КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра математичної інформатики

Кваліфікаційна робота
на здобуття ступеня магістра

за спеціальністю 122 Комп'ютерні науки

на тему:

ДІАГНОСТУВАННЯ АФЕКТИВНИХ РОЗЛАДІВ НА ОСНОВІ АНАЛІЗУ ТЕКСТІВ ПРИРОДНОЇ МОВИ

Виконала студентка 2-го курсу магістратури
Федоренко Галина Ігорівна _____
(підпис)

Науковий керівник:
професор, доктор технічних наук
Заславський Володимир Анатолійович _____
(підпис)

Засвідчую, що в цій роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент _____ (підпис)

Роботу розглянуто й допущено
до захисту на засіданні кафедри
математичної інформатики
«____» _____ 2021 р., протокол № ___
Завідувач кафедри
Терещенко В. М. _____
(підпис)

Київ – 2021

РЕФЕРАТ

Обсяг роботи 50 сторінок, 9 ілюстрацій, 2 таблиці, 18 джерел посилань.

АФЕКТИВНІ РОЗЛАДИ, ВЕЛИКИЙ ДЕПРЕСИВНИЙ РОЗЛАД, ГЛИБОКЕ НАВЧАННЯ, НАЇВНИЙ КЛАСИФІКАТОР БАЄСА, ВКЛАДАННЯ, КЛАСИФІКАЦІЯ, МОДЕЛЬ, НЕЙРОННІ МЕРЕЖІ.

Об’єктом дослідження є діагностування афективних розладів у пацієнтів.

Предмет дослідження - діагностування афективних розладів на основі аналізу природної мови.

Метою дослідження є пошук та реалізація декількох методів та моделей діагностування афективних розладів на основі аналізу текстів природної мови.

В результаті роботи було реалізовано дві моделі для визначення депресивного забарвлення тексту з точністю d1 = 0.9628078863891132 та d2 = 0.9272.

Дані моделі можуть бути використані у додатках для зв’язку з терапевтами та чатах для психологічної допомоги.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

NLP - natural language processing, аналіз текстів природної мови

ML - machine learning, машинне навчання

МКХ - міжнародна класифікація хвороб

DSM - Diagnostic and Statistical Manual of mental disorders, Діагностично-статистичному посібнику психічних розладів

ВДР - Великий депресивний розлад

PHQ - The Patient Health Questionnaire, анкета здоров’я пацієнта

ROC - Receiver Operating Characteristic, робоча характеристика приймача

AUC - Area Under the Curve, площа під кривою

ВСТУП

Багато досліджень, проведених різноманітними експертами, а також ВООЗ, прогнозують, що уже у 20-х роках XXI століття афективні розлади випередять хвороби серця, за числом хворих та вийдуть на перше місце на планеті. Депресія з уніполярним плином займе друге місце після ішемічної хвороби серця.

Пандемія коронавірусної інфекції 2019 (COVID-19) також пов’язана з проблемами психічного здоров’я. Це спричинено високою захворюваністю та смертністю, в результаті хвороби, та діями щодо пом’якшення наслідків, включаючи вплив фізичного дистанціювання та постійного перебування вдома. Симптоми тривоги та депресивний розлад значно зросли протягом квітня – червня 2020 року у порівнянні з тим же періодом 2019 року.

Депресія, зокрема, є одним з найбільш поширених захворювань у всьому світі. Ризик захворювання на депресію протягом життя становить близько 20%. Депресія є поширеною хворобою, на яку в усьому світі страждає понад 264 мільйони людей [1]. Особливо коли тривала і з помірною або важкою інтенсивністю, депресія може стати серйозним станом здоров'я. У гіршому випадку депресія може призвести до самогубства. Близько 800 000 людей помирають через самогубство щороку. Самогубство є другою причиною смерті у віці 15-29-річних. До цього ж, у 70% випадків після першого депресивного епізоду спостерігаються рецидиви захворювання [1].

Хоча відомі ефективні методи лікування психічних розладів, від 76% до 85% людей у країнах з низьким та середнім рівнем доходу не отримують лікування від свого розладу.

Актуальність проблеми афективних розладів обумовлена їх поширеністю, суттєвим впливом на якість життя, соціальне функціонування людини.

