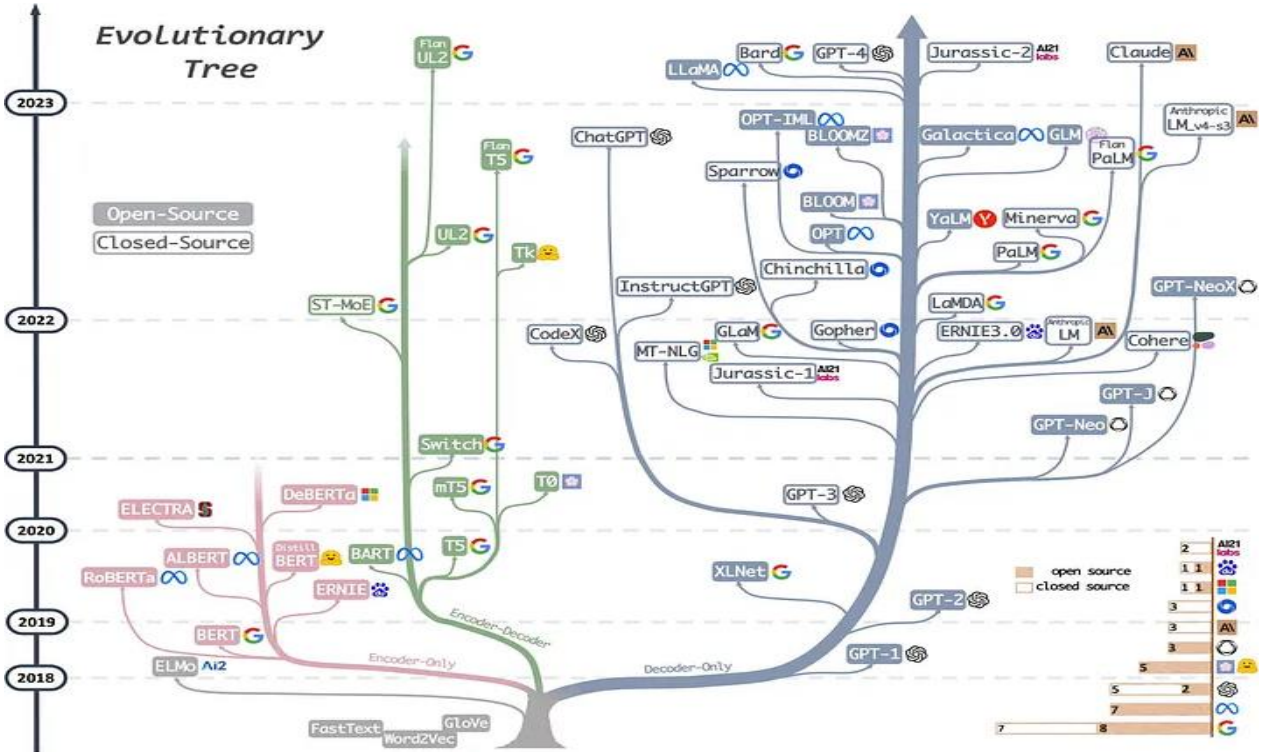


Рисунок 1.4 – Дерево еволюції трансформерних моделей на 2023 рік [25]

1.4.1 Енкодерні моделі

Енкодерні моделі використовують лише енкодер трансформаторної моделі. На кожному етапі шари уваги мають доступ до всіх слів у початковому реченні. Ці моделі часто характеризуються як такі, що мають "двонаправлену" увагу, і їх часто називають моделями автокодування. [26]

Моделі енкодерів найкраще підходять для завдань:

- класифікація тексту;
- розпізнавання іменованих сутностей;
- відповіді на питання.

Серед цих моделей можна виділити такі як:

- BERT;
- ALBERT;
- RoBERTa;
- DIstilBERT;

1.4.2 Декодерні моделі

Моделі з декодером використовують лише декодер моделі-трансформера. На кожному етапі для певного слова шари уваги мають доступ лише до слів, розташованих перед ним у реченні. Ці моделі часто називають авторегресивними моделями і підходять для завдань пов’язаних з генерацією тексту. Особливо великі моделі цього класу називають великими мовними моделями (LLM), з-за можливості використання у багатьох завданнях NLP. [27]

До декодерного типу моделей належать:

- GPT;
- LLaMa;
- Claude;
- Gemini.

Хоча і не вказано, що моделі цього типу використовуються для завдання NER, але за свого рівня узагальнення і кількості параметрів моделі цього типу можуть використовуватись і для цього.

Проте їх використання обмежено можливістю доступу до API компаній-власників LLM, ціною на запити до API та швидкістю відповіді самих моделей. Порівняння LLM приведено на рис 1.5.

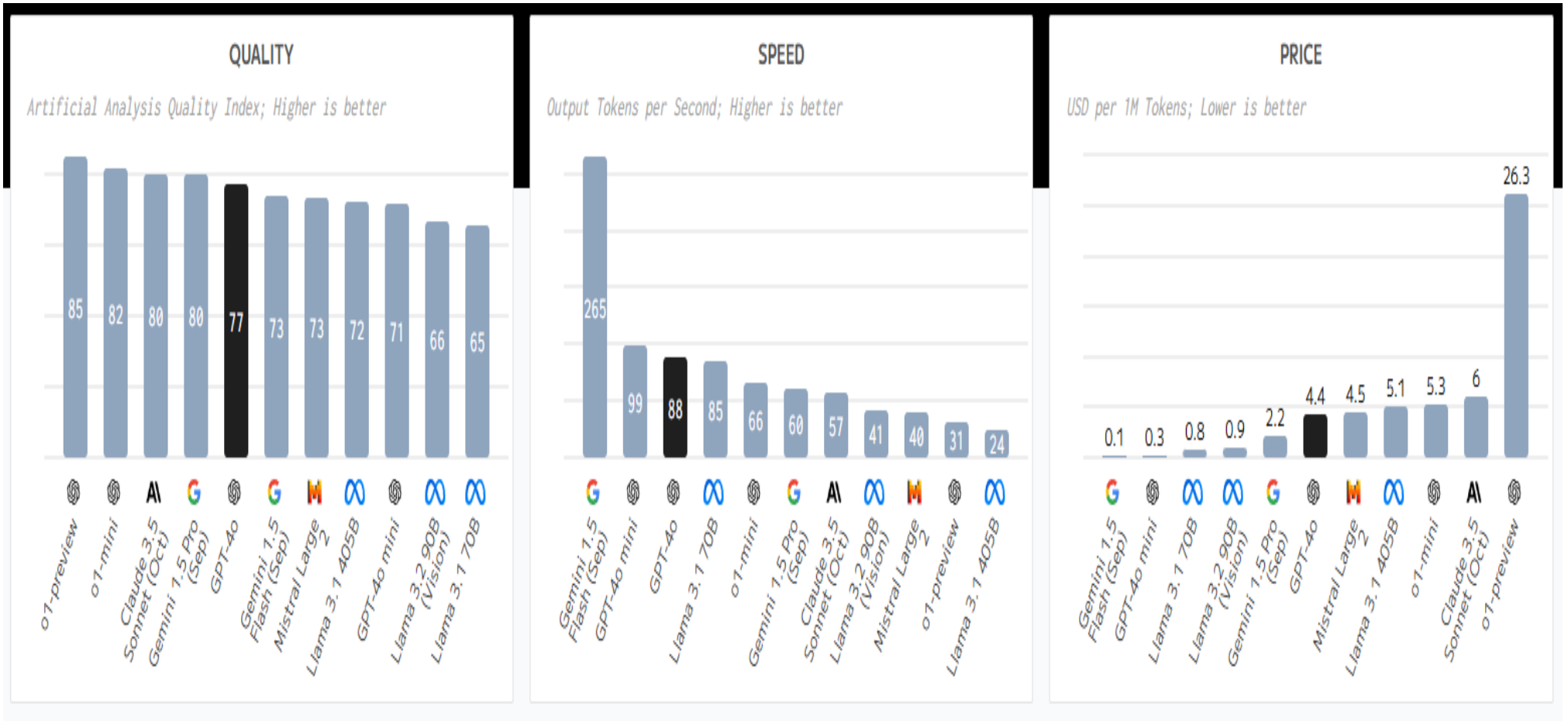


Рисунок 1.5 – Порівняння аналогів GPT за якістю, швидкістю і ціною [28]

1.4.3 Моделі «від послідовності до послідовності» або енкодер-декодер моделі

Моделі енкодер-декодер (також названі моделями "від послідовності до послідовності" або S2S) використовують обидві частини архітектури трансформатора. На кожному етапі шари уваги енкодера мають доступ до всіх слів у початковому реченні, тоді як шари уваги декодера мають доступ лише до слів, розташованих перед даним словом у вхідному тексті.

Послідовно-послідовні моделі найкраще підходять для завдань:

- узагальнення;
- переклад;
- генерування відповідей на запитання.

До моделей енкодер-декодерного типу належать:

- Bart;
- mBART;
- Marian;
- T5;

Однак потрібно зауважити, що моделі S2S не використовуються для завдання виявлення іменованих сутностей, тому моделі цього типу в цій роботі не розглядаються. [29]

Таблиці порівняння енкодерних, декодерних, декодерних з енкодерними та аналогами GPT-4 приведено в таблицях А.3-А.5 додатку А.

1.5 Висновок з огляду технологій класифікації текстів

Отже, виходячи із даних порівняння для досягнення зазначеної мети і вирішення задачі NLP класифікації тексту завдання NER методу глибокого машинного навчання можливо використовувати трансформерні моделі енкодерного класу, що найкраще показали себе на датасеті загального знання

Commonsense QA та мають навчену україномовну модель, або моделі декодерного класу, використання яких фінансово вигідно в Україні.

1.6 Постановка задачі

Для досягнення зазначеної мети потрібно зазначити, що помилки першого та другого роду різним чином відносяться до задач виявлення та класифікації.

1.6.1 Особливості мінімізації помилок першого і другого роду для задачі виявлення онейрологічних образів

Для задачі виявлення розглянемо матрицю збентеження (табл. 1.1) для варіантів виявлення-невиявлення ознаки сну. [30]

Таблиця 1.1. Матриця збентеження для виявлення ознак сну

Істинні значення	Передбачення	
	Виявлено	Не виявлено
Виявлено	TP	FN
Не виявлено	FP	TN

Представимо, що існує ознака сну, що складається з трьох слів.

Припустимо, що модель зробила помилку і не додала слово, що є частиною ознаки, тобто припустила помилку другого роду. В такому випадку при подальшій обробці модель може з меншим шансом правильно визначити клас ознаки.

Далі, представимо, що припустила помилку першого роду, додавши до контексту декілька слів що не є частиною ознаки. В такому випадку модель потенційно лише розширить контекст для свого розуміння тим самим

збільшуючи точність для подальшої класифікації, особливо, якщо використовується невелика кількість навчальних даних.

Проте, якщо модель додасть до однієї ознаки слова з усього тексту,то тоді неможливо буде визначити окрему ознаку сну. В такому випадку можливо сприймати увесь текст у вигляді одного, пусть навіть нечіткого, але онейрологічного образу.

В протилежному варіанті, модель взагалі не визначить ознаку, що є значно гіршим ніж визначення однієї ознаки сну.

Тому важливо, щоб модель визначила контекстуально основну частину ознаки і не відбракувала її, з-за цього потрібна саме мінімізація помилок другого роду. Присутність ж додаткових слів в контекстуально важливій частині ознаки сну (помилки першого роду) не є критичним до того моменту, поки вони не змінюють контекст усієї ознаки і навіть тоді можливо хоч і не чітко, але виявити ознаку сну.

Отже, на етапі виявлення ознак сну першочергово потрібно проводити мінімізацію помилок другого роду і за залишковим принципом ознак першого роду.

1.6.2 Особливості мінімізації помилок першого і другого роду для задачі класифікації онейрологічних образів

Для класифікації ознак сну потрібно використовувати матрицю збентеження для виявлення кожного класу за можливими схемами OVO (див. табл. 1.2) або OVR (див. табл. 1.3) [31].

Таблиця 1.2. Матриця збентеження для завдання класифікації по методу OVO

Істинні значення	Передбачення	
	$Клас_1$	$Клас_2$
$Клас_1$	TP	FN
$Клас_2$	FP	TN

Таблиця 1.3. Матриця збентеження для завдання класифікації по методу OVR

Істинні значення	Передбачення	
	$Клас_1$	Інший клас
$Клас_1$	TP	FN
Інший клас	FP	TN

Для задачі класифікації припустимо, що в нас є наближена до ідеалу модель виявлення і образ, що має два класи.

При відкиданні одного чи двох класів (помилка другого роду) інформація по сутності ознаки сну буде втрачено, що є неприйнятним в контексті цієї роботи і може завадити подальшій інтерпретації ознаки сну.

Але і при додаванні неправильних класів (помилка першого роду) ми можемо неправильно охарактеризувати ознаку сну, зробивши її спотвореною для інтерпритації людиною. Іншим можливим наслідком помилок першого роду

Отже, аналогічно до задачі виявлення при задачі класифікації втрата типу ознаки виявляється більш важливою, ніж мінімізація помилок першого роду.

1.6.3 План дослідження

Так як основною метою роботи є вдосконалення існуючих методів для досягнення цілі дослідження потрібно сформувати план його проведення:

1) збір даних із зовнішніх джерел у форматі необхідному для даної роботи, а саме: присутність записів снів та визначених експертом онейрологічних образів;

2) визначення можливих рішень на основі аналогічних систем машинного навчання;

3) визначення необхідних метрик для оцінки моделей;

4) визначити план проведення експериментів для потенційних рішень;

5) провести експерименти і оцінити моделі за обраними метриками;

6) визначити найкращі рішення за метриками;

7) провести опитування для виявлення кращого рішення.

2 ОПИС СИРИХ ДАНИХ ДЛЯ ФОРМУВАННЯ ПРИКЛАДНОГО ДАТАСЕТУ

Необхідність створення датасету випливає з відсутності у відкритому доступі україномовних датасетів по даній тематиці. З цієї причини було прийнято рішення створити власний датасет.

Джерелом даних виступає власний зовнішній інтернет ресурс, в якому реалізовано метод експортування даних у вигляді архіву з двома типами файлів metadata та dreams. Структура зовнішнього ресурсу не розглядається в даній роботі. [10]

Файл metadata описує дані і складається з таких полів:

- total_dreams – загальна кількість снів;
- total_users – кількість користувачів;
- users – перелік анонімізованих користувачів з трьох можливих полів:
  - id – ідентифікатор користувача в зовнішньому ресурсі;
  - time_zone – часова зона в форматі UTC, в якій знаходиться користувач;
  - birth_date – дата народження користувача;
- total_markers – кількість типів ознак;
- markers – перелік типів ознак, на основі яких відбувається виявлення та класифікація онейрологічних образів:
  - id – ідентифікатор маркера в системі;
  - name – назва типу ознаки;
  - description – опис ознаки;
  - category_name – назва категорії типу ознаки;
- date_range – інтервал дат, на основі яких сформовано пакет даних;
- export_date – дата екпорту даних;
- export_type – тип експортування даних (дані самого користувача або всіх користувачів в системі.

На основі цього файлу можливо загально описати дані, що складається з 315 власних снів. Ознаки цих снів класифіковано за декількома типами ознак.

Назва кожного типу ознак складається з назви самого типу і назви категорії типу ознаки за наступним форматом: НАЗВА ТИПУ ОЗНАКИ (НАЗВА КАТЕГОРІЇ).

Перелік типів ознак з їх описом:

1) Відчуття (Внутрішня обізнаність): Ви можете відчути параліч, або «вийти» з тіла, або несподівано відчути сильне сексуальне збудження;

2) Сприйняття (Внутрішня обізнаність): Сприйняття може бути незвично ясним або розпливчастим, або ви можете бачити чи чути що-небудь, що недоступне сприйняттю, коли ви не спите;

3) Емоції (Внутрішня обізнаність): Емоція може здатися вам недоречною;

4) Думки (Внутрішня обізнаність): Думка може бути незвичайною, з тих, що спадають на думку тільки уві сні або з тих, що можуть чарівним чином вплинути на світ сну;

5) Істот (Дії): Оточуючі вас люди або інші істоти роблять щось незвичайне або неможливе в стані неспання, але не пов'язане з вашими думками чи відчуттями;

6) Об'єктів (Дії): Об'єкти, що вас оточують виконують незвичні для них дії або мають незвичні для них властивості. Приклад: мій диван, наче пес, побіжав кудись;

7) Мої (Дії): Ви робите щось незвичайне або неможливе в стані неспання, але не пов'язане з вашими думками чи відчуттями;

8) Моя (Форма): Ваш вигляд незвичайний для вас. Він деформується або трансформується. Наприклад: ви розумієте, що голі на важливому зібранні чи у вас більше пальців аніж має бути або починають випадати зуби;

9) Істот (Форма): Вигляд інших персонажів незвичайний, деформується або трансформується. Аномаліями форми вважаються також незвичайний одяг або зачіска. Приклад: оточуючі вас не мають обличчя чи воно розпливчасте;

10) Об'єктів (Форма): Ваш вигляд навколишніх предметів незвичайний, деформується або трансформується. Приклад: я сидів на зеленому камні, що підозріло був схожий на телепата;

11) Місця (Форма): Вигляд чи форма місця, в якому ви перебуваєте уві сні здається вам незвичною. Приклад: у домі було більше кімнат, ніж у реальності; ринок ставав палацом;

12) Час (Обставини): Події відбуваються у недоречний їм час, чи в минулому або майбутньому;

13) Моя роль (Обставини): Ви виступаєте в невластивій для себе ролі;

14) Роль істот (Обставини): Будь хто інший взяв незвичну для себе роль. Приклад: батько став королем;

15) Ситуація (Обставини): Відчуття недоречності навколишнього. Наприклад: вам відома історія вигаданого світу; зненацька почалася війна із киборгами; вам поручили доставити вантаж на Аляску;

16) Істот (Місцезнаходження): Люди, тварини чи інші істоти опинились там де б ви їх ніколи не побачили;

17) Об'єктів (Місцезнаходження): Предмети, будівлі чи інші об'єкти знаходяться не на своїх місцях чи перебувають у неможливих положеннях. Як наприклад, церква на дорозі;

18) Моє (Місцезнаходження): Ви опинилися десь, де, найімовірніше, не були в реальному житті;

Частотний розподіл довжин текстів снів показано на рисунку 2.1. Сни в датасеті можуть мати доволі широкий розподіл від декількох десятків і сотень символів, що приблизно дорівнює малим реченням з пари слів або малим текстам.

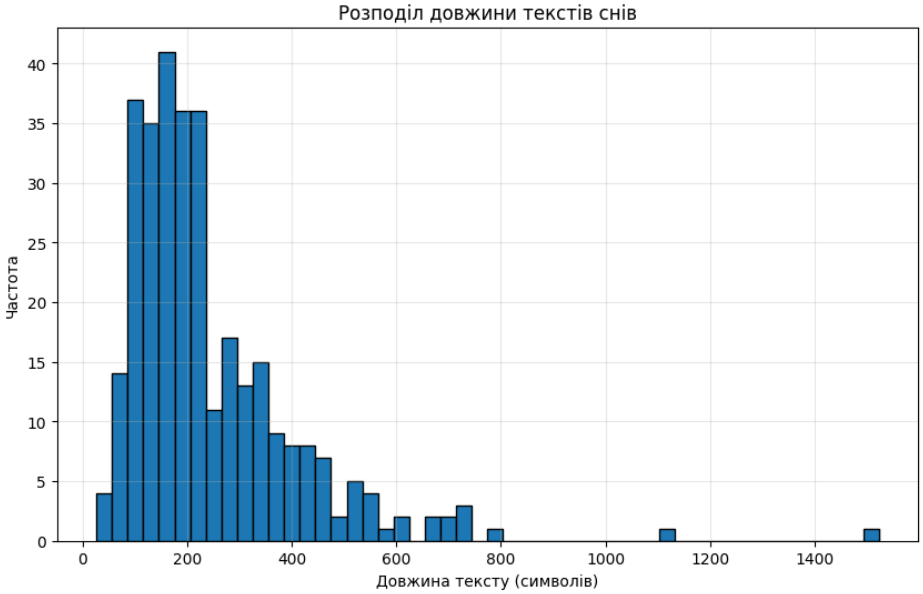


Рисунок 2.1 – Розподіл довжини текстів в символах

За розподілом довжин ознак (рисунок 2.2) можливо помітити, що більша частина ознак має довжину близько 20 – 60 символів. За середньої довжини слів української мови близько 5 символів на слово за простим розрахунком очевидно, що довжина ознак в словах коливається від 4 до 20, що підпадає під категорії довгих словосполучень та довгих речень художної мови. Проте, хоч і їх не так багато у порівнянні з основною групою, також присутні і ознаки, що займають більшу частину тексту.[32][33]

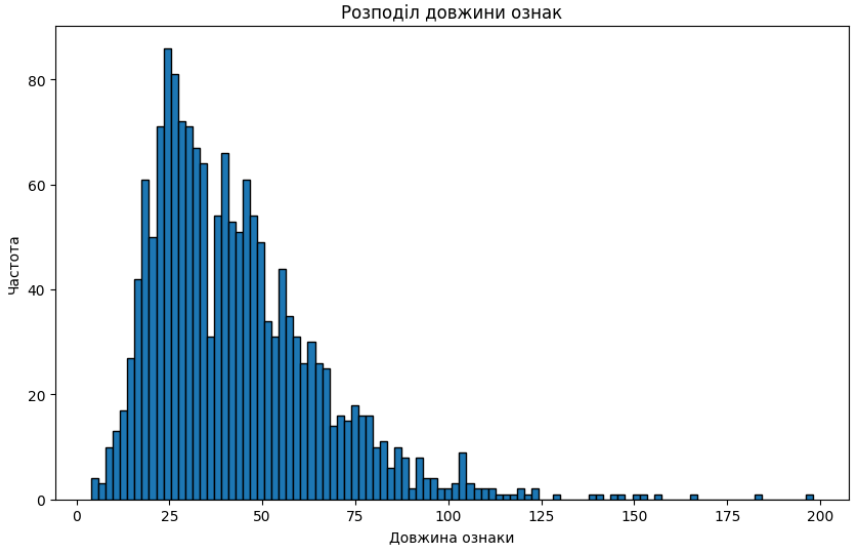


Рисунок 2.2 – Розподіл довжин ознак снів (символи)

Проте, спостерігаючи розподіл класів ознак (рис. 2.3) ясно, що серед даних присутні ознаки, кількість яких перевищує середню у багато разів. Цими ознаками є (за частотою зустрічі): мої дії, моє місцезнаходження, дії істот, ситуація та місцезнаходження істот.

Отже, з-за такої нерівномірності даних превалюючі ознаки будуть доволі сильно впливати на результуючі моделі. Також очевидно, що цих даних може бути недостатньо для навчання моделі для загального випадку. Проте, за відсутності інших подібних датасетів та даних, нічого не залишається окрім як використовувати ці дані для створення власного датасету.

