НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

На правах рукопису
УДК 004.852

До захисту допущено
В.о. зав. кафедри ШІ
_____О.І. Чумаченко
«___»_____2022 р.

Магістерська дисертація

на здобуття ступеня магістра зі спеціальності 122 «Комп'ютерні науки»

на тему: «Застосування методів машинного навчання для виявлення психічних захворювань»

Виконав:
студент ІІ курсу, групи КІ-11мп
Цупрун Ілля Юрійович _____

Керівник:
доцент кафедри ММСА,
к.ф-м.н., доц. Стусь О.В. _____

Рецензент:
доцент кафедри системного проектування
КПІ ім. Ігоря Сікорського, к.т.н., Безносик О. Ю. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент (*підпис*): _____

Київ
2022

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

Рівень вищої освіти — другий (магістерський)
Спеціальність (ОПП) — 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
В.о. зав. кафедри ШІ
_____ О.І. Чумаченко
«___» _____ 2022 р.

ЗАВДАННЯ
на магістерську дисертацію студенту Цупрун Ілля Юрійовичу

1. Тема дисертації: «Застосування методів машинного навчання для виявлення психічних захворювань», науковий керівник роботи Стусь Олександр Вікторович, к.ф-м.н., доцент кафедри ММСА, затверджено наказом наказом по університету від «03» листопада 2022 р. № 4046-с.

2. Термін подання студентом дисертації 12.12.2022 р.

3. Об'єкт дослідження: Дописи користувачів соцмережі Reddit що мають психічні захворювання.

4. Предмет дослідження: Підходи та методи машинного навчання для вирішення задачі класифікації текстів.

5. Перелік завдань, які потрібно зробити:

1) здійснити огляд технічної літератури за темою роботи;

2) дослідити актуальність обраної теми;

3) ознайомитись із існуючими методами та моделями класифікації тексту;

4) здійснити порівняльний аналіз наявних методів, виявити їх переваги та недолі

5) розробити та реалізувати систему, що використовує апарат методів машинного навчання, та вирішує задачу ідентифікації психічний захворювань на основі дописів користувачів у соцмережах;

6) провести експеримент, що засвідчує працеспроможність запропонованої моделі, виконати аналіз результатів;

7) провести аналіз ринкових можливотсей запуску стартап проекту;

8) розробити концептуальні висновки;

9) підготувати ілюстративний матеріал;

10) оформити пояснювальну записку.

6. Перелік ілюстративного матеріалу: використані метрики, опис набору даних, побудова моделі, значення метрик.

7. Орієнтований перелік публікацій: Цупрун І.Ю., Стусь О.В. «Ідентифікація психічних захворювань методами обробки природної мови на основі дописів користувачів у соцмережах» / 1-а Всеукраїнська Науково-практична конференція «Системні науки та інформатика», секція «Системи і методи штучного інтелекту», КПІ ім. Ігоря Сікорського, м. Київ, 2022.

8. Дата видачі завдання: 31 січня 2022 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів роботи	Примітка
1.	Вивчення літератури за темою роботи.	01.09.2022 – 14.09.2022	Виконано
2.	Підготовка першого розділу.	14.09.2022 – 16.09.2022	Виконано
3.	Підготовка другого розділу.	16.09.2022 – 23.09.2022	Виконано
4.	Розробка програмного продукту.	01.10.2022 – 01.11.2022	Виконано
5.	Підготовка третього розділу	01.11.2022 – 10.11.2022	Виконано
6.	Підготовка частини стартап-проєкту	11.11.2022 – 13.11.2022	Виконано
9.	Концептуальні висновки. Перспективи розвитку отриманих рішень	13.11.2022 – 15.11.2022	Виконано
10.	Оформлення пояснювальної записки	15.11.2022 – 21.11.2022	Виконано

Студент  Ілля ЦУПРУН

Науковий керівник дисертації  Олександр СТУСЬ

РЕФЕРАТ

Магістерська дисертація: 127 с., 28 табл., 33 рис., 23 джерел, 1 додаток.

ОБРОБКА ПРИРОДНОЇ МОВИ, ГЛИБОКА МЕРЕЖА, КЛАСИФІКАЦІЯ ТЕКСТУ, ІДЕНТИФІКАЦІЯ ПСИХІЧНИХ ЗАХВОРЮВАНЬ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, СОЦІАЛЬНІ МЕРЕЖІ, МЕТОДИ КЛАСИФІКАЦІЇ, МАШИННЕ НАВЧАННЯ, АНАЛІЗ ДАНИХ, ГЛИБОКЕ НАВЧАННЯ

Об’єктом дослідження є зібрані дописи користувачів із соцмережі Reddit.

Предметом дослідження є підходи та методи машинного навчання та обробки природної мови для вирішення задачі класифікації текстів.

Метою даної роботи є створення системи для ідентифікації психічних захворювань на основі дописів користувачів у соцмережах, тобто вирішити задачу класифікації текстів у вказаній предметній області.

Методи обробки природної мови в останні роки отримали великий поштовх для розвитку та на сьогоднішній день, їх використання призвело до приголомшливих досягнень у різних сферах, від медицини до науки та розваг. У роботі було описано та використано методи необхідні для розв’язання задачі класифікації тексту, а також оглянуто вже існуючі дослідження, що стосуються даної теми.

В результаті було побудовано 8 моделей бінарної класифікації для ідентифікації 8 видів психічних захворювань та досягнуто точності ідентифікації на рівні 70-85% для більшості моделей, що покращує існуючі на сьогоднішній день досягнення у вирішенні посталеної проблеми.

ABSTRACT

Master’s thesis: 127 p., 28 tab., 33 fig., 23 references, 1 appendix.

NATURAL LANGUAGE PROCESSING, DEEP NETWORK, TEXT CLASSIFICATION, MENTAL ILLNESS IDENTIFICATION, CONVOLUTIONAL NEURAL NETWORKS, SOCIAL NETWORKS, CLASSIFICATION METHODS, MACHINE LEARNING, DATA MINING, DEEP LEARNING

The object of the study is the collected user posts from the social network Reddit.

The subject of research is approaches and methods of machine learning and natural language processing to solve the problem of text classification.

The aim of this work is to create a system for identifying mental illnesses based on user posts in social networks, that is, to solve the problem of text classification in mentioned subject area.

Natural language processing techniques have received a great impulse for development in recent years. Today, their use has led to amazing achievements in various fields, from medicine to science and entertainment. Methods required to solve the problem of text classification were described and used in the paper, as well as existing studies related to this topic were reviewed.

As a result, 8 binary classification models were built to identify 8 types of mental illnesses and the identification accuracy of 70-85% was achieved for most models, which improves the existing achievements in solving the problem.

ЗМІСТ

ВСТУП

В сучасному світі, де розвиток технологій біжить нелюдськими темпами, все більше і більше людей страждає від психічних захворювань. Так, за даними Всесвітньої Організації Охорони Здоров'я (далі ВООЗ), у 2019 р. кожна восьма людина планети, тобто. загалом 970 мільйонів чоловік, страждають на психічні розлади, причому найбільш поширеними є тривожні та депресивні розлади. У 2020 р. на тлі пандемії COVID-19 кількість людей, які страждають на тривожні та депресивні розлади, значно зросла. За попередніми оцінками лише за рік поширеність тривожних і серйозних депресивних розладів збільшилася на 26% та 28% відповідно. Незважаючи на наявність ефективних методів профілактики та лікування, більшість людей із психічними розладами не мають доступу до ефективної медичної допомоги. Багато хто з них також стикається зі стигматизацією, дискримінацією та порушенням прав людини. Крім того, є люди які бояться звернутись по допомогу або просто не розуміють що потребують її. [1]

Метою даної роботи є дослідження методів машинного навчання, які допоможуть створити систему для ідентифікації психічних розладів на основі публікацій людини в соцмережах та повідомлень у месенджерах. Таким чином, після виявлення того чи іншого психічного захворювання система матиме можливість у не нав'язливій формі запропонувати людині допомогу, що дуже важливо з огляду на страх людей звертатись по допомогу та ділитись своїми проблемами.

Так, наприклад, уявіть що модель ідентифікувала, що людина X страждає панічними атаками або тривогою. В такому випадку система може, наприклад, надіслати людині на телефон ненав'язливе пуш-повідомлення наступного характеру: “Відчуваєш тривогу - дізнайся що робити в такому випадку!” з посиланням на статтю про боротьбу з тривогою, де розповідають

як правильно себе вести в такому випадку, до якого спеціаліста звернутись, що є ще дуже багато таких самих як і ти.

Тематика першого розділу присвячена огляду теоретичних аспектів проблеми класифікації текстів та її вирішення за допомогою технік аналізу природної мови з використанням нейронних мереж. В цьому розділі були розглянути наступні методи: методи відображення слів у вектори дійсних чисел для вилучення з тексту корисних ознак, методи класифікації текстів за допомогою згорткових нейронних мереж та методи для оцінки якості роботи моделей машинного навчання у вирішенні задач класифікації.

У другому розділі розглянуто існуючі досягнення в даній предметній області, а саме дослідження пов’язані з визначенням тих чи інших психічних порушень на основі публікацій у соцмережах. Було проаналізовано результати цих досліджень, досліджена повнота вирішення зазначеної проблеми, а також підкреслено основні аспекти, які вплинули на цю роботу.

Третій розділ присвячений розробці власної моделі машинного навчання для ідентифікації психічних захворювань на основі публікацій у соцмережах. В даному розділі описано комплексний підхід до повного розв’язку поставленої проблеми, а саме розроблено модель яка вміє виявляти всі основні 8 категорій психічних розладів, які виділяє ВООЗ:

- тривожні розлади;
- депресія;
- біполярні розлади;
- посттравматичні стресові розлади;
- шизофренія;
- розлади харчової поведінки;
- асоціальна поведінка та дисоційовані розлади;
- порушення розвитку центральної нервової системи (аутизм).

Крім того, в третьому наведено детальне порівняння ефективності різних методів машинного навчання за допомогою яких вирішувалась поставлена задача, було проаналізовано результати.

Загальну поставноку задачі можна описати наступним чином: вирішити задачу класифікації основних видів психічних захворювань за допомогою методів обробки природної мови та методів машинного навчання, покращити існуючі результати в даній предметній області. Аналітичну постановку задачі та методи для її вирішення розглянуто в розділах 1 та 2, в третьому розділі запропоновано розв’язок поставленої задачі за допомогою описаних методів.

1 ТЕОРІЯ МЕТОДІВ ОБРОБКИ ПРИРОДНОЇ МОВИ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ КЛАСИФІКАЦІЇ ТЕКСТІВ З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ

1.1 Аналітична постановка задачі. Огляд задач які вирішує обробка природної мови. Основні терміни NLP

Обробка природної мови (Natural Language Processing, NLP) - це галузь штучного інтелекту (ШІ), яка робить людську мову зрозумілою для машин. NLP поєднує в собі розділи лінгвістики та комп'ютерних наук для вивчення правил і структури мови, а також створення інтелектуальних систем (на основі машинного навчання та алгоритмів NLP), здатних розуміти, аналізувати та витягувати сенс з тексту та мови.

НЛП використовується для розуміння структури та значення людської мови шляхом аналізу різних аспектів, таких як синтаксис, семантика, прагматика та морфологія. Потім комп'ютерні науки перетворюють ці лінгвістичні знання на алгоритми машинного навчання, засновані на правилах, які можуть вирішувати конкретні проблеми і виконувати бажані завдання.

Візьмемо, наприклад, електронну пошту Gmail. Електронні листи автоматично класифікуються як рекламні, соціальні, первинні або спам саме завдяки методам NLP, які використовують вилучення ключових слів. "Читаючи" слова в темах листів і пов'язуючи їх із заздалегідь визначеними тегами, машини автоматично дізнаються, до якої категорії віднести електронні листи.

Метою даної роботи є створити систему для ідентифікації психічних захворювань на основі текстів користувачів у соцмережах, отже потрібно вирішити задачу класифікації тексту. Саме теоретичні основи методів обробки природної мови необхідні для вирішення задачі класифікації текстів і будуть розглянуті в даному розділі.

Для кращого розуміння подальшого змісту необхідно ввести термінологію, що використовується в обробці природної мови. Основними термінами NLP є: корпус, документ, токен, словник. Спочатку текст розбивається на текстові одиниці (токени), наприклад, символи, слова, словосполучення, речення, абзаци тощо. Найчастіше розбивають на слова. Токени утворюють словник, який може бути відсортований за алфавітом. Також у NLP застосовуються терміни "документ" і "корпус". Документ - це сукупність токенів, які належать одній смисловій одиниці. Як документ може виступати речення, коментар або пост користувача. Корпус - це генеральна сукупність усіх документів.

Розглянемо приклад. Припустимо, є два речення: "Пес сів на пень", "Кіт сів на ялину". Виберемо як токени слова, тоді вийде такий словник: {Пес, Кот, ялина, на, сів, пень} і два документи, які складають корпус: [Пес, сів, на, пень] та [Кіт, сів, на, ялина]. Тоді корпус має наступний вигляд: [[Пес, сів, на, пень], [Кіт, сів, на, смерека]].

Класифікація тексту - це процес розуміння сенсу неструктурованого тексту та організації його за заздалегідь визначеними категоріями (тегами). Одним з найпопулярніших завдань класифікації тексту є аналіз настроїв, який має на меті класифікувати неструктуровані дані за настроями. Інші завдання класифікації включають виявлення намірів, моделювання тем і виявлення мови.

Математично задачу класифікації тексту можно описати наступним чином. Нехай є множина класів C:

$$\mathrm{C} = \{c_1, \dots, c_{|C|}\}, \quad (1)$$

та множина документів D:

$$D = \{d_1, \dots, d_{|D|}\}, \quad (2)$$

невідома цільова функція Φ:

$$\Phi = C \text{ x } D -> \{0, 1\}. \quad (3)$$

Необхідно побудувати класифікатор Φ', який найкраще апроксимує Φ. Є деяка початкова колекція розмічених документів $R \subset C * D$, для яких відомі значення Ф. Зазвичай її ділять на "навчальну" і "перевірочну" частини. Першу використовують для навчання класифікатора, другу - для незалежної перевірки якості його роботи. Класифікатор може видавати точну відповідь або ступінь подібності:

$$\Phi' : \mathrm{C} \times D \to \{0,1\}, \quad (4)$$

$$\Phi' : \mathrm{C} \times D \to [0,1]. \quad (5)$$

Процес вирішення задачі класифікації текстів зазвичай складається з трьох основних етапів:

- попередня обробка тексту;
- векторизація слів;
- вирішення задачі класифікації.

Саме методи обробки природної мови для вирішення цих трьох основних етапів і буде розглянуто в даному розділі.

1.2 Методи попередньої обробки текстових даних

Попередня обробка даних - це заключний етап збору та підготовка даних, який включає в себе трансформацію даних у доступний для розуміння формат. В обробці природної мови виділяють такі етапи попередньої обробки

даних: чистка даних, зміна регістру, токенізація, видалення стоп слів, стематизація, лематизація, Н-грами. Всі ці методи будуть розглянуті та описані далі в цьому підрозділі.

1.2.1 Чистка даних

Чистка даних - це процес видалення з вихідних даних особливих знаків, символів, пунктуації, тегів html тощо, які не містять жодної корисної для моделі інформації та лише додають шум у дані. Що видаляти з вихідних даних, а що ні залежить від постановки завдання. Наприклад, при аналізі тексту зі сфери економіки або бізнесу, знаки типу $ або інші символи валют можуть містити приховану інформацію, але в більшості випадків такі сущності повинні бути видалені.

Зазвичай виконують наступні етапи чистки даних: видалення розділових знаків, видалення часто та рідко вживаних слів, видалення емодзі та смайликів, які в сучасному світі стали невід'ємною частиною багатьох неформальних текстів. [2]

1.2.2 Зміна регістру

Використання нижнього регістру є поширеною технікою попередньої обробки тексту. Ідея полягає в тому, щоб перетворити вхідний текст в однаковий формат, щоб, наприклад, слова "текст", "Текст" і "ТЕКСТ" розглядалися однаково. Це також потрібно для виростання методів

характеризації тексту побудованих на частоті вживання слів, наприклад, TF-IDF, оскільки це допомагає об'єднати однакові слова разом, тим самим зменшивши дублювання та отримавши правильні підрахунки значення метрики частоти. Такі методи будуть більш детально описані в розділі 1.3 “Методи векторизації текстових даних”.

За замовчуванням, нижній регістр використовується більшістю сучасних векторизаторів і токенізаторів, наприклад TfidfVectorizer і Keras Tokenizer в пакеті sklearn мови програмування python.

1.2.3 Токенізація

Токенізація - це один з найважливіших кроків у підготовці текстових даних. Це процес розбиття потоку текстових даних на слова, терміни, речення, символи або інші значущі елементи, які називаються токенами. Для виконання процесу токенізації існує безліч інструментів з відкритим вихідним кодом. Токенізація - це один з найперших кроків у вирішенні будь-якої задачі обробки природної мови. Він має важливий вплив на пайплайн вирішення задачі. Токенізатор розбиває неструктуровані дані і текст природною мовою на фрагменти інформації, які можна розглядати як дискретні елементи. Вміст токенів у документі можна використовувати безпосередньо як вектор, що представляє цей документ.

Завдяки цьому неструктурований рядок (текстовий документ) можна перетворити в числову структуру даних, придатну для машинного навчання, яка може бути зрозуміла безпосередньо комп'ютером. Токенізація може відокремлювати речення, слова, символи або словосполучення. Розбиття тексту на речення називається токенізацією речень, на слова - токенізацією слів. [3]

1.2.4 Видалення стоп-слів

Стоп-слова - це слова, що часто зустрічаються в мові, такі як "you, he, she, in, a, has, are, etc." тощо. У більшості випадків їх можна видалити з тексту, оскільки вони не надають цінної інформації для подальшого аналізу. У випадку визначення частини мови їх видаляти не потрібно, оскільки вони надають дуже цінну інформацію про частину мови. [4]

Список стоп-слів вже складені для різних мов і їх можна використовувати. Так, наприклад, список стоп-слів для англійської мови з пакету nltk мови програмування Python наведено на Рисунку 1.1.

```
"i, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, you'd, your, your
s, yourself, yourselves, he, him, his, himself, she, she's, her, hers, herself, it, it's, it
s, itself, they, them, their, theirs, themselves, what, which, who, whom, this, that, that'l
l, these, those, am, is, are, was, were, be, been, being, have, has, had, having, do, does,
did, doing, a, an, the, and, but, if, or, because, as, until, while, of, at, by, for, with,
about, against, between, into, through, during, before, after, above, below, to, from, up, d
own, in, out, on, off, over, under, again, further, then, once, here, there, when, where, wh
y, how, all, any, both, each, few, more, most, other, some, such, no, nor, not, only, own, s
ame, so, than, too, very, s, t, can, will, just, don, don't, should, should've, now, d, ll,
m, o, re, ve, y, ain, aren, aren't, couldn, couldn't, didn, didn't, doesn, doesn't, hadn, ha
dn't, hasn, hasn't, haven, haven't, isn, isn't, ma, mightn, mightn't, mustn, mustn't, needn,
needn't, shan, shan't, shouldn, shouldn't, wasn, wasn't, weren, weren't, won, won't, wouldn,
wouldn't"
```

Рисунок 1.1 - Стоп-слова англійської мови з пакету nltk мови програмування Python

1.2.5 Стемінг

Ідея стемінгу полягає в тому, щоб звести різні форми вживання слова до його кореневої форми. Наприклад, "drive", "drove", "driving", "driven", "driver" є похідними від слова "drive", і дуже часто дослідники хочуть прибрати цю

варіативність зі свого корпусу. Приклад операції стемінгу наведено на рисунку 1.2.

Існує декілька типів алгоритмів стемінгу і одним з найбільш відомих є Стеммер Портера, який широко використовується. Стеммер Портера - алгоритм стемінгу, опублікований Мартіном Портером у 1980 році. Оригінальна версія стеммера була призначена для англійської мови і була написана мовою BCPL. Згодом Мартін створив проект "Snowball" і, використовуючи основну ідею алгоритму, написав стеммери для поширених індоєвропейських мов, зокрема для російської. Алгоритм не використовує базу основ слів, а лише, застосовуючи послідовно низку правил, відтинає закінчення і суфікси, ґрунтуючись на особливостях мови, у зв'язку з чим працює швидко, але не завжди безпомилково. [5]

Алгоритм був дуже популярним і тиражованим, до нього часто вносилися зміни різними розробниками, причому не завжди вдалі. У 2000 році Портер вирішив "заморозити" проєкт і надалі розповсюджувати одну-єдину реалізацію алгоритму (кількома популярними мовами програмування) зі свого сайту.

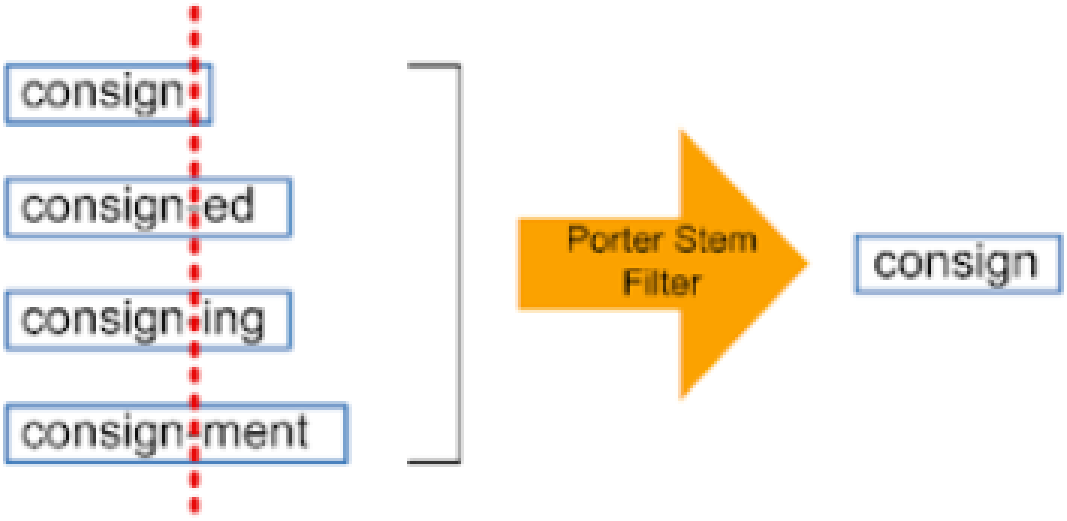

Рисунок 1.2 - Приклад роботи алгоритму стемінгу

1.2.6 Лематизація

Лематизація дещо схожа зі стемінгом. Різниця полягає в тому, що лематизація враховує морфологічний аналіз слів. Для цього необхідно мати детальні словники, які алгоритм може переглянути, щоб зв'язати форму з її лемою. Приклад лематизації в англійській мові наведено на Рисунку 1.3. Ще одна важлива відмінність, яку слід підкреслити, полягає в тому, що лема є базовою формою для всіх його флексійних форм, тоді як основа не є такою. Ось чому звичайні словники - це списки лем, а не основ. Так, наприклад, лема для слова "краще" - "хороший". Цей зв'язок не враховується під час стемінгу, оскільки для цього потрібно шукати його у словнику. Слово "прогулянка" є іншої формою для слова "гуляти", а отже, воно збігається і за словотвірною, і за лематичною структурою. [6]

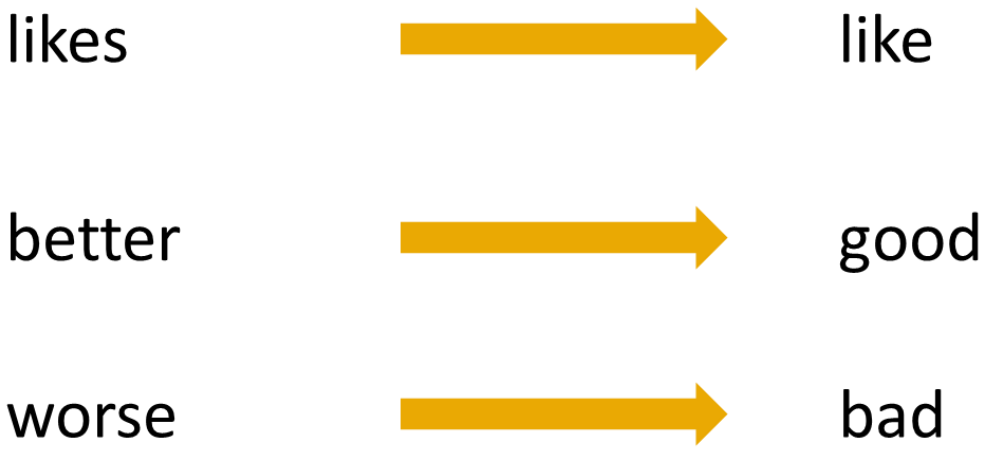

Рисунок 1.3 - Приклад роботи алгоритму лематизації

1.2.7 N-грами

N-грами - це комбінації з кількох слів, що використовуються разом. N-грами, де N=1 називаються уніграмами. Подібним же чином, біграми (N=2), триграми (N=3) і так далі. N-грами використовуються, коли потрібно зберегти якусь послідовність даних, наприклад, яке слово найчастіше слідує за заданим словом. Уніграми не містять ніякої послідовності даних, оскільки кожне слово береться індивідуально. [7] Приклад використання N-грамів продемонстровано на малюнку 1.4.