Сучасні інформаційні технології не стоять на місці й переймають усе більше задач, в тому числі у сфері охорони здоров’я. Комп’ютер є незамінним

для обробки великих об’ємів даних та може значно спрощувати життя як лікарям, так і пацієнтам.

Машинне навчання (ML) вже допомагає у різних ситуаціях в галузі охорони здоров'я. ML у галузі охорони здоров’я допомагає аналізувати тисячі різних точок даних та пропонувати результати, забезпечувати своєчасні оцінки ризику, точний розподіл ресурсів та має багато інших застосувань. Для правильної діагностики захворювань потрібні роки медичної підготовки. Діагностика часто є важким, трудомістким процесом. У багатьох сферах попит на експертів значно перевищує наявну пропозицію. Це створює навантаження на лікарів і часто затримує життєво важливу діагностику пацієнтів. Машинне навчання, особливо алгоритми глибокого навчання, останнім часом досягло величезного прогресу в автоматичному діагностуванні захворювань, зробивши діагностику дешевшою та доступнішою. Алгоритми машинного навчання можуть навчитися бачити закономірності аналогічно тому, як їх бачать лікарі.

Отже, **об’єктом дослідження** є діагностування афективних розладів у пацієнтів.

Предмет дослідження - діагностування афективних розладів на основі аналізу природної мови.

Метою дослідження є пошук та реалізація декількох методів та моделей діагностування афективних розладів на основі аналізу текстів природної мови.

Для досягнення мети були поставленні наступні завдання:

- дослідити класифікацію та симптоматику афективних розладів та обрати один чи два для подальшого дослідження;
- дослідити специфіку діагностування обраного розладу без використання технологій;
- розглянути задачі наближені до задачі діагностування афективних розладів;
- вибрати один чи декілька методів для реалізації;

- обрати мову, фреймворк, бібліотеки для реалізації та тестування моделі;
- знайти та за потреби модифікувати дані для дослідження та навчання моделі;
- реалізація моделей, їх налагодження, навчання та перевірка на тестових даних;
- якщо можливо, порівняння результатів відносно різних підходів, та/або різних розладів.

З ростом популярності текстових комунікацій, з’являється дедалі більше додатків для спілкування з психотерапевтами в режимі онлайн у текстовому форматі. Результати даної роботи можуть бути використані, як частина платформи для комунікації терапевта та пацієнта для постійного моніторингу стану пацієнта та прогресу лікування.

РОЗДІЛ І
ДІАГНОСТУВАННЯ АФЕКТИВНИХ РОЗЛАДІВ ПАЦІЄНТІВ ТА ВИКОРИСТАННЯ АНАЛІЗУ ТЕКСТІВ ПРИРОДНОЇ МОВИ ДЛЯ ЇХ ДОСЛІДЖЕННЯ. СУЧАСНИЙ СТАН

1.1 Афективні розлади та їх класифікація

Афективний розлад – це вагома група психічних захворювань, що мають ендогенну і хронічну природу, вони класифікуються згідно МКХ і підрозділяються на безліч категорій. Головне прояв їх в особливостях настрою. Варто позначити, що настрій здатне проявляти варіативність, не обов'язково у вигляді зниження або підвищення. Ця група сильно поширюється в розвинених урбанізованих країнах, де не прийнято яскраво або бурхливо виражати емоції.

До афективних розладів відносять такі порушення, при яких провідна психічна патологія міститься у зміні афекту або настрою, найчастіше у бік його пригнічення чи підвищення. Ця зміна переважно супроводжується зміною загального рівня активності, а більшість інших симптомів має вторинний характер. Як правило, ці розлади мають тенденцію до повторюваності, а початок окремих епізодів порушень нерідко пов'язаний зі стресовими подіями чи ситуаціями.

Типи афективних розладів розподіляються за МКХ 10 і мають чітку класифікацію зі своїми критеріями діагностик [2]. Залежно від «знаку» афекту, відрізняють маніакальні (F30.0-F30.9), депресивні (F32.0-F32.9) та змішані, біполярні (F31.0-F31.9) розлади настрою. Досить часто ці розлади мають тенденцію до свого повторювання. В цих випадках застосовуються коди F33.0-F33.9. Їх повторне виникнення може настати після перенесення людиною будь-якої стресової ситуації, але частіше виникає саме по собі. До афективних розладів відносяться і випадки, коли ступінь зниження настрою

(F34.1 – дистимія) або його полярні зміни (F34.0 – циклотимія) є недостатньо виразними, але мають хронічний характер [2].