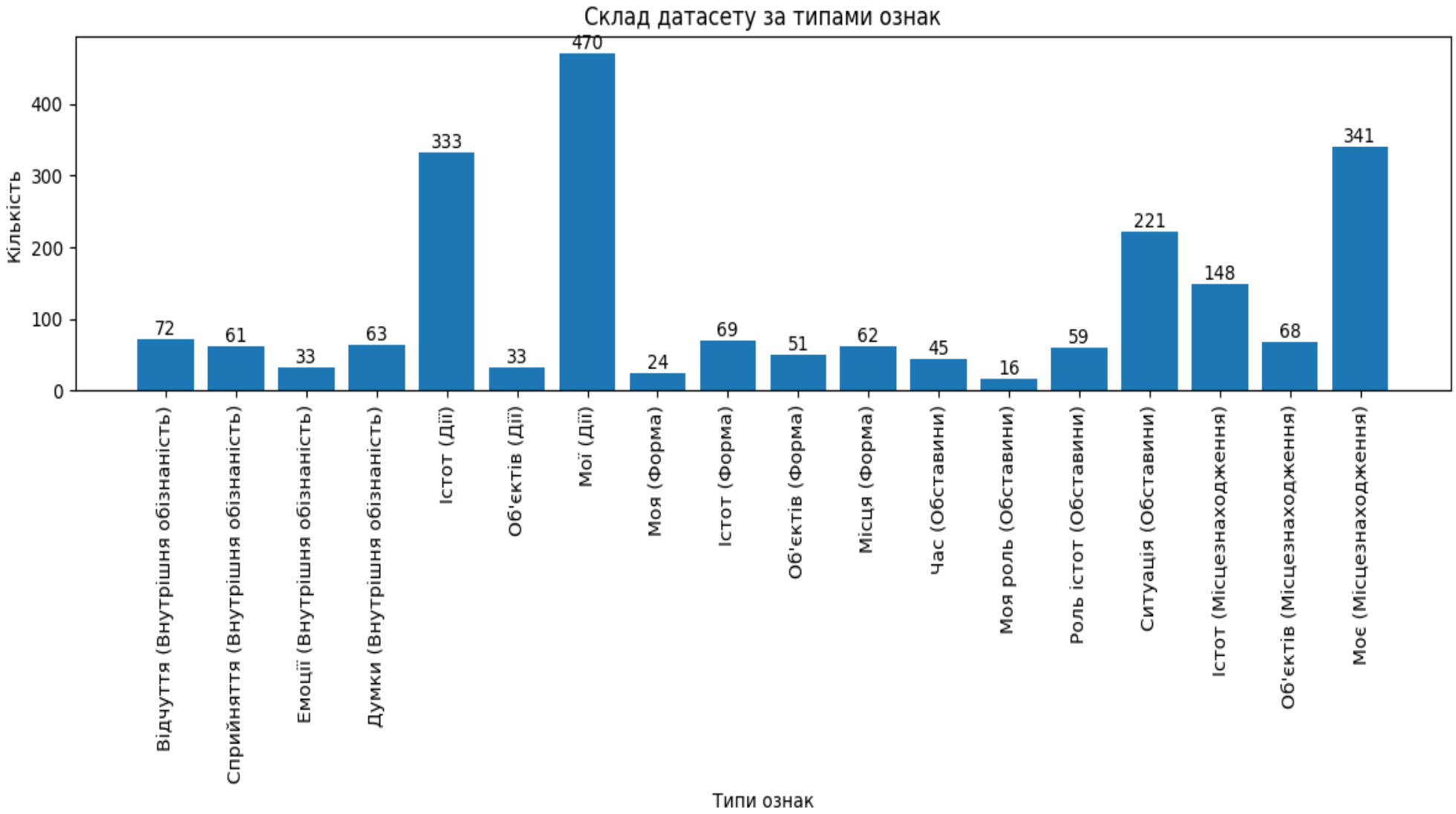


Рисунок 2.3 – Склад за типами ознак

3 ОПИС МОДЕЛЕЙ

3.1 Обчислювальні ресурси

Моделі сімейства BERT донавчено за допомогою наступних обчислювальних ресурсів локального середовища:

- Ноутбук ASUS TUF F15;
- Дванадцятиядерний Intel Core i5-12500H (2.5 - 4.5 ГГц);
- 32 ГБ DDR4 ОЗП;
- M2 SSD 1TB;
- Nvidia 4060 TI Mobile з 8 ГБ відеопам’яті;

При неможливості навчати моделі у локальному середовищі використовувалось хмарне середовище Google Collab з наступними характеристиками:

- 12.7 ГБ ОЗП;
- Сховище 112 ГБ + підключений Google Disk;
- Nvidia T4 Tensor Chip з 16 ГБ відеопам’яті;

3.2 Опис основних бібліотек

Для досягнення зазначеної мети, використовуючи трансформерні моделі енкодерного та декодерного класів, протестовано декілька видів архітектур, що включають: одну універсальну модель та дві трансформерні моделі різних класів для виконання розділеного виявлення та класифікації іменованих образів.

Вільний доступ до енкодерних моделей сімейства BERT надає середовище Hugging Face. Навчання проводиться за допомогою бібліотеки PyTorch та CUDA ядер графічного процесору для розподілених обчислень.

Однак моделі декодерного типу не використовуються за замовчуванням для завдання NER і цю проблему можливо вирішити або створенням деталізованого запиту (промпту) до великої мовної моделі, або використанням бібліотеки, що надає програмний інтерфейс для виконання зазначеного завдання.

Існує пакет spacy-llm на МП Python, який інтегрує великі мовні моделі (LLM) у конвеєри для перетворення неструктурованих відповідей на надійні результати для різних завдань NLP. [34]

У пакеті доступні наступні LLM моделі:

- GPT-4o;
- GooglePaLM;
- Claude 3;
- Mistral;
- Dolly;
- LLaMa 2.

Конфігурувати та донавчати великі мовні моделі перебором гіперпараметрів, як наприклад для моделей сімейства BERT, неможливо з-за відсутності прямого доступу до моделі окрім API або потреби в обчислювальних потужностях декількох датацентрів із найсучаснішим обладнанням.

Навчання таких моделей можливо завдяки zero-shot або few-shot навчанню, що відповідає сценарію, при якому модель спроможна виявляти та катагоризувати об’єкти та концепції без необхідності навчатись або донавчаючись на декількох прикладах. [35][36]

Таким чином цей пакет пропонує створення конфігураційного файлу для заданої моделі, а потім його використання у відповідному скрипті (див. лістінги 2.1-2.3). Тобто завдання NER можливо виконати і великими мовними моделями.

```
[nlp]
lang = "ua"
pipeline = ["llm"]

[components]

[components.llm]
factory = "llm"

[components.llm.task]
@llm_tasks = "spacy.NER.v3"
labels = ["PERSON", "ORGANISATION", "LOCATION"]

[components.llm.model]
@llm_models = "spacy.GPT-4.v1"
```

Лістінг 2.1 – Приклад конфігураційного файлу для завдання NER моделі GPT-4

```
from spacy_llm.util import assemble
nlp = assemble("/path/to/config.cfg")
doc = nlp("Дмитро і Михайло проїхались на велосипеді по Дюковському парку")
print([(ent.text, ent.label_) for ent in doc.ents])
```

Лістінг 2.2 – Приклад скрипту, що застосовує модель на тексті.

```
[
   ['Дмитро', 'PERSON‘],
   ['Михайло', 'PERSON‘],
   ['Дюковському парку', 'LOCATION‘]
]
```

Лістінг 2.3 – Результат виконання скрипту

3.3 Модель розділеного виявлення та класифікація

Як описано в порівнянні завдань класифікації тексту, традиційно моделі машинного навчання навчання використовують одне із завдань класифікації тексту, але в контексті цієї роботи і обмежень кожного з них прийнято рішення об’єднати їх в загальну структуру. Проте, зробити це у рамках однієї моделі

потребує додаткового дослідження, тому прийнято рішення розділити ці завдання між двома моделями. [10]

Перша модель виступає в ролі детектора онейрологічних образів за допомогою бінарної класифікації присутній-неприсутній, проте така схема має обмеження. Це обмеження пов'язане з неможливістю коректного розділення сусідніх ознак при використанні простої бінарної класифікації.

Розглянемо формальний опис проблеми. Нехай маємо токенізоване речення, що складається з n токенів t_i:

$$[t_1, t_2, t_3, t_4 \dots t_n], \qquad i \in [1, n], \mathrm{n} \in Z \tag{3.1}$$

В випадку бінарної класифікації кожному токену t_i буде співставлено клас c_0 або c_1, де c_0 – клас відсутності ознаки, c_1 – клас присутності ознаки. Тоді результуюча послідовність класів матиме вигляд:

$$[c_{j_0}, c_{j_1}, c_{j_3}, c_{j_4} \dots c_{j_n}] \tag{3.2}$$

де j_i – визначений моделлю клас (c_0 або c_1) для токену з індексом $i \in [1, n], \mathrm{n} \in Z$.

Проблема виникає при наявності двох сусідніх ознак у тексті. Розглянемо послідовність токенів, де присутні дві різні ознаки:

$$[t^1, t^1, t^2, t^2 \dots] \tag{3.3}$$

де t^1та t^2– токени, що належать до різних ознак.

При бінарній класифікації обидві ознаки будуть позначені як одна:

$$[c_1, c_1, c_1, c_1 \dots] \quad (3.4)$$

Це призводить до втрати інформації про межі між окремими ознаками та унеможливлює їх подальшу окрему класифікацію.

Для вирішення проблеми злиття сусідніх ознак пропонується модифікувати схему класифікації токенів шляхом додавання додаткового класу, що позначає порядок токенів у межах однієї ознаки. Замість бінарної класифікації (ознака присутня/відсутня) використовується система мультикласифікації на трьох класах:

- c_0 – відсутність ознаки;
- c_1 – початок нової ознаки;
- c_2 – продовження попередньої ознаки;

При такому підході послідовність токенів, що належить до однієї ознаки, буде мати вигляд:

$$[c_1, c_2, c_2, c_2 \dots] \quad (3.5)$$

У випадку наявності декількох ознак у тексті, що знаходяться поруч, послідовність класів матиме вигляд:

$$[\dots c_0, c_1, c_2, c_1, c_2, c_0 \dots] \quad (3.6)$$

Таким чином проблему детекції буде вирішено.

Тринарна система класифікації токенів дозволяє ефективно розділити текст на окремі ознаки. Процес відновлення текстових фрагментів ознак відбувається наступним чином:

1) Знаходження токену класу c_1, що позначає початок ознаки;

2) Послідовне об'єднання всіх наступних токенів класу c_2 до:

а. появи нового токену класу c_1 (початок нової ознаки);

б. або токену класу c_0 (кінець ознаки);

Наприклад, для послідовності класів та відповідних їм токенів (див. лістінг 3.1)

```
Класи: [c₁, c₂, c₂, c₂, c₂, c₁, c₂, c₂]
Токени: [Над, столом, летів, чорний, кіт, неподалік, танцювала, свічка]
```

Лістінг 3.1 – Приклад співпадіння класів токенам

Можна виділити дві окремі ознаки:

1) "Над столом летів чорний кіт"

2) "неподалік танцювала свічка"

Проте такий формат викликає іншу проблему (див. таблицю 3.1). Так як тепер присутні більше класів, помилка у визначенні початку ознаки чи середини сну хоч першого хоч другого роду призведе до кардинальної зміни самої ознаки, що може допомогти моделі зменшити довжину ознаки, роблячи класифікацію ознаки більш чіткою. Проте, якщо така помилка призведе до розділу неділимої ознаки, то це призведе до марної класифікації такого фрагменту.

Таблиця 3.1. Матриця збентеження для виявлення початку ознак сну по тринарній системі класифікації з класом початку ознаки як цільовим класом

Істинні значення	Передбачення		
	Початок ознаки	Середина ознаки	Не виявлено
Початок ознаки	TP	FN	FN
Середина ознаки	FP	TN	TN
Не виявлено	FP	TN	TN

Отримані текстові фрагменти далі подаються на вхід класифікатора, який виконує multilabel класифікацію – тобто кожній ознаці може бути присвоєно декілька класів з визначеного набору (наприклад, "Дії істот", "Форма істот" тощо).

Для multilabel класифікації виділених ознак використовується метод one hot encoding, де кожному класу відповідає бінарне значення (0 або 1) в векторі довжиною 18 (за кількістю можливих класів ознак). Для кожної ознаки формується вектор, де 1 позначає приналежність до відповідного класу, а 0 - відсутність класу: [37]

$$\vec{y} = [y_1, y_2, \ldots, y_{18}], \text{ де } y_i \in \{0,1\} \quad (3.7)$$

Наприклад, для ознаки "над столом летів чорний кіт", що одночасно належить до класів "Дії істот" (позиція 5), "Форма істот" (позиція 9) та «Місцезнаходження істот» (позиція 16), вектор матиме вигляд:

$$\vec{y} = [0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,1,0,0] \quad (3.8)$$

Таке представлення по-перше підходить для сприйняття енкодерних моделей сімейства BERT, яким необхідно передавати вектори статичних значень. По-друге, такий формат використовується у бібліотечних методах оцінки моделей, що також потребують однорідні масиви для своєї коректної роботи.

3.4 Повноцінна модель

В контексті даної роботи під повноцінною моделлю розуміється end-to-end підхід з використанням великої мовної моделі (LLM). Такий підхід

передбачає виконання всього процесу виділення та класифікації онейрологічних образів в рамках однієї моделі, на відміну від розділеного підходу з використанням окремих моделей для виявлення та класифікації. [38]

На відміну від розділених моделей цілісна модель має краще розуміти контекст тексту, тим самим зменшуючи кількість потенційних помилок на різних етапах обробки.

Якщо в моделях сімейства BERT текст потрібно явно доводити, до однакової довжини, щоб коректно навчити модель, то для LLM проводити такі дії є зайвим.

Також, моделі декодерного класу не вигідно використовувати як моделі розділеного виявлення та класифікації по причині їх в декілька разів меншої швидкості генерації тексту.

Так як модель виконує завдання, що повністю не належить ні класифікації токенів, ані послідовностей. Тому, навіть використовуючи бібліотеку spacy, потрібно описати формат власного завдання.

Для цього в бібліотеці потрібно зарєструвати власний формат за допомогою декоратора, якому потрібно передати назву нового завдання: `@registry.llm_tasks("DreamMomentCatTask.v1")`. [39]

І застосувати його на функції-фабриці завдання (див. лістінг 3.2) [40]

```
@registry.llm_tasks("DreamMomentCatTask.v1")
def make_dream_moment_cat_task(
    labels: List[str],
    template: str = DEFAULT_DDC_TEMPLATE,
    description: Optional[str] = None,
    label_definitions: Optional[Dict[str, str]] = None,
    prompt_examples: Optional[List[Dict[str, Any]]] = None
) -> DreamMomentCatTask:
    return DreamMomentCatTask(
        labels=labels,template=template,description=description,
        label_definitions=label_definitions,
        prompt_examples=prompt_examples
    )
```

Лістінг 3.2 – Фабрика завдання класифікації ознак сну

Аргументи фабрики зазначеної в лістінгу описують:

- labels – перелік типів ознак сну (необхідне);
- template – шаблон jinja, в який будуть підставлятись текстові дані;
- description – додатковий опис для моделі;
- label_definitions – опис конкретних типів ознак;
- prompt_examples – приклади відповідей для моделі;

В класі самого завдання нас цікавлять функції `genetate_prompts` та `parse_responses`.

Так функція `genetate_prompts` перетворює внутрішнє представлення віхдних текстів від spaCy у формат, який передається великій мовній моделі (див. Лістінг 3.3).

```
def generate_prompts(self, docs: Iterable[Doc]) -> Iterable[str]:
        for doc in docs:
            prompt = self.template.render(
                text=doc.text,
                labels=self.labels,
                description=self.description,
                label_definitions=self.label_definitions,
                prompt_examples=self.prompt_examples
            )
            yield prompt
```

Лістінг 3.3 – Функція класу DreamMomentCatTask, що генерує промпти

Функція `parse_responses` перетворює текст, що було згенеровано моделлю у перелік ознак з їх типами (див. Лістінг 3.4).

```
def parse_responses(
        self, docs: Iterable[Doc], responses: Iterable[str]
    ) -> Iterable[Doc]:
        parsed_responses = parse_responses(responses, self)

        for doc, moments in zip(docs, parsed_responses):
            doc.spans["dream_moments"] = []
            for moment in moments:
                try:
```

Лістінг 3.4 – Функція класу DreamMomentCatTask, що перетворює дані від моделі в текст

```
                start = doc.text.index(moment["text"])
                end = start + len(moment["text"])
                span = doc.char_span(start, end)
                span._.can_classify = moment["can_classify"]
                span._.labels = moment["labels"]
                doc.spans["dream_moments"].append(span)
            except ValueError:
                continue
    yield doc
```

Лістінг 3.4, аркуш 2 – Функція класу DreamMomentCatTask, що перетворює дані від моделі в текст

Сам шаблон промпту у форматі Jinja наведено в додатку Б. Сам шаблон написано англійською як основною мовою для створення промптів, на основі промптів, що використовуються самими розробниками бібліотеки. [41][42]

Розроблений шаблон промпту представляє собою структурований документ, що складається з декількох логічних частин, кожна з яких виконує специфічну роль у процесі аналізу снів.

Перша частина шаблону встановлює контекст взаємодії з моделлю. У ній визначається роль моделі як експертної системи з аналізу снів та окреслюється основне завдання – виявлення та класифікація окремих моментів сну в тексті сну.

Наступна частина містить детальні інструкції щодо обробки тексту. Вона пояснює, що ознаки сну не повинні перетинатися між собою, можуть варіюватися за довжиною від одного речення до декількох абзаців. Також тут описується процес класифікації: модель повинна визначити, чи можливо класифікувати конкретну ознаку, і якщо так – надати відповідні типи ознак з наданого переліку. У випадку неможливості класифікації передбачено спеціальну позначку: `==NONE==`.

Важливою складовою шаблону є секція, що визначає формат виводу. Вона забезпечує уніфікований спосіб представлення результатів, де кожен виділений момент сну супроводжується порядковим номером, флагом

можливості класифікації та переліком релевантних міток (див. лістінги 3.5 та 3.6). Такий структурований формат полегшує подальшу автоматичну обробку результатів.

```
NUMBER. DREAM MOMENT TEXT | FLAG | Label1, Label2, Label3
```

Лістінг 3.5 – Загальний шаблон для великої мовної моделі

```
1. Я йшов по темному лісі. | True | Моє місцезнаходження
2. Раптом побачив світло між деревами. | True | Місцезнаходження об'єктів
3. Підійшовши ближче, я зрозумів, що це була хатинка. | True | Місцезнаходження об'єктів, Мої думки
```

Лістінг 3.6 – Шаблон прикладів для великої мовної моделі

Шаблон включає кілька умовних секцій, які можуть бути присутніми залежно від конкретного використання. Серед них – додатковий опис завдання, що може надавати специфічний контекст для конкретного випадку застосування. Також може бути присутній перелік можливих міток для класифікації та їх детальні визначення, що допомагає моделі точніше розуміти критерії класифікації. Проте, з-за того, що сам датасет є неоднорідним, тому моделі була надана інструкція не сприймати цей датасет як вичерпні інструкції, а лише як загальні рекомендації.

Особливу роль відіграє секція з навчальними прикладами. Вона реалізує механізм few-shot навчання, де модель може спиратися на готові приклади правильної обробки та класифікації снів.

У випадку відсутності користувацьких прикладів, шаблон містить стандартний приклад українською мовою, який демонструє очікуваний формат роботи з різними типами онейрологічних образів для zero-shot навчання.

Шаблон спроектований з урахуванням можливості масштабування – він підтримує додавання нових параметрів класифікації, зміну форматів вхідних

даних та розширення системи класифікації. При цьому зберігається консистентність у обробці даних та форматі виводу, що важливо для стабільної роботи всієї системи аналізу снів.

Також для повноцінної моделі, що одночасно виконує завдання виявлення та класифікації онейрологічних образів, важливість мінімізації обох типів помилок набуває комплексного характеру.

Помилки першого роду у контексті повноцінної моделі проявляються у двох аспектах. З одного боку, це виявлення неіснуючих ознак або включення надлишкового контексту до реальних ознак, а з іншого – присвоєння ознакам невластивих їм класів. Такі помилки призводять до суттєвих практичних наслідків. Збільшується час обробки через необхідність аналізу надлишкових даних, відбувається потенційне спотворення статистики використання різних типів ознак. Крім того, виникає можливість плутанини при подальшому використанні результатів аналізу, а також зростають витрати обчислювальних ресурсів.

Помилки другого роду також мають подвійний характер, проявляючись як у пропуску існуючих ознак або важливих частин їх контексту, так і в невизначенні релевантних класів для правильно виявлених ознак. Наслідки таких помилок виявляються більш критичними для загального результату роботи моделі. Відбувається безповоротна втрата інформації про важливі елементи сну, що призводить до неповноти аналізу та може суттєво вплинути на подальшу інтерпретацію. Це знижує практичну цінність результатів для користувача та потенційно спотворює загальну картину сновидіння.

З огляду на це, для повноцінної моделі оптимальною є стратегія, що першочергово спрямована на мінімізацію помилок другого роду для забезпечення повноти аналізу, але при цьому тримає під контролем рівень помилок першого роду для збереження практичної застосовності результатів.

3.5 Метрики

При оцінці якості роботи моделі детекції онейрологічних образів важливо використовувати комплексний набір метрик, кожна з яких висвітлює різні аспекти її функціонування. Нижче описано основні метрики, що використовуються для оцінки.

Точність (Accuracy) – це частка правильних прогнозів моделі серед усіх зроблених прогнозів. Для завдання детекції ця метрика показує, наскільки часто модель правильно визначає наявність або відсутність ознаки у тексті.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.9}$$

При наближенні метрики Accuracy до одиниці кількість помилок першого і другого родів зменшується. Проте в контексті поставленої задачі важливо звернути особливу увагу на мінімізацію помилок другого роду, оскільки втрата інформації є критичнішою ніж її надлишковість. Саме тому доцільно розглянути окремо метрики Precision та Recall. [43]

Precision (точність передбачення) вимірює частку правильно визначених позитивних випадків серед усіх випадків, які модель визначила як позитивні:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.10}$$

Чим краща ця оцінка, тим менше помилок першого роду зроблено моделлю.

Recall (повнота) показує, яку частку від усіх позитивних випадків модель змогла виявити:

$$Recall = \frac{TP}{TP + FN} \tag{3.11}$$

При наближенні до одиниці показує зменшення помилок другого роду.

F_β(прийнято, що β=1) – гармонійне середнє між Precision та Recall, де β визначає відносну важливість повноти відносно точності:

$$F_\beta = (1 + \beta^2)\,\frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall} \tag{3.12}$$

де β =1 вказує на рівнозначність Precision та Recall у кінцевій оцінці.

Таким чином, хоча загальна точність (Accuracy) враховує обидва типи помилок, для нашої задачі особливу увагу слід приділити метриці Recall, яка безпосередньо відображає здатність моделі мінімізувати помилки другого роду. При цьому не можна повністю ігнорувати Precision, оскільки надмірна кількість помилок першого роду може ускладнити практичне використання моделі. Тому доцільно використовувати і метрику F_β. [44]

Hamming Loss (відстань Хеммінга) вимірює частку неправильних передбачень: [45]

$$\text{Hamming Loss} = \frac{1}{nL}\sum_{i=1}^{n}\sum_{j=1}^{L} I\left(y_{ij} \neq \acute{y}_{ij}\right) \tag{3.13}$$

де n – кількість зразків;

L – кількість міток;

y_{ij} – істинне значення;

$\acute{y}_{ij}$ – передбачене значення;

I – індикаторна функція.

ROC AUC Score (площа під ROC-кривою) вимірює здатність моделі розрізняти класи: [46][47]

$$AUC = \int_0^1 ROC(t)\, dt \tag{3.14}$$

де ROC – функція, що задає значення кривої ROC, у залежності від порогу відсікання t, відносно TPR (частка правильно визначених позитивно з позитивних випадків) і FPR (частка неправильно визначених позитивно з негативних випадків).

Таким чином, обраний набір метрик дозволяє всебічно оцінити якість роботи моделей, приділяючи особливу увагу здатності мінімізувати помилки другого роду при збереженні загальної ефективності класифікації. При цьому використання декількох взаємодоповнюючих метрик забезпечує більш об'єктивну та повну оцінку результатів роботи моделей.