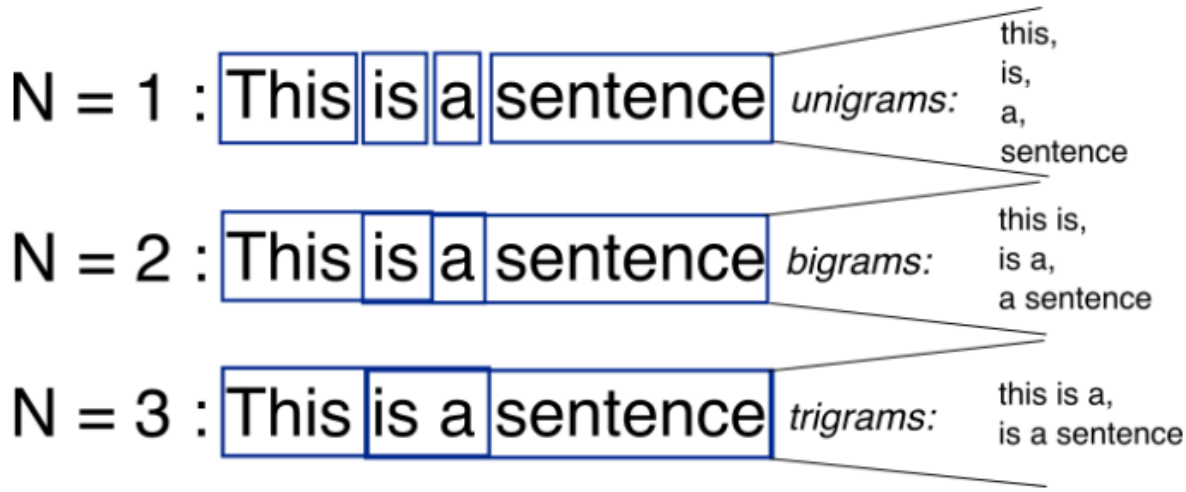

Рисунок 1.4 - Приклад використання N-грамів

1.3 Векторизація текстових даних, класичні методи векторизації

Як відомо, багато алгоритмів машинного навчання і майже всі моделі глибокого навчання не здатні обробити простий текст в сирому вигляді. Для виконання будь-яких завдань, таких як класифікація, регресія, кластеризація і т.д., їм потрібні числові дані в якості вхідних даних. Крім того, з величезної кількості даних, які присутні в текстовому форматі, необхідно витягти якісь

знання. Отже, для побудови будь-якої моделі машинного навчання підготовлені дані бути в числовій формі, оскільки моделі не розуміють текстові або графічні дані безпосередньо так, як це робить людина. Саме для перетворення текстових даних у числові і були вигадані методи векторизації тексту.

Існує багато способів виконання векторизації, починаючи від наївних бінарних функцій входження слів до просунутих контекстно-залежних представлень. Залежно від випадку використання та моделі, будь-який з них може бути здатний виконати необхідну задачу. Крім того, методи векторизації сильно відрізняються між собою обчислювальної ефективністю.

В даному та наступному розділах будуть розглянуті фундаментальні методи векторизації та їх еволюція протягом часу до найсучасніших та більш ефективних для вирішення практичних задач методів.

1.3.1 Пряме кодування

Пряме кодування (one-hot encoding) вважається найпростішим способом перетворення токенів в тензори або вектори. При прямому кодуванні кожне слово (або навіть символ), що входить до складу заданих текстових даних, записується у вигляді векторів, що складаються тільки з чисел 1 і 0. Таким чином, векторизація методом прямого кодування - це створення набору векторів, елементами яких є тільки 1 і 0, причому кожен такий вектор є унікальним. Це дозволяє однозначно ідентифікувати слово за його вектором і навпаки, тобто жодні два слова не матимуть однакового представлення. Приклад прямого кодування продемонстровано на рисунку 1.5.

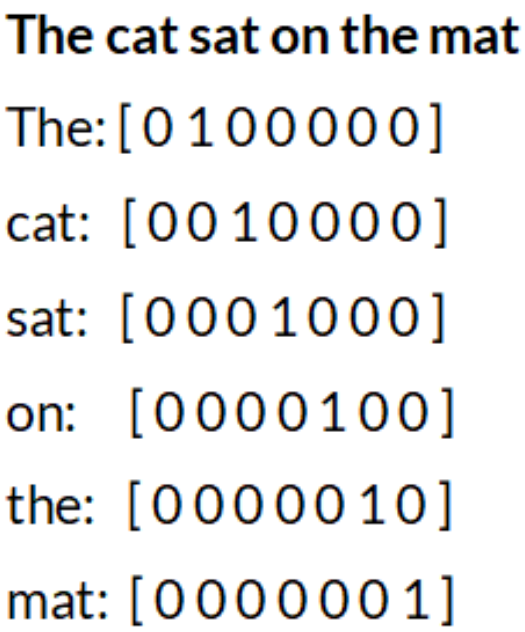

Рисунок 1.5 - Приклад векторизації тексту методом прямого кодування

Проте, в багатьох сучасних задачах, де кількість тестів які підлягають аналізу сягає мільйонів, словник може досягти багато мільйонів слів. Кількість слів в англійській мові дуже грубо становить мільйон - матриця спільних зустрічаємостей тільки пар слів буде 10^6 x 10^6. Така матриця навіть зараз не дуже лізе в пам'ять комп'ютерів, а, скажімо, 10 років тому про таке можна було не мріяти. Крім того, метод прямого кодування ні як не враховує контекст у якому були вжиті слова. Саме ці проблеми звели нанівець ефективність прямого кодування у практичному використанні, але саме цей метод став фундаментом для побудови більш сучасних алгоритмів векторизації тексту. [8]

1.3.2 Мішок слів (Bag-of-words, BOW)

Модель мішок слів (bag-of-words, BOW)) - це представлення, яке перетворює довільний текст у вектори фіксованої довжини, підраховуючи, скільки разів з'являється кожне слово. Незважаючи на те, що це все ще досить проста модель, саме це дозволило стати їй досить широко використовуваною у вирішенні задач природної мови. Bag of words вирішує проблему розмірності за однією віссю завдяки чому було отримано покращення обчислювальної ефективності порівняно з методом прямого кодування. Однак, як і в методі прямого кодування, цей метод не враховує порядок слів та контекст у якому були вжиті слова. [9] Приклад векторизації текстових даних методом BOW наведено на рисунку 1.6.

Document	the	cat	sat	in	hat	with
the cat sat	1	1	1	0	0	0
the cat sat in the hat	2	1	1	1	1	0
the cat with the hat	2	1	0	0	1	1

Рисунок 1.6 - Приклад векторизації текстових даних методом Bag-of-words

1.3.3 Векторизація на основі метрики TF-IDF

Як вже було зазначено, метод BOW є простим і добре працює, але проблема полягає в тому, що він однаково ставиться до всіх слів. В результаті, він не може розрізнити дуже поширені слова або рідкісні слова. Отже, щоб вирішити цю проблему було розроблено метрику TF-IDF. Вона складається з

двох основних частин: Term Frequency (TF) та Inverse Document Frequency (IDF) та розраховується для кожного слова у документі наступним чином:

$$TF{\sim}IDF_{td} = TF_{td} * IDF_{tc} \quad (6)$$

Term Frequency означає частоту входження слова (токена) в документ. Для певного слова вона визначається як відношення кількості разів, коли слово з'являється в документі, до загальної кількості слів у документі. Тобто, це відсоток кількості разів, коли слово (t) зустрічається в певному документі (d), поділений на загальну кількість слів у цьому документі. Term Frequency розраховується для кожного слова у документі наступним чином:

$$TF_{td} = \frac{N(t|d)}{N_d}, \quad (7)$$

де:

$N(t|d)$ - це кількість входжень слова t в документ d,

N_d - загальна кількість слів в документі d.

Inverse Document Frequency (IDF) - це метрика яка вимірює важливість слова у корпусі з декількох документів. Вона представляє собою логарифмічне відношення кількості всіх документів до кількості документів з певним словом. IDF розраховується для кожного слова у корпусі наступним чином:

$$IDF_{tc} = \frac{N_c}{N_{(t|c)}}, \quad (8)$$

де:

N_c- це кількість документів в корпусі c,

$N_{(t|c)}$ - це кількість документів що містять слово t в корпусі c.

IDF метрика допомагає зробити більший акцент на рідкісні для документу слова. Так, якщо слово зустрічається багато разів у багатьох документах, то знаменник df буде збільшуватися, зменшуючи значення другого доданка. [10] Приклад векторизації корпусу текстів за допомогою метрики TF-IDF наведено на рисунку 1.7.

Токен	TF		IDF	TF-IDF	
	Док. №1	Док. №2		Док. №1	Док. №2
Кот	0	$\frac{1}{4}$	$\log\frac{2}{1}$	0	0.17
Пес	$\frac{1}{4}$	0	$\log\frac{2}{1}$	0.17	0
ель	0	$\frac{1}{4}$	$\log\frac{2}{1}$	0	0.17
на	$\frac{1}{4}$	$\frac{1}{4}$	$\log\frac{2}{2}=0$	0	0
сел	$\frac{1}{4}$	$\frac{1}{4}$	$\log\frac{2}{2}=0$	0	0
пень	$\frac{1}{4}$	0	$\log\frac{2}{1}$	0.17	0

Рисунок 1.7 - Приклад векторизації документів за допомогою метрики TF-IDF

В якості висновку, можна виділити наступні основні особливості та відмінності методу векторизації тексту на основі метрики TF-IDF:

- подібно до методу прямого кодування, в методі TF-IDF генерується матриця токенів документа, кожен стовпець якої представляє окреме унікальне слово;
- відмінність методу TF-IDF полягає в тому, що кожна комірка не вказує частоту токена, а містить значення ваги, яке означає, наскільки важливим є слово для окремого текстового повідомлення або документа;
- цей метод відрізняється від попередніх тим, що враховує не тільки входження слова в окремому документі, але і в усьому корпусі;

- TF-IDF надає більшої ваги подіям, що трапляються рідше, і меншої ваги очікуваним подіям;
- TF-IDF слова вказує на те, як часто лексема зустрічається в документі і наскільки унікальною є ця лексема для всього корпусу документів.

1.4 Нова епоха векторизації текстових даних

Описані в попередньому розділі методи були, є і залишаються популярними в задачах, де кількість текстів мала і словник обмежений за розміром. Але з появою інтернету і збільшенням кількості інформації, яку можна використовувати в задачах обробки природної мови, проблеми вже згаданих методів ставали критичними та не дозволяли розв’язувати сучасні задачі. По-перше, для їх використання були потрібні неймовірні обчислювальні потужності, що робило вирішення задач економічно неефективними. По-друге, так і не була вирішена проблема розуміння слів у контексті.

Так, наприклад, при використанні методів TF-IDF або BOW, слова "хороший" і "великий" відрізняються так само, як "день" і "мати", що не відповідає дійсності. Навіть при дуже гарній попередній обробці даних, якісній лематизації, залишається варіативність і слова “король” та “імператор” відрізняються так само, як “король” та “крокодил”. Отже, постала задача знайти метод векторизації який допоможе створити схожі вектора для схожих за значенням словам. Математично, косинус кута між такими векторами повинен бути близьким до 1, тобто кут близький до 0.

У 2013 році було запропоновано рішення за принципом "той, хто нам заважає, той нам допоможе". А саме, у 2013 році тоді мало кому відомий чеський аспірант Томаш Міколов запропонував свій підхід до векторизації

слів, який він назвав word2vec. Його підхід ґрунтується на іншій важливій гіпотезі, яку в науці заведено називати гіпотезою локальності - "слова, які зустрічаються в однакових оточеннях, мають близькі значення". Близькість у цьому разі розуміють дуже широко, як те, що поруч можуть стояти тільки слова, які поєднуються. Наприклад, для нас звичне словосполучення "наводити будильник". А сказати "наводити апельсин" ми не можемо - ці слова не поєднуються. [11]

1.4.1 Ідея метода word2vec. Модель CBOW.

Ідея метода word2vec полягає у визначенні слова за його контекстом або контексту який відповідає слову за допомогою використання нейронних мереж. Ці два протилежні принципи отримали назви Skip Gram та Continuous Bag Of Words (CBOW) відповідно.

Модель CBOW використовує контекст кожного слова як вхідні дані і намагається передбачити слово, що відповідає контексту. Розглянемо як приклад речення “Бажаю чудового дня”. Нехай на вхід нейронної мережі подається слово "чудовий". В такому випадку, в якості цільового слова для нейронної мережі буде слово “день”, використовуючи єдине контекстне вхідне слово "чудовий". Якщо бути точніше, на вхід в мережу подається вектор слова "чудовий" закодований за допомогою прямого кодування або іншого методу згаданого вище. В процесі прогнозування цільового слова буде отримано векторне представлення цільового слова. Архітектуру найпростішої нейронної мережі CBOW наведено на рисунку 1.8.

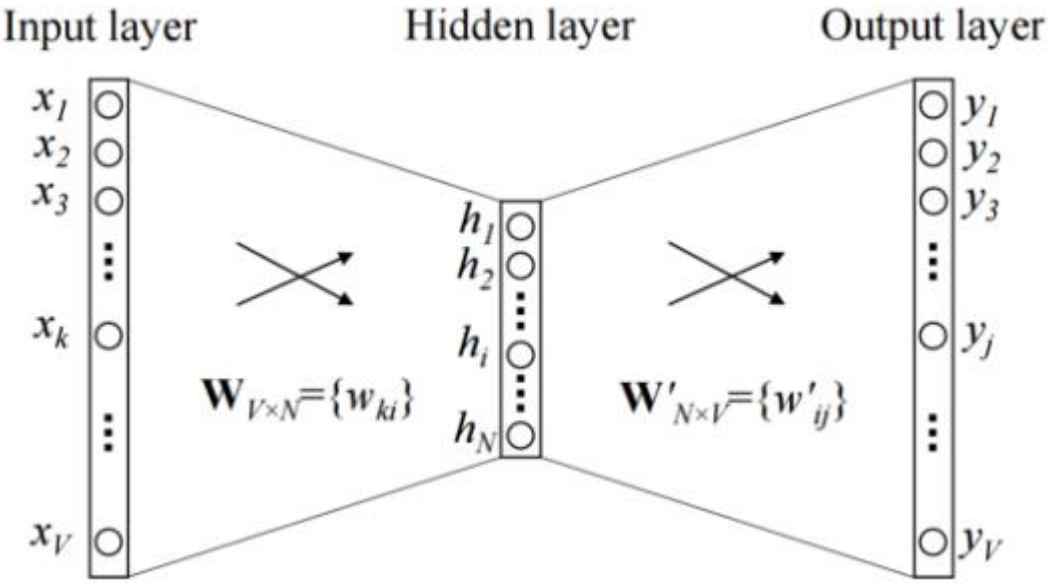

Рисунок 1.8 - Проста модель CBOW з одним словом контексту

Якщо більш детально розглянути архітектуру наведеної мережі, то вхід або контекстне слово - це одне слово перетворене у вектор розміром V за допомогою прямого кодування. Прихований шар містить N нейронів, а вихід - це знову вектор довжиною V, елементами якого є значення функції активації softmax. $W(V\text{x}N)$ - це вагова матриця, яка відображає вектор вхідних даних X у прихований шар (матриця розмірністю V*N). $W'(N\text{x}V)$ - вагова матриця, яка відображає виходи прихованого шару на кінцевий вихідний шар (матриця розмірності N*V). Нейрони прихованого шару просто копіюють зважену суму входів до наступного шару. В них немає функції активації. Єдиною нелінійністю є обчислення softmax у вихідному шарі.

Але, вищевказана модель використовувала одне контекстне слово для прогнозування цілі. Ідея ж CBOW полягає у можливості використовувати кілька контекстних слів. Рисунок 1.9 демонструє повноцінну модель CBOW з використанням кількох слів контексту для прогнозування слова, що йому відповідає.

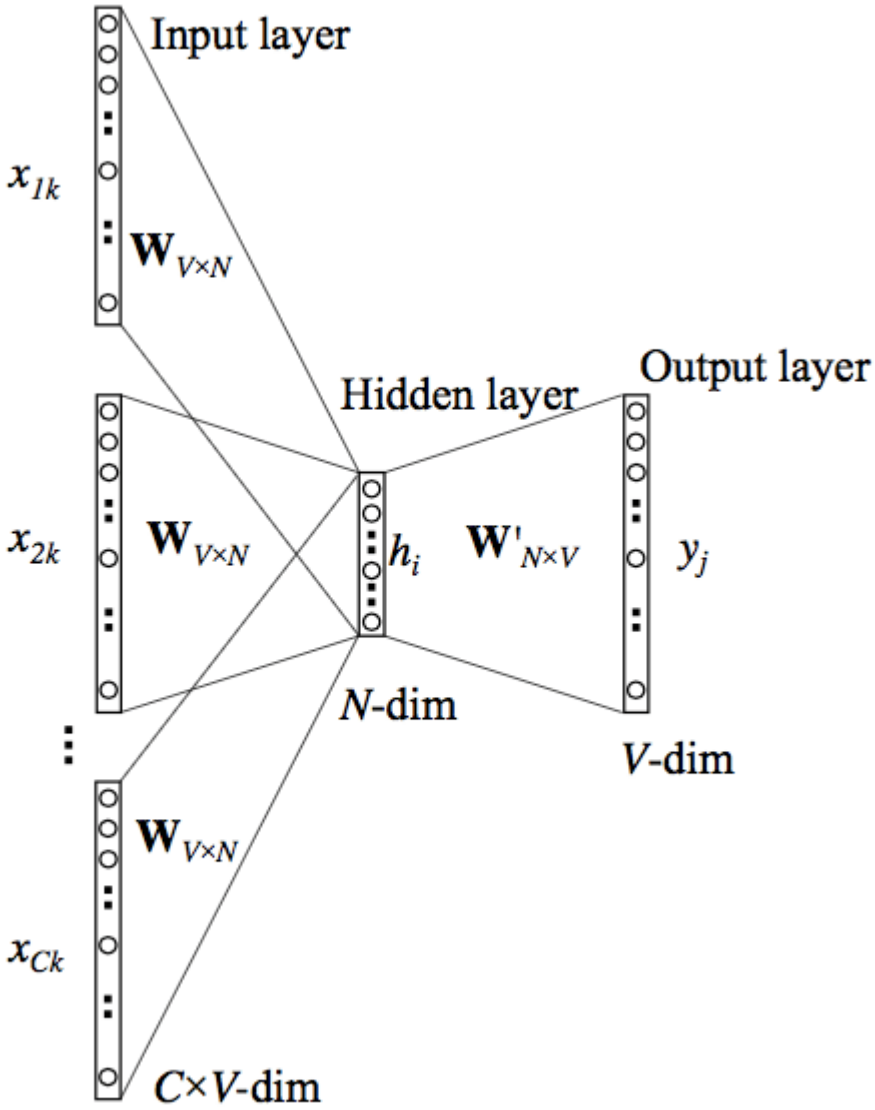

Рисунок 1.9 - Модель CBOW з кількома словами контексту

У наведеній вище моделі використовується C контекстних слів. Коли $W(V\text{x}N)$ використовується для обчислення входів прихованого шару, на вхід подається середнє значення для всіх цих C контекстних слів.[12]

1.4.2 Skim-gram як протилежність до CBOW

В попередньому підрозділі було оглянуто метод генерації представлення слів за допомогою контекстних слів. Але, існує ще один спосіб зробити те ж саме - використовувати цільове слово (представлення якого ми хочемо згенерувати) для прогнозування контексту. Такий метод називається моделлю Skip Gram. Архітектура моделі Skim Gram представлена на рисунку 1.10.