•F30 – маніакальний епізод, який проявляється якимись окремими симптомами з тріадою Ясперса. До неї належать покращений настрій, прискорення мислення і підвищення рухової активності.

• F31 – біполярний афективний розлад має у своєму складі симптоматику як маніакального, так і депресивного епізодів.

• F32 – депресивний епізод має у своєму складі традиційну тріаду Ясперса для депресій, зниження настрою, рухів і мислення.

• F33 – рекурентний депресивний розлад, має у своєму складі лише депресивні епізоди різного ступеня тяжкості.

• F34 – постійні розлади настрою, в яких на передній план виходять депресивні епізоди, що стають нормою.

• F38 – інші розлади афективного спектра, що можуть мати у своєму складі змішані епізоди, які можуть характеризуватись ознаками і депресії, і манії одночасно, але не усіма класичними симптомами одразу..

У цій роботі буде розглянуто детальніше лише депресивні розлади (F32.0-F32.9).

1.1.1 Великий депресивний розлад та його симптоматика

Великий депресивний розлад, який ще часто називають клінічною депресією, є захворюванням, яке може впливати на багато сфер життя хворого. Прикладом впливу є зміни настрою і поведінки, а також проблеми з такими фізичними функціями, як апетит і сон.

Лікар або експерт психічного здоров’я може поставити діагноз великого депресивного розладу на основі симптомів, почуттів та поведінки пацієнта. Щоб поставити діагноз ВДР, потрібно відповідати критеріям симптомів, переліченим у Діагностично-статистичному посібнику психічних розладів (DSM).

Відповідно до його критеріїв:

- пацієнт має зазнати змін у своєму попередньому функціонуванні
- симптоми повинні бути присутні протягом 2 і більше тижнів
- хоча б одним симптомом має бути або пригнічений настрій, або втрата зацікавленості

Пацієнти також повинні відчувати 5 або більше таких симптомів протягом 2-тижневого періоду:

- депресивний настрій, підвищена дратівливість.
- зменшення цікавості до речей, які колись подобались.
- зміни в апетиті та/або вазі тіла.
- сонливість або безсоння.
- почуття неспокою.
- постійна втома та брак енергії.
- безпричинне відчуття провини та меншовартості.
- проблеми із зосередженням та прийняттям рішень.
- суїцидальні думки або спроби самогубства.

Головним інструментом для скринінгу депресії є PHQ-9. PHQ-9 - це інструмент із 9 запитаннями, що надається пацієнтам, які перебувають у стаціонарі первинної медичної допомоги, для обстеження на наявність та тяжкість депресії. Це депресійна шкала депресії з опитувальника здоров'я пацієнта.

Неможливо назвати одну причину депресії, адже депресія розвивається в результаті складної взаємодії соціальних, психологічних та біологічних факторів.

Дуже часто поштовхом до депресії стають негативні події. Своєю чергою, депресія може послабити стресостійкість, порушити нормальну життєдіяльність, погіршити життєву ситуацію, в результаті чого депресія

переходить у більш важку форму. Існує взаємозв'язок між депресією та фізичним здоров'ям. Наприклад, серцево-судинні хвороби можуть призвести до розвитку депресивного стану та навпаки.

Легку форму депресії можна вилікувати без медикаментів, за допомогою певних психотерапевтичних методів. Але при середній та важкій формі може знадобитись медикаментозне лікування і психотерапія з кваліфікованими спеціалістами.

1.2 Класифікація текстів та визначення інтонаційного забарвлення

Найбільш наближеною до діагностування афективних розладів на основі мови є задача визначення інтонаційного забарвлення, яка у свій час базується на задачі класифікації текстів [3].

Алгоритми класифікації є основою техніки видобування тексту. Як правило, методи класифікації можна поділити на статистичні підходи та підходи з машинним навчанням (ML). Класифікація методів зображена на рис 1.1.

Статистичні методи лише задовольняють попередньо проголошені гіпотези, отже, і необхідність для цих алгоритмів менша, але техніки ML були спеціально винайдені для автоматизації. Алгоритми в основному поділяються на контрольовані, неконтрольовані та напівконтрольовані категорії відповідно до критеріїв навчання.