4 ОПИС ЕКСПЕРИМЕНТІВ

Всього в роботі присутні три групи експериментів:

1) Експерименти, для навчання моделей сімейства BERT як моделі виявлення ознак сну;

2) Експерименти, для навчання моделей сімейства BERT як моделі класифікації ознак сну;

3) Експеременти, для навчання LLM для виявлення і класифікації ознак сну;

Кожен експеримент складається з наступних секцій:

- секція з підготовки датасету для поточної задачі;
- секція підготовки даних з датасету під конкретні моделі;
- секція із налаштування моделі та підготовки моделі до навчання;
- секція із описом метрик та особливостей їх розрахунку
- секція із описом ідеї експерименту чи експериментів.

Код створення датасетів представлено у додатку В.

4.1 Виявлення ознак сну за допомогою моделей сімейства BERT

Для опису використано приклад роботи з моделю youscan/ukr-roberta-base.

Основний код потрібний для проведення експериментів над моделями виявлення наведено в додатку Г.

4.1.1 Підготовка датасету

Для завдання виявлення ознак сну використовується спеціальна схема розмітки тексту на рівні токенів, що дозволяє точно визначати межі ознак у

тексті. Замість простої бінарної класифікації (є ознака / немає ознаки) застосовується схема розмітки з трьома класами:

- `NONE` (код 0) - токен не належить до ознаки сну;
- `B-EXIST` (код 1) - токен є початком нової ознаки сну;
- `I-EXIST` (код 2) - токен є продовженням поточної ознаки сну;

Така схема розмітки відома як BIO та широко використовується в завданнях розпізнавання іменованих сутностей (NER). Вона дозволяє чітко розділяти послідовні ознаки в тексті та уникати їх злиття. При цьому кожному токену присвоюється одна з трьох міток: `NONE` для токенів, що не належать до ознак сну, `B-EXIST` для токенів, що починають нову ознаку, та `I-EXIST` для токенів, що продовжують поточну ознаку. [48

]

Процес підготовки датасету включає наступні етапи:

- завантаження вихідних даних з JSON файлів;
- токенізація текстів снів з використанням простого токенізатора на основі регулярних виразів, що розділяє текст на слова та знаки пунктуації;
- розмітка токенів відповідно до схеми BIO шляхом зіставлення з наявними ознаками;
- перевірка якості розмітки шляхом порівняння кількості виявлених початків ознак (`B-EXIST`) з кількістю ознак у вихідних даних.

Етап токенізації відбувається за допомогою функції, представленою у лістінгу 4.1.

```
def tokenize(text):
    return re.findall(r'\w+|[^\w\s]', text.lower())
```

Лістінг 4.1 – Функція токенізації тексту на слова та знаки пунктуації

Після відбувається розмітка токенів шляхом пошуку співпадаючих послідовностей ознак сну із токенами тексту. Пошук співпадаючих областей

відбувається за допомогою функції, показаною в лістінгу 4.2. В функції також додано пороговий параметр, для відсічення неправильно розмічених ознак сну.

```
def is_moment_equal(tokens, start_idx, moment_tokens, threshold=0.8):
    incorrect_tokens = 0
    moment_tokens_length = len(moment_tokens)
    for j in range(start_idx, len(tokens)):
        if j - start_idx >= moment_tokens_length:
            break
        if tokens[j] != moment_tokens[j - start_idx]:
            incorrect_tokens += 1
    return incorrect_tokens / moment_tokens_length <= 1 - threshold
```

Лістінг 4.2 – Функція пошуку співпадаючих послідовностей

Для забезпечення якості розмітки використовується механізм нечіткого співставлення токенів з пороговим значенням 0.8, що дозволяє коректно обробляти випадки з незначними відмінностями в написанні слів.

Підготовлений датасет зберігається у форматі Hugging Face datasets, що забезпечує зручний інтерфейс для подальшої роботи з даними при навчанні моделей сімейства BERT. Кожен запис у датасеті містить:

- id: унікальний ідентифікатор сну;
- tokens: список токенів тексту;
- labels: список міток для кожного токена згідно схеми BIO .

В датасет також додано назви міток, для зручності роботи з ним.

Аналіз якості розмітки показав високу точність - лише для 2 снів з 315 було виявлено розбіжності в кількості ознак, що становить менше 1% від загального обсягу датасету. Такий результат свідчить про надійність обраного підходу до розмітки даних.

4.1.2 Підготовка даних

Підготовка даних для навчання моделі включає декілька етапів обробки та форматування вхідних текстів та міток. Для роботи з моделлю сімейства

BERT використовується спеціальний токенізатор, що є частиною самої моделі. Наприклад, для української моделі youscan/ukr-roberta-base відповідає токенізатор RobertaTokenizer. Він відповідає за розбиття тексту на токени у форматі, що підходить для обробки моделлю.

Процес підготовки даних складається з наступних кроків:

- токенізація текстів;
- встановлення максимальної довжини послідовності, наприклад, у 512 токенів;
- додавання спеціальних токенів в кінці послідовності, для унифікації довжин текстів для кращого сприйняття моделлю;
- перетворення слів у відповідні індекси зі словника токенізатора;
- створення маски уваги (attention mask) для розрізнення значущих токенів від паддінгу.

В процесі токенізації також використовується схема розмітки BIO.

Особлива увага приділяється обробці підслів, які виникають при токенізації, та встановленню спеціального значення -100 для службових токенів та токенів заповнення.

При цьому також потрібно встановлювати це значення для токенів, що належать слову, окрім першого. Це потрібно, щоб модель уникала ситуацій, коли одна частина слова належить ознаці сну, а інша – ні.

Після підготовки токенізованих текстів та їх міток відбувається формування навчального та валідаційного наборів даних. Датасет розділяється у співвідношенні 80% на навчання та 20% на валідацію, що є стандартною практикою в машинному навчанні.

Дані перетворюються у формат, сумісний з бібліотекою HuggingFace Datasets, що забезпечує зручну роботу з ними під час навчання моделі. Для ефективної обробки даних під час навчання використовується спеціальний колектор даних `DataCollatorForTokenClassification`, який забезпечує правильне формування батчів (зібрань тренувальних даних). [49]

4.1.3 Аргументи для тренування моделі

Для налаштування моделі використовуються два основні набори аргументів: базові параметри моделі та параметри навчання.

При ініціалізації базової моделі AutoModelForTokenClassification задаються наступні ключові параметри:

- model_name = "youscan/ukr-roberta-base" – наприклад, базова українська модель RoBERTa;
- num_labels = 3 – кількість класів для класифікації токенів, що відповідає схемі розмітки BIO (`NONE, B-EXIST, I-EXIST`);
- id2label та label2id - словники відповідності між числовими індексами та текстовими мітками класів.

Параметри навчання моделі налаштовуються через клас TrainingArguments, який включає такі стандартні параметри:

- output_dir – шлях для збереження результатів навчання та контрольних точок моделі;
- num_train_epochs = 10 – кількість епох навчання;
- evaluation_strategy = "epoch" – стратегія проведення оцінки моделі після кожної епохи;
- per_device_train_batch_size = 8 – розмір батчу для навчання;
- per_device_eval_batch_size = 8 – розмір батчу для валідації;
- learning_rate = 2E-5 – швидкість навчання;
- weight_decay = 0.01 – коефіцієнт регуляризації для запобігання перенавчання;

4.1.4 Особливості розрахунку критеріїв оцінки моделі

Для оцінки якості роботи моделі виявлення ознак сну використовується комплексний підхід з кількома метриками, реалізований через функцію compute_metrics.

Для обчислення метрик використовується бібліотеки scikit-learn, спеціалізовані для оцінки моделей. Процес оцінки включає наступні етапи:

- перетворення вихідних логітів моделі у ймовірності через функцію softmax;
- визначення передбачених міток на основі максимальних ймовірностей;
- фільтрація службових токенів та токенів паддінгу (позначених як -100);
- обчислення метрик з використанням бібліотеки scikit-learn.

Логіт – сирі, ненормалізовані передбачення від моделі, які генеруються нею перед використанням функції активації.[50]

Функція активації, що використовується усіма моделями окрім LLM – softmax, як основна функція, що використовується для випадку мультикласової класифікації: [51][52]

$$Softmax = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}} \quad (4.1)$$

де: z_i – вхід для і-го з K класів функції softmax.

Для кожної метрики проведено її усереднення, так як у контексті цієї роботи нас цікавить середня точність на датасеті, ніж точність на кожному класі

Так для випадку мультикласифікації рекомендовано в функціях розрахунку метрик бібліотеки scikit-learn встановити значення `weighted` параметру `average`. Це означає, що для кожної метрики буде розраховуватись взвішане середнє між кожним класом, так як кількість токенів різних класів може відрізнятись за природи самих класів.

Для розрахунку функції ROC AUC Score використовується стратегія один проти одного, як стандартна практика для мультикласифікаці. Крім того, саме для цієї метрики передаються не самі класи, а розраховані методом softmax вірогідності, що дозволяє визначити значення під кривою ROC AUC незалежно від порогу відсікання. [53]

4.1.5 Опис експерименту

Для моделей цього класу експеримент проводиться шляхом навчання моделі на тренувальних даних і оцінки на тестових.

Результат експерименту – набір метрик на момент останньої ітерації.

4.2 Класифікація ознак сну за допомогою моделей сімейства BERT

Для опису використано приклад роботи з моделю youscan/ukr-roberta-base.

Основний код потрібний для проведення експериментів над моделями класифікації наведено в додатку Д.

Опис експерименту проводиться по аналогії з моделями виявлення.

4.2.1 Підготовка датасету

Для задачі класифікації ознак сну датасет формується з текстових фрагментів, що містять онейрологічні образи, та відповідних їм міток класів. На відміну від завдання виявлення ознак, де використовується схема розмітки

BIO на рівні окремих токенів, для класифікації застосовується multilabel підхід, де кожному текстовому фрагменту може відповідати декілька класів одночасно.

Процес підготовки датасету починається з завантаження вихідних даних з JSON файлів, які містять записи снів та метадані.

Для зручності роботи з класами створюються словники відповідності між текстовими назвами класів та їх числовими ідентифікаторами. Спочатку формується словник markers_dict, що зв'язує повні назви класів (включаючи категорію) з їх ідентифікаторами з метаданих. Далі створюється маппінг marker_id_to_int, який перетворює оригінальні ідентифікатори в послідовні цілі числа від 0 до кількості класів мінус один. Також формуються допоміжні словники id2label та label2id для двостороннього перетворення між числовими індексами та текстовими назвами класів.

Основна обробка даних відбувається у функції `process_dreams`, яка витягує з кожного запису сну окремі ознаки разом з їх маркерами класів. Для кожної ознаки зберігається її текст, перелік класів у вигляді числових індексів, ідентифікатор сну та позиції початку і кінця ознаки в тексті. Результати зберігаються у DataFrame бібліотеки Polars для подальшої обробки (див. лістінг 4.3).

```
def process_dreams(dreams):
  processed_dreams = []
  for dream in dreams:
      dream_moments = dream['moments']
      for moment in dream_moments:
          moment_text = moment['text']
          moment_markers    =    [marker_id_to_int[marker_id]    for
marker_id in moment['markers']]
          processed_dreams.append({
              'text': moment_text,
              'labels': moment_markers,
              'dream_id': dream['id'],
              'start': moment['index'],
              'end': moment['index'] + moment['length']
          })
  df = pl.DataFrame(processed_dreams)
  return df
```

Лістінг 4.3 – Функція для конвертації масиву снів у масив ознак сну

Наступним важливим етапом є перетворення міток класів у формат one-hot encoding.

Після чого підготовлений датасет зберігається у спеціальному форматі бібліотеки Hugging Face datasets. При цьому, по аналогії з моделями виявлення, зберігається інформація про структуру даних, включаючи опис формату міток як списку класів з фіксованим набором можливих значень (міток класів).

4.2.2 Підготовка даних

Для підготовки даних до навчання моделі використовується, по аналогії з моделю виявлення, спеціалізований токенізатор, наприклад, `RobertaTokenizer`.

Процес токенізації реалізовано через функцію `tokenize` (дивитись лістінг 4.4), яка приймає текстові приклади ознак сну та виконує наступні операції:

- розбиває текст на токени відповідно до словника моделі;
- обрізає послідовності до максимальної довжини MAX_LENGTH = 256 токенів;
- доповнює короткі послідовності спеціальними токенами паддінгу до максимальної довжини;
- створює маску уваги (attention mask), яка позначає реальні токени одиницями, а токени паддінгу нулями.

```
def tokenize(examples, tokenizer):
    return tokenizer(examples['text'],
                     truncation=True,
                     padding='max_length',
                     max_length=MAX_LENGTH,
                     add_special_tokens=False)
```

Лістінг 4.4 – Функція для токенізації ознак сну

Далі для обробки всього датасету використовується метод map бібліотеки datasets, який застосовує функцію токенізації до всіх прикладів в батчевому режимі. Результатом є токенізований датасет, що містить наступні поля:

- input_ids – числові індекси токенів;
- attention_mask – маска уваги;
- labels – вектори міток класів у форматі one-hot encoding;

Для забезпечення коректної роботи з мітками класів проводиться додаткове перетворення типів даних. Мітки приводяться до типу float32 за допомогою функції `cast_column`. Це необхідно для правильного обчислення функції втрат при навчанні моделі.

Після токенізації з датасету видаляються надлишкові поля, такі як вихідний текст та метадані про позиції ознак у тексті, оскільки вони не потрібні для навчання моделі. Залишаються тільки поля з токенізованими даними та мітками класів.

Далі датасет розділяється на навчальну та валідаційну частини у співвідношенні 80% на 20% за допомогою функції random_split з бібліотеки PyTorch, по аналогією з моделями виявлення.

4.2.3 Аргументи для тренування моделі

Аргументи для тренування майже не відрізняються від моделі виявлення окрім значень batch_size та eval_batch_size, що дорівнюють 16 та 64 відповідно. Збільшення розміру батчів для моделі класифікації обумовлено меншою кількістю даних для обробки, оскільки модель працює з уже виділеними ознаками, а не з повними текстами снів.

Всі інші параметри залишено без змін для перевірки роботи моделей в загальному випадку та забезпечення справедливого порівняння з результатами моделі виявлення.

4.2.4 Особливості розрахунку критеріїв оцінки моделі

Для оцінки моделі використовуються загальні метрики, по аналогії з моделями виявлення ознак. Однак, на відміну від моделі виявлення ознак сну для моделі класифікації використовується multilabel класифікація, тому для моделі додатково потрібно визначити ще і гіперплощину, що відділяє більшість FP випадків, за допомогою порогу відсікання.

Найкращий поріг визначається максимізацією середнього гармонічного, яке доречно використовувати саме для метрик у протилежність середньому арифметичному чи геометричному, усіх метрик, включаючи (1 – Hamming Loss). [54] [55]

$$HM = \frac{n}{\sum_{i=1}^{n}\left(\frac{1}{x_i}\right)} \tag{4.2}$$

де n – кількість снів, для яких розраховується метрика;

x_i – значення метрики для і-го сну.

4.3 Виявлення та класифікація ознак сну за допомогою LLM

Для цього розділу використано приклад з LLM від Open AI – GPT 4o mini.

Основний код потрібний для проведення експериментів над моделями класифікації наведено в додатку Е.

4.3.1 Підготовка датасету

Для навчання великої мовної моделі потрібно підготувати датасет у спеціальному форматі, який відрізняється від форматів для моделей сімейства BERT. Процес створення такого датасету складається з декількох етапів.

Першим етапом є завантаження та попередня обробка вихідних даних. З JSON файлів завантажуються два типи даних: безпосередньо записи снів (dreams.json) та метадані (metadata.json), що містять інформацію про типи ознак та їх категорії.

На другому етапі відбувається створення словників відповідності між різними представленнями даних. Створюється словник `marker_id2name`, який зіставляє числові ідентифікатори типів ознак з їх текстовими назвами, включаючи категорію. Наприклад, ознака "Відчуття" з категорії "Внутрішня обізнаність" матиме повну назву "Відчуття (Внутрішня обізнаність)". Також формується розширений словник `markers_dict`, який додатково включає описи кожного типу ознаки.

Третій етап включає трансформацію записів снів у формат, придатний для обробки мовною моделлю. Кожен запис сну перетворюється в структуру, що містить:

- Повний текст сну
- Масив виділених моментів, де кожен момент характеризується:
  - Текстом конкретної ознаки
  - Булевим прапорцем можливості класифікації
  - Списком ідентифікаторів застосовних типів ознак
  - Позиціями початку та кінця ознаки в тексті

Четвертий етап передбачає створення окремого набору даних для типів ознак. Використовуючи бібліотеку Polars, формується DataFrame, що містить для кожного типу ознаки:

- повну назву типу ознаки з категорією;
- унікальний числовий ідентифікатор;
- детальний опис типу ознаки.

Фінальним етапом є збереження підготовлених даних у файлову систему. Створюється окрема директорія для даних LLM моделі, де зберігаються:

- датасет з типами ознак у форматі Parquet для ефективного зберігання та швидкого завантаження;
- трансформований датасет зі снами у форматі JSON, який зберігає всю необхідну структуру даних;

Такий формат даних дозволяє ефективно використовувати їх для навчання великої мовної моделі, забезпечуючи всю необхідну інформацію як про самі сни, так і про систему класифікації ознак. При цьому зберігається можливість розширення датасету новими записами та модифікації системи типів ознак без необхідності змінювати структуру даних.

4.3.2 Підготовка даних

Процес підготовки даних для великої мовної моделі включає декілька етапів перетворення та форматування вхідних даних для забезпечення коректної роботи моделі.

Першим етапом є завантаження підготовленого датасету з файлової системи, описаного в попередньому розділі.

Наступним етапом є створення словників відповідності між різними представленнями даних для забезпечення узгодженої роботи з типами ознак:

- словник labels_dict, що зіставляє порядкові номери (від 0 до 17) з описами типів ознак;
- словник id2label для перетворення оригінальних ідентифікаторів типів у послідовні числа.

Третім етапом є трансформація даних у формат, придатний для обробки мовною моделлю. Для кожного запису в датасеті:

- копіюється текст сну
- для кожного моменту в сні:
  - зберігається текст ознаки;
  - конвертуються ідентифікатори типів ознак у послідовні числа;
  - зберігаються позиції початку та кінця фрагменту в тексті.

4.3.3 Аргументи для тренування моделі

Для тренування моделі використовується конфігураційний файл бібліотеки spaCy для великих мовних моделей (див. лістінг 4.5)

```
[nlp]
lang = "uk"
pipeline = ["llm"]

[components]

[components.llm]
factory = "llm"

[components.llm.task]
@llm_tasks = "DreamMomentCatTask.v1"

[components.llm.model]
@llm_models = "spacy.GPT-4.v3"
name = "gpt-4o-mini"
config = {"temperature": 0.0}
```

Лістінг 4.5 – Файл конфігурації великої мовної моделі GPT-4o-mini

Як зазначено в файлі конфігурації, створюється конвеєр під українську мову з компонентом для LLM. Йому призначається користувацьке завдання `"DreamMomentCatTask.v1"` і модель, якою виступає, наприклад GPT-4o-mini. Моделі також надається конфігурація з параметром, що відповідає за креативність моделі.

Під креативністю моделі мається на увазі спроможність моделі обирати менш вірогідні токени. Так модель з креативністю 0 буде обирати токени абсолютно детерміновано, тоді як – з креативністю 1 буде обирати їх навмання. [56]

Щоб звернутись до моделі за допомогою бібліотеки spaCy потрібно зібрати конвеєр за допомогою конфігураційного файлу та перевантажень деяких об’єктів конфігурації, такими як `labels`, `labels_definitions` та `prompt_examples`, що дозволяє модифікувати конфігураційний файл під конкретний експеремент та збільшити чи зменшити кількість токенів у

промпті. Це допомагає перевірити точність моделі при різних експериментах від zero-shot до few-shot навчання і також регулює споживання ресурсів API, яке не є безкоштовним.

4.3.4 Особливості розрахунку критеріїв оцінки моделі

На відміну від моделей сімейства BERT для LLM не потрібно вказувати функцію активації, бо текст оброблюється за допомогою засобів бібліотеки spacy. Тому лише потрібно вибрати необхідні дані для ознаки сну з результуючого об’єкту, а саме текст ознаки та її класи.

Проте для оцінки отриманих результатів привести ці дані у формат, який сприймають функції метрик бібліотеки scikit-learn, тому їх оброблено по аналогії з даними на етапі підготовки датасету моделей виявлення ознак:

1) токенізовано текст ознаки та сну до рівня слів та знаків пунктуації за допомогою методу `tokenize`;

2) вектор класів ознак переведено у формат one-hot encoding;

3) токенізований текст сну співставлено його токенізованим ознакам і кожному токену тексту призначено відповідні форматовані вектори класів ознак.

Однак на відміну від даних для моделей сімейства BERT довжини векторів текстів сну не є рівними за довжиною, тому методи scikit-learn не можуть розрахувати метрики для різних снів, з-за цього вирішено усереднити значення метрик від кожного сну за допомогою середнього гармонійного.

Треба також зауважити, що розрахунок метрики ROC AUC немає сенсу для цього класу моделей з-за відсутності явного розподілу вірогідностей для кожного токену.

Далі, метрика F_1 згідно визначенню є середнім гармонійним між Precision та Recall, тому для цього класу моделей доречно також буде

визначати окрім середнього F_1, ще і середнє F_1 на основі усереднених Precision та Recall.

Таким чином для великих мовних моделей використовуються усередненені метрики на їх наборах даних.

Параметри функції усерднення для кожної метрики залишаються незмінними.

Для оцінки моделей використано 10 випадкових снів із датасету.

4.3.5 Опис експерименту

Для LLM, що кардинально відрізняються від моделів сімейства BERT, заплановано інший план. Так спочатку буде використано методику zero-shot навчання при різних значеннях параметру креативності, що визначає кількість шуму накладеного на вихідні вірогідності вибору токенів моделю, що може спричинити досить заплутану

Цей параметр буде перевірено на проміжку від 0 до 0.3 (найбільша креативність рекомендована для аналітичних завдань) з кроком 0.1 на 10 випадкових снах. [57]

Таким чином буде визначено креативність, за якої моделі будуть показувати найкращі результати.

Далі буде проведено ще три експерименти на 10 попередньо обраних випадкових снах:

1) без навчальних даних за методикою zero-shot навчання;

2) з навчальними даними, що складаються з 10 випадково обраних снів;

3) з навчальними даними, що складаються з 10 снів, які повністю покривають класи ознак сну.

5 РЕЗУЛЬТАТИ ЕКСПЕРЕМЕНТІВ

5.1 Розділене виявлення та класифікація

Експеримент виявлення та класифікації проведено на наступних моделях середовища HuggingFace, що підтримують українську мову. Кожній моделі середовища відповідає назва базової моделі, яку буде використано у якості відмітки в результатах:

- RoBERTa – youscan/ukr-roberta-base;
- BERT – google-bert/bert-multilangual-cased;
- DistilBERT – distilbert/distilbert-base-multilingual-cased.

5.1.1 Результати моделей сімейства BERT на етапі виявлення ознак сну

Результати перевірки моделей сімейства BERT для завдання виявлення онейрологічних образів наведено у таблиці 5.1, де видно, що більшість найкращих метрик належить моделі DistilBERT. Модель продемонструвала найвищу точність (accuracy) на рівні 86.84%, найкращу точність передбачення (precision) – 86.06%, найвищий показник повноти (recall) – 86.84% та, відповідно, найкращий F1-score – 86.31%. Також модель показала найнижчий показник Hamming Loss - 13.15%, що свідчить про найменшу кількість помилкових передбачень серед усіх протестованих моделей. Єдиним показником, за яким DistilBERT поступається іншим моделям, є ROC AUC Score (90.15% проти 90.65% у BERT та 90.59% у RoBERTa), що вказує на трохи меншу спроможність моделі правильно розрізняти істинно позитивні випадки від хибно позитивних.