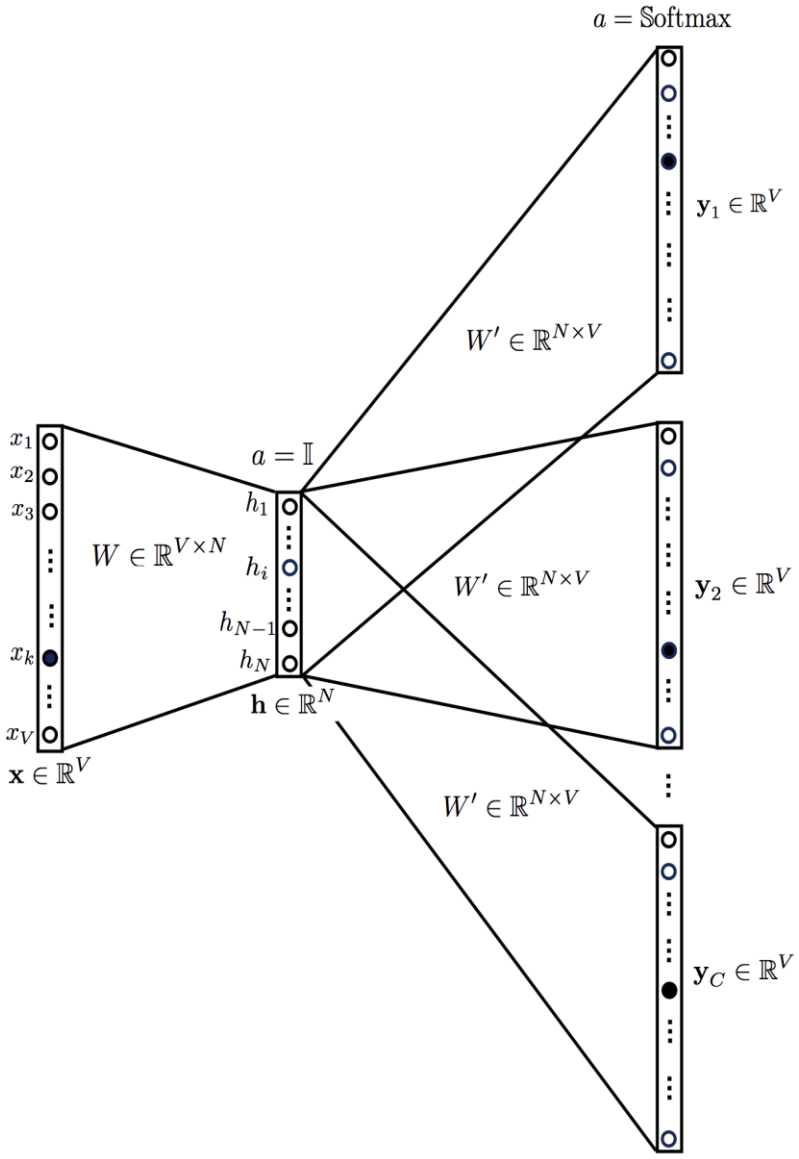

Рисунок 1.10 - Архітектура моделі Skim Gram

Модель Skim Gram дуже схожа на перевернуту модель CBOW. Певною мірою це так. На вхід моделі подається цільове слово, а модель видає C розподілів ймовірності. Тобто, для кожної позиції контексту (для кожного слова) модель повертає C розподілів ймовірностей з V ймовірностей, по одному для кожного слова.

В обох випадках мережа метод зворотного розповсюдження помилки для навчання. Це можливо за рахунок використання SoftMax в якості цільової функції, таким чином функція помилки таких стає диференційованою. Процес тренування влаштований наступним чином: береться послідовно (2k+1) слів, слово в центрі є тим словом, яке має бути передбачене, а навколишні слова є контекстом довжини по k з кожного боку. Кожному слову в моделі зіставлено унікальний вектор, який ми змінюється в процесі навчання нашої моделі.

1.4.3 Покращення методу word2vec. Negative sampling та Hierarchical SoftMax.

З 2013 року з’явилось багато покращень для класичної моделі word2vec. Основна особливість таких алгоритмів - покращення обчислювальної складності алгоритму word2vec. Два найбільш відомих - Negative Sampling та Hierarchical Softmax і будуть описані в даному розділі.

Як вже було зазначено, модель word2vec використовує активацію Softmax на останньому шарі, яка генерує мультиноміальний розподіл ймовірностей. Вихідний вектор, який було отримано, розглядається як векторне представлення слова (контекстного слово або просто вихідне слово, в залежності від моделі). Функція активації Softmax виглядає наступним чином:

$$p(w_j|w_I) = y_j = \frac{\exp(u_j)}{\sum_{j`=1}^{n} \exp(u_{j`})}, \quad (9)$$

де кожен з елементів активованого вихідного вектора - це ймовірність того, що слово дорівнює j-му слову в словнику при заданому вхідному слові I. Також варто зазначити, що сума елементів вихідного вектора дорівнює 1 і кожен з його елементів відображається в діапазон [0,1]. Обчислювальна складність цього алгоритму, що обчислюється прямолінійно, дорівнює розміру словника, O(V), проте її можна суттєво зменшити, використовуючи структуру бінарного дерева.

1.4.3.1 Hierarchical softmax

Основна мотивація даної методології зводиться до зменшення обчислювальної складності до логарифмічної, що кардинально кардинально зменшує кількість операцій необхідну для роботи алгоритму.

$$O(V) \rightarrow O(\log_2 V), \quad (10)$$

де:

V - розмір словника.

Для цього використовується бінарне дерево, де листки представляють ймовірності слів, точніше, листок з індексом j є ймовірністю j-го слова і має позицію j у вихідному векторі softmax. До кожного зі слів можна дістатися шляхом від кореня через внутрішні вузли, які представляють собою функції ймовірностей на цьому шляху. Ці значення обчислюються за допомогою

простої сигмоїдної функції, а шлях, який обчислюється, є просто добутком цих функцій ймовірностей, визначених наступним чином:

$$\sigma(x) = \frac{1}{1 + \exp(-x)}, \tag{11}$$

де:

x - обчислюється як скалярний добуток вхідного і вихідного векторних зображень слова, з яким ми працюємо:

$$x = u_{n(w,j)} = v`_{n(w,j)^T} v_{w_1}, \tag{12}$$

де:

$n(w,j)$ - j-й вузол на шляху від кореня до w, для якого обчислюється функція ймовірності. Тобто, можна замінити сигмоїдний запис на ймовірнісний: для кожного з внутрішніх вузлів вибираємо одну довільну дочірню вершину (ліву або праву) і приписуємо (як правило, це ліва дочірня вершина) додатне значення сигмоїдної функції. При збереженні цих обмежень сигмоїдну функцію можна трактувати наступним чином:

$$p(n, left) = \sigma\left(v`_n{}^T h\right) = \sigma\left(v`_n{}^T v_{w_1}\right), \tag{13}$$

для лівого нащадка вершини n і аналогічно для правого нащадка вершини n:

$$p(n, right) = \sigma\left(-v`_n{}^T h\right) = \sigma\left(-v`_n{}^T v_{w_1}\right). \tag{14}$$

Остаточне обчислення функції складається з усіх попередніх кроків разом з булевою перевіркою для випадку з довільно обраним вузлом (правим або лівим):

$$p(w|w_1) = \prod_{j=1}^{L(w)-1} \sigma\left(\{n(w,j+1) == ch(n(w,j))\} v`_n{}^T v_{w_I}\right), \quad (15)$$

де:

{ } - булева перевірка істинності або хибності випадку,

$L(w)$ - глибина дерева,

$ch(n)$ - нащадок вузла n.

Приклад такого бінарного дерева наведено на рисунку 1.11. Тут, якщо дерево має кореневий вузол, 2 внутрішні вузли та листові вузли, то очевидно, що виконується всього 3 кроки обчислень, що є достатнім зменшенням кількості операцій, які виконуються. [13]

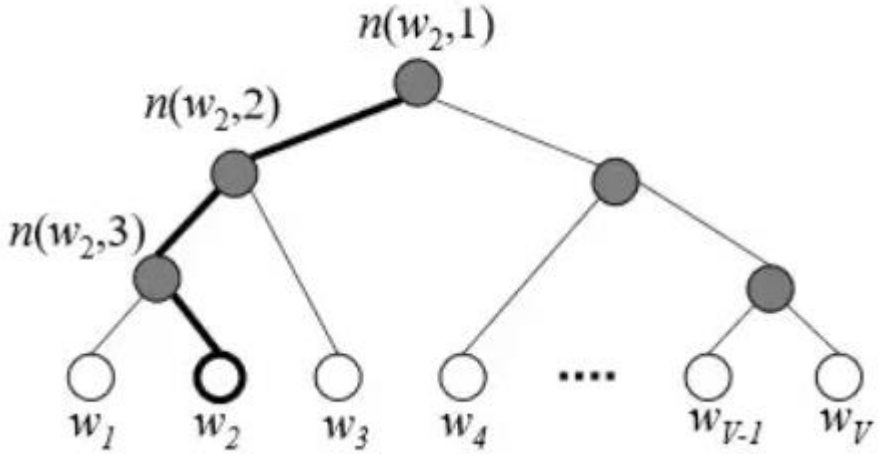

Рисунок 1.11 - Приклад бінарного дерева для векторизації слів

1.4.3.2 Negative sampling

Ідея Negative Sampling ґрунтується на концепції контрастного оцінювання шуму (подібно до генеративних змагальних мереж), яка полягає в тому, що хороша модель повинна відрізняти фальшивий сигнал від справжнього за допомогою логістичної регресії. Крім того, подібна до стохастичного градієнтного спуску: замість того, щоб кожного разу змінювати всі ваги з урахуванням усіх тисяч спостережень, використовується лише K з них, що також різко підвищує обчислювальну ефективність (залежить від кількості від'ємних зразків). Цільова функція для одного спостереження має такий вигляд:

$$\log p(w|w_1) = \log \sigma\left(v`_w v_{w_I}\right) + \sum_{i=k}^{K} E_{w_i} \sim P_n(w)[log\, \sigma\left(-v`^T_w v_{w_I}\right)] \qquad (16)$$

Як можна помітити, відмінність від стохастичного градієнтного спуску полягає в тому, що до уваги береться не одне спостереження, а K спостережень. Розподіл ймовірностей, який використовується, є розподілом шуму. Причиною використання цього розподілу шуму є завдання відрізнити реальні дані від фальшивих даних, яке вирішується. Відповідним розподілом шуму є уніграмний розподіл U(w), який визначається як:

$$P(w_i) = \frac{f(w_i)^{\frac{3}{4}}}{\sum_{j=0}^{n}(f(w_j)^{\frac{3}{4}})}, \qquad (17)$$

де:

$^{3}/_{4}$ - це параметри визначений під час проведення експериментів на великих масивах даних,

f(w) - частота входження слова в корпус.

В моделі Skim Gram, негативні приклади з вибірки - це слова, які не є контекстними словами, а позитивні приклади - це, звичайно, контекстні слова.

1.5 Методи класифікації для вирішення задачі класифікації текстів

Як вже було згадано - алгоритми класифікації з використанням машинного навчання є останнім етапом у вирішенні задачі класифікації тексту, після попередньої обробки даних та векторизації, тобто приведення текстових даних до числових векторів. Після того, як було сформоване векторне представлення всіх розмічених текстових документів, їх можна використовувати для навчання класифікатора. Векторне представлення текстових документів передається класифікатору з правильними категоріями. Модель вивчає асоціації між різними лінгвістичними ознаками в тексті та категоріями, як це схематично зображено на рисунку 1.12.

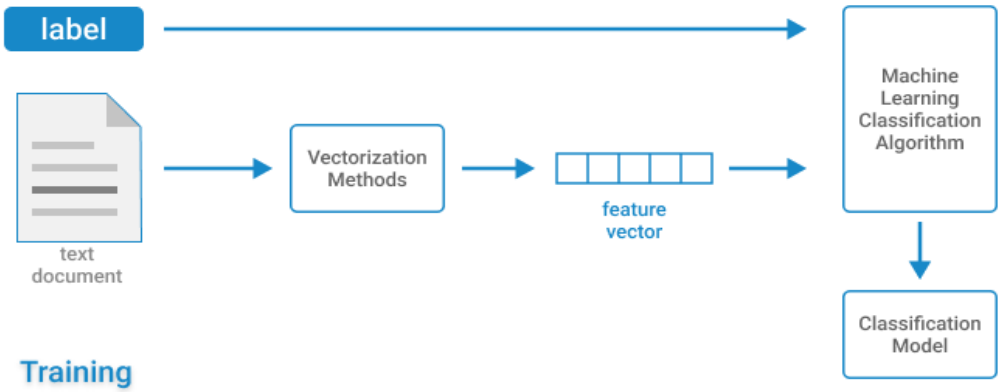

Рисунок 1.12 - Приклад навчання системи для класифікації текстових даних

Після того, як модель навчена до необхідних параметрів продуктивності, вона може бути використана для точного прогнозування. Той самий метод вилучення ознак використовується для створення векторного представлення нових текстових документів. Модель класифікації використовує ці вектори ознак для прогнозування класів документів, як це зображено на рисунку 1.13.

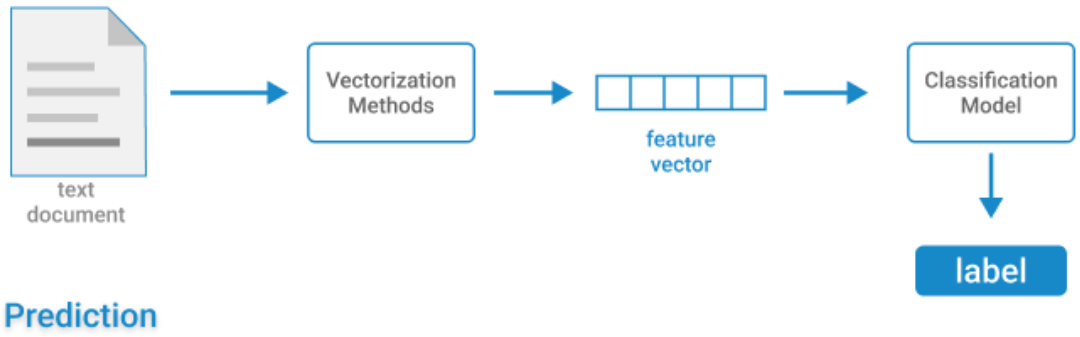

Рисунок 1.13 - Приклад прогнозування класу тексту за допомогою моделі машинного навчання

1.5.1 Вибір моделі класифікації

Сьогодні існує безліч моделей машинного та глибокого навчання та всі вони можуть бути використані під час вирішення тієї чи іншої задачі. Проте, вибір моделі для вирішення задачі - неоднозначне питання. Дуже складно ще до проведення експериментів сказати яка модель краще себе покаже. Враховуючи, що найкращі варіанти можуть бути неочевидними, наївним рішенням було б ретельно перепробувати всі можливі варіанти, відкидаючи деякі варіанти інтуїтивно. Однак, це не є ефективно, особливо з точки зору економічної ефективності.

Зазвичай, вибір моделі повинен бути заснованим на відповідях на наступні питання: "Як представлені текстові дані алгоритму, який очікує на

числовий вхід?", "Який тип моделі має бути використаним?", "Які параметри конфігурації мають бути використані для моделі?". На щастя, завдяки десятиліттям досліджень, на сьогоднішній день є доступ до великої кількості варіантів попередньої обробки даних і конфігурації моделі. Найкраще такі дані зведені у блок-схемі від компанії Google, яка допомагає звузити вибір моделей для експериментів базуючись на згаданих вище питаннях. Цю блок-схему продемонстровано на рисунку 1.14.

На наведеній нижче блок-схемі жовті прямокутники позначають процеси підготовки даних і моделі. Сірі та зелені прямокутники позначають варіанти вибору для кожного процесу. Зелені прямокутники позначають рекомендований компанією вибір для кожного процесу. Проте, цю схему рекомендовано використовувати лише як початковий варіант для первинного експерименту, оскільки вона забезпечує гарну точність за низьких обчислювальних втратах. Потім же модель все одно потребуватиме покращення.

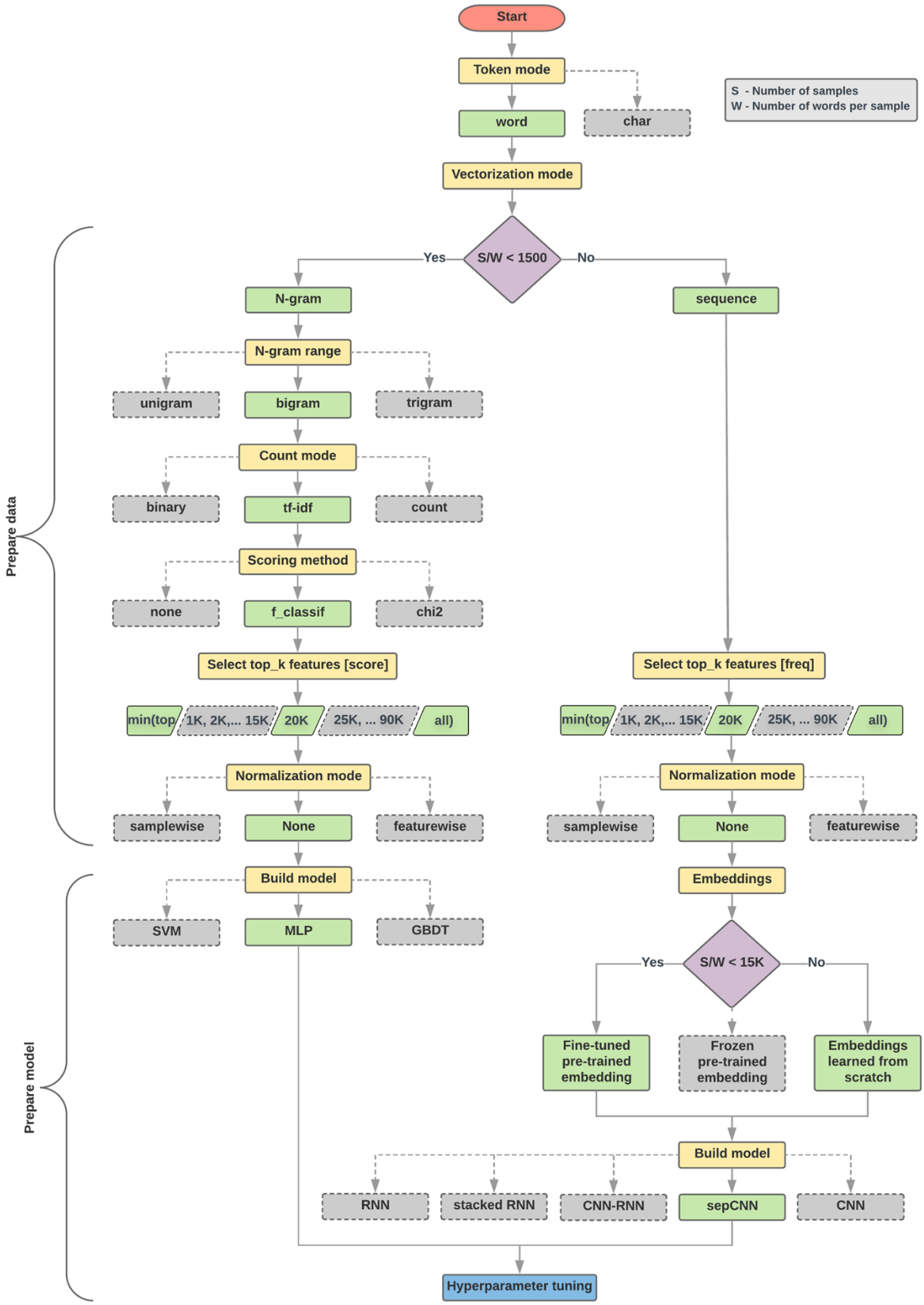

Рисунок 1.14 - Блок-схема вибору моделі класифікації тексту

Ця блок-схема допомагає визначити наступні два ключові моменти:

1. Яку модель машинного навчання потрібно використовувати?
2. Як повинні бути підготовлені дані для ефективного вивчення взаємозв’язку між текстом та міткою класу?

Відповідь на друге запитання залежить від відповіді на перше запитання, тобто метод обробки даних повинен залежати від вибору моделі, а не навпаки. Самі моделі можна розділити на дві категорії: ті, які використовують інформацію про порядок слів (моделі послідовності), і ті, які просто бачать текст як "мішки" (набори) слів (моделі n-грам). Типи моделей послідовностей включають згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN) та їхні різновиди. Типи моделей n-грам включають логістичну регресію, прості багатошарові персептрони (MLP або повнозв'язні нейронні мережі), моделі градієнтного бустинга дерев рішень і методи опорних векторів.

Ключовою метрикою для вибору категорії моделі розробники компанії визначили відношення "кількості вибірок" (S) до "кількості слів у вибірці" (W). Коли значення цього відношення невелике (<1500), невеликі багатошарові перцептрони, що приймають n-грами як вхідні дані, працюють краще або, принаймні, не гірше, ніж моделі послідовностей. MLP легко визначити і зрозуміти, і вони вимагають набагато менше часу для обчислень, ніж моделі послідовностей. Коли значення цього відношення велике (>= 1500) рекомендується використовувати модель послідовності. [14]

Забігаючи наперед до власного дослідження варто зазначити, що в даній роботі у якості текстових даних будуть використані дописи користувачів соцмережі Reddit. Приблизна кількість дописів яка буде використана дорівнює 1 мільйон, проте після чистки даних ця цифра може бути зменшена майже в два рази, а для моделі бінарної класифікації навіть у 5 разів. Середня кількість слів у очищеному дописі користувача налічує 100 слів (дуже маленькі дописи були видалені). Тоді метрика відношення кількості дописів до середньої

кількості слів у дописі може приймати значення від 2000 до 10000. У такому випадку рекомендовано використовувати моделі послідовностей та методи векторизації, що враховують порядок слів, наприклад, word2vec.

Проте, для перевірки зазначеною на блок-схемі гіпотези та врахувавши, що значення метрики для вибору не набагато вище 1500, в даній роботі також буде розглянута класифікація за допомогою моделей, які бачать текст як мішок слів (Bag-of-words).

У якості моделі послідовностей була вибрана вже знайома за темою дипломної роботи згорткова нейронна мережа (CNN) та word2vec метод векторизації. Як n-грам моделі буде виступати модель класифікації на основі градієнтного бустингу дерев рішень, яка отримала у спільноті назву XgBoost та вважається однією з найбільш універсальних для різних задач та в той же час ефективних. Для підготовки даних для неї буде використано векторизацію на основі метрики TF-IDF, як покращений варіант алгоритму Bag-of-words. Зазначені моделі класифікації будуть кратко описані дали в цьому розділі.

1.5.2 Алгоритм градієнтного бустингу для класифікації, пакет XgBoost.

Градієнтний бустінг - один з найпопулярніших алгоритмів машинного навчання для табличних наборів даних. Він досить потужний, щоб знайти будь-яку нелінійну залежність між цільовою функцією моделі та ознаками, і має чудову зручність використання, що дозволяє впоратися з відсутніми значеннями, викидами без будь-якої спеціальної обробки. Градієнтний бустинг - це один з варіантів ансамблевих методів, коли відбувається об’єднання кількох слабких моделей, щоб отримати кращу продуктивність в цілому.

Ідея алгоритма заключається у навчанні кожної наступної слабкої моделі передбачати помилки попередньої слабкої моделі, щоб вирівняти прогноз під час їх комбінації. Спочатку будується модель на навчальному наборі даних, потім будується друга модель, яка виправляє помилки, допущені в першій моделі, потім третя модель, яка виправляє помилки другої і так далі.

Повний алгоритм градієнтного бустингу для вирішення задачі бінарної класифікації виглядає наступним чином:

Першим кроком є створення початкового значення прогнозу F_0. L - це функція втрат, яка розраховується як перехресна ентропія:

$$F_0(x) = argmin_\omega \sum_{i=1}^{n} L(y_i, \omega), \tag{18}$$

$$L = -(y_i * \log(p) + (1 - y_i) * \log(1 - p)), \tag{19}$$

де:

y_i - це цільовий клас класифікації, який дорівнює або 0, або 1,

p - прогнозована ймовірність класу 1. Можна помітити, що L приймає різні значення в залежності від цільового класу,

ω - ваги які навчаються під час мінімізації.