Серед контрольованих алгоритмів класифікації є дві категорії, а саме параметричні та непараметричні, поділені за перевагою параметрів у даних. Логістична регресія та наївний класифікатор Баєса є найбільш широко використовуваними алгоритмами параметричної класифікації. Підтримка Vector Machine (SVM), дерево рішень, індукція правил, K-nn та нейронні мережі є непараметричними аналоги. Нечітка кластеризація c-середніх, кластеризація методом k середніх та ієрархічна кластеризація - це непідконтрольні підходи до навчання [4].

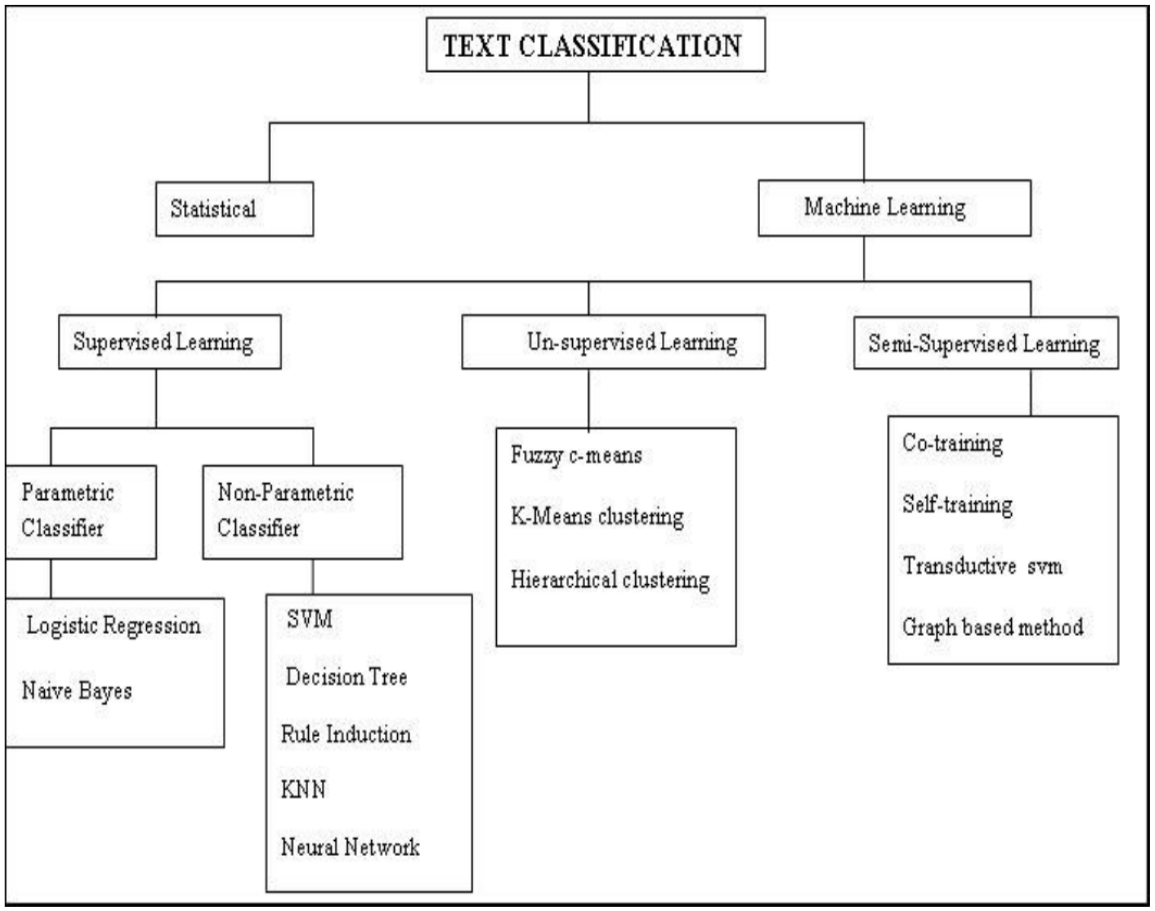

Рисунок 1.1 - Класифікація методів класифікації текстів [4]

1.2.1.1 Наївний класифікатор Баєса

Наївний текстовий класифікатор Баєса широко застосовується для завдань категоризації документів з 1950-х років. Наївний класифікатор Баєса теоретично заснований на теоремі Байєса, яка була сформульована Томасом Байєсом. Нещодавні дослідження широко застосовували цю техніку для пошуку інформації. Цей прийом є генеративною моделлю, яка є найбільш традиційним методом категоризації тексту. Ми починаємо з найосновнішої версії НБК, яка була розроблена за допомогою TF (bag-of-word), техніки вилучення особливостей, яка підраховує кількість слів у документі.