У таблицях результатів напівжирним шрифтом відмічено найкращі метрики.

Таблиця 5.1. Порівняльна таблиця результатів по виявленню ознак сну моделями сімейства BERT у %

Назва моделі	Accuracy	Precision	Recall	F1	Hamming Loss	ROC AUC Score
RoBERTa	85.76	85.09	85.76	85.33	14.24	90.59
BERT	85.28	84.59	85.28	84.86	14.72	**90.65**
DistilBERT	**86.84**	**86.06**	**86.84**	**86.31**	**13.15**	90.15

Незважаючи на доволі гарні результати виявлення онейрологічних образів у тексті модель іноді неправильно визначала їх межі (див. лістінг 5.1)

```
[…,B-EXISTS, NONE, I-EXISTS, … ]
[…,I-EXISTS, NONE, I-EXISTS, … ]
```

Лістінг 5.1 – Приклади неправильного визначення послідовностей токенів

Для уникнення таких ситуацій результуючі послідовності моделі мають перевірятись на подібні помилки.

Отже, за результатами модель DistilBERT показує найкращі результати у мінімізації помилок першого і другого роду.

5.1.2 Результати моделей сімейства BERT на етапі класифікації виявлених ознак сну

Результати перевірки моделей на завданні класифікації, наведено у таблиці 5.2. Згідно цих результатів модель RoBERTa виявилась найкращою для цього завдання. Окрім цього помітно, що порогове значення різниться від моделі до моделі.

Таблиця 5.2. Порівняльна таблиця результатів по класифікації ознак сну моделями сімейства BERT у %

Назва моделі	Accuracy	Precision	Recall	F1	Hamming Loss	ROC AUC Score	Порогове значення
RoBERTa	**35.87**	**55.20**	**53.17**	**54.00**	**6.60**	**82.18**	**0.15**
BERT	29.18	41.00	40.37	38.69	6.64	74.76	0.20
DistilBERT	26.75	31.35	43.98	40.94	7.90	76.81	0.15

5.1.3 Оцінка можливих варіантів комплексів моделей сімейства BERT для виявлення та класифікації ознак сну

Для комплексної оцінки ефективності різних комбінацій моделей сімейства BERT необхідно розглянути всі можливі пари моделей виявлення та класифікації.

Для оцінки конвеєра з двома роздільними моделями взято гармонічне середнє між двома метриками, на перетині між двома моделями.

Основними характеристиками для такої оцінки визначено Accuracy, F1, Hamming Loss та ROC AUC Score.

Таблиця 5.3. Таблиця оцінки гіпотетичного конвеєру між двома моделями виявлення та класифікації ознак сну моделями сімейства BERT на основі метрики Accuracy у %

Назва моделі виявлення	Назва моделі класифікації		
	RoBERTa	BERT	DistilBERT
RoBERTa	50,58	43,54	40,78
BERT	50,50	43,48	40,73
DistilBERT	**50,77**	43,68	40,90

Таблиця 5.5. Таблиця оцінки гіпотетичного конвеєру між двома моделями виявлення та класифікації ознак сну моделями сімейства BERT на основі метрики F1 у %.

Назва моделі виявлення	Назва моделі класифікації		
	RoBERTa	BERT	DistilBERT
RoBERTa	66,14	53,24	55,33
BERT	66,00	53,15	55,23
DistilBERT	**66,31**	53,35	55,45

Таблиця 5.6. Таблиця оцінки гіпотетичного конвеєру між двома моделями виявлення та класифікації ознак сну моделями сімейства BERT на основі метрики Hamming Loss у %.

Назва моделі виявлення	Назва моделі класифікації		
	RoBERTa	BERT	DistilBERT
RoBERTa	9,02	9,06	10,16
BERT	9,11	9,15	10,28
DistilBERT	**8,79**	8,82	9,87

Таблиця 5.7. Таблиця оцінки гіпотетичного конвеєру між двома моделями виявлення та класифікації ознак сну моделями сімейства BERT на основі метрики ROC AUC Score у %

Назва моделі виявлення	Назва моделі класифікації		
	RoBERTa	BERT	DistilBERT
RoBERTa	**86,18**	81,92	83,13
BERT	**86,21**	81,94	83,16
DistilBERT	85,98	81,74	82,95

Отже, за збалансованою метрикою кількості помилок першого та другого і 3-ма додатковими метриками комплексна модель виявлення та класифікації DistilBERT-RoBERTA має виявитись найліпшою.

5.2 Повноцінна модель

Так як у роботі використовуються великі мовні моделі, які, по-перше, неможливо навіть запустити при доступних обчислювальних ресурсах. По-друге, вони доступні лише через пропрієтарний API, саме тому доводиться оцінювати споживання ресурсів.

Так, якщо провести 10 експериментів для zero-shot навчання по 10 снів кожен, що буде відповідати 100 запитам до API по 2000 токенів на вході (довжина промпту без прикладів) і 250 токенів на виході (довжина середньої відповіді від LLM), тоді ціна використання буде відповідати наведеній на рис. 5.1.

Calculate by: Tokens | Words | Characters

Input tokens: 2000

Output tokens: 250

Number of API calls: 100

Provider	Model	Input price for 1M tokens	Output price for 1M tokens	Price per API call	Total price
OpenAI	gpt-4o	$5.00	$15.00	$0.0138	**$1.38**
OpenAI	gpt-4o-mini	$0.15	$0.60	$0.0004	**$0.04**
OpenAI	gpt-4-turbo	$10.00	$30.00	$0.0275	**$2.75**
OpenAI	gpt-4	$30.00	$60.00	$0.0750	**$7.50**
OpenAI	gpt-3.5-turbo	$0.50	$1.50	$0.0014	**$0.14**
Anthropic	claude-3.5-sonnet	$3.00	$15.00	$0.0097	**$0.97**
Anthropic	claude-3-haiku	$0.25	$1.25	$0.0008	**$0.08**
Google	gemini-1.5-flash	$0.07	$0.30	$0.0002	**$0.02**
Google	gemini-1.5-pro	$3.50	$10.50	$0.0096	**$0.96**
Mistral	mistral-large	$3.00	$9.00	$0.0083	**$0.83**

Рисунок 5.1 – Ціна використання API для великих мовних моделей при zero-shot навчанні[58]

Наступним чином будуть йти 2 експерименти, по 10 снів на експеримент. Проте тренувальні дані будуть відрізнятись. Для першого експерименту це 10 снів для першого і 20 для другого. Для зручності розраховуємо, що робиться 20 запитів по 8000 токенів на вході і 250 токенів на виході. В цьому випадку ціни на використання API будуть відповідати рисунку 5.2.

Provider	Model	Input price for 1M tokens	Output price for 1M tokens	Price per API call	Total price
OpenAI	gpt-4o	$5.00	$15.00	$0.0437	**$0.88**
OpenAI	gpt-4o-mini	$0.15	$0.60	$0.0013	**$0.03**
OpenAI	gpt-4-turbo	$10.00	$30.00	$0.0875	**$1.75**
OpenAI	gpt-4	$30.00	$60.00	$0.2550	**$5.10**
OpenAI	gpt-3.5-turbo	$0.50	$1.50	$0.0044	**$0.09**
Anthropic	claude-3.5-sonnet	$3.00	$15.00	$0.0278	**$0.56**
Anthropic	claude-3-haiku	$0.25	$1.25	$0.0023	**$0.05**
Google	gemini-1.5-flash	$0.07	$0.30	$0.0007	**$0.01**
Google	gemini-1.5-pro	$3.50	$10.50	$0.0306	**$0.61**
Mistral	mistral-large	$3.00	$9.00	$0.0262	**$0.53**

Рисунок 5.2 – Ціна використання API для LLM при few-shot навчанні

Однак, використовуючи моделі від різних компаній, потрібно придбати дозвіл використання API в кожної. Таким чином розраховано використання API для найкращих моделей від кожної компанії (Claude 3.5 Sonnet, GPT-4o та Gemini 1.5-Pro без GPT o1, як відсутньої в відкритому доступі): [59]

- Anthropic – 1.53 USD або 63 UAH;
- OpenAI – 2.26 USD або 93 UAH;
- Google –1.57 USD або 65 UAH.

Для міні-моделей (GPT-4o-mini, Gemini 1.5-Flash) ціни будуть на порядки менші:

- Anthropic – 0.13 USD або 5.35 UAH;
- OpenAI – 0.07 USD або 2.88 UAH;
- Google –0.03 USD або 1.24 UAH.

На відміну від моделей розділеної класифікації неможливо точно визначити якість виявлення ознаки без приведення до одного масштабу у тексті (рівень абзаців, речень, слів, токенів тощо). Таким чином надані моделлю класифікації ознаки приводяться до визначення класифікації слів частини тексту, що примножує кількість помилок на кількість слів. Так неправильне визначення двох класів на ознаці сну, складеної з 5 слів призведе до 10 помилок.

5.2.1 Результати визначення параметру креативності

Для визначення найкращого значення параметру креативності використано лише модель GPT-4o-mini, якої достатньо для визначення цього параметру.

Результати наведено у таблиці 5.6, де найкращим виявилось значення 0, при якому значення F1 середнього та середнє F1 виявилось найбільшим.

Крім того таке значення параметру креативності зробить подальші експерименти більш відтворюваними через відсутність випадкового фактору на результат.

Таблиця 5.8. Порівняльна таблиця результатів у % по виявленню та класифікації ознак сну моделлю GPT-4o-mini при різних рівнях креативності

Креативність	Accuracy	Precision	Recall	F1 Середнього	Середнє F1	Hamming Loss
0.00	19.70	**50.66**	15.67	**34.08**	**32.02**	6.35
0.10	17.99	46.09	**24.56**	32.04	30.28	6.53
0.20	**20.44**	47.07	17.34	25.34	24.19	6.03
0.30	18.52	45.97	24.18	32.54	30.80	**6.00**

5.2.2 Результати моделі GPT-4o-mini

У таблиці 5.8 наведено результати проведення експериментів, при чому зрозуміло, що найкращі результати модель GPT-4o-mini показала на 10 снах з випадкового датасету з середнім часом відповіді в 27.2 секунди. Зокрема, модель продемонструвала найкращу точність передбачення (precision) – 28.19%, і досить високий показник повноти (recall) – 32.13%, разом з найнижчим показником Hamming Loss (9.00.

Таблиця 5.8. Порівняльна таблиця результатів у % по виявленню та класифікації ознак сну моделлю GPT-4o-mini для трьох експериментів

Назва експерименту	Accuracy	Precision	Recall	F1 Середнього	Середнє F1	Hamming Loss
Zero-shot	12.83	5.99	18.59	9.06	8.97	9.78
Випадковий	**17.00**	**28.19**	**32.13**	**30.03**	**28.58**	**9.00**
З повними прикладами	13.62	3.05	15.72	5.11	5.06	10.29

5.2.3 Результати моделі GPT-4o

За результатами експериментів у таблиці 5.9, модель GPT-4o виявилась лише трохи кращою за свою mini версію і найкращі результати показала на випадковому датасеті з середнім часом відповіді в 50.3 секунди. Незважаючи на значно більший розмір моделі та майже вдвічі довший час обробки, покращення показників виявилось незначним.

Таблиця 5.9. Порівняльна таблиця результатів у % по виявленню та класифікації ознак сну моделлю GPT-4o для трьох експериментів

Назва експерименту	Accuracy	Precision	Recall	F1 Середнього	Середнє F1	Hamming Loss
Zero-shot	14.21	27.99	30.87	29.36	28.60	9.24
Випадковий	**14.49**	**18.49**	26.70	21.85	21.63	**8.68**
З повними прикладами	13.93	17.93	**31.12**	**22.75**	**22.14**	9.61

5.2.4 Результати моделі Claude 3.5 Haiku

Згідно з таблицею 5.10, mini модель від Antropic – Claude 3.5 Haiku, показала себе краще ніж велика модель від OpenAI на випадковому наборі снів з середнім часом відповіді в 36 секунд.

Таблиця 5.10. Порівняльна таблиця результатів у % по виявленню та класифікації ознак сну моделлю Claude 3.5 Haiku для трьох експериментів

Назва експерименту	Accuracy	Precision	Recall	F1 Середнього	Середнє F1	Hamming Loss
Zero-shot	14.92	13.37	21.02	16.34	16.13	11.99
Випадковий	14.18	26.24	**39.58**	**31.56**	**30.60**	**8.31**
З повними прикладами	**14.56**	**26.42**	37.37	30.95	29.67	8.71

5.2.5 Результати моделі Claude 3.5 Sonnet

Аналіз результатів роботи моделі Claude 3.5 Sonnet, представлених у таблиці 5.11, демонструє її перевагу над молодшою версією Claude 3.5 Haiku за більшістю ключових метрик. Найкращі показники модель продемонструвала при тестуванні на наборі з 10 випадково обраних снів, що забезпечило репрезентативну вибірку для оцінки її можливостей.

Модель також продемонструвала кращий баланс між точністю та повнотою, що відображається у показнику F1-score на рівні 32.61%, що перевищує результат Haiku (31.56%).

Такі результати підтверджують, що модель Claude 3.5 Sonnet показала себе краще ніж Haiku, з найкращими результатами на 10 випадкових снах з середнім часом відповіді в 38 секунд.

Таблиця 5.11. Порівняльна таблиця результатів у % по виявленню та класифікації ознак сну моделлю Claude 3.5 Sonnet для трьох експериментів.

Назва експерименту	Accuracy	Precision	Recall	F1 Середнього	Середнє F1	Hamming Loss
Zero-shot	10.87	18.39	27.26	21.95	21.79	11.23
Випадковий	**23.61**	**27.83**	**39.39**	**32.61**	**31.32**	**8.77**
З повними прикладами	15.16	22.10	35.63	27.28	26.87	8.96

5.2.6 Найкращі результати великих мовних моделей

За таблицею 5.12, модель Claude 3.5 Sonnet є кращою серед LLM для виконання поставленого завдання моделей за поточних метрик.

Таблиця 5.12. Порівняльна таблиця результатів у % по виявленню та класифікації ознак сну найкращими LLM моделями.

Назва моделі	Accuracy	Precision	Recall	F1 Середнього	Середнє F1	Hamming Loss
GPT-4o-mini	17.00	**28.19**	32.13	30.03	28.58	9.00
GPT-4o	14.49	18.49	26.70	21.85	21.63	8.68
Claude 3.5 Haiku	14.18	26.24	**39.58**	31.56	30.60	**8.31**
Claude 3.5 Sonnet	**23.61**	27.83	39.39	**32.61**	**31.32**	8.77

5.3 Порівняння результатів найкращих моделей з двох різних рішень

Останнім кроком для порівняння різних моделей вирішено провести експеримент над композицією моделей виявлення та класифікації ознак сну DistilBERT-RoBERTa за методикою повноцінної моделі з середнім часом обробки в 1.2 секунди.

Результати такого експерименту у порівнянні з найкращою LLM моделлю наведено у таблиці 5.13.

Таблиця 5.13. Таблиця результатів у % по виявленню та класифікації ознак сну найкращими моделями різних класів.

Назва моделі	Accuracy	Precision	Recall	F1 Середнього	Середнє F1	Hamming Loss
DistilBERT-RoBERTa	**27.86**	**72.09**	**80.44**	**76.04**	**74.86**	**3.22**
Claude 3.5 Sonnet	23.61	27.83	39.39	32.61	31.32	8.77

Таким чином комплексна модель DistilBERT-RoBERTa у контексті задачі виявлення та класифікації онейрологічних образів значно превалює над LLM моделями за всіма ключовими метриками. Особливо помітна різниця у показниках повноти (recall) - 80.44% проти 39.39% у найкращої LLM моделі Claude 3.5 Sonnet, що вказує на суттєво кращу здатність виявляти релевантні ознаки в текстах снів.

6 ОПИТУВАННЯ

6.1 Обґрунтування необхідності проведення опитування

При розробці систем машинного навчання для обробки природної мови важливо враховувати не лише кількісні метрики ефективності моделей, але й якісні характеристики їх роботи з точки зору кінцевих користувачів. У контексті даного дослідження виникла необхідність вирішення фундаментального питання: чи слід надавати перевагу моделі, яка демонструє найкращі результати на конкретному наборі даних, але може бути менш гнучкою при зміні контексту, або обрати більш універсальну модель, здатну до кращої генералізації, незважаючи на дещо нижчі показники на тестовому датасеті.

Композиція моделей DistilBERT-RoBERTa продемонструвала значно кращі кількісні показники на тестовому наборі даних порівняно з моделлю Claude 3.5 Sonnet. Проте виникає питання, чи ці метрики повністю відображають практичну цінність моделей для користувачів. Велика мовна модель, хоч і показала нижчі результати за стандартними метриками, може мати певні переваги завдяки своїй здатності краще розуміти контекст та працювати з різноманітними формулюваннями.

Саме тому було прийнято рішення провести опитування серед потенційних користувачів системи, щоб:

1) визначити, які характеристики роботи моделей є найбільш важливими з точки зору практичного застосування;

2) оцінити суб'єктивне сприйняття якості роботи різних моделей;

3) виявити можливі переваги та недоліки кожного підходу, які могли бути не враховані при кількісному оцінюванні. [10]

6.2 Підготовка даних для опитування

6.2.1 Дані для опитування частини виявлення

Весь перелік даних для опитування наведено в додатку Ж.

Для проведення якісного опитування було підготовлено спеціальний набір даних, що складається з 10 унікальних снів. Кожен сон був оброблений двома моделями: компонентою комплексної моделі DistilBERT та Claude 3.5 Sonnet. Загальна кількість виявлених онейрологічних образів склала 60 одиниць.

При аналізі результатів роботи моделей виявлено два типи ситуацій (див. табл Ж.1 та Ж.2) :

1) повна узгодженість – випадки, коли обидві моделі однаково визначили межі онейрологічного образу (29 випадків, що складає 48.3% від загального числа);

2) розбіжність у визначенні – випадки, коли моделі по-різному визначили межі образу (31 випадок, що складає 51.7% від загального числа).

Для проведення опитування було вирішено зосередитись саме на випадках розбіжності, оскільки вони представляють найбільший інтерес для дослідження та дозволяють краще зрозуміти особливості роботи кожної моделі. В результаті було сформовано 19 питань, що відповідають ситуаціям, де моделі демонструють різні підходи до виділення онейрологічних образів.

6.2.2 Дані для опитування частини класифікації

Окрім оцінки якості виявлення онейрологічних образів, важливою частиною дослідження стало порівняння підходів до їх класифікації. Для цього було відібрано 24 ознаки з перших п'яти снів, виявлених моделлю Claude 3.5 Sonnet. Такий підхід був обраний через неможливість використання

ідентичних образів, виявлених моделлю DistilBERT, що могло б ускладнити порівняльний аналіз.

Відібрані ознаки класифіковані двома моделями: RoBERTa (представник спеціалізованого підходу) та Claude 3.5 Sonnet (представник універсального підходу).

Аналіз результатів класифікації виявив суттєві відмінності у підходах моделей. Claude 3.5 Sonnet частіше визначає емоційні та когнітивні аспекти ("Емоції (Внутрішня обізнаність)", "Думки (Внутрішня обізнаність)"). RoBERTa більше фокусується на фізичних діях та просторовому розташуванні ("Мої (Дії)", "Місцезнаходження об'єктів");

Моделі по-різному інтерпретують складні ситуації, де присутні одночасно декілька аспектів (наприклад, фізичні дії з емоційним забарвленням).

З метою оптимізації процесу опитування та уникнення втоми респондентів вирішено обмежитись аналізом снів з найбільш показовими прикладами розбіжностей у класифікації. На основі виявлених розбіжностей сформовано 24 питання, що дозволяють оцінити, який підхід до класифікації є більш природнім та зрозумілим для користувачів.

6.3 Структура опитування

6.3.1 Демографічний блок

Демографічний блок опитування є важливим інструментом для розуміння складу респондентів та потенційного впливу їхніх характеристик на оцінку роботи моделей. Збір такої інформації дозволяє не тільки описати вибірку учасників, але й виявити можливі закономірності у сприйнятті різних аспектів роботи моделей різними демографічними групами.

Перший блок опитування спрямований на збір базової демографічної інформації про респондентів, що дозволить проаналізувати можливі

відмінності у сприйнятті роботи моделей різними групами користувачів. Блок містить два обов'язкових питання.

Перше питання виявляє стать респондента з такими варіантами відповідей:

- чоловік;
- жінка;
- не можу сказати;
- інше.

Друге питання відносить респондента до вікової групи з діапазонами:

- 11-20 років;
- 21-30 років;
- 31-40 років;
- 41-50 років;
- 51-60 років;
- 61-70+ років;

6.3.2 Блок оцінки виявлення ознак

Другий блок присвячений оцінці якості виділення онейрологічних образів у текстах снів. На початку блоку респондентам надається визначення ознаки сну як "достатньо яскравого образу, на який людина може звернути увагу уві сні" та пояснення, що все незвичне або абсурдне може виступати ознакою сну.

Блок містить 19 питань, сформованих на основі випадків розбіжності між моделями у визначенні меж ознак. Всього існують питання трьох типів.

Так перший тип стосується оцінки коректності виділення окремих фрагментів, у випадку коли одна модель виділила ознаку, яку не виділила інша модель.

Формулювання: "Чи виділили би ви у якості ознаки фразу '...' ? "

Варіанти відповідей: Так чи Ні

Другий тип питань стосується ситуацій, коли одна з моделей об’єднала ознаки, розділені другою моделлю. При цьому додатково вказуються ті ознаки, які можливо об’єднати.

Формулювання: "Чи об'єднали би ви ці ознаки в одну ? "

Варіанти відповідей: так чи ні

Третій тип відповідає ситуації, при якій моделі по різному виділили одну і ту саму ознаку.

Формулювання: "Який варіант ознаки вам подобається більше ? "

Варіанти відповідей: перший або другий або жодний

6.3.3 Блок оцінки класифікації

Третій блок фокусується на оцінці якості класифікації виявлених ознак. Респондентам надається повний перелік можливих класів ознак з їх детальними описами та прикладами. Блок містить 24 питання, де для кожної ознаки представлені варіанти класифікації від обох моделей.

У цьому блоці присутні два види питань.

Перший опитує доречність класифікації, у випадку тотожної класифікації ознаки обома моделями.

Другий блок дає можливість вибрати чотири варіанти, щоб виявити точку зору респондента на класифікацію ознаки сну. В описі питання вказано саму ознаку сну і класифікацію від кожної моделі.

Кожен блок супроводжується необхідними поясненнями та інструкціями, що допомагають респондентам краще зрозуміти завдання та надати більш точні відповіді.

Увесь перелік питань для опитування наведено в додатку К.

6.4 Результати опитування

6.4.1 Демографічні характеристики респондентів

В опитуванні взяли участь 13 респондентів, що представляють різні демографічні групи. Гендерний розподіл учасників виявився майже рівномірним: по 6 респондентів (46.2%) чоловічої та жіночої статі, також один респондент (7.7%) обрав варіант "Інше" (див. рис. 6.1.) (див. табл Л.1).

Рисунок 6.1 – Гендерний склад респондентів

Віковий розподіл учасників показав переважання молодої аудиторії (див рис. 6.2.) (див. табл Л.2): більшість респондентів (8 осіб, 61.5%) належать до вікової групи 21-30 років. Другою за чисельністю стала група 11-20 років (3 особи, 23.1%). По одному респонденту (7.7%) представляли вікові групи 31-40 та 51-60 років.