Оскільки - log(x) є спадною функцією від x, то чим кращий прогноз (тобто збільшення p для y_i=1), тим менші будуть втрати.

На другому етапі починається ітеративна побудова M слабких моделей (дерев рішень), кожна з яких призначена виправити помилки попередньої. Всі наступні процеси кроку 2 повторюються M разів, m - індекс окремого дерева.

Спочатку обчислюються залишки як похідна функції втрат по відношенню до попереднього прогнозу F помножена на -1:

$$r_{im} = -\left[\frac{\partial L\left(y_i, F(x_i)\right)}{\partial F(x_i)}\right], i = 1..n, \quad (20)$$

де:

$F(x) = F_{m-1}(x)$ прогноз попередньої моделі, на першій ітерації це F_0,

i - індекс підвибірки,

m - індекс моделі.

Ця величина є від'ємним градієнтом, який показує напрямок і величину, в який функція втрат може бути мінімізована. Далі навчається регресійне дерево, де у якості ознак виступає той самий вектор x, проте у якості цілі вже виступають залишки r та утворюються листки R_{jm} $j = 1\ldots J_m$, де j позначає листок в дереві, m позначає індекс дерева, а велика літера J позначає загальну кількість листків.

На третьому етапі виконується мінімізація функції втрат у кожному листку дерева з метою знайти ваги:

$$\omega_{jm} = argmin_{\omega} \sum_{x_i \epsilon R_{jm}} L(y_i, F_{m-1}(x_i) + y), j = 1..J_m, \quad (21)$$

де:

$\sum_{x_i \epsilon R_{jm}} L$ - означає агрегацію втрат на всіх x_i, які належать до термінального вузла R_{jm}.

На останньому кроці виконується оновлення прогнозу комбінованої моделі:

$$F_m(x) = F_{m-1}(x) + v \sum_{j=1}^{J_m} (\omega_{jm} | x \epsilon R_{jm}), \quad (22)$$

де:

v - гіпер параметр швидкості навчання,

$(\omega_{jm} | \, x \in R_{jm})$ - означає що значення ω_{jm} враховується коли заданий x потрапляє в листок R_{jm}.

Оскільки всі листки є виключними, будь-який заданий єдиний x потрапляє лише в одну термінальну вершину і відповідне значення ω_{jm} додається до попереднього прогнозу F_{m-1}, а потім робиться оновлений прогноз. Менша швидкість навчання зменшує вплив додаткового прогнозу дерева, але вона також зменшує ймовірність того, що модель перенавчиться. [15] Рисунок 1.15 демонструє та підсумовує роботу алгоритму.

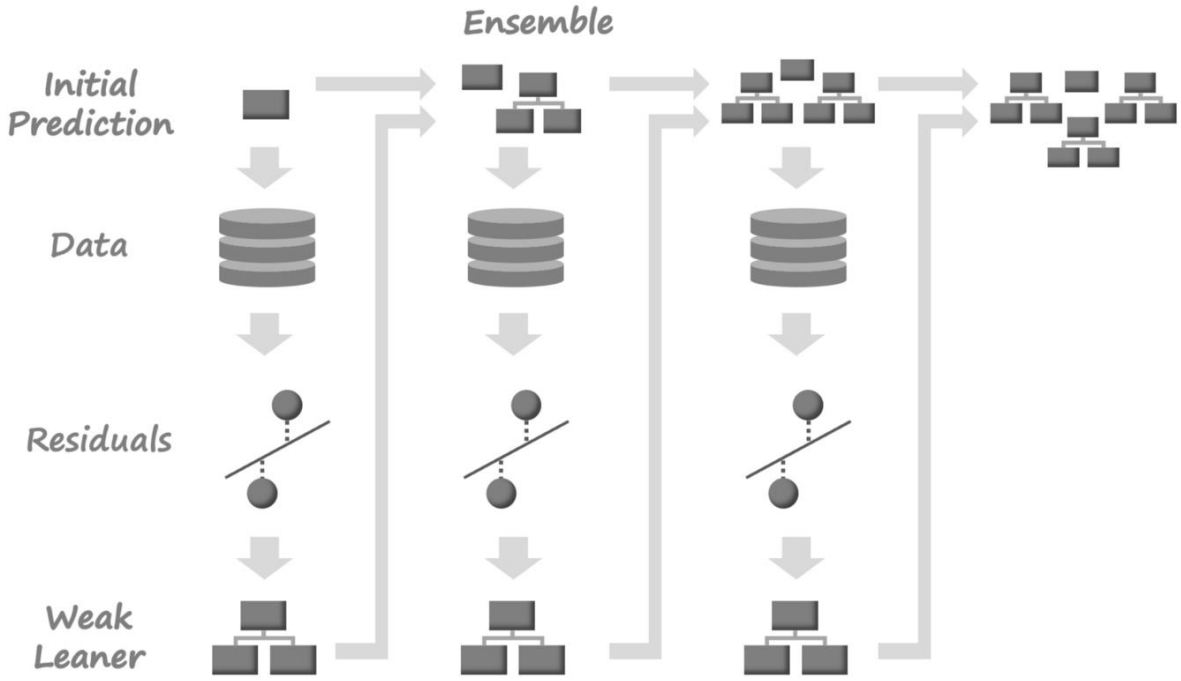

Рисунок 1.15 - Алгоритм градієнтного бустингу

1.5.3 Згорткові нейронні мережі в задачах класифікації тексту

Згорткові нейронні мережі (CNN) є найбільш широко вживаними архітектурами глибокого навчання в обробці зображень та розпізнаванні образів. Враховуючи їх домінування в області технічного зору, цілком природно, що їх намагаються впроваджувати в різних сферах машинного

навчання. Далі в цьому розділі буде розглянуто використання згорткових нейронних мереж в задачах обробки природної мови.

Згорткова нейронна мережа зазвичай складається з шарів згортки, об’єднання (pooling) та повнозв’язаних шарів. Концепція ковзання або згортання заздалегідь визначеного вікна (фільтра) вагів є центральною ідеєю, яка лежить в основі того, чому ці мережі так називають. Ілюстрація цієї концепції наведена нижче на рисунку 1.16.

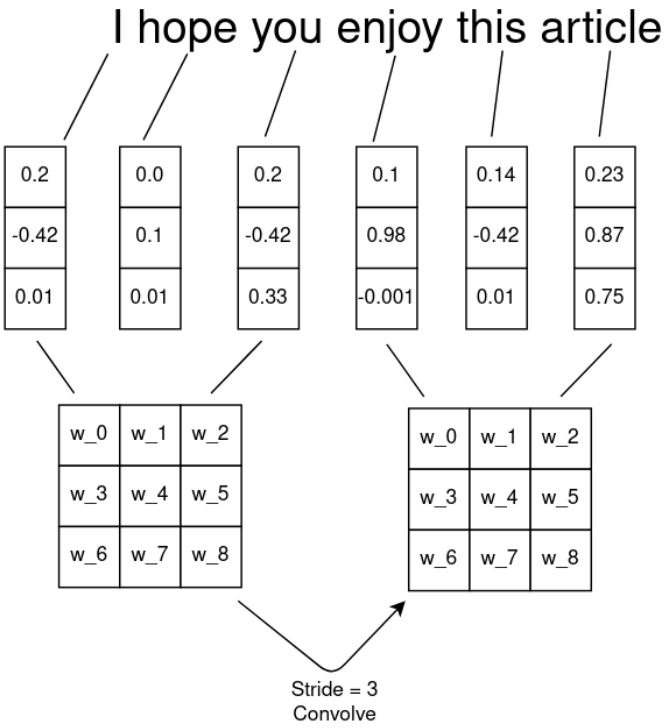

Рисунок 1.16 - Концепція роботи CNN в задачах NLP

Перше, на що слід звернути увагу, це метод, за допомогою якого кожне слово (лексема) представляється у вигляді 3-вимірних векторів слів. Такі методи було розглянуто у попередньому розділі. Далі, вагова матриця 3x3 переміщується горизонтально по реченню на один крок (або іншого заданого розміру), захоплюючі по три слова за раз. Ця вагова матриця називається фільтром; кожен фільтр також складається з функції активації, подібної до тих, що використовуються в нейронних мережах прямого поширення. Завдяки деяким математичним властивостям, функція активації ReLU (rectified linear unit - випрямлена лінійна одиниця) здебільшого використовується в ШНМ та

глибоких нейронних мережах. Повертаючись до класифікації зображень, загальна інтуїція цих фільтрів полягає в тому, що кожен фільтр може виявляти різні особливості зображення. Чим глибше фільтр, тим більша ймовірність того, що він захопить більш складні деталі. Так, наприклад, найперші фільтри у згортковій мережі виявляють прості особливості, такі як краї та лінії, але в самому кінці можуть бути здатні виявити певні типи тварин. Навчання вагів у фільтрах відбувається за допомогою методу зворотного розповсюдження помилки.

Наступним важливим кроком є розрахунок результату операції згортки (виходу шару згортки). Для більш якісного пояснення операції згортки буде використано рисунок 1.17.

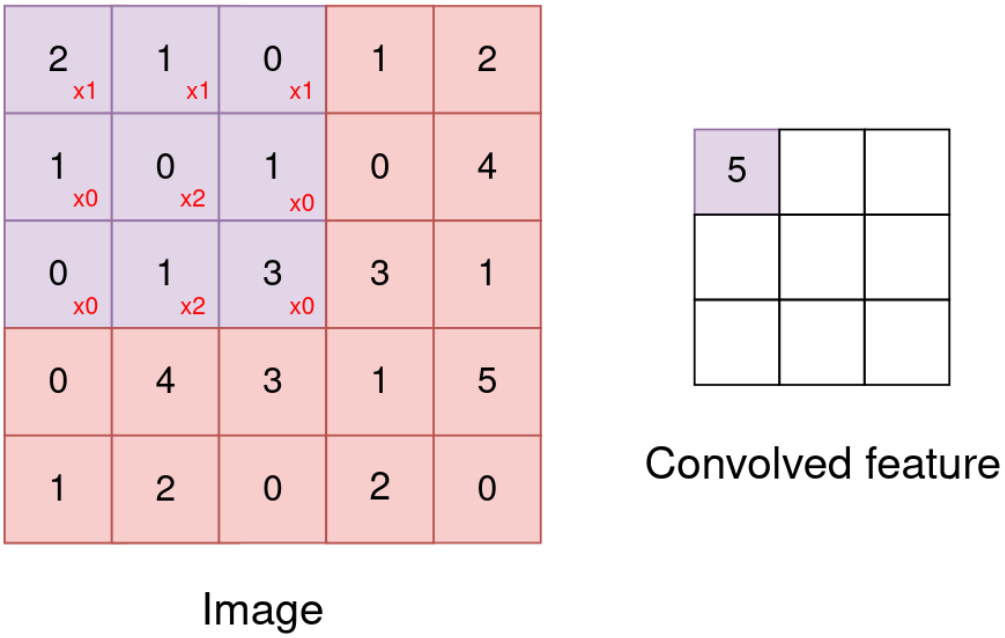

Рисунок 1.17 - Операція згортки

На рисунку 1.17 представлено зображення 5*5, тобто квадратну матрицю, і фільтр 3*3. Вихідний шар обчислюється шляхом поелементного добутку між кожним елементом фільтра та вхідним тензором. Такий розрахунок необхідно провести в кожному розташуванні тензора і підсумовувати для отримання вихідного значення у відповідному положенні

вихідного тензора, що називається картою ознак. Наведений вище приклад ілюструє, як розраховується перша комірка вихідного шару; червоні числа на зображенні представляють ваги у фільтрі. [16]

При застосуванні до тексту зазвичай використовується одновимірний фільтр. Рисунок 1.18 демонструє приклад згортки текстових даних одновимірним фільтром, який ковзає на 3 кроки по горизонталі.

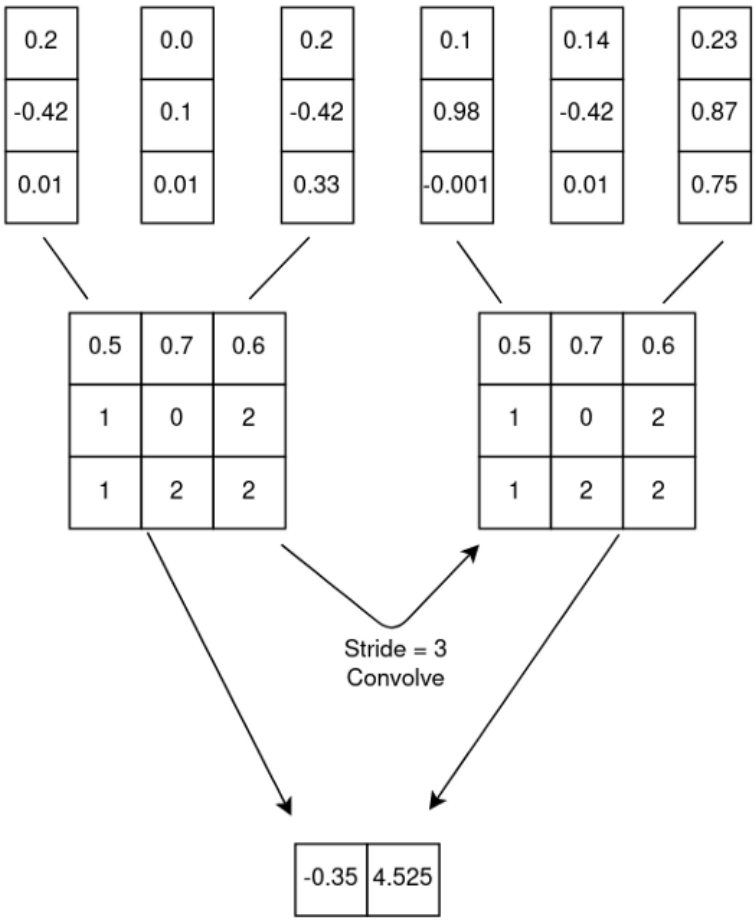

Рисунок 1.18 - Приклад застосування згортки на текстових даних

Варто зазначити, що в останніх двох прикладах розмір вихідного файлу був меншим за розмір вхідного. Також не важко уявити собі випадки, коли фільтр не зовсім точно відповідає матриці із заданою кількістю кроків. Для подолання цих ускладнень можна використовувати операцію доповнення (padding) двома способами:

- заповнити зовнішні краї вхідного тензора нульовими векторами (zero-padding);
- ігнорувати частину матриці, яка не підходить під фільтр (valid padding).

Тип падінгу є гіпер параметром моделі на кожному шарі згортки. Рисунок 1.19 демонструє операцію доповнення.

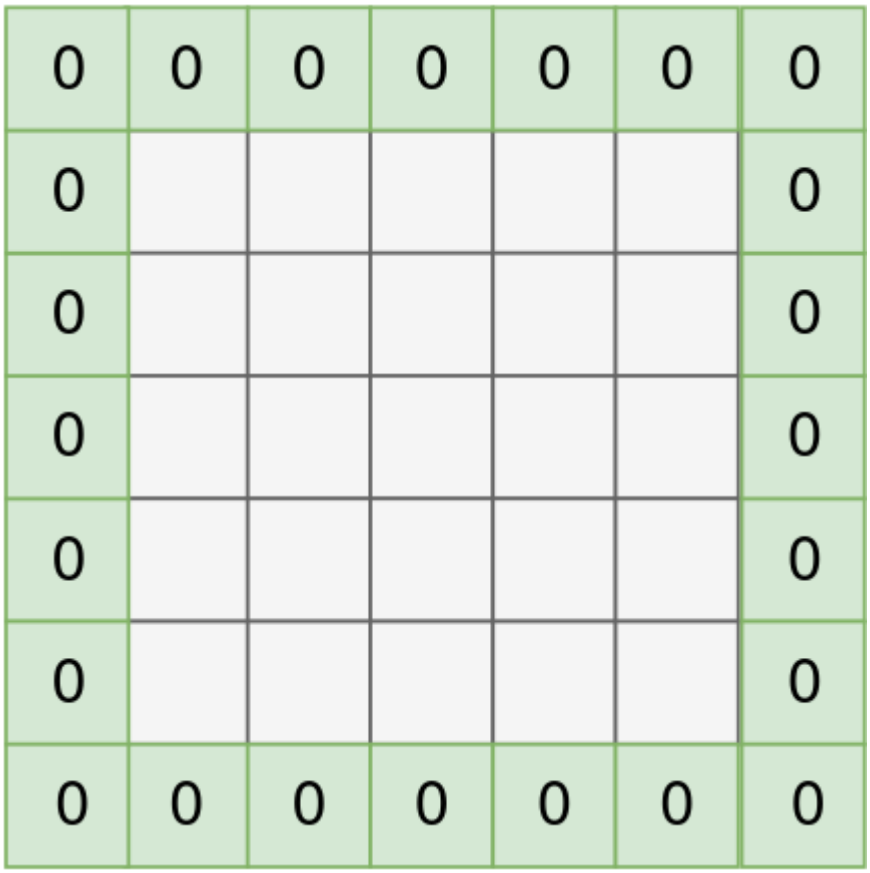

Рисунок 1.19 - Операція доповнення (Padding)

Іншою важливою операцією в згорткових нейронних мережах є операція об’єднання. Основна ідея полягає в тому, щоб розділити вихідні шари на підсекції і обчислити значення, яке найкраще представляє вихідні дані. Причина, чому це настільки ефективно, полягає в тому, що це допомагає алгоритму вивчати представлення даних більш високого порядку, зменшуючи при цьому кількість параметрів. Виділяють наступні типи операції об'єднання:

- сумарне об'єднання;
- максимальне об'єднання;

- середнє об'єднання.

Демонстрація операції максильного об’єднання наведена на рисунку 1.20.

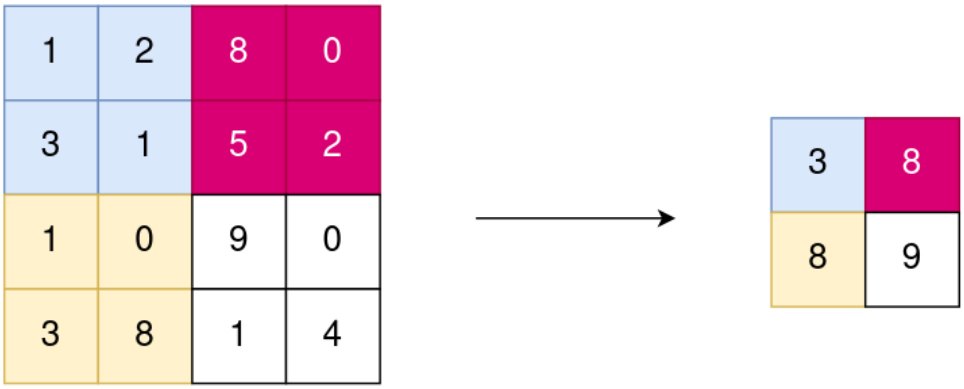

Рисунок 1.20 - Операція об’єднання (Max pooling)

Останній шар згорткової нейронної мережі представлено повністю пов'язаним шаром. Він отримує вхідні дані від попередніх шарів об'єднання та згортки, після чого виконує завдання класифікації. Для цього останній нейрон використовує сигмоїдну функцію активації (повертає значення між (0,1)) у випадкі бінарної класифікації або софтмакс у випадку багато класової задачі. Рисунок 1.21 демонструє архітектуру класичної згорткової нейронної мережі.

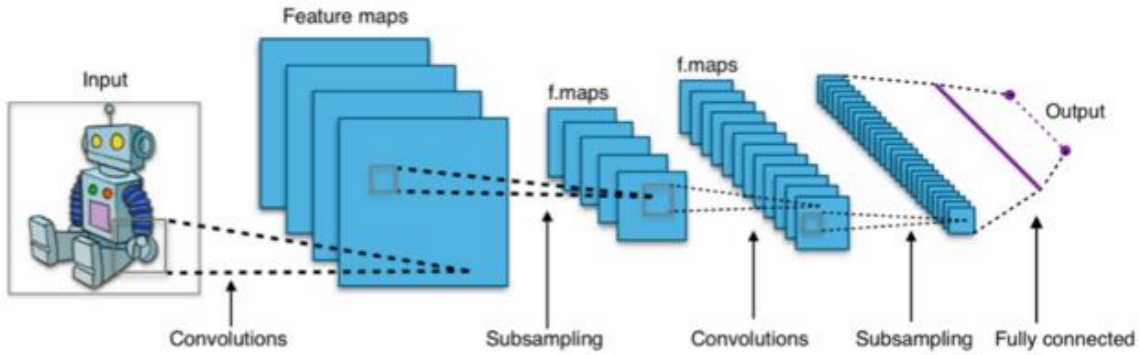

Рисунок 1.21 - Архітектура згорткової нейронної мережі

1.6 Висновки

Методи обробки природної мови в останні роки отримали великий поштовх для розвитку та на сьогоднішній день, їх використання призвело до приголомшливих досягнень у різних сферах, від медицини до науки та розваг.

В даному розділі було розглянуто основні теоретичні відомості, що стосуються вирішення найбільш популярної задачі обробки природної мови - класифікації тексту. Було розглянуто методи попередньої обробки текстових даних, методи векторизації текстових даних та моделі класифікації, які є також необхідними при роботі з текстовими даними та як правильно вибирати потрібну модель.

Було розглянуто переваги та недоліки кожного із методів, проблеми та рішення, які можна зустріти при вирішенні конкретних задач. Крім того, було акцентовано увагу на теоретичних аспектах які будуть використані та проблемах, які будуть вирішені при розробці власної моделі машинного навчання для ідентифікації психічних розладів на основі текстів у соцмережах.

2 ВИКОРИСТАННЯ ПОПЕРЕДНЬО НАВЧЕНИХ МОДЕЛЕЙ ДЛЯ ВЕКТОРИЗАЦІЇ ТЕСТОВИХ ДАНИХ. ДОСЯГНЕННЯ В СФЕРІ ВИКОРИСТАННЯ МЕТОДІВ ОБРОБКИ ПРИРОДНОЇ МОВИ ДЛЯ ВИЯВЛЕННЯ ПСИХІЧНИХ ЗАХВОРЮВАНЬ.

2.1 Використання попередньо навчених моделей для векторизації текстових даних

2.1.1 Ідея передавального навчання (Transfer learning)

Передавальне навчання - це повторне використання попередньо навченої моделі на новій проблемі. В даний час воно дуже популярне в глибокому навчанні, оскільки може навчати глибокі нейронні мережі з порівняно невеликим обсягом даних. Це дуже корисно, оскільки більшість реальних проблем, як правило, не мають мільйонів маркованих точок даних для навчання таких складних моделей.

При передавальному навчанні, знання вже навченої моделі машинного навчання застосовуються до іншої, але пов'язаної з нею проблеми. Наприклад, якщо простий класифікатор навчився передбачати, чи містить зображення кота, то можна використовувати знання, які модель отримала під час навчання, для розпізнавання інших об'єктів, таких як конкретні породи котів.