Якщо кількість документів (n) входить до k категорій, де k ∈ {c1, c2, ..., ck}, передбачуваний клас як вихід є c ∈ C. Наївний алгоритм Байєса можна описати наступним чином [5]:

$$P(c \mid d) = \frac{P(d \mid c)\, P(c)}{P(d)},$$

де d це документ (текст) і c позначає класи.

$$C_{\text{MAP}} = arg\,max_{c \in C} P(d \mid c)\, P(c) =$$
$$= arg\,max_{c \in C} P(x_1, x_2, \ldots, x_n \mid c) p(c)$$

Ця модель слугує основою для багатьох досліджень і загально визначається таким чином:

$$P(c_j \mid d_i; \hat{\theta}) = \frac{P(c_j \mid \hat{\theta}) P(d_i \mid c_j; \hat{\theta}_j)}{P(d_i \mid \hat{\theta})}$$

Хоча наївний класифікатор Баєса є одним із найстаріших, він є досить простим та на диво ефективним, тому і досі застосовується для вирішення багатьох завдань. Через широку сферу використання існує багато модифікацій даного методу.

1.2.1.2 Поліноміальний наївний класифікатор Баєса

Якщо кількість документів (n) входить до k категорій, де k ∈ {c1, c2, ..., ck}, передбачуваний клас як вихід є c ∈ C. Поліноміальний алгоритм наївного Байєса можна описати наступним чином:

$$P(c \mid d) = \frac{P(c) \prod_{w \in d} P(d \mid c)^{n_{wd}}}{P(d)},$$

де n_{wd} позначає кількість випадків, коли слово w зустрічається в документі, а P (w | c) - ймовірність спостереження слова w для класу c [6].

P (w | c) розраховується як:

$$P(w \mid c) = \frac{1 + \sum_{d \in D_c} n_{wd}}{k + \sum_{w'd \in D_c} n_{w'd}}$$

Наївний алгоритм Байєса також має кілька обмежень. НБК робить тверде припущення щодо форми розподілу даних. НБК також обмежений дефіцитом даних [5].

1.2.1.3 Нейронні мережі

Штучні нейронні мережі працюють так само як і мозок людини, приймаючи рішення. Ройовий інтелект та алгоритм еволюції використовуються для узагальнення моделі нейронної мережі. Це працює на чесноті навчання та еволюції з мінімальним або відсутнім втручанням людини. Нейронні мережі також популярні серед випадків, коли необхідно застосувати ієрархічний багатозначний класифікаційний підхід. Цей вид класифікації є складним, оскільки кожна вибірка може належати до кількох класів і прогнози одного рівня подаються як вхідні дані на наступний рівень для прийняття остаточного рішення.Нейронна мережа формує основу ансамблю за допомогою композитних пнів. Нейронні мережі мають хорошу цінність застосування, потенціал розвитку, і також не потрібно навчати окремі двійкові класифікатори для багатокласових задач, тому вони утворюють кращі базові класифікатори з ансамблевим підходом.

Глибоке навчання відображає вхідні дані на вихідні дані та знаходить кореляції. Також відоме як "універсальний апроксиматор", оскільки може навчитися апроксимувати невідому функцію f (x) = y між будь-яким входом x і будь-яким вихідним y, припускаючи, що вони взагалі пов'язані (наприклад, за

допомогою кореляції або причинно-наслідкових зв'язків). У процесі навчання нейронна мережа знаходить правильний f або правильний спосіб перетворення x в y, будь то f (x) = 3x + 12 або f (x) = 9x - 0,1. Ось кілька прикладів того, що може зробити глибоке навчання.

Поглиблене навчання - це назва, яку ми використовуємо для “нейромереж із накопиченням”; тобто мережі, що складаються з декількох шарів.