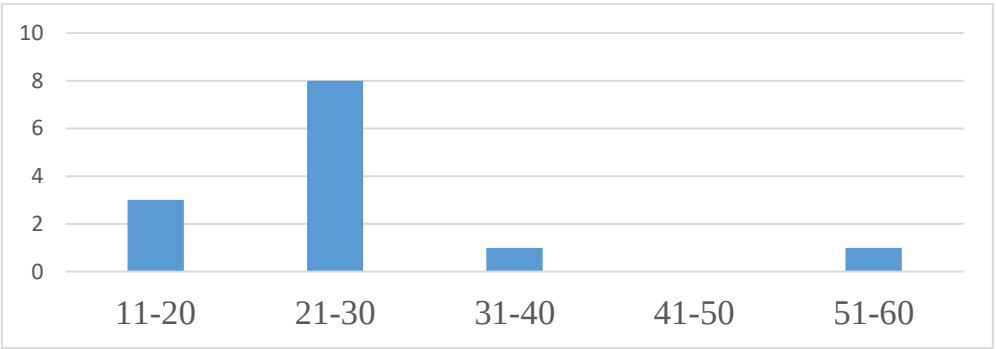


Рисунок 6.2 – Віковий склад респондентів

6.4.2 Оцінка схильності до моделей виявлення ознак

При оцінці якості виявлення онейрологічних образів респонденти надали невелику перевагу моделі Claude 3.5 Sonnet, яка отримала 122 позитивних оцінки порівняно з 120 оцінками для моделі DistilBERT (див. рис. 6.3.) (див. табл Л.3).

Лише в трьох випадках респонденти відзначили, що жодна з моделей не впоралась із завданням задовільно. Така близькість результатів свідчить про те, що обидві моделі демонструють порівнянний рівень якості при виявленні ознак сну, незважаючи на відмінності в їхній архітектурі та підходах до обробки тексту.

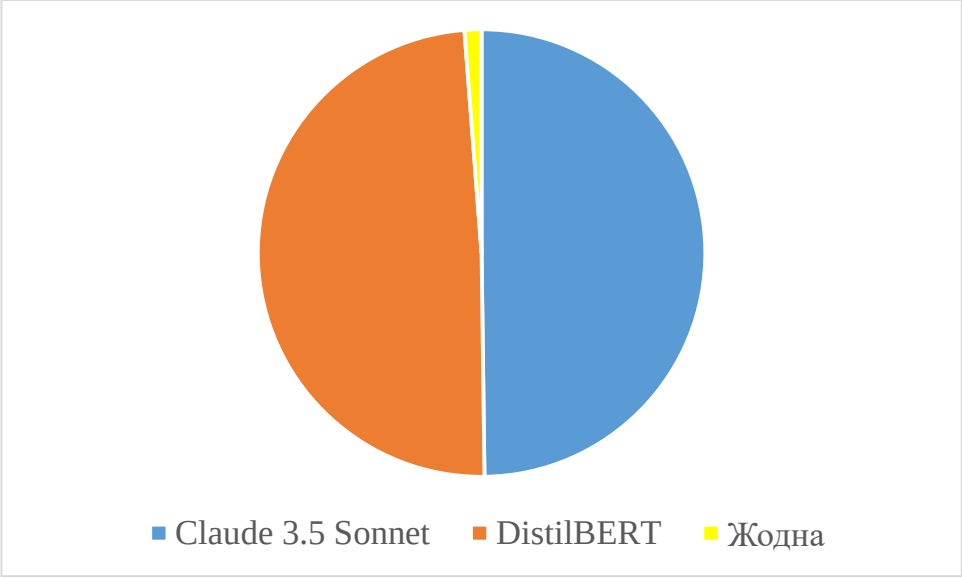


Рисунок 6.3 – Розподіл схильностей респондентів до моделей виявлення ознак

6.4.3 Оцінка схильності до моделей якості класифікації

У завданні класифікації виявлених ознак спостерігається більш чітка диференціація результатів. Модель Claude 3.5 Sonnet отримала 105 позитивних оцінок проти 61 для моделі RoBERTa (див. рис. 6.4) (див. табл Л.4). Примітно, що в 122 випадках респонденти визнали прийнятною класифікацію обох

моделей, що свідчить про загальну адекватність їхніх підходів до категоризації ознак. У 24 випадках жодна з моделей не задовольнила очікування респондентів щодо якості класифікації.

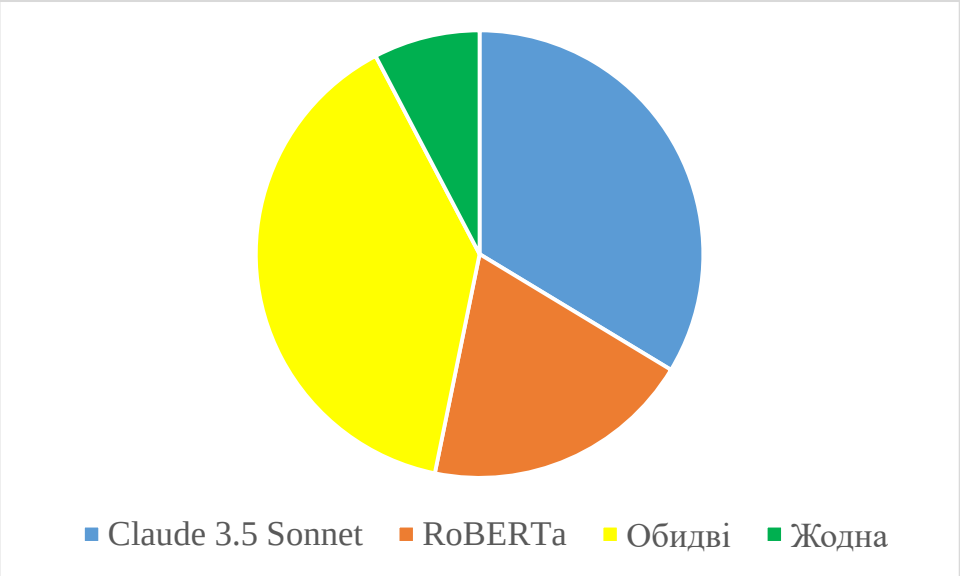


Рисунок 6.4 – Розподіл схильностей респондентів до моделей класифікації ознак

6.4.4 Загальні результати

Підсумовуючи всі оцінки, включаючи випадки, коли респонденти схвалили роботу обох моделей, Claude 3.5 Sonnet отримала 349 позитивних відгуків проти 303 для комбінації моделей DistilBERT-RoBERTa (див. рис. 6.5.). У 27 випадках жодна з моделей не задовольнила критеріям респондентів. Такий розподіл оцінок вказує на певну перевагу моделі Claude 3.5 Sonnet в суб'єктивному сприйнятті користувачів, особливо в завданні класифікації ознак

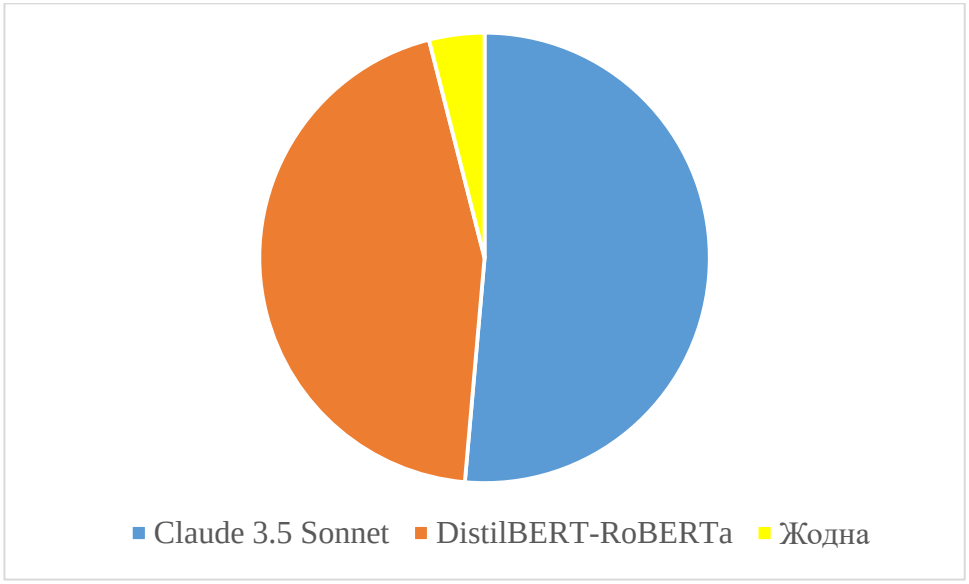


Рисунок 6.5 – Розподіл схильностей респондентів до моделей виявлення та класифікації ознак

Ці результати становлять цікавий контраст з кількісними метриками, де DistilBERT демонструвала кращі показники. Це може свідчити про те, що стандартні метрики оцінки не повністю відображають якість роботи моделей з точки зору кінцевих користувачів, особливо в контексті такого суб'єктивного завдання як аналіз снів.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи досліджено методи та технології для вирішення задачі виявлення та класифікації онейрологічних образів у текстах українською мовою. Основною метою було зменшення помилок другого роду під час цього процесу шляхом налаштування моделей машинного навчання.

Для досягнення поставленої мети проведено порівняльний аналіз різних підходів до обробки природньомовних текстів. В результаті аналізу обрано два основних напрямки: використання розділеного підходу на основі моделей сімейства BERT та застосування єдиної великої мовної моделі (LLM).

Під час розробки підходsd виявилось, що окрім помилок другого роду на результати також впливають і помилки першого роду, які теж необхідно мінімізувати.

В рамках розділеного підходу реалізовано та протестовано кілька комбінацій моделей для виявлення та класифікації ознак сну. Найкращі результати показала композиція моделей DistilBERT для виявлення (точність передбачення – 86.06%, повнота 86.84% F1-score 86.31%) та RoBERTa для класифікації (повнота 53.17%, F1-score 54.00%). При оцінці комплексної роботи цих моделей досягнуто наступних показників: точність – 27.86%, точність передбачення – 72.09%, повнота – 80.44%, F1-score 76.04% та hamming loss 3.22%.

Для підходу з використанням єдиної моделі протестовано кілька сучасних великих мовних моделей, включаючи GPT-4o, GPT-4o-mini, Claude 3.5 Haiku та Claude 3.5 Sonnet. Найкращі результати продемонструвала модель Claude 3.5 Sonnet з наступними показниками: точність – 23.61%, точність передбачення – 27.83%, повнота – 39.39%, F1-score – 32.61% та hamming loss – 8.77%. Експерименти показали, що моделі краще працюють при нульовому рівні креативності та з використанням невеликого набору випадкових прикладів для few-shot навчання.

Порівняння двох підходів демонструє, що розділений підхід з використанням спеціалізованих моделей значно перевершує єдину велику мовну модель за всіма метриками загалом і зокрема за якістю визначення помилок другого роду (різниця за повнотою у 41.05%), помилок першого роду (різниця за точністю передбачення 44.26%) та загальною точністю (різниця в 4.25%).. Це можна пояснити тим, що спеціалізовані моделі краще адаптуються до конкретних завдань завдяки можливості точного налаштування під специфіку даних.

Результати опитування виявили цікавий парадокс між об'єктивними метриками ефективності моделей та їх суб'єктивним сприйняттям користувачами. Хоча композиція моделей DistilBERT-RoBERTa продемонструвала значно кращі кількісні показники в експериментах, учасники опитування віддали невелику перевагу моделі Claude 3.5 Sonnet, особливо в задачі класифікації ознак сну. Це може пояснюватися кількома факторами: по-перше, велика мовна модель могла краще враховувати контекстуальні нюанси та семантичні зв'язки, які важливі для людського сприйняття; по-друге, її класифікація могла виглядати більш природною та інтуїтивно зрозумілою для користувачів, навіть якщо вона не завжди відповідала формальним критеріям оцінки.

Крім того, значна кількість випадків, коли респонденти схвалили роботу обох моделей (122 випадки при класифікації), свідчить про те, що обидва підходи мають свої переваги і можуть успішно застосовуватися для аналізу онейрологічних образів.

Це вказує на потенційну користь розробки гібридних рішень, які б поєднували сильні сторони обох підходів для створення більш ефективних систем аналізу снів.

Практична цінність роботи полягає в створенні та оцінки методології обробки україномовних текстів снів, яка може бути застосована для розробки систем автоматичного ведення щоденників сновидінь. Крім того, розроблені підходи можуть бути адаптовані для інших завдань обробки природньомовних

текстів з подібними вимогами до виявлення та класифікації текстових фрагментів.

Це підтверджується тим, що роботу рекомендовано до впровадження в компанії VTM Group для виявлення та класифікації ознак продукції в текстах повідомлень від лідів (див. додаток М).

За результатами даної роботи опубліковано тези на XXI всеукраїнській конференції студентів і молодих науковців.

Подальші напрямки досліджень можуть включати:

- розробку гібридних підходів, що поєднують переваги обох методів;
- вдосконалення системи класифікації ознак сну для підвищення точності категоризації;
- розширення датасету для покращення якості навчання моделей;
- оптимізацію архітектури моделей для підвищення ефективності обробки текстів;
- дослідження можливостей застосування розроблених рішень для інших мов та предметних областей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hirshkowitz M, Whiton K, Albert SM, Alessi C, Bruni O, DonCarlos L, Hazen N, Herman J, Adams Hillard PJ, Katz ES, Kheirandish-Gozal L, Neubauer DN, O'Donnell AE, Ohayon M, Peever J, Rawding R, Sachdeva RC, Setters B, Vitiello MV, Ware JC. National Sleep Foundation's updated sleep duration recommendations: final report. Sleep Health. 2015 Dec;1(4):233-243. doi: 10.1016/j.sleh.2015.10.004. Epub 2015 Oct 31. PMID: 29073398.

2. Revision of World Population Prospects. United Nations. Department of Economic and Social Affairs. Population Division. 2022 [Електронний ресурс] Режим доступу: https://population.un.org/wpp/.

3. Lucid dreams overview. Minesh Khatri. Kris Martins. 2022 [Електронний ресурс] Режим доступу: https://www.webmd.com/sleep-disorders/lucid-dreams-overview.

4. Aakash K. Patel; Vamsi Reddy; Karlie R. Shumway; John F. Araujo. Physiology, Sleep Stages. Treasure Island (FL): StatPearls Publishing; 2024 Jan.

5. Laberge, S. (1980). Lucid dreaming: An exploratory study of consciousness during sleep (PhD thesis). Stanford University.

6. Laberge S., Levitan L. / Validity Established of DreamLight Cues for Eliciting Lucid Dreaming / Lucidity Letter Journal. - 1986. - Vol. 5, No. 3.

7. Jay Summer / How to Keep a Dream Journal: Tips and Techniques // Sleep Foundation. - 2023. - [Електронний ресурс] Режим доступу: https://www.sleepfoundation.org/dreams/dream-journal

8. Лаберж С. Свідомий сон. Практика свідомого сну. Видавництво «Софія» - 1996.

9. Top 10 Lucid Dreaming Apps For iOS & Android. Steven Voser. 2022. [Електронний ресурс] Режим доступу: https://www.zamnesia.com/blog-the-best-lucid-dreaming-apps-n1347

10. Виділення та класифікації онейрологічних образів в природномовному тексті / М.Ю. Жар, Є.В. Малахов // Інформатика, інформаційні системи та

технології: тези доповідей двадцять першої всеукраїнської конференції студентів і молодих науковців. Одеса, 26 квітня 2024 р. - Одеса, 2024.– С. 121-123.

11. Comparison Study Between Token Classification and Sequence Classification In Text Classification.

12. What are tokens and how to count them [Електронний ресурс] // OpenAI Help Center. – 2024. – Режим доступу: https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them

13. A Guide to Unlocking the Power of Sequence Classification, Mostafa Ibrahim, 2023

14. Token Classification. Hugging Face. [Електронний ресурс] Режим доступу: https://huggingface.co/tasks/token-classification.

15. Token Classification in Python with HuggingFace. Reyansh Bahl. 2023. [Електронний ресурс] Режим доступу: https://reybahl.medium.com/token-classification-in-python-with-huggingface-3fab73a6a20e

16. Named Entity Recognition: The Mechanism, Methods, Use Cases, and Implementation Tips. [Електронний ресурс] Режим доступу: https://www.altexsoft.com/blog/named-entity-recognition/

17. Deep Learning vs. Machine Learning – What's The Difference? Arne Wolfewicz. 2023. [Електронний ресурс] Режим доступу: https://levity.ai/blog/difference-machine-learning-deep-learning

18. What is deep learning? IBM. [Електронний ресурс] Режим доступу: https://www.ibm.com/topics/deep-learning

19. Recurrent Neural Networks (RNN) Tutorial: RNN Training, Advantages & Disadvantages (Complete Guidance) Akshay Tondak 2023

20. LSTM — Implementation, Advantages and Diadvantages, Prudhviraju Srivatsavaya, 2023

21. IBM. What is transformer model ? [Електронний ресурс] Режим доступу: https://www.ibm.com/topics/transformer-mode

22. Attention is all you need. A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, AN Gomez, Ł Kaiser, I Polosukhin. Advances in neural information processing systems, 2017

23. How Transformers Work: A Detailed Exploration of Transformer Architecture. Datacamp. 2024. [Електронний ресурс] Режим доступу: https://www.datacamp.com/tutorial/how-transformers-work

24. From RNNs to Transformers. Baeldung. Eric Martin. 2024ю [Електронний ресурс] Режим доступу: https://www.baeldung.com/cs/rnns-transformers-nlp

25. Yang, J., Jin, H., Tang, R., Han, X., Feng, Q., Jiang, H., Yin, B., & Hu, X. (2023). Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond. ACM Trans. Knowl. Discov. Data, 18, 160:1-160:32.

26. NLP Course Chapter 1: Introduction to NLP [Електронний ресурс] // Hugging Face. – 2024. – Режим доступу: https://huggingface.co/learn/nlp-course/chapter1/5

27. NLP Course: Decoder models [Електронний ресурс] // Hugging Face. – 2024. – Режим доступу: https://huggingface.co/learn/nlp-course/chapter1/6?fw=pt

28. GPT-4 Turbo: Quality, Performance & Price Analysis. [Електронний ресурс] Режим доступу: https://artificialanalysis.ai/models/gpt-4o-2024-08-06

29. NLP Course: Sequence-to-sequence models [Електронний ресурс] // Hugging Face. – 2024. – Режим доступу: https://huggingface.co/learn/nlp-course/chapter1/7?fw=pt

30. Banerjee, A., Chitnis, U. B., Jadhav, S. L., Bhawalkar, J. S., & Chaudhury, S. (2009). Hypothesis testing, type I and type II errors. Industrial psychiatry journal, 18(2), 127–131. https://doi.org/10.4103/0972-6748.62274

31. Wadhwa T. Multi-Class Classification: One vs All & One vs One [Електронний ресурс] / T. Wadhwa // Medium. – 2023. – Режим доступу: https://wadhwatanya1234.medium.com/multi-class-classification-one-vs-all-one-vs-one-993dd23ae7ca

32. Українська мова: Енциклопедія [Електронний ресурс] // Літопис. – 2000. – Режим доступу: http://litopys.org.ua/ukrmova/um219.htm

33. Висоцька В. А. Особливості лінгвістичного моделювання україномовних текстових масивів [Електронний ресурс] / В. А. Висоцька, Л. В. Чирун, Л. Б. Чирун // Вісник Національного університету "Львівська політехніка". – 2018. – С. 139-148. – Режим доступу: https://science.lpnu.ua/sites/default/files/journal-paper/2018/jun/13011/ilovepdfcom-139-148.pdf

34. Документація Spacy LLM. [Електронний ресурс] Режим доступу: https://spacy.io/usage/large-language-models.

35. What is zero-shot learning? IBM. 2024. [Електронний ресурс] Режим доступу: https://www.ibm.com/topics/zero-shot-learning

36. What is few-shot learning? IBM. 2024. [Електронний ресурс] Режим доступу: https://www.ibm.com/topics/few-shot-learning

37. One Hot Encoding in Machine Learning [Електронний ресурс] // GeeksforGeeks. – 2023. – Режим доступу: https://www.geeksforgeeks.org/ml-one-hot-encoding/

38. End-to-End Learning: Definition and Applications [Електронний ресурс] // Clickworker AI Glossary. – 2024. – Режим доступу: https://www.clickworker.com/ai-glossary/end-to-end-learning/

39. Декоратори в Python [Електронний ресурс] // ACode Ukraine. – 2023. – Режим доступу: https://acode.com.ua/decorators-python/?-#3-8 https://refactoring.guru/design-patterns/factory-method/python/example

40. Factory Method // Refactoring.Guru: Design Patterns / під ред. Alexander Shvets. - 2024. - [Електронний ресурс] Режим доступу: https://refactoring.guru/design-patterns/factory-method

41. Spacy-LLM Templates [Електронний ресурс] // GitHub Repository explosion/spacy-llm. – 2024. – Режим доступу: https://github.com/explosion/spacy-llm/blob/main/spacy_llm/tasks/templates/

42. Robino G. Non-English Languages in Prompt Engineering: Trade-offs and Best Practices [Електронний ресурс] / Giorgio Robino // LinkedIn. – 2023. –

Режим доступу: https://www.linkedin.com/pulse/non-english-languages-prompt-engineering-trade-offs-giorgio-robino

43. Classification: Accuracy, Precision, Recall [Електронний ресурс] // Google Developers. – 2024. – Режим доступу: https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

44. Scikit-learn: F-beta score [Електронний ресурс] // Scikit-learn Documentation v1.5. – 2024. – Режим доступу: https://scikit-learn.org/1.5/modules/generated/sklearn.metrics.fbeta_score.html

45. Scikit-learn: Hamming Loss [Електронний ресурс] // Scikit-learn Documentation v1.5. – 2024. – Режим доступу: https://scikit-learn.org/1.5/modules/generated/sklearn.metrics.hamming_loss.html

46. Understanding ROC Curves in Machine Learning [Електронний ресурс] // Evidently AI Documentation. – 2024. – Режим доступу: https://www.evidentlyai.com/classification-metrics/explain-roc-curve

47. ROC Curve and AUC [Електронний ресурс] // Google Machine Learning Crash Course. – 2024. – Режим доступу: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

48. Maurya, M. Name Entity Recognition and Various Tagging Schemes [Електронний ресурс] / M. Maurya // Medium. – 2023. – Режим доступу: https://medium.com/@muskaan.maurya06/name-entity-recognition-and-various-tagging-schemes-533f2ac99f52

49. Token Classification [Електронний ресурс] // Hugging Face Transformers Documentation. – 2024. – Режим доступу: https://huggingface.co/docs/transformers/tasks/token_classification

50. Aina T. Understanding Logits: A Fundamental Concept in Machine Learning [Електронний ресурс] / Temiloluwa Aina // LinkedIn. – 2023. – Режим доступу: https://www.linkedin.com/pulse/understanding-logits-fundamental-concept-machine-aina-temiloluwa-7d3ff

51. Sigmoid and Softmax Functions Explained [Електронний ресурс] // Towards Data Science. – 2023. – Режим доступу:

https://towardsdatascience.com/sigmoid-and-softmax-functions-in-5-minutes-f516c80ea1f9

52. A Complete Guide to the Softmax Activation Function [Електронний ресурс] // SingleStore Blog. – 2024. – Режим доступу: https://www.singlestore.com/blog/a-guide-to-softmax-activation-function/

53. Brownlee J. One-vs-Rest and One-vs-One for Multi-Class Classification [Електронний ресурс] / Jason Brownlee // Machine Learning Mastery. – 2023. – Режим доступу: https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/

54. Brownlee J. Arithmetic, Geometric and Harmonic Means for Machine Learning [Електронний ресурс] / Jason Brownlee // Machine Learning Mastery. – 2023. – Режим доступу: https://machinelearningmastery.com/arithmetic-geometric-and-harmonic-means-for-machine-learning/

55. Harmonic Mean: Definition, Formula & Examples [Електронний ресурс] // CueMath. – 2024. – Режим доступу: https://www.cuemath.com/data/harmonic-mean/

56. Claude API Reference [Електронний ресурс] // Anthropic Documentation. – 2024. – Режим доступу: https://docs.anthropic.com/en/api/complete

57. Understanding GPT-3 Temperature Parameter [Електронний ресурс] // GPT for Work Guides. – 2024. – Режим доступу: https://gptforwork.com/guides/openai-gpt3-temperature

58. OpenAI ChatGPT API Pricing Calculator [Електронний ресурс] // GPT for Work Tools. – 2024. – Режим доступу: https://gptforwork.com/tools/openai-chatgpt-api-pricing-calculator

59. Курс валют на 01.11.2024 // Мінфін. – 2024. – [Електронний ресурс] Режим доступу: https://minfin.com.ua/currency/usd/

ДОДАТОК А
ПОРІВНЯННЯ МЕТОДІВ ТА ТЕХНОЛОГІЙ

Таблиця А.1. Порівняльна таблиця класів методів NER.