Загальна ідея полягає в тому, щоб використовувати знання, отримані моделлю при виконанні завдання з великою кількістю доступних маркованих навчальних даних, в новому завданні, в якому не так багато даних. Замість того, щоб починати процес навчання з нуля, завдяки даному підходу, навчання починається з шаблонів, отриманих при вирішенні спорідненої задачі.

В моїй бакалаврський роботі вже було розглянуто основні принципи передавального навчання у вирішенні задач комп’ютерного зору. Там, нейронні мережі зазвичай намагаються виявити низькорівневі ознаки в верхніх шарах, більш складні форми в середньому шарі і деякі специфічні для

завдання особливості в останніх шарах. При передавальному навчанні зазвичай виконується передача верхніх та середніх вагів мережі і перенавчаються лише останні шари.

Такий самий підхід можна застосувати і для вирішення задач обробки природної мови, а саме, для векторизації текстів контекстними методами, як word2vec, що побудовані саме на використанні глибоких штучних нейронних мереж для прогнозування контексту у якому було вжите слово. [17]

2.1.2 Передавальне навчання в задачі векторизації тексту методом word2vec

Як вже було зазначено у розділі 1, основна ідея методу word2vec полягає у навчанні представлень слів або символів у текстах з метою споріднити саме схожі за використанням у деякому контексті слова. Представлення слова - це щільний вектор, який представляє слово. За замовчуванням, вектори представлень ініціалізуються випадковим чином, потім поступово покращуються під час фази навчання з алгоритмом градієнтного спуску на кожному кроці зворотного поширення, так, що схожі слова або слова в одному лексичному полі, або зі спільною основою будуть в кінцевому підсумку близькі з точки зору відстані в новому векторному просторі. Приклад результату такого навчання представлень слів наведено на рисунку 2.1

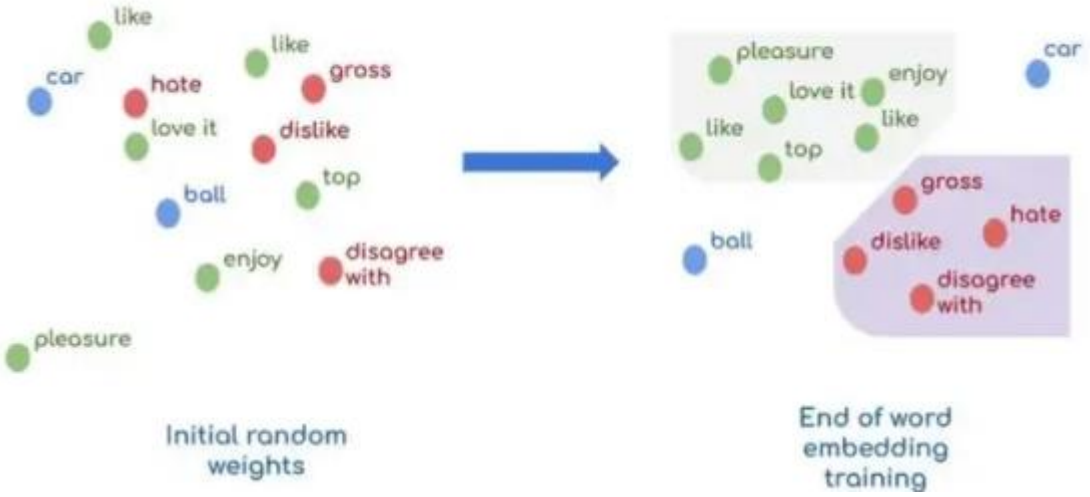

Рисунок 2.1 - Приклад навчання представлень слів методом word2vec

Проблемою при вирішенні задач контекстної векторизації є те, що набори даних, зібрані для вирішення конкретної задачі, мають словник слів та їх використання у контексті заздалегідь звужено до конкретної предметної області. Тому, навчання представлень слів на таких наборах даних може бути недостатньо ефективним через звужений простір використання слів у контексті. Так, наприклад, при навчанні на “вузькому” наборі даних, мережа може вивчити дуже специфічні використання слів у контексті, що притаманні саме набору даних, при цьому інші варіанти використання деякого слова у контексті, які зустрічаються у світі за межами набору даних, мережа виявити не зможе.

Саме для вирішення даної проблеми і використовується передавальне навчання. Воно полягає у використанні вже попередньо навчених на дуже великих та різноманітних наборах даних представлення слів, що виступають в якості початкових представлень слів під час навчання на більш специфічному наборі даних. Таким чином, мережа не тільки не втрачає всі можливі контекстні варіанти використання слова, а ще і зближує слова, схожі за контекстним використання, у більш специфічному наборі даних, який використовується саме для вирішення поставленої задачі. [18]

2.1.3 Найсучасніші попередньо навчені моделі векторного представлення слів

Представлення тестів можна умовно розділити на 2 класи: представлення на рівні слів та на рівні символів. Найбільш відомими мережами, що навчені для представлення символів є ELMo та Flair, проте в даній роботі я хочу розглянути три найбільш популярні моделі заздалегідь навчених представлень на рівні слів: Word2Vec від Google, GloVe від Стенфордського університету та fasttext від Facebook.

2.1.3.1 Word2Vec від Google

Word2Vec Google - один з найпопулярніших та найбільших наборів попередньо навчених представлень слів, розроблених компанією Google. Він був отриманий завдяки навчанню звичайної нейронної мережі прямого поширення з одним прихованим шаром з використанням підходів CBOW та Skim Gram парадигми word2vec. Ця модель є у відкритому доступі, її може завантажити кожен та вона важить всього 1.5 гігабайти. Вона включає вектори слів для словника з 3 мільйонів слів та словосполучень, які були навчені на приблизно 100 мільярдах слів з набору даних Google News. Довжина вектора - 300 ознак. В мові програмування python реалізація даної моделі доступна з пакетом для обробки природної мови gensim. [19]

2.1.3.2 GloVe від Стенфордського університету

Global Vectors (GloVe) - це алгоритм навчання без вчителя, який використовується для контекстного представлення слів у вектори, проте він дещо відрізняється від підходу word2vec. Не зважаючи на це, такі векторні представлення можна використовувати як початкові дані для навчання будь-якої мережі. За допомогою цього алгоритму був створений великий набір даних, що містить вектори англійських слів, попередньо навчені на дуже великому корпусі текстів з Вікіпедії. Усі лексеми пишуться малими літерами. Цей набір даних містить 50-вимірні, 100-вимірні, 200-вимірні та 300-вимірні попередньо підготовлені вектори слів.

GloVe тісно асоціюється з Word2Vec: алгоритми з'явилися приблизно в один і той самий час і спираються на інтерпретованість векторів слів. Модель GloVe намагається вирішити проблему ефективного використання статистики збігів. GloVe мінімізує різницю між добутком векторів слів і логарифмом ймовірності їхньої спільної появи за допомогою стохастичного градієнтного спуску. Отримані уявлення відображають важливі лінійні підструктури векторного простору слів: вдається пов'язати разом різні супутники однієї планети або поштовий код міста з його назвою.

У Word2Vec частота спільної зустрічальності слів не має великого значення, вона лише допомагає генерувати додаткові навчальні вибірки. GloVe враховує спільну зустрічальність, а не покладається тільки на контекстну статистику. Вектори слів групуються разом на основі їхньої глобальної схожості.

Перевагами метода Glove є проста архітекстура, що працює без нейронної мережі, модель працює та навчається набагато швидше. Крім того, GloVe покращує Word2Vec. Вона додає частоту зустрічальності слів і випереджає Word2Vec на більшості тестів.

Проте, хоча матриця спільної зустрічаємості надає глобальну інформацію, GloVe залишається навченою на рівні слів і дає небагато даних про речення та контекст, у якому слово використовується. Отже, Glove більше використовується саме для створення початкових векторних представлень у глобальному просторі, які навчаються іншими методами на більш специфічних наборах даних. [20]

2.1.3.3 Fasttext

Бібліотека fastText - ще один серйозний крок у розвитку моделей природної мови. У її розробці взяв участь Томаш Міколов, який розробив принцип Word2Vec. Для векторизації слів використовуються одночасно і skim-gram, і negative sampling, і алгоритм CBOW.

До основної моделі Word2Vec додано модель символьних n-грам. Кожне слово представляється композицією декількох послідовностей символів певної довжини. Наприклад, слово they залежно від гіпер параметрів може складатися з "th", "he", "ey", "the", "hey". По суті, вектор слова - це сума всіх його n-грам. Результати роботи класифікатора добре підходять для слів з невеликою частотою зустрічальності, оскільки вони поділяються на n-грами. На відміну від Word2Vec і Glove, модель здатна генерувати представлення для невідомих слів. У відкритому доступі є попередньо навчені моделі для 290 різних мов. Проте, навчання в даній моделі відбувається на рівні слів та їх частин, отже немає інформації про речення або контекст, у якому використовується слово. [21]

2.2 Досягнення в сфері використання методів обробки природної мови для виявлення психічних захворювань.

Після еволюції та великого прогресу методів обробки природної мови після 2013 року, з'явилось дуже багато досліджень, що стосуються застосування згаданих методів у різних предметних областях. Не пройшли зроблені відкриття і повз аналіз психічного здоров'я людей. Перші дослідження були пов'язані з аналізом емоціонального забарвлення текстів з метою вирізняти депресію та суїцидальні схильності за текстами у соцмережах. Проте, з розвитком методів обробки природної мови, накопиченням великих об'ємів текстових даних та методів які попередньо на них навчені, з'явилися і нові більш серйозні дослідження. Найкращу систематизацію всіх досліджень на цю тему у хронологічному порядку було зроблено Кітом Гарріганом, який створив публічний репозиторій на платформі Github та зібрав близько 50 досліджень, систематизувавши їх за джерелами даних, видами психічних розладів та роками видання. [22] У даній роботі я хотів би виділити дослідження в якому було запропоновано дуже неординарний підхід до збору даних, що стосується психічних захворювань, в якому продемонстровано найбільш комплексне вирішення задачі з використання сучасних методів обробки природної мови.

Таким дослідженням є робота 'A deep learning model for detecting mental illness from user content on social media' випущена відділом науки про взаємодію з університету Сонгюнгван, Сеул, Корея. В даній роботі досить чітко піднімається проблема необхідності ранньої ідентифікації психічних захворювань та запропоновано комплексний підхід до класифікації зібраних у Reddit даних за допомогою моделей глибокого навчання з методами обробки природної мови для виявлення користувачів з потенційними психічними захворюваннями на основі їхніх дописів.

В даному дослідженні були запропоновано підхід до збору даних з публічної соціальної мережі Reddit, яка відрізняється від персональної стрічки сервісів соціальних мереж у вираженні емоцій користувачів. Завдяки орієнтованості на тематичні канали даної соцмережі пов’язані з психічними захворюваннями, їм вдалося зібрати великий набір вже розмічених даних. Так, хоча публікації в соціальних мережах не можуть чітко вказувати на симптоми порівняно з публікаціями на особистих сторінках користувачів, які можуть свідчити про наявність у них клінічних психічних захворювань, соціальні мережі можуть бути використані для ідентифікації осіб, які страждають на психічні розлади, оскільки вони відносно точно діляться своїми симптомами в умовах напіванонімності.

Крім того, в даному дослідженні запропоноване досить комплексне вирішення проблеми та моделі для бінарної класифікації шести психічних захворювань: емоційно нестабільний розлад особистості, депресія, тривожні розлади, шизофренія, аутизм та біполярні розлади. З використанням глибоких нейронних мереж у поєднанні з методом word2vec їм вдалося досягти високої точності у вирішенні задачі класифікації, проте за мірою F1, яка більш повноцінно оцінює результати бінарної класифікації, лише депресія була ідентифікована з точністю 79%. Для всіх інших моделей дана міра була на рівні дещо вищому або меншому за 50%, що вказує на не найкращу ефективність моделей.

Проте, ідеї закладені у даному дослідженні є дуже цікавими та мають право на розвиток, тому деякі з них були покладені в основу мого власного дослідження з метою покращити запропонований підхід та виправити його недоліки. [23]

2.3 Висновки

Використання контекстних методів векторизації текстів потребує великої кількості дуже різноманітних даних. При їх використанні на наборі даних з певної предметної області - висока ймовірність недонавчання контекстних представлень слів для роботи за межами набору даних. Для вирішення даної проблеми в даному розділі було розглянуто методи передавального навчання в обробці природної мови та розглянуто три найбільш сучасні моделі, що попередньо навчені на дуже великих та різноманітних наборах даних. Усі ці моделі мають багато спільного, але кожна з них має використовуватися у відповідному контексті.

Крім того, в даному розділі були розглянуті досягнення в ідентифікації психічних захворювань методами машинного навчання та було підкреслено основні аспекти, які вплинули на цю роботу.

3 РОЗРОБКА ВЛАСНОЇ МОДЕЛІ МАШИННОГО НАВЧАННЯ ДЛЯ ІДЕНТИФІКАЦІЇ ПСИХІЧНИХ ЗАХВОРЮВАНЬ НА ОСНОВІ ПУБЛІКАЦІЙ У СОЦМЕРЕЖАХ.

На сьогоднішній день практично кожна людина користується соціальними мережами. Користувачі соціальних мереж часто діляться своїми почуттями, емоційним станом у своїх публікаціях, постах та повідомлення. У цьому розділі було розроблено модель глибокого навчання, щоб визначити психічний стан користувача на основі інформації, яку він публікує. Для вирішення даної задачі були зібрані публікації людей у спільнотах пов’язаних з психічним здоров’ям в соціальній мережі Reddit. В даному розділі буде проаналізовано та вивчено розмічені дані, написані користувачами та буде запропонована модель, яка може з високою точністю визначити, чи належить публікація користувача до певного психічного розладу. ВООЗ виділяє 8 основних класів психічних захворювань: депресію, тривогу, біполярний розлад, посттравматичний розлад, шизофренію, розлади травлення, аутизм та асоціальні розлади. Саме на ідентифікації цих основних класів і буде побудовано подальше дослідження. Розроблена модель може допомогти ідентифікувати потенційних людей із психічними захворюваннями на основі їхніх публікацій. У цьому розділі також будуть проаналізовано результати запропонованої моделі.

3.1 Збір даних. Опис даних та статистика. Дослідження специфіки вхідних даних, попередня підготовка даних, пояснювальний аналіз даних.

3.1.1 Збір даних

Для збору даних було обрано соцмережу Reddit. Її особливістю, яка задовольняє потреби даного дослідження є те, що вона налічує безліч тематичних каналів (англ. subreddit) в яких люди діляться своїми думками, проблемами, почуттями згідно обраної теми. Крім того, вона надає зручний програмний інтерфейс, за допомогою якого можна зручно завантажити дописи користувачів.

Враховуючи специфіку теми, було обрано 26 тематичних каналів, по одному-два канали для кожного психічного розладу, декілька більш загальних каналів на тему психічного здоров'я, а також канали зовсім іншої тематики. З кожного такого каналу було завантажено всі публікації користувачів, після чого знайдені саме такі, в яких люди зізнаються у своїй хворобі. Зробивши класифікацію користувачів соцмережі за розладами було отримано вибірку з розміченими парами типу "Користувач - Психічний розлад". Треба врахувати, що абсолютна більшість користувачів даної соціальної мережі пишуть свої дописи в інших тематичних каналах. Тому, для кожного користувача з отриманої вибірки було завантажено всі дописи, які вони створили в даній соцмережі. Таким чином було отримано вибірку розмічених текстів виду "Допис користувача - Психічний розлад користувача". Пости з прямим згадуванням хвороби були видалені з вибірки, щоб навчитись ідентифікувати хворобу саме по загальним ознакам мовлення. Використаний алгоритм збору даних візуалізує малюнок 3.1.

Головне припущення, яке було використано для збору даних можна сформулювати наступним чином: людина, яка хвора на певне психічне захворювання, пише пост у канал, в якому люди діляться як жити з цим

психічним захворюванням. Якщо людина ділиться тим, що її також турбує ця проблема - всі її пости позначаються міткою цього захворювання.

Рисунок 3.1 - Алгоритм збору даних

В результаті зібрані дані були представлені у вигляді реляційних таблиць у .csv файлах, які містять наступні поля:

- subreddit - тематика каналу, мітка психічного захворювання;
- author - автор допису;
- date - дата допису;
- title - заголовок допису;
- post - текст допису;
- meta - мета дані допису.

Приклад організації зібраних даних наведено на рисунку 3.2.

subreddit	author	date	post
addiction	SearchSmegmaongoogle	2018/01/01	Deciding to go of tramadol Well after never taking a tablet before in my life, I got these for a back injury end up a I'll still take them after I detox for a while because I do need them for my back but I'll only be sticking to x1 200M The only thing is, the withdrawels are horrible, worse than herion according to many H addicts, wish my luck, as Wish me luck
addiction	420somthing	2018/01/02	My vyvanse addiction... It has gotten pretty bad yesterday I went without taking it, it was a living hell I got meg
addiction	PurposedPorpoise	2018/01/02	Quitting coke and nicotine I'm gonna start by saying I've been doing really well and am very proud of myself. I ha As for nicotine, I'd been vaping for 2 or 3 years. My vape broke recently and instead of buying another one I figu Unfortunately I have been undergoing an immensely stressful period because of my work, which is a bad time to I guess I just wanted to vent a bit. I just had a cigarette on my break and I smell awful. I can reliably get a few d Thanks for reading.
addiction	iamauserunknown	2018/01/02	Is it OK to leave a drug addict you love? Many details now. My gf (of a little over five years) became addicted t One final detail to note is that she is very resistant to any kind of talk therapy, and my impression based on her What do I need to do / should I be doing. I'm about at my wit's end, and I'm about to kick her out, but I do have v TL;DR: What is my move with a long-term gf who has emotional/personality issues, failed rehab, and continues

Рисунок 3.2 - Приклад організації зібраних даних

Всього було зібрано 1 000 000 публікацій, всі англійською мовою. Розподіл зібраних даних за тематикою наведено в таблиці 3.1.

Таблиця 3.1 - Розподіл зібраних даних за тематикою наведено.

Оригінальна назва тематичного каналу	Тематика	Психічний розлад	Загальне здоров'я	Інша тематика	Кількість текстів
legaladvice	Юридичні поради	НІ	НІ	ТАК	164233
depression	Депресія	ТАК	НІ	НІ	117331
personalfinance	Фінанси	НІ	НІ	ТАК	102848
jokes	Жарти	НІ	НІ	ТАК	94505
suicidewatch	Самогубства	ТАК	НІ	НІ	66161

anxiety	Тривожні розлади	ТАК	НІ	НІ	57671
fitness	Фітнес	НІ	НІ	ТАК	49844
adhd	Синдром порушення активності та уваги	ТАК	НІ	НІ	45631
mentalhealth	Психічне здоров’я	НІ	ТАК	НІ	45332
parenting	Виховання дітей	НІ	НІ	ТАК	33868
conspiracy	Змови та чутки	НІ	НІ	ТАК	29847
bpd	Емоційно нестабільний розлад особистості	ТАК	НІ	НІ	24294
lonely	Самотність	ТАК	НІ	НІ	23635
socialanxiety	Антисоціальні розлади	ТАК	НІ	НІ	22996
guns	Зброя	НІ	НІ	ТАК	22973
meditation	Медитація	НІ	ТАК	НІ	16475
EDAnonymous	Розлади харчової поведінки	ТАК	НІ	НІ	14577
divorce	Розлучення	НІ	НІ	ТАК	12594
autism	Аутизм	ТАК	НІ	НІ	8869

schizophrenia	Шизофренія	ТАК	НІ	НІ	8712
healthanxiety	Тривожні розлади	ТАК	НІ	НІ	8648
ptsd	Посттравматичні стресові розлади	ТАК	НІ	НІ	8643
addiction	Залежності	НІ	ТАК	НІ	7641
alcoholism	Алкоголізм	ТАК	НІ	НІ	5911
bipolarreddit	Біполярні розлади	ТАК	НІ	НІ	5780
covid19_support	Підтримка постраждалих від Ковід 19	НІ	ТАК	НІ	981

Як можна помітити, найбільше текстів було зібрано для депресії та тривожних розладів, 117 та 65 тисяч відповідно. Найменше - для біполярних розладів, шизофренії, аутизму та посттравматичних розладів. Крім того, 581 тисяча текстів не є пов’язаною з конкретним психічним захворюванням, проте буде використана у дослідженні для покращення результатів класифікації. Серед конкретних психічних захворювань є також ті, які не будуть брати прямої участь у їх ідентифікації, а лише будуть використані для більш якісного відділення інших класів: алкоголізм, самогубства, синдром порушення активності, емоційно нестабільний розлад особистості.

3.1.2 Процедура попередньої обробки даних

Процедура попередньої обробки даних для зібраних публікацій представлена на рисунку 3.3. Спочатку було проведено розширення всіх скорочень, які наявні у текстах. Це необхідно тому, що дані тексти є неформальними та користувачі соцмереж часто вдаються за використання скорочень. Для їх розширення була використана база даних найбільш популярних скорочень слів англійської мови. Під час подальшої обробки текстів публікацій було проведено токенізацію, а саме розбивку тексту на масив слів та символи. Для кожного масиву слів та символів було проведено видалення всієї пунктуації, пробілів та спецсимволів, всі слова було приведено до нижнього регістру, було видалено, так звані, стоп-слова - слова що є часто вживаними та не несуть великого сенсу. Всі наведені операції було виконано за допомогою набору інструментів для обробки природної мови (NLTK), реалізований на Python. Для приведення токенізованих слів до їх кореневої форми було використано алгоритм Портера Стеммера, який застосовує послідовний ряд правил, що відсікає закінчення та суфікси та не містить бази слів, тому спрацьовує швидко, але не завжди безпомилково.

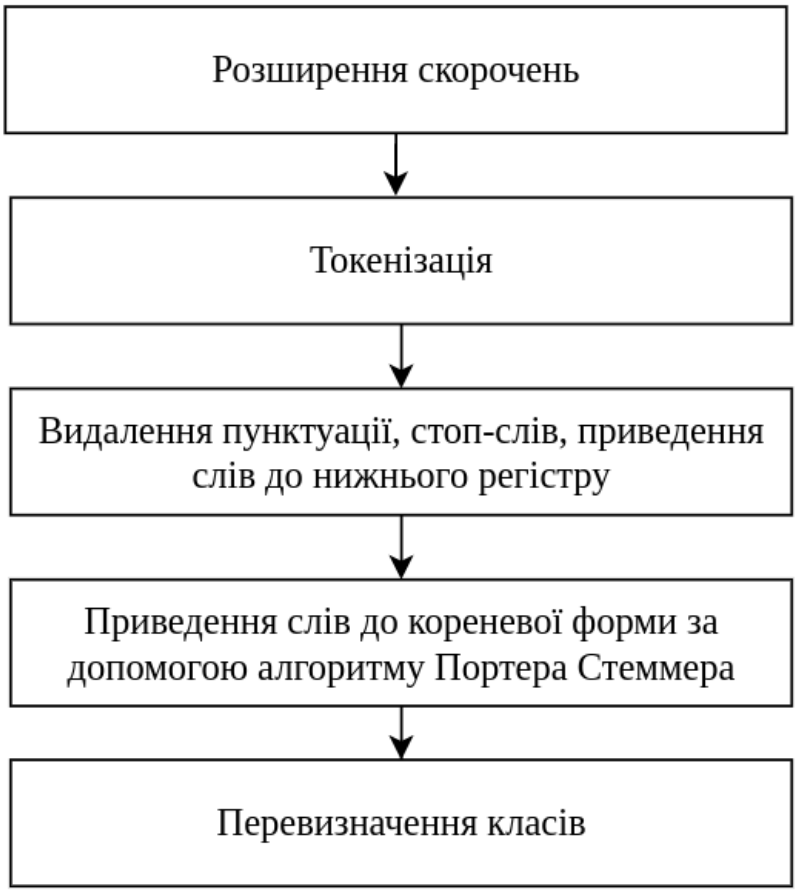

Рисунок 3.3 - Процедура попередньої обробки даних

Фінальним етапом попередньої обробки стала розмітка даних на нові класи: дописи, які стосуються загального психічного здоров’я та дописи на інші тематики було агреговано у два класи, які доповнили 8 класів для кожного з психічних захворювань. Також було проведено розмітку даних для майбутньої бінарної класифікації, тобто у відповідність кожному тексту була поставлена мітка 1 або 0 для кожного психічного розладу. Розподіл кількості дописів для кожного з класів наведено в таблиці 3.2.

Таблиця 3.2 - Розподіл кількості дописів для кожного з класів.

Назва класу	**Кількість дописів**
Пости на іншу тему	510712

Інші психічні розлади	212426
Тривожні розлади	66319
Посттравматичні стресові розлади	8643
Біполярні розлади	5780
Антисоціальні розлади	46631
Депресія	117331
Аутизм	8869
Шизофренія	8712
Розлади харчової поведінки	14577

3.1.3 Пояснювальний аналіз даних

Після попередньої обробки даних було отримано очищені від очевидного шуму текстові дані для яких можна робити числові представлення за допомогою методів векторизації. Проте, з метою краще зрозуміти специфіку даних та виконати додаткову обробку - було проведено пояснювальний аналіз даних.

По-перше, було проаналізовано розподіл кількості слів у текстах. Це необхідно, в першу чергу, для звуження набору моделей, які можуть бути використані під час вирішення задачі класифікації. Так, середня довжина допису для всього набору даних складає 82.4 слова. Розподіл ймовірностей для

довжини допису на всьому наборі даних та для кожного з класів продемонстровано на рисунках 3.4 і 3.5 відповідно.

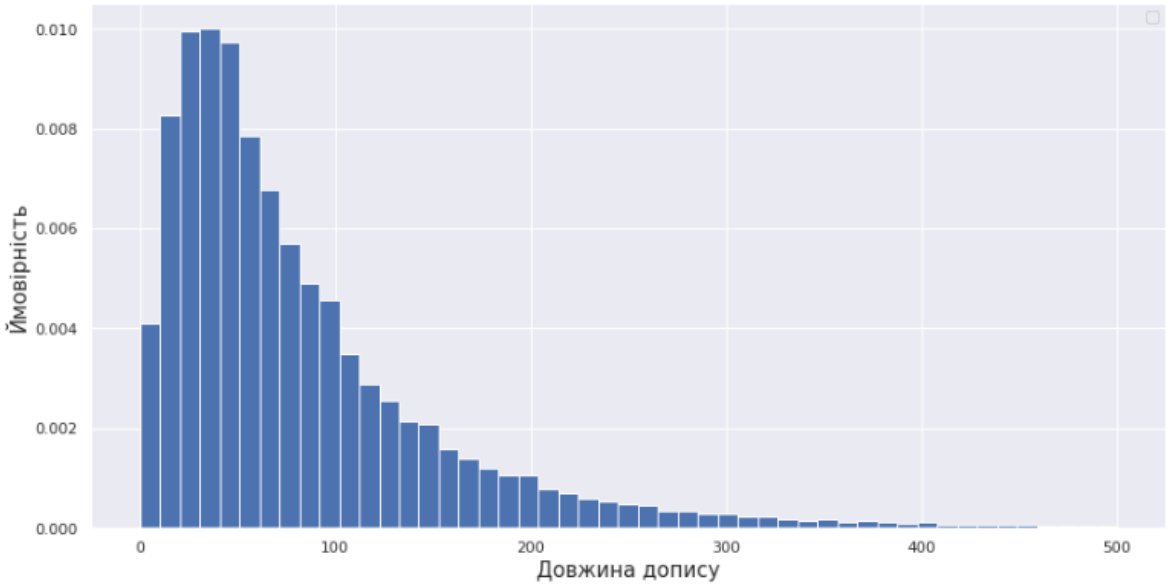

Рисунок 3.4 - Розподіл ймовірностей довжини допису на повному наборі даних

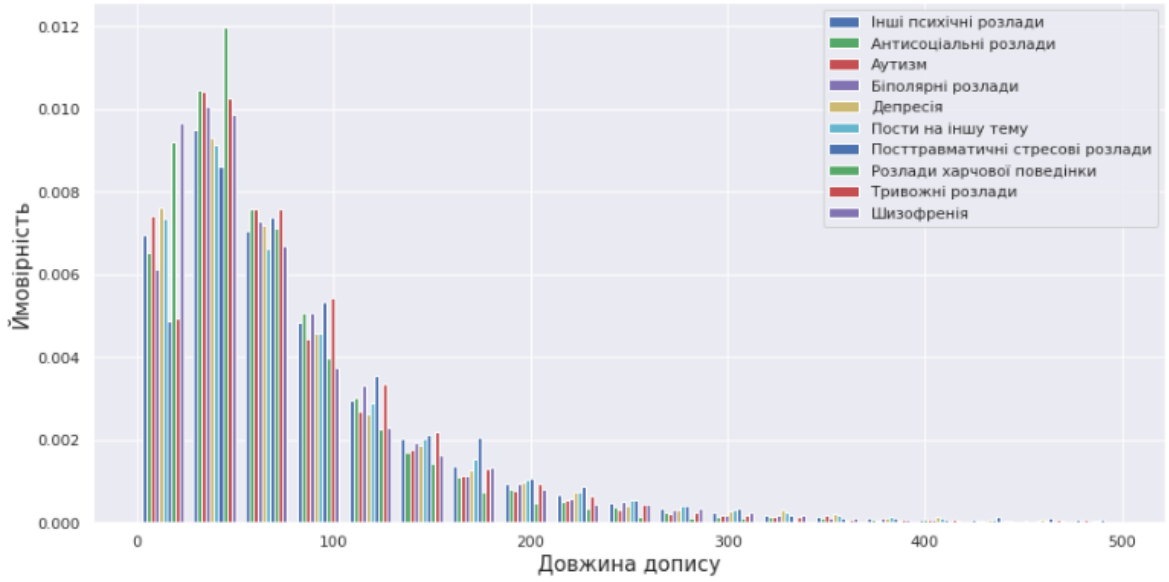

Рисунок 3.5 - Розподіл ймовірностей довжини допису для кожного з класів

На основі наведеного аналізу видно, що досить велика кількість дописів має 50 та менше слів. Такі дописи можуть бути недостатньо інформативними для моделі машинного навчання, тому було вирішено видалити всі приклади,

які містять 50 та менше слів. Крім того, варто зазначити що для більш довгих дописів з 50+ словами спостерігається дуже схожий розподіл довжини допису для різних класів, що вказує на збалансованість вибірки, а це також є важливим під час вирішення задачі класифікації. Також, з метою отримати більш збалансовану вибірку та полегшити обчислювальну складність алгоритму, було видалено дуже довгі дописи з 500 та більше слів, а дописи що залишились було перемішано таким чином, щоб дуже довгі дописи не стояли поруч. Розподіл ймовірностей довжини допису після згаданих перетворень наведено на рисунках 3.6 і 3.7 відповідно. Середня довжина допису склала 121.5 слів.

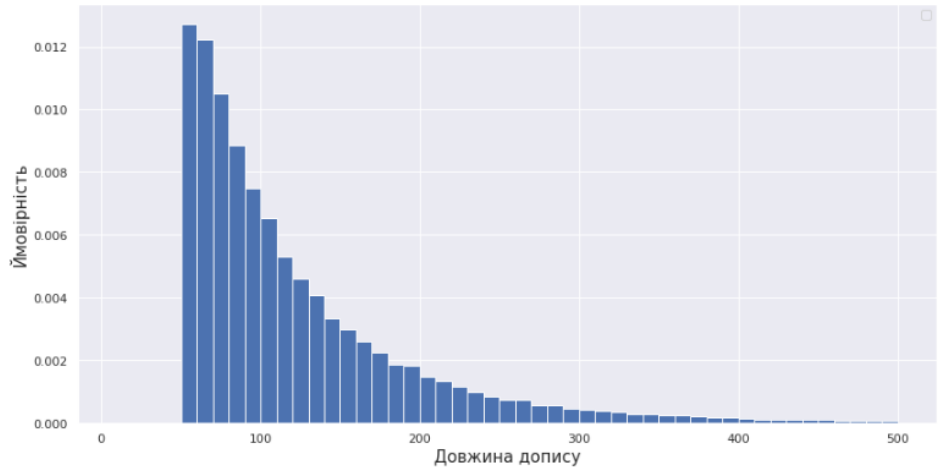

Рисунок 3.6 - Розподіл ймовірностей довжини допису після додаткової обробки на повному наборі даних

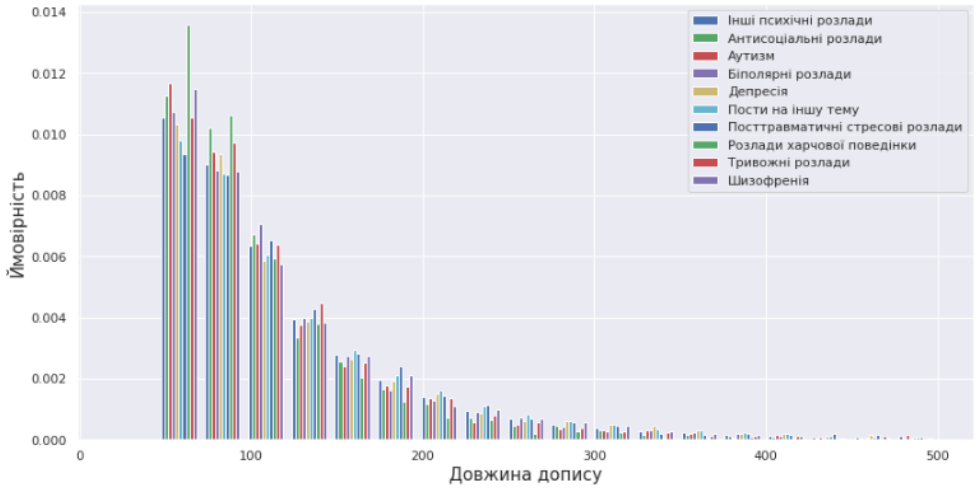

Рисунок 3.7 - Розподіл ймовірностей довжини допису після додаткової обробки даних по класам

Іншою важливою характеристикою текстових даних є частота вживання різних слів. З метою розрізнити словник, який притаманний конкретному психічному розладу, для кожного з класів була побудована хмара слів, яка демонструє 100 найбільш часто вживаних слів. Хмари слів для кожного з класів продемонстровано на рисунку 3.8.

Рисунок 3.8 - Хмари найбільш часто вживаних слів для кожного з класів

Як можна помітити, найбільш часто вживані слова для всіх класів дуже схожі. Це пов’язано з особливостями мовлення і такі слова також можна вважати схожими на стоп-слова, але з однією відмінністю. При використанні контекстних методів векторизації тексту, вивчається контекст у якому вживається те чи інше слово і такі слова є дуже важливими з точки зору розуміння інших слів у контексті. Проте, при використанні частотних методів векторизації, такі слова можна вважати зайвим шумом. Саме тому, з метою винайти слова притаманні кожному з класів для векторизації тексту методами на основі частоти вживання слів, було проведено ітеративне видалення тих слів, які однаково часто вживаються у текстах усіх класів. Крім того, під час процесу ітеративного видалення було виявлено, що серед часто вживаних слів з’явилися, власне, назви психічних захворювань. Враховуючи, що метою роботи є ідентифікувати психічні захворювання на ранніх стадіях, назви психічних захворювань також були видалені з усіх текстів. Хмари часто вживаних слів, які притаманні кожному з класів після додаткової обробки продемонстровано на рисунку 3.9.

Рисунок 3.9 - Хмари часто вживаних слів, які притаманні кожному з класів після додаткової обробки

Після проведених операцій загальна вибірка скоротилась до 551,1 тисяч дописів, а середня довжина допису склала 103 слова. Розподіл кількості текстів за класами після аналітичної обробки наведено в таблиці 3.3.

Таблиця 3.3 - Розподіл кількості текстів за класами після аналітичної обробки даних.

Назва класу	**Кількість дописів**
Пости на іншу тему	270677
Інші психічні розлади	123207
Тривожні розлади	40455
Посттравматичні стресові розлади	5704
Біполярні розлади	3399
Антисоціальні розлади	26113
Депресія	65705
Аутизм	4789
Шизофренія	4182
Розлади харчової поведінки	6909

3.2 Розробка моделі машинного навчання для ідентифікації психічних захворювань на основі публікацій у соцмережах

Для вирішення поставленої задачі було розроблено вісім моделей бінарної класифікації, кожна з яких відносить конкретний допис користувача до одного з наступних класів:

- тривожні розлади;
- депресія;
- біполярні розлади;
- посттравматичні стресові розлади;
- шизофренія;
- розлади харчової поведінки;
- асоціальна поведінка та дисоційовані розлади;
- порушення розвитку центральної нервової системи (аутизм).

Такий підхід було обрано через те, що деякі користувачі соцмереж, публікації яких були використані для навчання класифікатора, можуть мати відразу кілька симптомів. Через це модель може постраждати від зашумлених даних. Саме тому було розроблено вісім незалежних моделей бінарної класифікації для кожного розладу, щоб покращити ефективність. Виконавши таке розбиття була отримана можливість більш точно визначити потенційний психічний стан користувача.

Великою проблемою для вирішення задачі класифікації є сильний дисбаланс класів. Для вирішення цієї проблеми було використано метод збільшення кількості прикладів міноритарного класу за допомогою алгоритму SMOTE 19. Так, простим вирішенням даної проблеми є дублювання прикладів міноритарного класу або видалення прикладів мажоритарного класу, але в такому випадку або втрачається велика кількість даних, або існує велика

ймовірність перенавчання моделі. Алгоритм SMOTE 19 частково вирішує дану проблему та синтетично генерує приклади схожі на приклади міноритарного класу та дещо модифікує їх, зберігаючи розподіл.

Набір даних було розбито на навчальну та тренувальну вибірку у відношенні 80% та 20% відповідно. Для вирішення задачі класифікації було порівняно алгоритм класифікації XGBoost та згорткову нейронну мережу (CNN).

Для отримання математичного представлення кожного допису слова з навчальної вибірки було переведено в числові вектори.

3.2.1 Вилучення із тексту корисних ознак для моделі машинного навчання. Переведення словника слів у матрицю числових векторів.

Як було зазначено у розділі 1, завдяки великій кількості експериментів в різних задачах та на різних наборах даних, які проводила компанія Google, була знайдена залежність між ефективністю деяких методів машинного навчання та відношенням розміру вибірки до середньої кількості слів у текстах. Так, рекомендується використовувати частотні методи векторизації та неглибокі моделі класифікації, якщо зазначене відношення менше ніж 1500. В іншому випадку рекомендується використовувати методи контекстної векторизації та глибокі нейронні мережі для класифікації.

Враховуючи дуже різні розміри класів для різних психічних захворювань, відношення розміру вибірки до кількості слів у вибірки для кожної моделі бінарної класифікації варіювалося від 900 до 3500. Саме тому було вирішено спробувати вирішення задачі двома підходами та порівняти їх ефективність.

В якості неглибокого класифікатора було обрано алгоритм XGBoost, який являє собою ансамбль слабких моделей та вважається одним з найбільш ефективних методів класифікації. Тут, для представлення даних у числовому вигляді було використано векторизатор на основі метрики TF-IDF для перетворення слів у n-вимірні вектори.

Для класифікації за допомогою глибоких нейронних мереж були обрані згорткові нейронні мережі, які вміють дуже гарно витягувати інформативні ознаки та є дуже універсальними. Вектори слів в даному випадку були отримані з використання підходу word2vec. Було проведено два досліди:

1) навчання представлень слів на навчальному наборі даних, зібраному для поточного дослідження, з використанням моделей неперервного представлення мішків слів (CBOW);

2) навчання представлень слів з використанням передавального навчання, використовуючи модель word2vec від Google. Було проведено додаткове навчання даної попередньо навченої моделі на зібраному наборі даних.

Кількість ознак для кожного вектора представлення дорівнювала 300. Максимальну кількість слів було встановлено на рівні 150 слів. Отже, для кожного тексту було отримано матрицю розмірністю (150, 300). Реалізацію було написано за допомогою пакетів TF-IDF scikit-learn та gensim в мові програмування python. Також варто зауважити, що у випадку частотної векторизації методом TF-IDF були використана більш очищена від часто вживаних слів вибірка, ніж під час векторизацію методом word2vec.

3.2.2 Побудова нейронної мережі для класифікації текстів. Архітектура мережі та параметри навчання.

Архітектура моделі була організована послідовністю шарів, що включає шар який генерує векторне представлення слова (Embedding), декілька шарів згортки (Convolutional), шар максимального об'єднання (Map pooling), декілька щільних шарів (Dense), що представлять повнозв’язну нейронну мережду та вихідний шар з функцією активації сигмоїд для бінарної класифікації. Першим шаром моделі є шар представлення, який генерує векторні представлення слів за допомогою метода word2vec. Кожен допис перетворюється на матрицю з 150 векторів на 300 ознак. Ваги даного шару ініціалізуються випадковим чином або за допомогою попередньо навченого векторизатору в залежності від експерименту. Другий, третій та четвертий шари - шари згортки, на вхід яких подаються вектори слів. Розмір кожного фільтра згортки дорівнює три, було використано 256, 128 та 64 карти ознак відповідно на кожному з рівнів. Наступний шар - шар максимального об'єднання (Max Pooling), який зменшує дискретність карт ознак після проходження згорткової частини мережі. Вихід шару максимального об’єднання пропускається через три повнозв’язних шари (Dense), а останній шар представлено одним нейроном з функцією активації сигмоід, який повертає ймовірність належання до класу в межах від 0 до 1. Між шарами з активацією було додано шари пакетної нормалізації (Batch Normalization), які нормалізують дані перед потраплянням у наступний шар. Крім того, для запобігання перенавчання в мережу було додано шар з коефіцієнтом відсіву (Dropout). Цей шар з деякою ймовірністю робить вхід в наступний шар нульовим. Коефіцієнт відсіву було встановлено на рівні 25%. Рисунок 3.10 та 3.11 демонструють архітектуру нейронної мережі, що була розроблена під час дослідження.

Для навчання нейронної мережі було використано бінарну функцію втрат - перехресну кросенторпію та оптимізатор Адам. Було проведено тренування гіпермараметрів мережі за допомогою гратчастого пошуку. Оптимальний параметр швидкості дорівнював 0,001. Модель навчалася протягом 50 епох, а розмір батчу був встановлений на рівні 64 екземпляри.

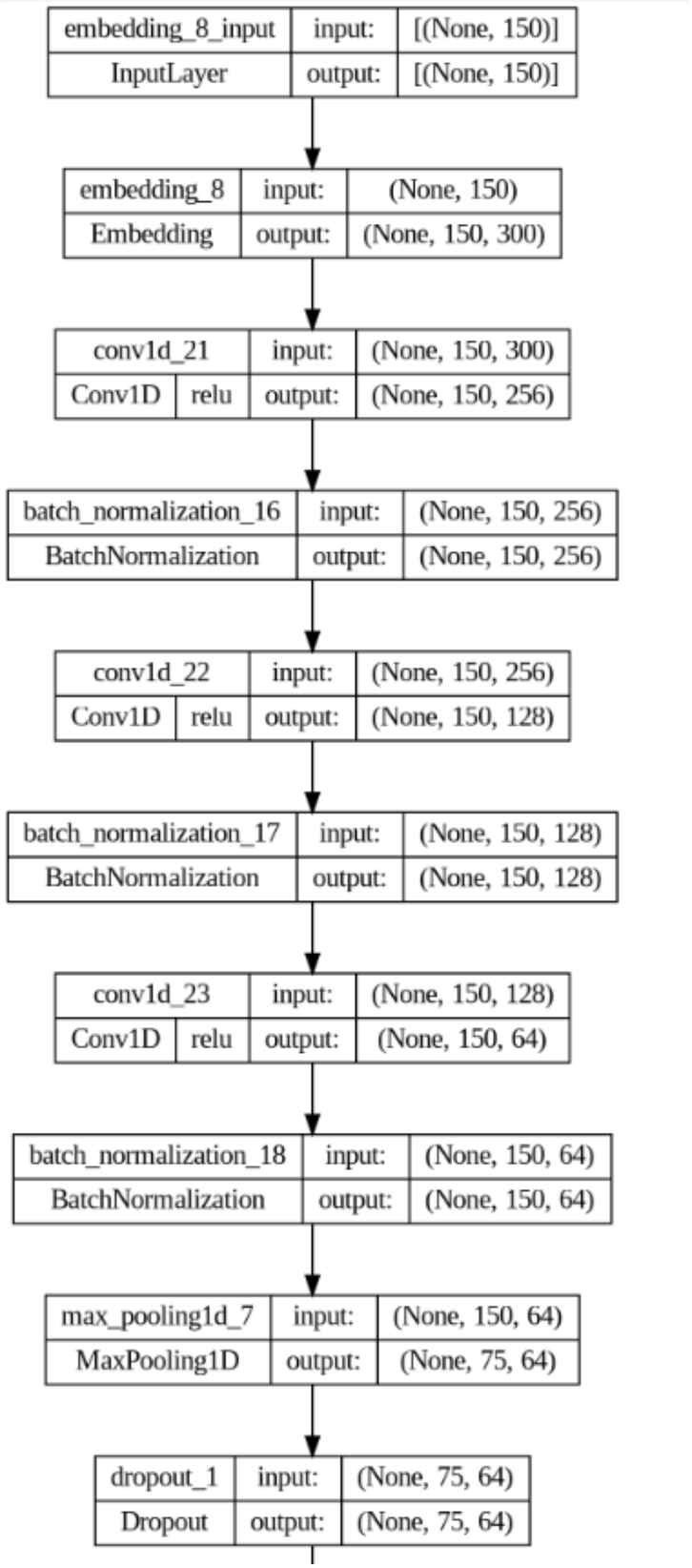

Рисунок 3.10 - Архітектура розробленої нейронної мережі (векторизатор та згорткова частина)