	Правило-спрямовані	Машинне навчання	Глибоке навчання
Обробка нечітких даних	Дуже погано	Погано	Гарно
Необхідний контроль над навчанням	Повний	Навчальні дані	Відсутній
Потреба у високопродуктивному обладнанні	Відсутня	Середня	Висока

Таблиця А.2. Порівняльна таблиця класів методів NER.

	RNN	LSTM	Трансформери
Обробка нечітких даних	Добре	Добре	Дуже гарно
Розуміння контексту	Погане	Погане	Присутнє
Швидкість навчання	Погана	Дуже погана	Гарна
Україномовні моделі	Невідомі	Невідомі	Присутні
Потреба у високопродуктивному обладнанні	Висока	Висока	Гігантська

Таблиця А.3. Порівняння енкодерних трансформерів.

	Bert-LG	Roberta-LG	XLNET LG	ALBERT LG V2	Людина
Дані для тренування	Book Corpus + Англійська вікіпедія (16 GB)	CommonCrawl Новини, OpenWebText та інше (160 GB)	Book Corpus + Англійська вікіпедія (16 GB)	32 GB	?
Кількість параметрів	340M	355M	340M	17M	?
Кількість шарів	24	24	24	24	?
Кількість голів	24	16	16	16	1
Commonsense QA, %	55.9	72.1	75.3	76.5	~89

Таблиця А.4. Порівняння ALBERT LG V2 з GPT різних версій.

	ALBERT LG V2	GPT-2 XL	GPT-3	GPT-4
Дані для тренування	32 GB	40 GB	75 TB	1 PB
Кількість параметрів	17M	1.5B	175B	1.76T
Кількість шарів	12	48	96	120
Вага	71 MB	6.43 GB	17 GB	45 GB
Commonsense QA, %	76.5	< 38.0	38.0	> 81.0
Ціна за API	Open Source	Open Source	-	30$ за 1M токенів на вході 60$ за 1M токенів на виході

Таблиця А.5. Порівняння аналогів GPT.

	GPT-4	Gemini	Llama 2	Claude 3
Дані для тренування	1 PB	?, 540 PB	75 TB	5 – 15 T words
Кількість параметрів	1.76T	1.5T	70B	100T
Кількість шарів	120	?	80	80
Вага	45 GB	?	130 GB	?
Обладнання	> 3 NVIDIA A100 80GB	> 3 NVIDIA A100 80GB	4 x 48GB GPU	?
Ціна за API	30$ за 1M токенів на вході 60$ за 1M токенів на виході	Безкоштовно 50 RPD запитів на день 2 запитів у хв 32000 токенів у хв	Open Source	$15 за 1M токенів на вході $75 за 1M токенів на виході

ДОДАТОК Б
ПРОМПТ ДЛЯ LLM

```
You are an expert Dream Analysis system.
Your task is to identify distinct dream moments within a given dream narrative and classify each moment with relevant labels.
{# whitespace #}
Instructions:
Identify non-overlapping dream moments in the text.
These moments can vary in length from a single sentence to multiple paragraphs.
For each identified moment, provide the following:
   a. The text of the dream moment
   b. The bool flag indicating if you think you could classify this moment
   c.1 If you can classify this moment, provide a list of applicable labels from the following options: {{ labels|join(", ") }}
   c.2 If you cannot classify this moment, provide the label ==NONE==
{# whitespace #}
Output Format:
For each dream moment, provide the information in the following format:
NUMBER. DREAM MOMENT TEXT | FLAG | Label1, Label2, Label3
{# whitespace #}
Do not output anything besides entities in this output format.
Output entities in the order they occur in the input paragraph regardless of label.
{# whitespace #}
{%- if description -%}
{# whitespace #}
{{ description }}
{# whitespace #}
{%- endif -%}
{%- if labels -%}
{# whitespace #}
Use the following labels: {{ labels|join(", ") }}
{# whitespace #}
{%- endif -%}
{%- if label_definitions -%}
{# whitespace #}
Below are definitions of each label to help aid you in what kinds of moments you can classify.
Assume these definitions are not exhaustive, but are meant to give you a general idea of what each label means.
{# whitespace #}
{%- for label, definition in label_definitions.items() -%}
{{ label }}: {{ definition }}
{# whitespace #}
{%- endfor -%}
{# whitespace #}
{# whitespace #}
{%- endif -%}
{%- if prompt_examples -%}
{# whitespace #}
Q: Given the following dream narrative, extract the dream moments and classify each moment with relevant labels:
{# whitespace #}
```

Лістінг Б.1, аркуш 1 – Промпт для великих мовних моделей

```
{%- for example in prompt_examples -%}
{# whitespace #}
Dream Text:
{# whitespace #}
{{ example.dream }}
Answer:
{# whitespace #}
{%- for moment in example.moments -%}
{{ loop.index }}. {{ moment.text }} | {{ moment.flag }} | {% if
moment.flag %}{{ moment.labels|join(", ") }}{% else %}==NONE=={% endif %}
{# whitespace #}
{%- endfor -%}
{# whitespace #}
{%- endfor -%}
{%- else -%}
{# whitespace #}
Here is an example of the output format for a dream using different labels than
this task requires.
Use labels provided above instead of ones defined in the example below.
{# whitespace #}
Q: Given the following dream narrative, extract the dream moments and classify
each moment with relevant labels:

Dream Text: Я йшов по темному лісі. Раптом побачив світло між деревами.
Підійшовши ближче, я зрозумів, що це була хатинка. Коли я постукав у двері,
вони самі відчинилися. Всередині я побачив старого чоловіка, який сидів біля
каміна і читав велику книгу. Він подивився на мене і сказав: "Я чекав на тебе".
Потім він простягнув мені книгу, і коли я її відкрив, з неї вилетіли яскраві
метелики.
Answer:
1. Я йшов по темному лісі. | True | Моє місцезнаходження
2. Раптом побачив світло між деревами. | True | Місцезнаходження об'єктів
3. Підійшовши ближче, я зрозумів, що це була хатинка. | True | Місцезнаходження
об'єктів, Мої думки
4. Коли я постукав у двері, вони самі відчинилися. | True | Дії об'єктів
5. Всередині я побачив старого чоловіка, який сидів біля каміна і читав велику
книгу. | True | Форма істот, Місцезнаходження істот
6. Він подивився на мене і сказав: "Я чекав на тебе". | True | Дії істот
7. Потім він простягнув мені книгу, і коли я її відкрив, з неї вилетіли яскраві
метелики. | True | Форма істот, Місцезнаходження істот
{%- endif -%}
{# whitespace #}
{# whitespace #}
Q: Given the following dream narrative, extract the dream moments and classify
each moment with relevant labels:

Dream Text: {{ text }}
Answer:
```

Лістінг Б.1, аркуш 2 – Промпт для великих мовних моделей

ДОДАТОК В
СКРІПТИ СТВОРЕННЯ ДАТАСЕТІВ

```
import json
import re
import datasets
import polars as pl
import os
from sklearn.model_selection import train_test_split
pl.Config.set_fmt_str_lengths(100)
HF_TOKEN = os.environ.get("HF_TOKEN")
DATASET_FOLDER = "data/personal/personal-dataset-10-21-24"
with open(f'{DATASET_FOLDER}/dreams.json', 'r') as f:
    dreams_data = json.load(f)
with open(f'{DATASET_FOLDER}/metadata.json', 'r') as f:
    metadata = json.load(f)
LABELS = {
  "NONE": 0,
  "B-EXIST": 1,
  "I-EXIST": 2,
 }
labels2id = LABELS
id2labels = {v: k for k, v in labels2id.items()}
def tokenize(text):
    return re.findall(r'\w+|[^\w\s]', text.lower())
def is_moment_equal(tokens, start_idx, moment_tokens, threshold=0.8):
    incorrect_tokens = 0
    moment_tokens_length = len(moment_tokens)
    for j in range(start_idx, len(tokens)):
        if j - start_idx >= moment_tokens_length:
            break
        if  tokens[j] != moment_tokens[j - start_idx]:
            incorrect_tokens += 1
    return incorrect_tokens / moment_tokens_length <= 1 - threshold
def token_collection(moment_tokens_length):
    labels = [1] + [2] * (moment_tokens_length - 1)
    return labels
def create_token_classification_dataset(dreams_data):
    data = []
    THRESHOLD = 0.8

    for dream in dreams_data['dreams']:
        text = dream['text']
        tokens = tokenize(text)
        tokens_length = len(tokens)
        moments_tokens = []
        for moment in dream['moments']:
            moments_tokens.append(tokenize(moment['text']))
        labels = [LABELS['NONE']] * tokens_length
        i = 0
        while i < tokens_length:
            if not moments_tokens:
                break
```

Лістінг В.1 – Створення датасету для виявлення онейрологічних образів

```
moment_tokens, idx = next(((mt, idx) for idx, mt in enumerate(moments_tokens)
                                        if tokens[i] in mt and
                                            is_moment_equal(tokens,   i,   mt,
threshold=THRESHOLD)), (None, -1))
            if(moment_tokens):
                moment_tokens_length = len(moment_tokens)
                labels[i:i+moment_tokens_length]                              =
token_collection(moment_tokens_length)
                moments_tokens.pop(idx)
                i += moment_tokens_length - 1
                continue
            i += 1

        data.append({
            'id': dream['id'],
            'tokens': tokens,
            'labels': labels
        })

    return pl.DataFrame(data)
token_classification_dataset =
create_token_classification_dataset(dreams_data)
dataset = datasets.Dataset.from_polars(token_classification_dataset)
label_feature                                                                 =
datasets.LargeList(datasets.ClassLabel(names=list(LABELS.keys())))
dataset = dataset.cast_column("labels", label_feature)
print(dataset.features)
dataset.save_to_disk(f"{DATASET_FOLDER}/exists-dataset-3-labels.hf")
```

Лістінг В.1, аркуш 2 – Створення датасету для виявлення онейрологічних образів

```
import json
import re
import datasets
import polars as pl
from sklearn.model_selection import train_test_split
pl.Config.set_fmt_str_lengths(100)
DATASET_FOLDER = "data/personal/personal-dataset-10-21-24"
with open(f'{DATASET_FOLDER}/dreams.json', 'r') as f:
    dreams_data = json.load(f)
with open(f'{DATASET_FOLDER}/metadata.json', 'r') as f:
    metadata = json.load(f)
markers_dict                  =                {f"{marker['name'].strip()}
({marker['category_name'].strip()})":   marker['id']   for   marker   in
dreams_data['markers']}
marker_id_to_int    =    {marker_id:    i    for    i,    marker_id    in
enumerate(markers_dict.values())}
id2label = {marker_id_to_int[v]: k for k, v in markers_dict.items()}
label2id = {k: marker_id_to_int[v] for k, v in markers_dict.items()}
def process_dreams(dreams):
  processed_dreams = []
  for dream in dreams:
```

Лістінг В.2 – Створення датасету для класифікації онейрологічних образів

```
      dream_moments = dream['moments']
      for moment in dream_moments:
          moment_text = moment['text']
          moment_markers  =  [marker_id_to_int[marker_id]  for  marker_id  in
moment['markers']]
          processed_dreams.append({
              'text': moment_text,
              'labels': moment_markers,
              'dream_id': dream['id'],
              'start': moment['index'],
              'end': moment['index'] + moment['length']
          })
  df = pl.DataFrame(processed_dreams)
  return df
df = process_dreams(dreams_data['dreams'])
ids = list(id2label.keys())
df = df.with_columns(
    pl.col('labels').map_elements(lambda x: [1.0 if id in x else 0.0 for id in
ids], return_dtype=pl.List(pl.Float64))
)
text_classification_dataset = df
dataset =
datasets.Dataset.from_polars(text_classification_dataset)
label_names = list(label2id.keys())
label_feature = datasets.LargeList(datasets.ClassLabel(names=label_names))
dataset = dataset.cast_column("labels", label_feature)
dataset.save_to_disk(f"{DATASET_FOLDER}/text-classification-dataset.hf")
```

Лістінг В.2, аркуш 2 – Створення датасету для класифікації онейрологічних образів

```
import json
import polars as pl
from datasets import Dataset
import os
DATASET_FOLDER = "data/personal/personal-dataset-10-21-24"
with open(f'{DATASET_FOLDER}/dreams.json', 'r', encoding='utf-8') as f:
    dreams_data = json.load(f)
with open(f'{DATASET_FOLDER}/metadata.json', 'r', encoding='utf-8') as f:
    metadata = json.load(f)
marker_id2name          =          {marker['id']:          f"{marker['name']}
({marker['category_name'].strip()})" for marker in dreams_data['markers']}
markers_dict = {
    f"{marker['name']} ({marker['category_name'].strip()})": {
        "id": marker['id'],
        "description": marker['description'],
    } for marker in dreams_data['markers']
}
transformed_dreams = []
for dream in dreams_data['dreams']:
    transformed_dream = {
        "dream": dream['text'],
        "moments": []
    }
```

Лістінг В.3 – Створення датасету для повноцінної моделі

```
    for moment in dream['moments']:
        transformed_moment = {
            "text": moment['text'],
            "flag": len(moment['markers']) > 0,
            "labels": moment['markers'],
            "start": moment['index'],
            "end": moment['index'] + moment['length']
        }
        transformed_dream["moments"].append(transformed_moment)

    transformed_dreams.append(transformed_dream)
print(json.dumps(transformed_dreams[0], indent=2, ensure_ascii=False))
print(f"Total number of transformed dreams: {len(transformed_dreams)}")
label_df = pl.DataFrame(
    {
        "label": list(markers_dict.keys()),
        "id": [marker["id"] for marker in markers_dict.values()],
        "description":      [marker["description"]      for      marker      in
markers_dict.values()]
    }
)
LLM_DATASET_FOLDER = DATASET_FOLDER + "/llm"
os.makedirs(LLM_DATASET_FOLDER, exist_ok=True)
label_df.write_parquet(f"{LLM_DATASET_FOLDER}/labels_dataset.parquet")
with open(f"{LLM_DATASET_FOLDER}/dreams_dataset.json", "w", encoding="utf-8")
as f:
    json.dump(transformed_dreams, f, ensure_ascii=False)
```

Лістінг В.3, аркуш 2 – Створення датасету для повноцінної моделі

ДОДАТОК Г
ФУНКЦІЇ ТА ПРИКЛАД ЕКСПЕРИМЕНТУ НАД МОДЕЛЛЮ ВИЯВЛЕННЯ

```
def tokenize_and_align_labels(examples, tokenizer):
    tokenized_inputs = tokenizer(examples["tokens"], padding=True,
truncation=True, is_split_into_words=True)
    labels = []
    for i, label in enumerate(examples[f"labels"]):
        word_ids = tokenized_inputs.word_ids(batch_index=i)  # Map tokens to
their respective word.
        previous_word_idx = None
        label_ids = []
        for word_idx in word_ids:  # Set the special tokens to -100.
            if word_idx is None:
                label_ids.append(-100)
            elif word_idx != previous_word_idx:  # Only label the first token
of a given word.
                label_ids.append(label[word_idx])
            else:
                label_ids.append(-100)
            previous_word_idx = word_idx
        labels.append(label_ids)

    tokenized_inputs["labels"] = labels
    return tokenized_inputs
```

Лістінг Г.1 – Функція вирівнювання токенів під мітки оригінального датасету

```
def preprocess_dataset(dataset, tokenizer):
    tokenized_dataset = dataset.map(tokenize_and_align_labels, batched=True,
fn_kwargs={"tokenizer": tokenizer})
    tokenized_dataset = tokenized_dataset.remove_columns(['tokens'])
    tokenized_dataset = tokenized_dataset.cast_column("labels",
datasets.Sequence(datasets.Value("float32")))
    return tokenized_dataset
```

Лістінг Г.2 – Функція конвертації датасету у необхідний для моделі формат

```
def process_dataset(dataset, tokenizer):
    tokenized_dataset = preprocess_dataset(dataset, tokenizer)
    train_dataset, eval_dataset = split_dataset(tokenized_dataset)
    return train_dataset, eval_dataset
```

Лістінг Г.3 – Функція обробки датасету на набори для тренування та валідації

```
def compute_metrics(p):
    predictions, labels = p.predictions, p.label_ids
    probabilities   =   torch.nn.functional.softmax(torch.tensor(predictions),
dim=-1).numpy()
    predicted_labels = np.argmax(probabilities, axis=-1)
    mask = labels != -100
    filtered_predictions = [
        [p for (p, m) in zip(prediction, mask_row) if m]
        for prediction, mask_row in zip(predicted_labels, mask)
    ]
    filtered_labels = [
        [l for (l, m) in zip(label, mask_row) if m]
        for label, mask_row in zip(labels, mask)
    ]
    flat_predictions = [item for sublist in filtered_predictions for item in
sublist]
    flat_labels = [item for sublist in filtered_labels for item in sublist]
    precision      =      precision_score(flat_labels,      flat_predictions,
average='weighted', zero_division=0)
    recall = recall_score(flat_labels, flat_predictions, average='weighted',
zero_division=0)
    f1    =    f1_score(flat_labels,    flat_predictions,    average='weighted',
zero_division=0)
    accuracy = accuracy_score(flat_labels, flat_predictions)
    hamming = hamming_loss(flat_labels, flat_predictions)
    filtered_probabilities = [
        [p for (p, m) in zip(prob, mask_row) if m]
        for prob, mask_row in zip(probabilities, mask)
    ]
    flat_probabilities = np.array([item for sublist in filtered_probabilities
for item in sublist])
    roc_auc       =       roc_auc_score(flat_labels,       flat_probabilities,
average='weighted', multi_class='ovo')
    return {
        "precision": precision,
        "recall": recall,
        "f1": f1,
        "accuracy": accuracy,
        "hamming_loss": hamming,
        "roc_auc": roc_auc,
    }
```

Лістінг Г.4 – Функція оцінки моделі на етапі валідації

```
def get_tokenizer(model_name, model_max_length = 512):
    tokenizer   =   AutoTokenizer.from_pretrained(model_name,   use_fast=True,
add_prefix_space=True)
    tokenizer.model_max_length = model_max_length
    return tokenizer
```

Лістінг Г.5 – Функція конфігурації токенізатора

```
def get_model(model_name, label2id, id2label):
    model = AutoModelForTokenClassification.from_pretrained(
      model_name,
      num_labels=len(label2id),
      id2label=id2label,
      label2id=label2id
    )
    model.config.id2label = id2label
    model.config.label2id = label2id
    device = "cuda" if torch.cuda.is_available() else "cpu"
    model.to(device)
    return model
```

Лістінг Г.6 – Функція конфігурації моделі

```
def build_trainer(model, tokenizer, train_dataset, eval_dataset, results_dir):
    data_collator = DataCollatorForTokenClassification(tokenizer)
    epochs = 10
    eval_batch_size = 8
    batch_size = 8
    learning_rate = 2e-5
    weight_decay = 0.01
    training_args = TrainingArguments(
        output_dir=results_dir,
        num_train_epochs=epochs,
        evaluation_strategy="epoch",
        per_device_train_batch_size=batch_size,
        per_device_eval_batch_size=eval_batch_size,
        learning_rate=learning_rate,
        weight_decay=weight_decay,
    )
    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=train_dataset,
        eval_dataset=eval_dataset,
        tokenizer=tokenizer,
        data_collator=data_collator,
        compute_metrics=compute_metrics,
    )
    return trainer
```

Лістінг Г.7 – Функція конфігурації трейнера для моделі

```
import sys
import os
project_root = os.path.abspath(os.path.join("./src"))
sys.path.insert(0, project_root)
import dream_detection_classification_helpers.helpers.detection as detection
from   dream_detection_classification_helpers.helpers   import   read_dataset,
get_label_dicts
```

Лістінг Г.8 – Приклад навчання моделі DistillBERT для завдання виявлення онейрологічного образу

```
model_name = "distilbert/distilbert-base-multilingual-cased"
dataset_path  =  "data/personal/personal-dataset-10-21-24/detection-dataset-3-
labels.hf"
results_dir = f"./results/DISTILBERT/distilbert-multilang-cased-detection"
tokenizer = detection.get_tokenizer(model_name)
dataset = read_dataset(dataset_path)
train_dataset, eval_dataset = detection.process_dataset(dataset, tokenizer)
id2label, label2id = get_label_dicts(dataset)
model = detection.get_model(model_name, label2id, id2label)
trainer = detection.build_trainer(model, tokenizer, train_dataset, eval_dataset,
results_dir)
predictions = trainer.predict(eval_dataset)
print(predictions.metrics)
trainer.train()
trainer.save_model(f"./results/DISTILBERT/distilbert-multilang-cased-
detection")
```

Лістінг Г.8, аркуш 2 – Приклад навчання моделі DistillBERT для завдання виявлення онейрологічного образу

ДОДАТОК Д
ФУНКЦІЇ ТА ПРИКЛАД ЕКСПЕРИМЕНТУ НАД МОДЕЛЛЮ КЛАСИФІКАЦІЇ

```
def tokenize(examples, tokenizer, max_length=512):
    return tokenizer(examples['text'],
                     truncation=True, padding='max_length',
                     max_length=max_length, add_special_tokens=False)
```

Лістінг Д.1 – Функція токенізації для задачі класифікації онейрологічних образіів

```
def metrics_by_threshold(probabilities, y_true, threshold):
    y_pred = np.zeros(probabilities.shape)
    y_pred[np.where(probabilities>=threshold)] = 1
    f1 = f1_score(y_true, y_pred, average = 'weighted', zero_division=0)
    hamming = hamming_loss(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average = 'weighted',
zero_division=0)
    recall = recall_score(y_true, y_pred, average = 'weighted',
zero_division=0)
    accuracy = accuracy_score(y_true, y_pred)
    metrics = [f1, 1-hamming, precision, recall, accuracy]
    metrics = [x for x in metrics if x != 0]
    return {
        "f1": f1,
        "hamming": hamming,
        "precision": precision,
        "recall": recall,
        "accuracy": accuracy,
        "overall": statistics.harmonic_mean(metrics),
        "threshold": threshold
    }
```

Лістінг Д.2 – Функція обчислень метрик при різних порогах

```
def find_best_metrics_by_threshold(probabilities, y_true):
   best_metrics = None
   for threshold in np.arange(0.0, 1.0, 0.05, dtype=float):
        metrics = metrics_by_threshold(probabilities, y_true, threshold)
        if best_metrics is None or metrics["overall"] >
best_metrics["overall"]:
            best_metrics = metrics
   return best_metrics
def multi_labels_metrics(predictions, labels):
  probs = torch.nn.functional.softmax(torch.Tensor(predictions), dim=-1)
  y_true = labels
  roc_auc = roc_auc_score(y_true, probs, average = 'weighted',
multi_class='ovo')
  best_metrics = find_best_metrics_by_threshold(probs, y_true)
```