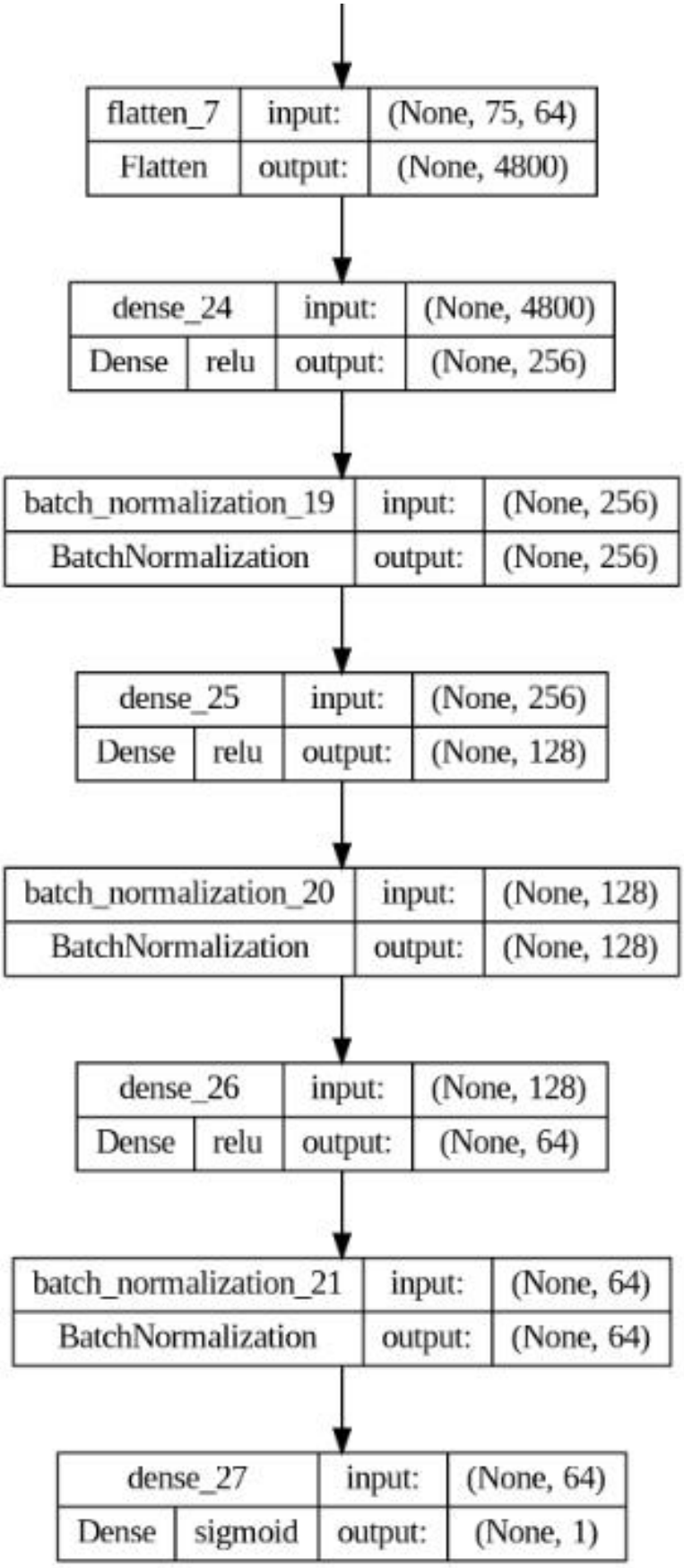

Рисунок 3.11 - Архітектура розробленої нейронної мережі (щільні шари)

3.2.3 Оцінка результатів

Для перевірки ефективності моделей було використано чотири метрики оцінки: точність (accuracy), влучність (precision), повнота (recall) та міра F1 (F1-Score). TP, FN, TN та FP означають відповідно істинно-позитивний, хибно-негативний, істинно-негативний та хибно-позитивний результати. Перевагою міри F1 є те, що вона балансує між точністю та влучністю на позитивному класі, в той час як точність враховує тільки правильно класифіковані спостереження, як позитивні, так і негативні. В умовах не найкращим чином збалансованої вибірки вона допомагає більш якісно оцінити результат бінарної класифікації.

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP}, \tag{23}$$

$$precision = \frac{TP}{TP+FP}, \tag{24}$$

$$recall = \frac{TP}{TP + FN}, \tag{25}$$

$$F1score = \frac{2 \times precision \times recall}{precision + recall}. \tag{26}$$

Загалом було проведено три досліди, в кожному було побудовано 8 моделей бінарної класифікації для кожного психічного захворювання:

1) класифікація методом XGBoost, коли тексти було векторизовано частотним методом з використанням метрики TF-IDF;

2) класифікація згортковою нейронною мережею, коли векторні представлення слів навчались за допомогою метода word2vec з нуля;

3) класифікація згортковою нейронною мережею, коли векторні представлення слів донавчались на попередньо навченій мережі word2vec від Google.

Результати кожного з дослідів продемонстровано в таблицях 3.4, 3.5 та 3.6 відповідно.

Таблиця 3.4 - Результат класифікації методом XGBoost.

Психічний розлад	Клас	XGBoost класифікатор	
		F1-Score	Точність (%)
Депресія	1	60.48	72.34
	0	80.54	
Тривожні розлади	1	70.73	75.31
	0	84.39	
Біполярні розлади	1	53.59	71.53
	0	88.06	
Посттравматичні стресові розлади	1	55.2	62.19
	0	91.68	
Шизофренія	1	43.74	68.25
	0	89.44	
Аутизм	1	36.23	69.93
	0	98.12	
Антисоціальні розлади	1	52.74	74.13
	0	90.34	
Розлади харчової поведінки	1	34.25	69.3
	0	94.43	

Таблиця 3.5 - Результат класифікації згортковими нейронними мережами при навчанні представлень слів з “нуля”.

Згорткова нейронна мережа CNN (навчання представлень слів методом word2vec з нуля)			
Психічний розлад	Клас	F1-Score	Точність (%)
Депресія	1	72.5	85,08
	0	96.64	
Тривожні розлади	1	79.04	87.36
	0	97.89	
Біполярні розлади	1	69.94	81.71
	0	96.45	
Посттравматичні стресові розлади	1	61.82	69.65
	0	86.05	
Шизофренія	1	66.73	79.12
	0	97.89	
Аутизм	1	69.33	77.08
	0	93.67	
Антисоціальні розлади	1	68.01	81.5
	0	91.52	
Розлади харчової поведінки	1	60.83	79.77
	0	95.92	

Таблиця 3.6 - Результат класифікації згортковими нейронними мережами при донавчанні представлень слів попередньо навченою мережею word2vec від Google.

Згорткова нейронна мережа CNN (донавчання представлень слів Google word2vec)			
Психічний розлад	Клас	F1-Score	Точність (%)
Депресія	1	81.55	89,48
	0	98.69	
Тривожні розлади	1	85.74	92.06
	0	99.11	
Біполярні розлади	1	62.94	78.71
	0	94.8	
Посттравматичні стресові розлади	1	71.38	77.82
	0	96.31	
Шизофренія	1	59.89	74.73
	0	98.26	
Аутизм	1	50.28	73.42
	0	96.51	
Антисоціальні розлади	1	74.45	84.17
	0	97.76	
Розлади харчової поведінки	1	77.38	86.93
	0	98.24	

Аналіз результатів потрібно почати з особливості, яка поєднала практично всі моделі, а саме незбалансованість роботи мережі на різних класах. Це пов’язано тим, що для балансування вибірки було використано синтетичне генерування прикладів, і хоча вибірка була номінально

збалансована, синтетично згенеровані приклади створювали додатковий шум. Загалом, серед восьми різних підкласів найвищу точність було отримано при ідентифікації тривожних розладів за допомогою згорткової нейронної мережі та попередньо навчених представлень слів. Точність класифікації дорівнювала 92%, а міра F1, яка описує здатність правильно ідентифікувати саме клас захворювання дорівнює 85.7%. Схожі результати були досягнуті для ідентифікації депресивних розладів, дещо гірші для ідентифікації розладів харчової поведінки. Для всіх інших захворювань у найкращих випадках вдалося досягти точності ідентифікації на рівні 66-80%.

Загалом, згорткові нейронні мережі показали вищу точність, ніж моделі XGBoost, за всіма показниками та на всіх класах. Куди більш цікава ситуація виникла під час порівняння результатів згорткових моделей з використання попередньо навчених представлень слів та представлень навчених з “нуля”. Кращі результати для ідентифікації біполярних розладів, аутизму та шизофренії були досягнуті при навчанні представлень слів з “нуля”, тоді як всі інші моделі краще проявили себе при навчанні з використанням попередньо навчених представлень слів. Це свідчить про те, що у людей з даними психічними розладами є особливості мовлення, які відрізняються від звичайних, засвоєних на дуже великих наборах даних особливостей. При навчанні представлень слів у контексті з нуля, такі особливості були засвоєні набагато краще, що покращило результат відповідних моделей.

Таким чином, запропоновані моделі можуть з досить високою точність ідентифікувати психічні захворювання на основі текстів у соцмережах. Варто зазначити, що дане дослідження було проведено на даних зібраних з однієї соцмережі Reddit, а поведінка моделей на даних, які можна зустріти в інших соцмережах додатково не вивчалася. Для подальшого покращення результатів необхідно провести додатковий збір даних з більшої кількості джерел, покращити збалансованість даних, врахувати та додати експертні оцінки психологів під час наповнення словника слів для кожного психічного розладу.

3.3 Висновки

Для вирішення задачі ідентифікації психічних захворювань було проведено розробку бінарних моделей класифікації текстів для кожного психічного захворювання. Для цього було проведено збір даних у соцмережі Reddit, виконано їх попередню обробку даних, досліджено специфіку даних, проведено додаткову аналітичну обробку текстових даних, після чого, з метою знайти найефективніший метод векторизації текстових даних та модель для вирішення задачі класифікації - було проведено декілька експериментів. В результаті було отримано вісім найкращих моделей бінарної класифікації текстів для кожного з психічних розладів. Найкращого результату вдалося досягти для ідентифікації тривожних розладів та депресії, точність за мірою F1 склада 85% та 81% відповідно. Всі інші моделі показали ефективність на рівні 70-80%. Найгірше вдалося вирішити задачу ідентифікації шизофренії - точність ідентифікації лише 66.7%.

Найкращі результати для кожного з психічних розладів було обгорнуто програмним інтерфейсом з метою надати користувачу можливість використовувати модель на практиці.

4 РОЗРОБКА СТАРТАП-ПРОЕКТУ

Останні роки довели, що стартапи сильніші, ніж більшість думала. Гнучкий та інноваційний підхід були ключовими, коли світ зіштовхнувся з такими проблемами, як віддалена робота, масштабні зміни в галузі та ринку, а також абсолютно нова реальність.

Здатність стартапів швидко змінюватися та адаптуватися і є ключовою властивістю даного виду бізнесу, не кажучи вже про те, що багато стартапів продовжували рости та розширювати свої команди.

Стартап — це бізнес-структура, що розвивається та працює на основі інновацій, створена для вирішення проблеми шляхом надання нової пропозиції в умовах надзвичайної невизначеності.

Власне, стартап – це бізнес, який:

– швидко росте;

– порушує ринок або галузь (щось нове, що змушує конкурентів удосконалюватись);

– вирішує проблему;

– працює в умовах надзвичайної невизначеності.

Багато підприємців і відомих бізнес-магнатів визначають стартап як культуру та менталітет побудови бізнесу на основі інноваційної ідеї для вирішення критичних проблем.

Одне, що відрізняє стартапи від інших компаній — це зв’язок між їхнім продуктом та його попитом. Стартапи мають продукти, орієнтовані на невикористаний ринок. Підприємці-початківці знають ідеальну стратегію, щоб створити продукт, який хоче ринок, а також охопити й обслуговувати їх усіх. Це викликає швидке зростання.

4.1 Опис ідеї проекту

В межах підпункту було проаналізовано і подано у вигляді таблиць:

- зміст ідеї;
- можливі напрямки застосування;
- можливі напрямки застосування;
- основні вигоди, що може отримати користувач товару;
- чим відрізняється від існуючих аналогів та замінників.

Перші три пункти подані у вигляді таблиці (таблиця 4.1) і дають цілісне уявлення про зміст ідеї та можливі базові потенційні ринки, в межах яких потрібно шукати групи потенційних клієнтів.

Таблиця 4.1 — Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Дана комплексна система дозволяє розвязати проблему ідентифікації психічних захворювань.	Ідентифікація психічних захворювань на ранніх стадіях	Не доспутити розвитку захворювання
	Інформування користувача та надання рекомендацій	Покращення самопочуття

Аналіз потенційних техніко-економічних переваг ідеї (чим відрізняється від існуючих аналогів та замінників) порівняно із пропозиціями конкурентів передбачає:

- визначення переліку техніко-економічних властивостей та характеристик ідеї;

– визначення попереднього кола конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та проводиться збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку;

– проводиться порівняльний аналіз показників: для власної ідеї визначаються показники, що мають:

– гірші значення (W, слабкі);

– аналогічні (N, нейтральні) значення;

– кращі значення (S, сильні) (табл. 4.2).

Таблиця 4.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Технікоекономічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Немає	Немає			
1.	Форма виконання	Надавання послуг	-	-			+
2.	Собівартість	Низька	-	-			+
3.	Функціонал	Широкий	-	-			+

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

4.2 Технологічний аудит ідеї проекту

В межах даного підрозділу було проведено аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару). Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (таблиця 4.3):

1. За якою технологією буде виготовлено товар згідно ідеї проекту?
2. Чи існують такі технології, чи їх потрібно розробити/доробити?
3. Чи доступні такі технології авторам проекту?

Таблиця 4.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології реалізації	Наявність технологій	Доступність технологій
1	Створення системи ідентифікації психічних захворювань	Використання мови програмування Python	Наявна	Доступна
		Використання мови програмування C#	Відсутні	Недоступні
		Використання мови C++	Відсутні	Недоступні
Обрана технологія реалізації ідеї проекту: мова програмування Python.				

За результатами аналізу таблиці зроблено висновок щодо можливості технологічної реалізації проекту. Технологічним шляхом реалізації проекту було обрано такі технології, як Python через доступність та безкоштовність.

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку було проведено аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (таблиця 4.4).

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	0
2	Загальний обсяг продаж, грн/ум.од	1000000000
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі, %	20%

За результатами аналізу таблиці 4.4 було зроблено висновок, що ринок є привабливим для входження.

Надалі були визначені потенційні групи клієнтів, їх характеристики та сформовано орієнтовний перелік вимог до товару для кожної групи (табл. 4.5).

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Ідентифікація психічних захворювань	Будь-яка людина	Велика кількість даних	Простота використання, доступність
Надання рекомендацій та допомоги	Психічно хворі люди	Цікавить простота у використанні, доступність підтримки системи	Швидкість створення, доступність

Після визначення потенційних груп клієнтів було проведено аналіз ринкового середовища: складено таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (табл. 4.6, 4.7).

Таблиця 4.6 – Фактори загроз

№ п/ п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Вихід на ринок продуктів з кращими характеристиками	Передбачити додаткові переваги власного програмного продукту (ПП) для того, щоб повідомити про них саме після виходу на ринок конкурентів. Вдосконалення технічних моментів власного продукту. Обрати нову цільову аудиторію і зосередитися на ній: зниження цін.
2	Зміна потреб користувачів	Користувачам необхідний сервіс з більшим/новим функціоналом.	Розроблення гнучкої архітектури програмного забезпечення для легшого впровадження нового функціоналу

Таблиця 4.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Гнучкі ціни	Зменшення ціни товару задля збільшення попиту	Введення власних гнучких цін
2	Поява нових методів квантування зображення	З’являться нові методи, що будуть швидше, та більш точно квантувати зображення	Покращити ПП додаванням нового функціоналу, розширення можливостей

Надалі було проведено аналіз пропозиції: визначили загальні риси конкуренції на ринку (таблиця 4.8).

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
1. Тип конкуренції- чиста	Існує величезна кількість конкурентів на ринку.	Якісно провести рекламу.
2. За рівнем конкурентної боротьби - міжнародний	Компанії-конкуренти з інших країн	Створити основу ПП таким чином, щоб можна було легко переробити даний ПП для використання у галузях інших країн.
3. За галузевою ознакою - міжгалузева	Продукт може використовуватись для різних галузей	Постійне вдосконалення продукту, що не має прив’язки до сфери
4. Конкуренція за видами товарів: - товарно-видова	Конкуренція між видами ПП, їх особливостями.	Створити ПП, враховуючи недозображення конкурентів
5. За характером конкурентних переваг - нецінова	Вдосконалення технології створення ПП, щоб собівартість була нижчою	Удосконалення моделі. Використання більш дешевих технологій для розробки, ніж використовують конкуренти, але тільки якщо ці технології відповідають необхідним вимогам якості.
6. За інтенсивністю - не марочна	Бренд присутній, але його роль незначна	Реклама, участь у конференціях, семінарах.

Було проведено аналіз конкуренції у галузі за моделлю М. Портера (табл. 4.9).

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товаризамінники
	Навести перелік прямих конкурентів	Визначити бар'єри входження в ринок	Визначити фактори сили постахальників	Визначити фактори сили споживачів	Фактори загроз з боку замінників
	Amazon Personalize	Наявність вже існуючих рішень	-	Контроль якості продукту	Наявність більш широкого функціоналу, зручнішого інтерфейсу та авторитет
Висновки:	Досить інтенсивна конкурентна боротьба з іншими гравцями	Є можливості і виходу на ринок, але є і конкуренти.	-	Клієнти диктують умови роботи на ринку: зручний інтерфейс	Необхідно випускати ПЗ не гірше, ніж у конкурентів та розширяти функціонал.

За результатами аналізу було зроблено висновок про можливість роботи на ринку з огляду на конкурентну ситуацію.

Цей висновок був врахований при формулюванні переліку факторів конкурентоспроможності у наступному пункті. На основі аналізу конкуренції, проведеного в таблиці, а також із урахуванням характеристик ідеї проекту (табл. 4.2), вимог споживачів до товару (табл. 4.5) та факторів маркетингового середовища (табл. 4.6, 4.7) визначається та обґрунтовується перелік факторів конкурентоспроможності.

Аналіз оформлено у (табл. 4.10).

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування
1	Ціна	Один із факторів для вибору продукту клієнтом.
2	Якість	Один із факторів для вибору продукту клієнтом.
3	Зручність роботи з програмою	Дозволяє користувачу легко працювати з програмою

За визначеними факторами конкурентоспроможності (табл. 4.10) проведено аналіз сильних та слабких сторін стартап-проекту (табл. 4.11).

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні						
			–3	–2	–1	0	+1	+2	+3
1	Ціна	15					*		
2	Якість	10			*				
3	Зручність роботи з програмою	15					*		

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (табл. 4.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (таблиця 4.11).

Перелік ринкових загроз та ринкових можливостей було складено на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення.

Таблиця 4.12 – SWOT-аналіз стартап-проекту

Сильні сторони: Якість Простота використання Висока швидкодія	Слабкі сторони: Дуже насичений ринок, мала кількість функціоналу, відсутня кросплатформеність.
Можливості: насичення ринку новим підходом до прогнозування; різноманітна клієнтура, вдосконалення системи	Загрози: Конкуренція

На основі SWOT-аналізу було розроблено альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок.

Визначені альтернативи були проаналізовані з точки зору строків та ймовірності отримання ресурсів (таблиця 4.13).

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	PR, просування бренду	50%	6 місяців
2	Перехід на безкоштовне розповсюдження	75%	3 місяців
3	Партнерство для об’єднання продукції	65%	2 місяці

Після аналізу було обрано альтернативу №2.

4.3 Аналіз ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: було проведено опис цільових груп потенційних споживачів (таблиця 4.14).

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Психічно хворі люди	Висока	Високий	Сильна	Просто
2	Розробники застосунків	Середня: велика конкуренція і можливість власних веб-відділів	Високий	Сильна	Складно
3	Розробники операційних систем	Низька	Низький	Слабка	Середня
Які цільові групи обрано:1,2,3					

За результатами аналізу потенційних груп споживачів було обрано цільові групу, для яких буде запропоновано даний товар, та визначено стратегію охоплення ринку - стратегію диференційованого маркетингу (компанія працює з декількома сегментами).

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку (таблиця 4.15).

Таблиця 4.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Постійне оновлення і покращення продукту	Ринкове позиціонування на індивіуальних користувачів	Швидкодія, якість продукту	Концентрований маркетинг

Наступним кроком обрано стратегію конкурентної поведінки (таблиця 4.16).

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні.	Компанія буде шукати нових споживачів та забирати існуючих у конкурентів	Буде копіювати, удосконалювати та створювати свої унікальні пропозиції	Зайняття конкурентної ніші

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (табл. 4.5), а також в залежності від обраної

базової стратегії розвитку (табл. 4.15) та стратегії конкурентної поведінки (таблиця 4.16) розроблено стратегію позиціонування (таблиця 4.17), що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 4.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартаппроекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
1	Легкість розуміння, зручний інтерфейс, надійний, швидкий, точний та достовірни ПП для генерації рекомендацій.	Стратегія диферен нціації	Позиція на основі порівняння фірми з товарами конкурентів; Відмінні особливості споживача	Економія часу; Зручність застосування; Практичність та точність результату

Результатом виконання підрозділу стала узгоджена система рішень щодо ринкової поведінки стартап-компанії, яка визначає напрями роботи стартап- компанії на ринку.

4.4 Розроблення маркетингової програми стартап-проекту

Сформовано маркетингову концепцію товару, який отримає споживач. Для цього підсумовано результати попереднього аналізу конкурентоспроможності товару (таблиця 4.18). Концепція товару – письмовий опис фізичних та інших характеристик товару, які сприймаються споживачем, і набору вигод, які він обіцяє певній групі споживачів.

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Швидкість отримання результату	Швидка видача рекомендацій наступної пропозиці	Необхідно покращити швидкість навчання моделі для видачі рекомендацій наступної пропозиці
2	Зручність застосування	Нативна підтримка мови програмування Python	Розробка зручного прикладного програмного інтерфейсу
3	Практичність та точність результату	Користувач отримує точні (з малою похибкою розбіжності) результати рекомендацій.	Користувач на виході роботи ПП отримує модель та прогноз, котрі відповідають необхідним показникам варіативності та точності.

Розроблено трирівневу маркетингову модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (таблиця 4.19).

1-й рівень. При формуванні задуму товару вирішується питання щодо того, засобом вирішення якої потреби і / або проблеми буде даний товар, яка його основна вигода. Дане питання безпосередньо пов'язаний з формуванням технічного завдання в процесі розробки конструкторської документації на виріб.

2-й рівень. Цей рівень являє рішення того, як буде реалізований товар в реальному/ включає в себе якість, властивості, дизайн, упаковку, ціну.

3-й рівень Товар з підкріпленням (супроводом) – додаткові послуги та переваги для споживача, що створюються на основі товару за задумом і товару в реальному виконанні (гарантії якості , доставка, умови оплати та ін).