Лістінг Д.3 – Функція обчислень метрик

```
  metrics = {
      "roc_auc": roc_auc,
      "hamming_loss": best_metrics["hamming"],
      "f1": best_metrics["f1"],
      "precision": best_metrics["precision"],
      "recall": best_metrics["recall"],
      "accuracy": best_metrics["accuracy"],
      "best_threshold": best_metrics["threshold"]
  }
  return metrics
```

Лістінг Д.3, аркуш 2 – Функція обчислень метрик

```
def compute_metrics(p:EvalPrediction):
  preds = p.predictions[0] if isinstance(p.predictions, tuple) else
p.predictions
  result = multi_labels_metrics(predictions=preds,
                                labels=p.label_ids)
  return result
```

Лістінг Д.4 – Функція обчислень метрик на основі виходів моделі (логітів)

```
def get_tokenizer(model_name, model_max_length = 512):
    tokenizer = AutoTokenizer.from_pretrained(model_name, use_fast=True,
add_prefix_space=True)
    tokenizer.model_max_length = model_max_length
    return tokenizer
```

Лістінг Д.5 – Функція створення токенізатора для класифікації тексту

```
def get_model(model_name, label2id, id2label):
    model = AutoModelForSequenceClassification.from_pretrained(
        model_name,
        num_labels=len(label2id),
        id2label=id2label,
        label2id=label2id,
        problem_type="multi_label_classification"
    )
    model.config.id2label = id2label
    model.config.label2id = label2id
    device = "cuda" if torch.cuda.is_available() else "cpu"
    model.to(device)
    return model
```

Лістінг Д.6 – Функція створення моделі для класифікації образів

```
def build_trainer(model, tokenizer, train_dataset, eval_dataset, results_dir):
    data_collator = DataCollatorWithPadding(tokenizer, padding='longest')
    epochs = 10
```

Лістінг Д.7 – Функція створення трейнера для моделі класифікації онейрологічних образів

```
    eval_batch_size = 8
    batch_size = 8
    learning_rate = 2e-5
    weight_decay = 0.01
    training_args = TrainingArguments(
        output_dir=results_dir,
        num_train_epochs=epochs,
        evaluation_strategy="epoch",
        per_device_train_batch_size=batch_size,
        per_device_eval_batch_size=eval_batch_size,
        learning_rate=learning_rate,
        weight_decay=weight_decay,
    )
    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=train_dataset,
        eval_dataset=eval_dataset,
        tokenizer=tokenizer,
        data_collator=data_collator,
        compute_metrics=compute_metrics,
    )
    return trainer
```

Лістінг Д.7, аркуш 2 – Функція створення трейнера для моделі класифікації онейрологічних образів

```
import sys
import os
project_root = os.path.abspath(os.path.join("./src"))
sys.path.insert(0, project_root)
import dream_detection_classification_helpers.helpers.classification as classification
from dream_detection_classification_helpers.helpers import read_dataset, get_label_dicts
from importlib import reload
reload(classification)
model_name = "google-bert/bert-base-multilingual-cased"
dataset_path = "data/personal/personal-dataset-10-21-24/text-classification-dataset.hf"
results_dir = f"./results/BERT/bert-multilang-cased-classification"
tokenizer = classification.get_tokenizer(model_name)
dataset = read_dataset(dataset_path)
train_dataset, eval_dataset = classification.process_dataset(dataset, tokenizer)
id2label, label2id = get_label_dicts(dataset)
model = classification.get_model(model_name, label2id, id2label)
trainer = classification.build_trainer(model, tokenizer, train_dataset, eval_dataset, results_dir)
trainer.train()
trainer.save_model(f"./results/BERT/bert-multilang-cased-classification")
```

Лістінг Д.8 – Тренування моделі BERT для завдання класифікації онейрологічних образів

ДОДАТОК Е
ФУНКЦІЇ ТА ПРИКЛАД ЕКСПЕРИМЕНТУ НАД ПОВНОЦІННОЮ МОДЕЛЛЮ

```
class DreamMomentCatTask:
    def __init__(
        self,
        labels: List[str],
        template: str = DEFAULT_DDC_TEMPLATE,
        description: Optional[str] = None,
        label_definitions: Optional[Dict[str, str]] = None,
        prompt_examples: Optional[List[Dict[str, Any]]] = None
    ):
        self.labels = labels
        self.template = Template(template)
        self.description = description
        self.label_definitions = label_definitions
        self.prompt_examples = prompt_examples
        logger.debug("DreamMomentCatTask initialized")
        logger.debug(self.template)

    def generate_prompts(self, docs: Iterable[Doc]) -> Iterable[str]:
        for doc in docs:
            prompt = self.template.render(
                text=doc.text,
                labels=self.labels,
                description=self.description,
                label_definitions=self.label_definitions,
                prompt_examples=self.prompt_examples
            )
            yield prompt
    def parse_responses(
        self, docs: Iterable[Doc], responses: Iterable[str]
    ) -> Iterable[Doc]:
        parsed_responses = parse_responses(responses, self)

        for doc, moments in zip(docs, parsed_responses):
            doc.spans["dream_moments"] = []
            for moment in moments:
                try:
                    start = doc.text.index(moment["text"])
                    end = start + len(moment["text"])
                    span = doc.char_span(start, end)
                    span._.can_classify = moment["can_classify"]
                    span._.labels = moment["labels"]
                    doc.spans["dream_moments"].append(span)
                except ValueError:
                    continue
            yield doc
```

Лістінг Е.1 – Завдання для виявлення та класифікації онейрологічного образу

```
def template_to_prompt(text: str, labels: List[str], template: str =
DEFAULT_DDC_TEMPLATE, description: Optional[str] = None, label_definitions:
Optional[Dict[str, str]] = None, prompt_examples: Optional[List[Dict[str,
Any]]] = None) -> str:
    return Template(template).render(text=text, labels=labels,
description=description,                label_definitions=label_definitions,
prompt_examples=prompt_examples
    )
```

Лістінг Е.2 – Функція для створення промпту із шаблону

```
@registry.llm_tasks("DreamMomentCatTask.v1")
def make_dream_moment_cat_task(
    labels: List[str],
    template: str = DEFAULT_DDC_TEMPLATE,
    description: Optional[str] = None,
    label_definitions: Optional[Dict[str, str]] = None,
    prompt_examples: Optional[List[Dict[str, Any]]] = None
) -> DreamMomentCatTask:
    return DreamMomentCatTask(
        labels=labels,
        template=template,
        description=description,
        label_definitions=label_definitions,
        prompt_examples=prompt_examples
    )
```

Лістінг Е.3 – Функція для реєстрації завдання

```
def parse_responses(responses: List[str], task) -> List[List[Dict[str, Any]]]:
    #print("Processing responses")
    #print(responses)
    parsed_responses = []
    for response in responses:
        moments = []
        for message in response:
            for line in message.split("\n"):
                #print("Processing line")
                #print(line, type(line))
                match = re.match(r'(\d+)\. (.*) \| (True|False) \| (.*)',
line.strip())
                if match:
                    index, text, can_classify, labels = match.groups()
                    does_can_classify = can_classify == "True"
                    moment = {
                        "text": text.strip(),
                        "can_classify": does_can_classify,
                        "labels": []
                    }
                    if does_can_classify:
                        labels = [label.strip() for label in labels.split(",")]
                        moment["labels"] = [label for label in labels if
label != "==NONE=="]
                    moments.append(moment)
        parsed_responses.append(moments)
    return parsed_responses
```

Лістінг Е.4 – Функція для обробки запиту

За причиною відсутності у бібліотеці інтерфейсу для взаємодії з API LLM Claude 3 – 3.5, його розроблено самотужки на основі оригінального класу-обгортки бібліотеки.

```
class Endpoints(str, Enum):
    MESSAGES = "https://api.anthropic.com/v1/messages"

class AnthropicV3(Anthropic):
   def      __call__(self,      prompts:      Iterable[Iterable[str]])      ->
Iterable[Iterable[str]]:
      headers = {
            **self._credentials,
            "model": self._name,
            "anthropic-version": self._config.get("anthropic-version", "2023-
06-01"),
            "Content-Type": "application/json",
        }
      all_api_responses: List[List[str]] = []

      for prompts_for_doc in prompts:
          api_responses: List[str] = []
          prompts_for_doc = list(prompts_for_doc)

          def _request(json_data: Dict[str, Any]) -> Dict[str, Any]:
              r = self.retry(
                  call_method=requests.post,
                  url=self._endpoint,
                  headers=headers,
                  json={**json_data, **self._config, "model": self._name},
                  timeout=self._max_request_time,
              )
              try:
                  r.raise_for_status()
              except HTTPError as ex:
                  res_content = srsly.json_loads(r.content.decode("utf-8"))
                  # Include specific error message in exception.
                  error = res_content.get("error", {})
                  error_msg = f"Request to Anthropic API failed: {error}"
                  if error["type"] == "not_found_error":
                      error_msg  +=  f".  Ensure  that  the  selected  model
({self._name}) is supported by the API."
                  raise ValueError(error_msg) from ex
              response = r.json()
              # c.f. https://console.anthropic.com/docs/api/errors
              if "error" in response:
                  if self._strict:
                      raise ValueError(f"API call failed: {response}.")
                  else:
                      assert isinstance(prompts_for_doc, Sized)
                      return {
                          "error":       [srsly.json_dumps(response)]       *
len(prompts_for_doc)
                      }
              return response
```

Лістінг Е.5 – Клас-обгортка API для бібліотеки spacyLLM

```
            # Anthropic API currently doesn't accept batch prompts, so we're
making
            # a request for each iteration. This approach can be prone to rate
limit
            # errors. In practice, you can adjust _max_request_time so that the
            # timeout is larger.
            responses = [
                _request(
                    {
                        "system": f"Please respond strictly according to the
first userprompt.",
                        "messages": [
                            {
                                "role": "user",
                                "content": f"{prompt}"
                            }
                        ]
                    }
                )
                for prompt in prompts_for_doc
            ]

            for response in responses:
                if "content" in response:
                    text = ""
                    for content in response["content"]:
                        if content["type"] == "text":
                            text += content["text"]
                    api_responses.append(text)
                else:
                    api_responses.append(srsly.json_dumps(response))

            assert len(api_responses) == len(prompts_for_doc)
            all_api_responses.append(api_responses)

        return all_api_responses
```

Лістінг Е.5, аркуш 2 – Клас-обгортка API для бібліотеки spacyLLM

```
def load_dataset(dataset_path: str):
    labels_parquet_path = os.path.join(dataset_path, 'labels_dataset.parquet')
    labels_df = pl.read_parquet(labels_parquet_path)
    dreams_json_path = os.path.join(dataset_path, 'dreams_dataset.json')
    with open(dreams_json_path, 'r') as f:
        dreams_data = json.load(f)
    return labels_df, dreams_data
```

Лістінг Е.6 – Функція завантаження датасету

```
def metadata(labels_df: pl.DataFrame):
    labels_dict = {i: desc for i, (_, desc) in enumerate(zip(labels_df['label'],
labels_df['description']))}
    id2label = {id: i for i, (id, _) in enumerate(zip(labels_df['id'],
labels_df['label']))}
    return labels_dict, id2label
```

Лістінг Е.7 – Функція створення описових масивів для даних

```
def process_dreams(dreams_data: dict, id2label: Dict[Any, Any]):
    transformed_dream_data: List[Dict[str, Any]] = []
    for dream in dreams_data:
        transformed_dream = dream.copy()
        transformed_moments = []
        for moment in dream['moments']:
            transformed_moment = moment.copy()
            transformed_moment['labels'] = [id2label[label_id] for label_id in
moment['labels'] if label_id in id2label]
            transformed_moments.append(transformed_moment)
        transformed_dream['moments'] = transformed_moments
        transformed_dream_data.append(transformed_dream)
    return transformed_dream_data
```

Лістінг Е.8 – Функція обробки датасету для подальшого використання моделлю

```
def find_minimal_covering_dreams(dreams, labels_length):
    selected_dreams = []
    covered_labels = set()
    remaining_dreams = list(enumerate(dreams))
    while len(covered_labels) < labels_length and remaining_dreams:
        dream_idx, dream = remaining_dreams.pop(0)
        dream_label_set = set()
        for moment in dream['moments']:
            for label in moment['labels']:
                dream_label_set.add(label)
        if dream_label_set - covered_labels:
            selected_dreams.append(dream_idx)
            covered_labels.update(dream_label_set)
    return [dreams[i] for i in selected_dreams]
```

Лістінг Е.9 – Пошук мінімального набору снів, що мають всі типи ознак сну

```
def to_metric_array(predictions, true_moments, metric: Callable[[np.array,
np.array], float]):
    return [metric(predictions[i], true_moments[i]) for i in
range(len(predictions))]
```

Лістінг Е.10 – Функція формування списку метрик, на основі передбачених та істинних даних

```
def filter_zeros(array: list[float]) -> list[float]:
    return [x for x in array if x != 0]
```

Лістінг Е.11 – Функція фільтрації нулів із масиву, для подальшого розрахунку середнього геометричного

```
def process_metric(predictions, true_moments, metric: Callable[[np.array,
np.array], float]):
    metric_arrays = to_metric_array(predictions, true_moments, metric)
    metric_arrays = filter_zeros(metric_arrays)
    return harmonic_mean(metric_arrays)
```

Лістінг Е.12 – Функція створення масивів метрик

```
def harmonic_evaluate_moments(predictions, true_moments):
    accuracy = process_metric(predictions, true_moments, lambda y_pred, y_true:
accuracy_score(y_pred, y_true))
    f1 = process_metric(predictions, true_moments, lambda y_pred, y_true:
f1_score(y_pred, y_true, average='weighted', zero_division=0))
    precision = process_metric(predictions, true_moments, lambda y_pred,
y_true: precision_score(y_pred, y_true, average='weighted', zero_division=0))
    recall = process_metric(predictions, true_moments, lambda y_pred, y_true:
recall_score(y_pred, y_true, average='weighted', zero_division=0))
    _hamming_loss = process_metric(predictions, true_moments, lambda y_pred,
y_true: hamming_loss(y_pred, y_true))

    result = {
        "accuracy": accuracy,
        "precision": precision,
        "recall": recall,
        "mean_f1": harmonic_mean([precision, recall]),
        "f1_mean": f1,
        "hamming_loss": _hamming_loss,
    }
    return result
```

Лістінг Е.13 – Функція обчислень гармонізованих метрик

```
def tokenize_moments(test_data, labels_dict):
   return [
       [{
       "text": tokenize(moment['text']),
       "labels": labels_to_one_hot(moment['labels'], len(labels_dict), lambda
x: int(x))
       } for moment in test_data[i]['moments']
       ] for i in range(len(test_data)) ]
```

Лістінг Е.14 – Функція токенізації ознак сну

```
def tokenize_result(result: list[Doc], labels_dict):
   return [[
       {   "text": tokenize(span.text),
           "labels": labels_to_one_hot(span._.labels, len(labels_dict), lambda
x: int(x))
       } for span in doc.spans["dream_moments"]] for doc in result]
```

Лістінг Е.15 – Функція токенізації результатів від LLM

Схожа функція використовується для визначення класів слів при підготовці датасету для виявлення онейрологічних образів.

```
def is_span_equal(words, start_idx, span_words, threshold=0.8):
    incorrect_tokens = 0
    span_words_length = len(span_words)
    for j in range(start_idx, len(words)):
        if j - start_idx >= span_words_length:
            break
        if  words[j] != span_words[j - start_idx]:
            incorrect_tokens += 1
    return incorrect_tokens / span_words_length <= 1 - threshold
```

Лістінг Е.15 – Функція пошуку послідовностей в тексті

```
def conform_spans(words, spans, labels_dict, threshold=0.8):
    conformed_span = [[0] * len(labels_dict)] * len(words)
    spans_queue = [{"words": span['text'], "labels": span['labels']} for span
in spans]
    i = 0
    while i < len(words):
        if not spans_queue:
            break
        existing_span,   idx   =   next(((mt,   idx)   for   idx,   mt   in
enumerate(spans_queue)
                                    if words[i] in mt['words'] and
                                        is_span_equal(words,  i,  mt['words'],
threshold=threshold)), (None, -1))
        if(existing_span):
            span_words_length = len(existing_span['words'])
            conformed_span[i:i+span_words_length]  =  [existing_span['labels']]
* span_words_length
            spans_queue.pop(idx)
            i += span_words_length - 1
            continue
        i += 1

    return conformed_span
```

Лістінг Е.15 – Функція назначення класів словам виділених ознак сну

```
def evaluate_result(test_data, result, labels_dict):
    tokenized_texts = [tokenize(dream['dream']) for dream in test_data]
    all_true_spans = tokenize_moments(test_data, labels_dict)
    all_predicted_spans = tokenize_result(result, labels_dict)
    conformed_true_spans          =          [conform_spans(tokenized_texts[i],
all_true_spans[i], labels_dict) for i in range(len(test_data))]
    conformed_predicted_spans       =       [conform_spans(tokenized_texts[i],
all_predicted_spans[i], labels_dict) for i in range(len(test_data))]
    return                harmonic_evaluate_moments(conformed_predicted_spans,
conformed_true_spans)
```

Лістінг Е.15 – Функція фінальної оцінки результатів від LLM

```
from spacy_llm.util import assemble
import sys
import os
import random
project_root = os.path.abspath(os.path.join("./src"))
sys.path.insert(0, project_root)
import   dream_detection_classification_helpers.models.anthropic.registry   as
anthropic_registry
import  dream_detection_classification_helpers.tasks.dream_moment_cat.task  as
dmc_task
import  dream_detection_classification_helpers.tasks.dream_moment_cat.registry
as dmc_registry
import dream_detection_classification_helpers.helpers.end_to_end as end_to_end
DATASET_PATH = './data/personal/personal-dataset-10-21-24/llm'
labels_df, dreams_data = end_to_end.load_dataset(DATASET_PATH)
index2label = {i: label for i, label in enumerate(labels_df['label'])}
labels_dict, id2label = end_to_end.metadata(labels_df)
transformed_dream_data = end_to_end.process_dreams(dreams_data, id2label)
prompt_examples_1 = end_to_end.random_frozen_sample()
prompt_examples_2 = end_to_end.full_frozen_sample()

sample = end_to_end.test_sample()
input_texts = [dream['dream'] for dream in sample]
llm_config_path = './data/configs/LLM/claude-3-5-sonnet.cfg'
overrides = {
    "components.llm.task.labels": list(labels_dict.keys()),
    "components.llm.task.label_definitions": labels_dict,
    "components.llm.task.prompt_examples": prompt_examples_1,
}
nlp = assemble(llm_config_path, overrides=overrides)
result = list(nlp.pipe(input_texts))
metrics = end_to_end.evaluate_result(sample, result, labels_dict)
processed_metrics = end_to_end.metrics_to_df(metrics)
print(processed_metrics)
```

Лістінг Е.16 – Приклад експерименту для моделі Claude 3.5 Sonnet з прикладом у вигляді 10 випадкових записів сну і 10 тестових випадкових снах

ДОДАТОК Ж
ДАНІ ДЛЯ ОПИТУВАННЯ

Сон №1 «Автомобіль»

Я їду на автомобілі по безкінечній дорозі, що веде через мальовничі пейзажі. Дорога звивається через гори та долини, а навколо мене розкидані квітучі поля та густі ліси. Відчуття свободи та пригоди наповнює моє серце, і я насолоджуюся кожною миттю цієї подорожі.

Сон №2 «Зорі»

Я лежу на траві під відкритим небом, дивлячись на зорі. Небо вкрите мільйонами яскравих зірок, що мерехтять і створюють неймовірні візерунки. Я відчуваю себе частиною всесвіту, і кожна зірка здається мені близькою та рідною. Це момент спокою та гармонії, коли я відчуваю зв'язок з космосом.

Сон №3 «Дочка»

Я бачу свою дочку, яка грається в саду. Вона сміється та бігає серед квітів, її очі сяють радістю. Я відчуваю гордість та любов, спостерігаючи за її безтурботністю та щастям. Вона моя маленька принцеса, і я готовий зробити все, щоб вона завжди була щасливою.

Сон №4 «Землетрус»

Я відчуваю, як земля під моїми ногами починає трястися. Будинки навколо мене хитаються, і я чую гучний гуркіт. Люди в паніці вибігають на вулиці, шукаючи безпечне місце. Я намагаюся зберігати спокій і допомагати іншим, але страх і невизначеність охоплюють мене. Це момент випробування, коли я повинен знайти силу та мужність, щоб подолати цю стихію.

Сон №5 «Казка»

Я опиняюся в чарівному лісі, де дерева говорять, а тварини співають. Я зустрічаю добру фею, яка дарує мені чарівний амулет. З його допомогою я можу здійснювати бажання та допомагати іншим. Я вирушаю в подорож, повну пригод та магії, де кожен крок відкриває нові таємниці та дивовижні

відкриття. Це казковий світ, де все можливо, і я відчуваю себе героєм цієї неймовірної історії.

Сон №6 «Комп’ютер»

Я сиджу за комп'ютером, занурений у віртуальний світ. Екран переді мною сяє яскравими кольорами, і я відчуваю, як кожен піксель оживає. Я програмую нову гру, і кожен рядок коду приносить мені задоволення. Це момент творчості та натхнення, коли я створюю щось нове та захоплююче.

Сон №7 «Полуниця»

Я знаходжуся на полуничному полі, де стиглі ягоди виблискують на сонці. Я збираю полуницю, відчуваючи її солодкий аромат. Кожна ягода здається ідеальною, і я насолоджуюся її смаком. Я відчуваю зв'язок з природою.

Сон №8 «Хатсуне-Міку»

Я на концерті Хатсуне-Міку, де вона виступає на сцені. Її голос лунає через весь зал, і я відчуваю, як музика проникає в моє серце. Я співаю разом з нею, і кожна нота наповнює мене енергією і я відчуваю себе частиною чогось великого.

Сон №9 «Болгари»

Я подорожую до Болгарії, де зустрічаю доброзичливих людей. Вони запрошують мене на традиційний обід, де я смакую смачні страви. Я відчуваю гостинність та тепло, і кожен момент цієї подорожі наповнює мене вдячністю. Це момент культурного обміну та нових відкриттів.

Сон №10 «Нирки»

Я знаходжуся в лікарні, де мені роблять операцію на нирках. Лікарі навколо мене працюють зосереджено, і я відчуваю їхню професійність. Я відчуваю страх, але також надію на одужання. Це момент випробування та сили, коли я повинен довіритися медичному персоналу та вірити в краще майбутнє.

Напівжирним шрифтом відмічено розбіжності між відповідями моделей, що використано для формування опитування.

Таблиця Ж.1. Виявлені ознаки сну різними моделями.

Ознаки моделей виявлення	
Claude Sonnet	DistilBERT
Сон №1 «Автомобіль»	
Я їду на автомобілі по безкінечній дорозі	Я їду на автомобілі по безкінечній дорозі
	що веде через мальовничі пейзажі
Дорога звивається через гори та долини, а навколо мене розкидані квітучі поля та густі ліси	**Дорога звивається через гори та долини**
Відчуття свободи та пригоди наповнює моє серце	**Відчуття свободи та пригоди наповнює моє серце**
	я насолоджуюся кожною миттю цієї подорожі.
Сон №2 «Зорі»	
Я лежу на траві під відкритим небом	**Я лежу на траві під відкритим небом, дивлячись на зорі**
Небо вкрите мільйонами яскравих зірок, що мерехтять і створюють неймовірні візерунки	Небо вкрите мільйонами яскравих зірок, що мерехтять і створюють неймовірні візерунки
Я відчуваю себе частиною всесвіту, і кожна зірка здається мені близькою та рідною	**Я відчуваю себе частиною всесвіту**
Це момент спокою та гармонії, коли я відчуваю зв'язок з космосом	Це момент спокою та гармонії, коли я відчуваю зв'язок з космосом.

Продовження таблиці Ж.1. Виявлені ознаки сну різними моделями.

Ознаки моделей виявлення	
Claude Sonnet	DistilBERT
Сон №3 «Дочка»	
Я бачу свою дочку, яка грається в саду	Я бачу свою дочку, яка грається в саду
Вона сміється та бігає серед квітів, її очі сяють радістю	Вона сміється та бігає серед квітів, її очі сяють радістю.
Я відчуваю гордість та любов, спостерігаючи за її безтурботністю та щастям	Я відчуваю гордість та любов, спостерігаючи за її безтурботністю та щастям.
Вона моя маленька принцеса	Вона моя маленька принцеса,
	я готовий зробити все, щоб вона завжди була щасливою.
Сон №4 «Землетрус»	
Я відчуваю, як земля під моїми ногами починає трястися.	Я відчуваю, як земля під моїми ногами починає трястися.
Будинки навколо мене хитаються, і я чую гучний гуркіт.	**Будинки навколо мене хитаються, і** **я чую гучний гуркіт.**
Люди в паніці вибігають на вулиці, шукаючи безпечне місце.	Люди в паніці вибігають на вулиці, шукаючи безпечне місце.
Я намагаюся зберігати спокій і допомагати іншим, але страх і невизначеність охоплюють мене.	Я намагаюся зберігати спокій і допомагати іншим, але страх і невизначеність охоплюють мене.

Продовження таблиці Ж.1. Виявлені ознаки сну різними моделями.

Ознаки моделей виявлення	
Claude Sonnet	DistilBERT
Сон №5 «Казка»	
Я опиняюся в чарівному лісі	Я опиняюся в чарівному лісі
дерева говорять, а тварини співають	де дерева говорять, а тварини співають
Я зустрічаю добру фею	**Я зустрічаю добру фею, яка дарує мені чарівний амулет**
яка дарує мені чарівний амулет	
З його допомогою я можу здійснювати бажання та допомагати іншим	З його допомогою я можу здійснювати бажання та допомагати іншим
Я вирушаю в подорож, повну пригод та магії	Я вирушаю в подорож, повну пригод та магії
кожен крок відкриває нові таємниці та дивовижні відкриття	
Це казковий світ, де все можливо, і я відчуваю себе героєм цієї неймовірної історії	**Це казковий світ, де все можливо,**
	я відчуваю себе героєм цієї неймовірної історії.
Сон №6 «Комп’ютер»	
Я сиджу за комп'ютером, занурений у віртуальний світ	Я сиджу за комп'ютером, занурений у віртуальний світ
Екран переді мною сяє яскравими кольорами, і я відчуваю, як кожен піксель оживає	**Екран переді мною сяє яскравими кольорами**
	я відчуваю, як кожен піксель оживає
Я програмую нову гру, і кожен рядок коду приносить мені задоволення	**Я програмую нову гру**
	кожен рядок коду приносить мені задоволення
Це момент творчості та натхнення, коли я створюю щось нове та захоплююче	Це момент творчості та натхнення, коли я створюю щось нове та захоплююче.

Продовження таблиці Ж.1. Виявлені ознаки сну різними моделями.

Ознаки моделей виявлення	
Claude Sonnet	DistilBERT
Сон №7 «Полуниця»	
Я знаходжуся на полуничному полі	Я знаходжуся на полуничному полі
стиглі ягоди виблискують на сонці	де стиглі ягоди виблискують на сонці
Я збираю полуницю, відчуваючи її солодкий аромат	Я збираю полуницю, відчуваючи її солодкий аромат
Кожна ягода здається ідеальною	Кожна ягода здається ідеальною,
	я насолоджуюся її смаком
Я відчуваю зв'язок з природою	Я відчуваю зв'язок з природою.
Сон №8 «Хатсуне-Міку»	
Я на концерті Хатсуне-Міку	Я на концерті Хатсуне-Міку
вона виступає на сцені	
Її голос лунає через весь зал	Її голос лунає через весь зал
я відчуваю, як музика проникає в моє серце	я відчуваю, як музика проникає в моє серце
Я співаю разом з нею, і кожна нота наповнює мене енергією	**Я співаю разом з нею**
	кожна нота наповнює мене енергією
я відчуваю себе частиною чогось великого	я відчуваю себе частиною чогось великого.
Сон №9 «Болгари»	
Я подорожую до Болгарії	Я подорожую до Болгарії,
зустрічаю доброзичливих людей. Вони запрошують мене на традиційний обід	**де зустрічаю доброзичливих людей**
	Вони запрошують мене на традиційний обід
я смакую смачні страви	
Я відчуваю гостинність та тепло, і кожен момент цієї подорожі наповнює мене вдячністю	**Я відчуваю гостинність та тепло**
	кожен момент цієї подорожі наповнює мене вдячністю.
Це момент культурного обміну та нових відкриттів	Це момент культурного обміну та нових відкриттів.

Продовження таблиці Ж.1. Виявлені ознаки сну різними моделями.

Моделі виявлення	
Claude Sonnet	DistilBERT
Сон №10 «Нирки»	
Я знаходжуся в лікарні, де мені роблять операцію на нирках	**Я знаходжуся в лікарні**
	мені роблять операцію на нирках
Лікарі навколо мене працюють зосереджено	Лікарі навколо мене працюють зосереджено
	я відчуваю їхню професійність
Я відчуваю страх, але також надію на одужання	Я відчуваю страх, але також надію на одужання
Це момент випробування та сили, коли я повинен довіритися медичному персоналу та вірити в краще майбутнє	Це момент випробування та сили, коли я повинен довіритися медичному персоналу та вірити в краще майбутнє.

Таблиця Ж.2. Класифіковані ознаки сну різними моделями.

Ознака сну	Модель класифікації	
	Claude Sonnet	RoBERTa
Сон №1 «Автомобіль»		
Я їду на автомобілі по безкінечній дорозі	Моє (Місцезнаходження)	Моє (Місцезнаходження)
Дорога звивається через гори та долини, а навколо мене розкидані квітучі поля та густі ліси	Місця (Форма)	Об'єктів (Місцезнаходження), Моє (Місцезнаходження)
Відчуття свободи та пригоди наповнює моє серце	Емоції (Внутрішня обізнаність)	Відчуття (Внутрішня обізнаність), Сприйняття (Внутрішня обізнаність), Ситуація (Обставини)
Сон №2 «Зорі»		
Я лежу на траві під відкритим небом	Моє (Місцезнаходження)	Моє (Місцезнаходження)
Небо вкрите мільйонами яскравих зірок, що мерехтять і створюють неймовірні візерунки	Сприйняття (Внутрішня обізнаність), Об'єктів (Форма)	Об'єктів (Форма)
Я відчуваю себе частиною всесвіту, і кожна зірка здається мені близькою та рідною	Емоції (Внутрішня обізнаність), Думки (Внутрішня обізнаність)	Відчуття (Внутрішня обізнаність)
Це момент спокою та гармонії, коли я відчуваю зв'язок з космосом	Емоції (Внутрішня обізнаність), Думки (Внутрішня обізнаність)	Відчуття (Внутрішня обізнаність), Сприйняття (Внутрішня обізнаність)

Продовження таблиці Ж.2. Класифіковані ознаки сну різними моделями.

Ознака сну	Модель класифікації	
	Claude Sonnet	RoBERTa
Сон №3 «Дочка»		
Я бачу свою дочку, яка грається в саду	Ситуація (Обставини)	Істот (Місцезнаходження)
Вона сміється та бігає серед квітів, її очі сяють радістю	Сприйняття (Внутрішня обізнаність), Емоції (Внутрішня обізнаність)	Істот (Дії)
Я відчуваю гордість та любов, спостерігаючи за її безтурботністю та щастям	Емоції (Внутрішня обізнаність)	Відчуття (Внутрішня обізнаність)
Вона моя маленька принцеса	Роль істот (Обставини)	Істот (Місцезнаходження)
Сон №4 «Землетрус»		
Я відчуваю, як земля під моїми ногами починає трястися.	Об'єктів (Дії)	Відчуття (Внутрішня обізнаність)
Будинки навколо мене хитаються, і я чую гучний гуркіт.	Об'єктів (Дії), Сприйняття (Внутрішня обізнаність)	Сприйняття (Внутрішня обізнаність)
Люди в паніці вибігають на вулиці, шукаючи безпечне місце.	Істот (Дії)	Істот (Дії)
Я намагаюся зберігати спокій і допомагати іншим, але страх і невизначеність охоплюють мене.	Емоції (Внутрішня обізнаність)	Відчуття (Внутрішня обізнаність), Мої (Дії)
Це момент випробування, коли я повинен знайти силу та мужність, щоб подолати цю стихію.	Думки (Внутрішня обізнаність)	Ситуація (Обставини)

Продовження таблиці Ж.2. Класифіковані ознаки сну різними моделями.

Ознака сну	Модель класифікації	
	Claude Sonnet	RoBERTa
Сон №5 «Казка»		
Я опиняюся в чарівному лісі	Моє (Місцезнаходження)	Моє (Місцезнаходження)
дерева говорять, а тварини співають	Істот (Дії), Об'єктів (Дії)	Істот (Дії)
Я зустрічаю добру фею	Істот (Місцезнаходження)	Мої (Дії), Моє (Місцезнаходження)
яка дарує мені чарівний амулет	Ситуація (Обставини)	Відчуття (Внутрішня обізнаність), Істот (Форма)
З його допомогою я можу здійснювати бажання та допомагати іншим	Думки (Внутрішня обізнаність), Мої (Дії)	Мої (Дії)
Я вирушаю в подорож, повну пригод та магії	Ситуація (Обставини)	Мої (Дії)
кожен крок відкриває нові таємниці та дивовижні відкриття	Ситуація (Обставини)	Ситуація (Обставини)
Це казковий світ, де все можливо, і я відчуваю себе героєм цієї неймовірної історії	Моя роль (Обставини), Ситуація (Обставини)	Відчуття (Внутрішня обізнаність)

ДОДАТОК К
ПЕРЕЛІК ПИТАНЬ ОПИТУВАННЯ

Розділ №1 визначення ознак сну

Сон №1 "Автомобіль"

1.1. Чи виділили би ви у якості ознаки фразу "що веде через мальовничі пейзажі"? – Так чи ні

1.2. Чи об'єднали би ви ці ознаки в одну? – Так чи ні

1) Відчуття свободи та пригоди наповнює моє серце

2) я насолоджуюся кожною миттю цієї подорожі.

Сон №2 "Зорі"

1.3. Який варіант ознаки вам подобається більше ? Перший, другий або жодний.

1) Я лежу на траві під відкритим небом

2) Я лежу на траві під відкритим небом, дивлячись на зорі

1.4. Який варіант ознаки вам подобається більше ? Перший, другий або жодний.

1) Я відчуваю себе частиною всесвіту, і кожна зірка здається мені близькою та рідною

2) Я відчуваю себе частиною всесвіту

Сон №3 "Дочка"

1.5. Чи виділили би ви у якості ознаки фразу " я готовий зробити все, щоб вона завжди була щасливою. " ? – Так чи ні

Сон №4 "Землетрус"

1.6. Чи об'єднали би ви ці ознаки в одну? – Так чи ні

1) Будинки навколо мене хитаються, і

2) я чую гучний гуркіт.

Сон №5 "Казка"

1.7. Чи об'єднали би ви ці ознаки в одну? – Так чи ні

1) Я зустрічаю добру фею

2) яка дарує мені чарівний амулет

1.8. Чи виділили би ви у якості ознаки фразу "кожен крок відкриває нові таємниці та дивовижні відкриття" ? – Так чи ні

1.9. Чи об'єднали би ви ці ознаки в одну? – Так чи ні

1) Це казковий світ, де все можливо,

2) я відчуваю себе героєм цієї неймовірної історії.

Сон №6 "Комп'ютер"

1.10. Чи об'єднали би ви ці ознаки в одну? – Так чи ні

1) Екран переді мною сяє яскравими кольорами

2) я відчуваю, як кожен піксель оживає

1.11. Чи об'єднали би ви ці ознаки в одну? – Так чи ні

1) Я програмую нову гру

2) кожен рядок коду приносить мені задоволення

Сон №7 "Полуниця"

1.12. Чи виділили би ви у якості ознаки фразу "я насолоджуюся її смаком" ? – Так чи ні

Сон №8 "Хатсуне-Міку"

1.13. Чи виділили би ви у якості ознаки фразу "вона виступає на сцені" ? – Так чи ні

1.14. Чи об'єднали би ви ці ознаки в одну? – Так чи ні

1) Я співаю разом з нею

2) кожна нота наповнює мене енергією

Сон №9 "Болгари"

1.15. Чи об'єднали би ви ці ознаки в одну? – Так чи ні

1) де зустрічаю доброзичливих людей

2) Вони запрошують мене на традиційний обід

1.16. Чи виділили би ви у якості ознаки фразу "я смакую смачні страви" ? – Так чи ні

1.17. Чи об'єднали би ви ці ознаки в одну? – Так чи ні

1) Я відчуваю гостинність та тепло

2) кожен момент цієї подорожі наповнює мене вдячністю.

Сон №10 "Нирки"

1.18. Чи об'єднали би ви ці ознаки в одну? – Так чи ні

1) Я знаходжуся в лікарні

2) мені роблять операцію на нирках

1.19. Чи виділили би ви у якості ознаки фразу "я відчуваю їхню професійність" ? – Так чи ні

Розділ №2 классифікація ознак сну

Сон №1 "Автомобіль"

2.1. Чи здається вам класифікація ознаки доречною ? – Так чи ні

Ознака:

Я їду на автомобілі по безкінечній дорозі

Класифікація:

Моє (Місцезнаходження)

2.2. Класифікація ознаки якої моделі здається вам більш доречною ? – Перша, друга, обидві або жодна

Ознака:

Дорога звивається через гори та долини, а навколо мене розкидані квітучі поля та густі ліси

Класифікація:

1) Місця (Форма)
2) Об'єктів (Місцезнаходження), Моє (Місцезнаходження)

2.3. Класифікація ознаки якої моделі здається вам більш доречною ? – Перша, друга, обидві або жодна

Ознака:

Відчуття свободи та пригоди наповнює моє серце

Класифікація:

1) Емоції (Внутрішня обізнаність)
2) Відчуття (Внутрішня обізнаність), Сприйняття (Внутрішня обізнаність), Ситуація (Обставини)

Сон №2 "Зорі"

2.4. Чи здається вам класифікація ознаки доречною ? – Так чи ні

Ознака:

Я лежу на траві під відкритим небом

Класифікація:

Моє (Місцезнаходження)

2.5. Класифікація ознаки якої моделі здається вам більш доречною ? – Перша, друга, обидві або жодна

Ознака:

Небо вкрите мільйонами яскравих зірок, що мерехтять і створюють неймовірні візерунки

Класифікація:

1) Сприйняття (Внутрішня обізнаність), Об'єктів (Форма)
2) Об'єктів (Форма)

2.6. Класифікація ознаки якої моделі здається вам більш доречною ? – Перша, друга, обидві або жодна

Ознака:

Я відчуваю себе частиною всесвіту, і кожна зірка здається мені близькою та рідною

Класифікація:

1) Емоції (Внутрішня обізнаність), Думки (Внутрішня обізнаність)
2) Відчуття (Внутрішня обізнаність)

2.7. Класифікація ознаки якої моделі здається вам більш доречною ? – Перша, друга, обидві або жодна

Ознака:

Це момент спокою та гармонії, коли я відчуваю зв'язок з космосом

Класифікація:

1) Емоції (Внутрішня обізнаність), Думки (Внутрішня обізнаність)
2) Відчуття (Внутрішня обізнаність), Сприйняття (Внутрішня обізнаність)

Сон №3 "Дочка"

2.8. Класифікація ознаки якої моделі здається вам більш доречною ? – Перша, друга, обидві або жодна

Ознака:

Я бачу свою дочку, яка грається в саду

Класифікація:

1) Ситуація (Обставини)

2) Істот (Місцезнаходження)

2.9. Класифікація ознаки якої моделі здається вам більш доречною ? – Перша, друга, обидві або жодна

Ознака:

Вона сміється та бігає серед квітів, її очі сяють радістю

Класифікація:

1) Сприйняття (Внутрішня обізнаність), Емоції (Внутрішня обізнаність)

2) Істот (Дії)

2.10. Класифікація ознаки якої моделі здається вам більш доречною ? – Перша, друга, обидві або жодна

Ознака:

Я відчуваю гордість та любов, спостерігаючи за її безтурботністю та щастям

Класифікація:

1) Емоції (Внутрішня обізнаність)

2) Відчуття (Внутрішня обізнаність)

2.11. Класифікація ознаки якої моделі здається вам більш доречною ? – Перша, друга, обидві або жодна

Ознака:

Вона моя маленька принцеса

Класифікація:

1) Роль істот (Обставини)

2) Істот (Місцезнаходження)

Сон №4 "Землетрус"

2.12. Класифікація ознаки якої моделі здається вам більш доречною ? – Перша, друга, обидві або жодна

Ознака:

Я відчуваю, як земля під моїми ногами починає трястися.

Класифікація:

1) Об'єктів (Дії)
2) Відчуття (Внутрішня обізнаність)

2.13. Класифікація ознаки якої моделі здається вам більш доречною ? – Перша, друга, обидві або жодна

Ознака:

Будинки навколо мене хитаються, і я чую гучний гуркіт.

Класифікація:

1) Об'єктів (Дії), Сприйняття (Внутрішня обізнаність)
2) Сприйняття (Внутрішня обізнаність)

2.14. Чи здається вам класифікація ознаки доречною ? – Так чи ні

Ознака:

Люди в паніці вибігають на вулиці, шукаючи безпечне місце.

Класифікація:

Істот (Дії)

2.15. Класифікація ознаки якої моделі здається вам більш доречною ? – Перша, друга, обидві або жодна

Ознака:

Я намагаюся зберігати спокій і допомагати іншим, але страх і невизначеність охоплюють мене.

Класифікація:

1) Емоції (Внутрішня обізнаність)
2) Відчуття (Внутрішня обізнаність), Мої (Дії)

2.16. Класифікація ознаки якої моделі здається вам більш доречною ? – Перша, друга, обидві або жодна

Ознака:

Це момент випробування, коли я повинен знайти силу та мужність, щоб подолати цю стихію.

Класифікація:

1) Думки (Внутрішня обізнаність)
2) Ситуація (Обставини)

Сон №5 "Казка"

2.17. Чи здається вам класифікація ознаки доречною ? – Так чи ні

Ознака:

Я опиняюся в чарівному лісі

Класифікація:

Моє (Місцезнаходження)

2.18. Класифікація ознаки якої моделі здається вам більш доречною ? – Перша, друга, обидві або жодна

Ознака:

дерева говорять, а тварини співають

Класифікація:

1) Істот (Дії), Об'єктів (Дії)
2) Істот (Дії)

2.19. Класифікація ознаки якої моделі здається вам більш доречною ? – Перша, друга, обидві або жодна

Ознака:

Я зустрічаю добру фею

Класифікація:

Істот (Місцезнаходження)

Мої (Дії), Моє (Місцезнаходження)

2.20. Класифікація ознаки якої моделі здається вам більш доречною ? – Перша, друга, обидві або жодна

Ознака:

яка дарує мені чарівний амулет

Класифікація:

1) Ситуація (Обставини)

2) Відчуття (Внутрішня обізнаність), Істот (Форма)

2.21. Класифікація ознаки якої моделі здається вам більш доречною ? – Перша, друга, обидві або жодна

Ознака:

З його допомогою я можу здійснювати бажання та допомагати іншим

Класифікація:

1) Думки (Внутрішня обізнаність), Мої (Дії)

2) Мої (Дії)

2.22. Класифікація ознаки якої моделі здається вам більш доречною ? – Перша, друга, обидві або жодна

Ознака:

Я вирушаю в подорож, повну пригод та магії

Класифікація:

1) Ситуація (Обставини)

2) Мої (Дії)

2.23. Чи здається вам класифікація ознаки доречною ? – Так чи ні

Ознака:

кожен крок відкриває нові таємниці та дивовижні відкриття

Класифікація:

Ситуація (Обставини)

2.24. Класифікація ознаки якої моделі здається вам більш доречною ? – Перша, друга, обидві або жодна

Ознака: Це казковий світ, де все можливо, і я відчуваю себе героєм цієї неймовірної історії.

Класифікація:

1) Моя роль (Обставини), Ситуація (Обставини)

2) Відчуття (Внутрішня обізнаність)

ДОДАТОК Л
РЕЗУЛЬТАТИ ОПИТУВАННЯ

Таблиця Л.1. Таблиця результатів за статтю респондентів

Статева група	Кількість
Чоловіча	6
Жіноча	6
Не можу сказати	0
Інша	1

Таблиця Л.2. Таблиця результатів за віком респондентів

Вікова група	Кількість
11-20	3
21-30	8
31-40	1
41-50	1
51-60	0
61-70+	0

Таблиця Л.3. Таблиця результатів по секції виявлення

Номер питання	Схильність до моделі		
	Claude 3.5 Sonnet	DistilBERT	Жодна
1.1	5	8	0
1.2	8	5	0
1.3	1	9	3
1.4	10	3	0
1.5	4	9	0
1.6	10	3	0
1.7	4	9	0
1.8	10	3	0
1.9	3	8	0
1.10	5	8	0
1.11	6	7	0
1.12	5	8	0
1.13	7	6	0
1.14	9	4	0
1.15	6	7	0
1.16	7	6	0
1.17	8	5	0
1.18	9	4	0
1.19	3	10	0
Загалом	122	120	3

Таблиця Л.4. Таблиця результатів по секції класифікації

Номер питання	Схильність до моделі			
	Claude 3.5 Sonnet	DistilBERT	Обидві	Жодна
2.1			10	3
2.2	4	6	3	
2.3	2	5	6	
2.4			11	2
2.5		7	5	1
2.6	6	2	5	
2.7	6	3	4	
2.8	5	4	3	1
2.9	7	1	3	2
2.10	6	2	5	
2.11	10		1	2
2.12	7	4	2	
2.13			11	2
2.14			12	1
2.15	3	4	6	
2.16	8	2	3	
2.17			13	
2.18	8	2	2	1
2.19	4	4	2	3
2.20	7	3	2	1
2.21	5	6	1	1
2.22		4	5	4
2.23			8	5
2.24		6	3	4
Загалом	105	61	122	24

ДОДАТОК М

ДОВІДКА О ВПРОВАДЖЕННІ

Виробництво
балансувальних
верстатів з 1973 року

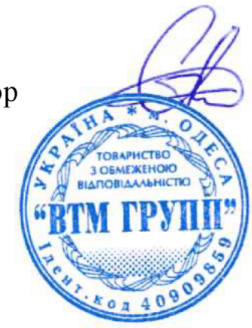
Товариство з обмеженою відповідальністю
«ВТМ ГРУПП»
www.vtm.group

Юридична адреса: 65005, Україна, Одеська обл.,м. Одеса, вул.Бугаївська,43А. Поштова адреса: 65005, Україна, Одеська обл., м. Одесса, а/с 137
р/р UA173071230000026006011140273 в ПАТ «БАНК ВОСТОК», Код ЄДРПОУ 40909859, ІПН 409098515520,
тел/факс (048) 777-06-80, 777-06-86; e-mail: micron@ukr.net

ДОВІДКА

про впровадження

Інформаційна технологія для виявлення та класифікації онейрологічних образів в природньомовному тексті, вдосконалена студентом Одеського національного університету імені І.І. Мечникова Жара Михайла Юрійовича під час виконання дипломної роботи магістра на кафедрі математичного забезпечення комп'ютерних систем, успішно пройшла виробничі випробування в компанії VTM Group та запланована до впровадження.

Генеральний директор 

Сергій ПОЛІЩУК