Таблиця 4.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Генерація палітри кольорів зображення за допомогою інтелектуальних систем		
II. Товар реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх/Тл/Е/Ор
	1. Індивідуальний підхід. 2. Низька ціна. 3. Простота у використанні.	1.Нм 2.Нм 3.Нм	1.Технологічна 2.Економічна 3.Технологічна
	Якість: тестування фірмами аудиторами		
	Пакування: відсутнє		
	Марка: ROSO		

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари субститути,

а також аналіз рівня доходів цільової групи споживачів (таблиця 4.20). Аналіз проведено експертним методом.

Таблиця 4.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	2000$	3700$	У всіх трьох груп високий рівень доходів	900$--

Наступним кроком є визначення оптимальної системи збуту, в межах якого було прийняте рішення (таблиця 4.21).

Таблиця 4.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Канал нульового рівня	Продаж	0(напряму)	Власна

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (таблиця 4.22).

Таблиця 4.22 – Концепція маркетингових комунікацій

№ п/ п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
	Інтеграція API у клієнтській системі	Інтернет	Низька ціна, простота використання, універсальність	Показати переваги рішення над конкурентами, виділити ключові особливості	Створення сайту продукту, розповсюдження інформації про продукт на спеціалізованих ресурсах.

Було визначено, що придбання продукту буде проводитись через мережу Інтернет або при безпосередньому спілкуванні із представниками компанії. Розповсюдження інформації про продукт буде проводитись виключно через Інтернет, адже аудиторія даного продукту активно користується всесвітньою мережею.

Результатом підрозділу стала ринкова (маркетингова) програма, що включає в себе концепції товару, збуту, просування та попередній аналіз можливостей ціноутворення, спирається на цінності та потреби потенційних клієнтів, конкурентні переваги ідеї, стан та динаміку ринкового середовища, в межах якого впроваджено проект, та відповідну обрану альтернативу ринкової поведінки.

4.5 Висновки

1. В даному розділі проведено підготовчий аналіз для впровадження розробленої системи в якості стартап проекту. Досліджено аналогічні конкурентні системи, встановлено сильні та слабкі сторони системи в порівнянні з ними. Також було досліджено можливі шляхи розповсюдження продукту та його ймовірну аудиторію, рівень доходів та ймовірну ціну продукту, що розробляється.

2. Було проведено аналіз потенційних ризиків і можливостей, а також розраховані основні фінансово-економічні показники проекту. Отримані результати кажуть про те, що реалізація проекту є доцільною. Було визначено сильні сторони проекту: зручність у використанні, ціна, якість та широкий функціонал. Серед слабких варто виділити повільне навчання. Варто відмітити можливість реклами продукту на спеціалізованих ресурсах із зазначенням сильних сторін проекту.

ВИСНОВКИ ПО РОБОТІ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

У магістерській дисертації піднята проблема необхідності створення системи для ідентифікації психічних захворювань з використанням «штучного інтелекту» та запропоновано вирішення проблеми, а саме розробка системи ідентифікації психічних захворювань на основі текстів користувачів у соцмережах.

В даній роботі досліджувалися різні методи машинного навчання та обробки природної мови з метою вирішити задачу класифікації текстів та створити модель машинного навчання, яка стане основою системи для ідентифікації психічних захворювань на основі публікацій користувачій соціальної мережі Reddit. Дані для проведення дослідження та побудови моделі машинного навчання були зібрані самостійно за допомогою програмного інтерфейсу соцмережі. Проведено порівняння різних підходів та їх ефективність для вирішення задачі векторизації текстових даних та класифікації тексту.

В результаті побудовано декілька моделей бінарної класифікації для кожного психічного розладу, а саме модель класифікації методом XGBoost з використанням частотних методів векторизації тексту на основі метрикиTF-IDF та згорткову нейронну мережу з використання методу word2vec для контекстної векторизації текстових даних. Виконано їх навчання з використанням підходів передавального навчання (Transfer Learning), після чого проведено порівняння результатів з метою обрати найкращу модель класифікації для кожного з психічних розладів. Для оцінки моделей використовувалась метрика F1-score та було досягнуто точності ідентифікації на рівні 85% при ідентифікації депресії та 82% при ідентифікації тривожних розладів. Точність ідентифікації інших психічних розладів, розглянутих у

даній роботі, була на рівні 65-80%. Найкращий результат продемонструвала модель класифікації з використанням згорткових нейронних мереж та контекстної векторизації текстів методом word2vec.

Розвиток даного проекту та його реалізацію потрібно продовжувати у двох основних напрямках:

1. Збір додаткової кількості даних з різних соціальних мереж та їх розмітка з метою покращити якість роботи моделей класифікації, вирішити проблему дисбалансу класів, навчити модель однаково гарно ідентифікувати психічні захворювання на даних з самих різних джерел.

2. Створити програмний продукт, який буде представляти собою сервіс, в основі якого буде покладено модель для ідентифікації психічних захворювань. Необхідно забезпечити відкристість даного сервісу для всіх бажаючих його покращити, забезпечити можливість використання сервісу сторонніми застосунками та можливість легкої інтеграції у мобільні додатки. Суть сервіса повинна полягати у інформуванні та допомозі людям після ідентифікації того чи іншого психічного захворювання: заспокоєння, ненав’язливе інформування з метою не погіршити психічний стан користувача, надання порад та рекомендацій для покращення психічного стану користувача, плавно підготувати людину до освідомлення необхідності лікування та підвищити її бажання звернутись до професіоналів лікарів, висвітливши «світлу» сторону роботи психоаналітиків.

ПЕРЕЛІК ПОСИЛАНЬ

1. Психические расстройства. *Всемирная организация здравоохранения*. URL: https://www.who.int/ru/news-room/fact-sheets/detail/mental-disorders (дата звернення: 09.09.2022).
2. Grandmaster K. Getting started with Text Preprocessing. *Kaggle*. URL: https://www.kaggle.com/code/sudalairajkumar/getting-started-with-text-preprocessing (date of access: 24.09.2022).
3. Menzli A. Tokenization in NLP: Types, Challenges, Examples, Tools. *Neptune Blog Logo*. URL: https://neptune.ai/blog/tokenization-in-nlp (date of access: 22.10.2022).
4. Mysiak K. NLP Part 2| Pre-Processing Text Data Using Python. *Medium*. URL: https://towardsdatascience.com/preprocessing-text-data-using-python-576206753c28 (date of access: 27.09.2022).
5. Стеммер Портера. *Википедия*. URL: https://ru.wikipedia.org/wiki/Стеммер_Портера (дата звернення: 16.11.2022).
6. What is the difference between lemmatization vs stemming? *Stackoverflow*. URL: https://stackoverflow.com/questions/1787110/what-is-the-difference-between-lemmatization-vs-stemming (date of access: 05.10.2022).
7. Плавное введение в Natural Language Processing (NLP). *Datastart*. URL: https://datastart.ru/blog/read/plavnoe-vvedenie-v-natural-language-processing-nlp (дата звернення: 14.09.2022).
8. Jeet. One Hot encoding of text data in Natural Language Processing. *Medium*. URL: https://medium.com/analytics-vidhya/one-hot-encoding-of-text-data-in-natural-language-processing-2242fefb2148 (date of access: 12.09.2022).
9. Zhou V. A Simple Explanation of the Bag-of-Words Model. *Medium*. URL: https://towardsdatascience.com/a-simple-explanation-of-the-bag-of-words-model-b88fc4f4971 (date of access: 28.10.2022).
10. GOYAL C. Part 5: Step by Step Guide to Master NLP – Word Embedding and Text Vectorization. *Analytics Vidhya*. URL: https://www.analyticsvidhya.com/blog/2021/06/part-5-step-by-step-

guide-to-master-nlp-text-vectorization-approaches/ (date of access: 07.09.2022).

11. Distributed Representations of Words and Phrases and their Compositionality / T. Mikolov et al. URL: https://arxiv.org/pdf/1310.4546.pdf (date of access: 23.11.2022).
12. Rong X. word2vec Parameter Learning Explained. URL: https://arxiv.org/pdf/1411.2738.pdf (date of access: 11.09.2022).
13. Oliinyk H. Hierarchical softmax and negative sampling: short notes worth telling. *Medium*. URL: https://towardsdatascience.com/hierarchical-softmax-and-negative-sampling-short-notes-worth-telling-2672010dbe08 (date of access: 08.11.2022).
14. Текстовая классификация. *Google Developers*. URL: https://developers.google.com/machine-learning/guides/text-classification/step-2-5 (дата звернення: 26.10.2022).
15. Masui T. All You Need to Know about Gradient Boosting Algorithm − Part 2. Classification. *Medium*. URL: https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-2-classification-d3ed8f56541e (date of access: 14.09.2022).
16. Binhuraib T. NLP with CNNs. *Medium*. URL: https://towardsdatascience.com/nlp-with-cnns-a6aa743bdc1e (date of access: 09.09.2022).
17. Цупрун І. Сервіс для розпізнавання об'єктів на основі супутникових даних та даних, отриманих з дронів: диплом бакалавра комп. наук: КПІ ім. Ігоря Сікорського. Київ, 2020, 133с. URL: https://ela.kpi.ua/handle/123456789/45474 (дата звернення: 17.11.2022).
18. Ghrib Z. Use Pre-trained Word Embedding to detect real disaster tweets. *Medium*. URL: https://towardsdatascience.com/pre-trained-word-embedding-for-text-classification-end2end-approach-5fbf5cd8aead (date of access: 15.09.2022).
19. Word2vec. *Google Code Archive*. URL: https://code.google.com/archive/p/word2vec/ (date of access: 01.11.2022).

20. Agarwal M. NLP: Stanford's GloVe for Word Embedding. *Datamahadev.com*. URL: https://datamahadev.com/nlp-stanfords-glove-for-word-embedding/ (date of access: 19.10.2022).
21. Enriching Word Vectors with Subword Information / P. Bojanowski et al. *Cornell University*. URL: https://arxiv.org/abs/1607.04606 (date of access: 22.10.2022).
22. Harrigian K. Mental Health Datasets. *GitHub*. URL: https://github.com/kharrigian/mental-health-datasets (date of access: 14.09.2022).
23. A deep learning model for detecting mental illness from user content on social media / J. Kim et al. *Scientific Reports*. URL: https://www.nature.com/articles/s41598-020-68764-y#Sec7 (date of access: 22.09.2022).

ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ

data_collection.py

```
import numpy as np
import pandas as pd

### Constants

folder_path = 'data/raw_data/'
file_prefix = '_features_tfidf_256.csv'

# the data is connected with COVID 19 periods
file_types = [
'2018', # Jan 1 to April 20, 2018
'2019', # Jan 1 to April 20, 2019
'pre', # Dec 2018 to Dec 2019
'post' # Jan 1 to April 20, 2020 called 'mid-pandemic'
]

subreddits = {
'specific_mental_health': {
'EDAnonymous': 0,
'addiction': 1,
'alcoholism': 2,
'adhd': 3,
'anxiety': 4,
'autism': 5,
'bipolarreddit': 6,
'bpd': 7,
'depression': 8,
'healthanxiety': 9,
'lonely': 10,
'ptsd': 11,
'schizophrenia': 12,
'socialanxiety': 13,
'suicidewatch': 14
},
'broad_mental_health': {
'mentalhealth': 15,
'COVID19_support': 16
},
```

```
'non_mental_health': {
'conspiracy': 17,
'divorce': 18,
'fitness': 19,
'guns': 20,
'jokes': 21,
'legaladvice': 22,
'meditation': 23,
'parenting': 24,
'personalfinance': 25,
'relationships': 26,
'teaching': 27
}
}

## Loading and preprocessing, concatenation

data = []

for subreddit_type in subreddits:
for topic, topic_code in subreddits[subreddit_type].items():
for file_type in file_types:
try:
file_path = folder_path + topic + '_' + file_type + file_prefix
file_data = pd.read_csv(file_path)
if subreddit_type == 'specific_mental_health':
file_data['is_specific_mental_health'] = 1
file_data['is_broad_mental_health'] = 0
file_data['is_not_mental_health'] = 0
elif subreddit_type == 'broad_mental_health':
file_data['is_specific_mental_health'] = 0
file_data['is_broad_mental_health'] = 1
file_data['is_not_mental_health'] = 0
else:
file_data['is_specific_mental_health'] = 0
file_data['is_broad_mental_health'] = 0
file_data['is_not_mental_health'] = 1

file_data['label'] = topic_code
file_data['label_name'] = topic
data.append(file_data)
except:
print('Failed to read:', file_path)
```

```
## Export necessary fields to .csv file.

data = pd.concat(data)

necessary_columns = [
'subreddit',
'author',
'date',
'post',
'is_specific_mental_health',
'is_broad_mental_health',
'is_not_mental_health',
'label',
'label_name']
data = data[necessary_columns]

data.to_csv('data/joined_data.csv')
```

data_preprocessing.py

```
import numpy as np
import pandas as pd

data = pd.read_csv('data/joined_data.csv')

arr = []
for label_name in np.unique(data['label_name']):
sub_df = data[data['label_name'] == label_name][:5000]
arr.append(sub_df)

data = pd.concat(arr)

import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
nltk.download('omw-1.4')
import string
```

```
import matplotlib.pyplot as plt
#import fasttext
import contractions
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords, wordnet
from nltk.stem import WordNetLemmatizer
plt.xticks(rotation=70)
pd.options.mode.chained_assignment = None
pd.set_option('display.max_colwidth', 100)
%matplotlib inline

data.drop('Unnamed: 0', axis=1, inplace=True)

for col in data.columns:
print(col, data[col].isnull().sum())

## Розширення скорочень

data['no_contract'] = data['post'].apply(lambda x: [contractions.fix(word) for word
in x.split()])
data.head()

data['post_str'] = [' '.join(map(str, l)) for l in data['no_contract']]
data.head()

## Tokenization

data['tokenized'] = data['post_str'].apply(word_tokenize)
data.head()

## Converting all characters to lowercase

data['tokenized'] = data['tokenized'].apply(lambda x: [word.lower() for word in x])
data.head()

## Removing punctiations

punc = string.punctuation
data['no_punc'] = data['tokenized'].apply(lambda x: [word for word in x if word not
in punc])
data.head()
```

```
## Removing stopwords

stop_words = set(stopwords.words('english'))
data['stopwords_removed'] = data['no_punc'].apply(lambda x: [word for word in x
if word not in stop_words])
data.head()

## Stemming

data['pos_tags'] = data['stopwords_removed'].apply(nltk.tag.pos_tag)
data.head()

def get_wordnet_pos(tag):
if tag.startswith('J'):
return wordnet.ADJ
elif tag.startswith('V'):
return wordnet.VERB
elif tag.startswith('N'):
return wordnet.NOUN
elif tag.startswith('R'):
return wordnet.ADV
else:
return wordnet.NOUN
data['wordnet_pos'] = data['pos_tags'].apply(lambda x: [(word,
get_wordnet_pos(pos_tag)) for (word, pos_tag) in x])
data.head()

wnl = WordNetLemmatizer()
data['lemmatized'] = data['wordnet_pos'].apply(lambda x: [wnl.lemmatize(word,
tag) for word, tag in x])
data.head()

# Saving preprocessed data

necessary_columns = [
'subreddit',
'author',
'date',
'is_specific_mental_health',
```

```
'is_broad_mental_health',
'is_not_mental_health',
'label',
'label_name',
'lemmatized'
]
data = data[necessary_columns]
export_df = data.rename(columns={"lemmatized": "post"})
export_df.head()

export_df.to_pickle('data/preprocessed_data.pkl')
```

exploratory_data_analysis.py

```
df = pd.read_pickle('data/preprocessed_data.pkl')
df['label_name'].value_counts()

df['post_len'] = df['post'].apply(lambda x: len(x))
df = df[df['post_len'] < 500]

df['post_len'].mean()

study_labels ={
0: (1, 'Розлади харчової поведінки'),
6: (2, 'Біполярні розлади'),
4: (3, 'Тривожні розлади'),
9: (3, 'Тривожні розлади'),
8: (4, 'Депресія'),
5: (5, 'Аутизм'),
12: (6, 'Шизофренія'),
11: (7, 'Посттравматичні стресові розлади'),
13: (8, 'Антисоціальні розлади')
}
other_mental_desorders = [1,2,3,7,10,14]
non_mental_disorders = [15,16,17,18,19,20,21,22,23,24,25,26,27]

def assign_label(old_label):
if old_label in other_mental_desorders:
return (9, 'Інші психічні розлади')
elif old_label in non_mental_disorders:
return (10, 'Пости на іншу тему')
else:
```

```
return study_labels[old_label]
df['label_name'] = df['label'].apply(lambda x: assign_label(x)[1])
df['label'] = df['label'].apply(lambda x: assign_label(x)[0])
df = df.drop(columns = ['is_specific_mental_health','is_broad_mental_health',
'is_not_mental_health'])

df['post_len'].mean()

# Post length distribution

# before preprocessing

import numpy as np
import matplotlib.pyplot as plt
values = df['post_len'].values
bins = np.linspace(10, 500, 50)

fig = plt.figure(figsize = (14,7))
plt.hist(values, bins, density=True)
plt.xlabel("Довжина допису", fontsize=15)
plt.ylabel("Ймовірність", fontsize=15)
plt.legend(loc='upper right')
plt.show()

import numpy as np
import matplotlib.pyplot as plt
plt.style.use('seaborn-deep')

df = df[df['post_len'] < 500]

values = []
label_names = []
for label_name in np.unique(df['label_name']):
values.append(df[df['label_name'] == label_name]['post_len'].values)
label_names.append(label_name)
bins = np.linspace(20, 500, 20)

fig = plt.figure(figsize = (14,7))
plt.hist(values, bins, label=label_names, density=True)
plt.xlabel("Довжина допису", fontsize=15)
plt.ylabel("Ймовірність", fontsize=15)
plt.legend(loc='upper right')
plt.show()
```

```
# after preprocessing

df = df[df['post_len'] > 50]

values = df['post_len'].values
bins = np.linspace(10, 500, 50)

fig = plt.figure(figsize = (14,7))
plt.hist(values, bins, density=True)
plt.xlabel("Довжина допису", fontsize=15)
plt.ylabel("Ймовірність", fontsize=15)
plt.legend(loc='upper right')
plt.show()

values = []
label_names = []
for label_name in np.unique(df['label_name']):
values.append(df[df['label_name'] == label_name]['post_len'].values)
label_names.append(label_name)
bins = np.linspace(50, 500, 20)

fig = plt.figure(figsize = (14,7))
plt.hist(values, bins, label=label_names, density=True)
plt.xlabel("Довжина допису, L", fontsize=15)
plt.ylabel("Ймовірність, P", fontsize=15)
plt.legend(loc='upper right')
plt.show()

#  words frequency
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from wordcloud import WordCloud
sns.set()
plt.rcParams['figure.figsize'] = [15, 20]

wordcloud = WordCloud(background_color="white", contour_width=0.1, scale=1,
contour_color="black", max_font_size=150, random_state=42,
colormap="Dark2")
```

```
for i, label_name in zip(range(10), np.unique(df['label_name'])):
df_part = df[df['label_name'] == label_name]
words = df_part['post']
allwords = []
for wordlist in words:
allwords += wordlist

mostcommon = FreqDist(allwords).most_common(100)
wordcloud.generate(text=str(mostcommon))
plt.subplot(5, 2, i+1)
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title(label_name, fontsize=15)
plt.show()

# remove frequent words

words_to_drop = ['go', 'get', 'feel', 'like', 'would', 'want', 'know', 'year', 'time', 'think',
'make',
'anxiety', 'autism', 'depression', 'ptsd', 'bipolar', 'schizophrenia', 'autistic',
'day', 'people', 'really', 'say', 'thing', 'take', 'life', 'even', 'one', 'work', 'help',
'try', 'see', 'tell', 'talk', 'friend', 'start']

def drop_words(x):
for word in x:
if word in words_to_drop:
x.remove(word)
return x
df['post'] = df['post'].apply(lambda x: drop_words(x))

# after preprocessing

sns.set()
plt.rcParams['figure.figsize'] = [15, 20]

wordcloud = WordCloud(background_color="white", contour_width=0.1, scale=1,
contour_color="black", max_font_size=150, random_state=42,
colormap="Dark2")

for i, label_name in zip(range(10), np.unique(df['label_name'])):
df_part = df[df['label_name'] == label_name]
words = df_part['post']
allwords = []
```

```
for wordlist in words:
allwords += wordlist

mostcommon = FreqDist(allwords).most_common(100)
wordcloud.generate(text=str(mostcommon))
plt.subplot(5, 2, i+1)
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title(label_name, fontsize=15)
plt.show()
```

modelling.py

```
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
from keras.layers import Embedding
from keras.layers import Flatten
from keras.layers import BatchNormalization

from keras.preprocessing.text import Tokenizer

import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import plot_confusion_matrix

from keras.constraints import maxnorm
from keras.layers import Dropout
from tensorflow.keras.utils import plot_model
import time
import warnings
warnings.filterwarnings("ignore")
path = '/content/drive/MyDrive/final_tidy_reddits_lemmetized.json' df =
pd.read_json(path, orient='table')

model = Sequential()
```

```
model.add(Embedding(150, 300, input_length=150))
model.add(Conv1D(256, 3, padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(Conv1D(128, 3, padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(Conv1D(64, 3, padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling1D())
model.add(Dropout(0.25))
model.add(Flatten())

model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(64, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(1, activation='sigmoid'))

plot_model(model, show_shapes=True, show_layer_activations=True,
to_file='model.png')
model.summary()


model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])


start = time.time() # fit the model to the training set
cnn_hist = model.fit(X_train, y_train, epochs = 5, validation_split = 0.2) #
evaluation of the model on the test set
scores = model.evaluate(X_test, y_test, verbose = 0)
print("Accuracy: %.2f%%" % (scores[1]*100))
warnings.filterwarnings("ignore")
end = time.time() total = (end-start) // 60
print('Training duration : {} minutes'.format(total))


history_dict = cnn_hist.history

# loss
loss_values = history_dict['loss']
```

```
val_loss_values = history_dict['val_loss']
epochs = range(1, len(loss_values) + 1)

# plot
plt.figure(figsize=(12, 8))
plt.plot(epochs, loss_values, 'green',label='Training loss')
plt.plot(epochs, val_loss_values, 'orange', label='Validation loss') plt.title('Training
and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

# accuracy
acc = history_dict['accuracy']
val_acc = history_dict['val_accuracy'] # range of X (no. of epochs)
epochs = range(1, len(acc) + 1)
plt.figure(figsize=(12, 8))
plt.plot(epochs, acc, 'green',label='Training accuracy') # orange is for "orange"
plt.plot(epochs, val_acc, 'orange', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# confusion matrix

preds = np.round(model.predict(X_test), 0)
cm = confusion_matrix(y_test, preds)
df_cm = pd.DataFrame(cm, index = ["anxiety", "non-anxiety"], columns =
["anxiety", "non-anxiety"])
plt.figure(figsize=(6, 4))
sb.heatmap(df_cm, annot=True, cmap="crest", fmt='d')
plt.xlabel('True Class')
plt.ylabel('Predicted Class')
```