Шари зроблені з вузлів. Вузол - це просто місце, де відбувається обчислення. Вузол поєднує вхідні дані з набором коефіцієнтів або вагових коефіцієнтів, які або підсилюють, або гасять цей вхідний сигнал, тим самим надаючи значення вхідним даним щодо характеристик, які намагається вивчити алгоритм; наприклад, який вхід є найбільш корисним для класифікації даних без помилок? Ці продукти вхідної ваги підсумовуються, а потім сума передається через так звану функцію активації вузла, щоб визначити, чи повинен і в якій мірі цей сигнал просуватися далі через мережу, щоб вплинути на кінцевий результат, скажімо, на акт класифікації. Якщо сигнали проходять, нейрон «активується».

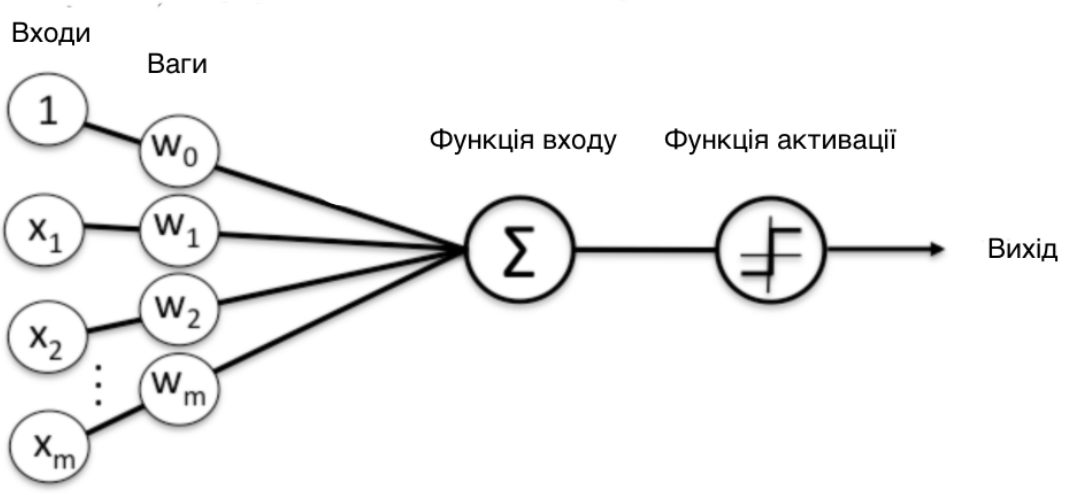

Рисунок 1.2 - Схема шару нейронної мережі

Шар вузлів(рис. 1.2) - це ряд тих нейроноподібних перемикачів, які вмикаються або вимикаються, коли вхід подається через мережу. Вихідні дані кожного шару є одночасно вхідними даними наступного шару, починаючи з початкового вхідного шару, що приймає дані.

Поєднання регульованих ваг моделі з функціями введення - це те, як ми надаємо значення цим характеристикам щодо того, як нейронна мережа класифікує та кластеризує введення. Мережі глибокого навчання відрізняються від більш загальноприйнятих одношарових нейронних мереж своєю глибиною; тобто кількість шарів вузлів, через які дані повинні проходити в багатоступеневому процесі розпізнавання шаблонів та характеристик.

Ранні версії нейронних мереж, такі як перші персептрони, були не глибокими, складалися з одного вхідного та одного вихідного шарів і мали не більше одного прихованого шару між ними. На сьогодні, мережі, що мають більш як три рівні (включаючи вхідні та вихідні дані) кваліфікуються як “глибоке” навчання.

У мережах глибокого навчання кожен рівень вузлів тренується за окремим набором функцій на основі результатів попереднього рівня. Чим далі ви просуваєтеся в нейронну мережу, тим складніші функції можуть розпізнавати ваші вузли, оскільки вони узагальнюють і рекомбінують функції попереднього шару.

Мережі глибокого навчання виконують автоматичне вилучення функцій без втручання людини, на відміну від більшості традиційних алгоритмів машинного навчання. Враховуючи, що видобуток функцій - це завдання, на виконання якого командам дослідників даних може знадобитися багато років, глибоке навчання - це спосіб обійти точки задухи обмежених експертів. Це збільшує повноваження малих команд з обробки даних, які за своєю природою не масштабуються [7].

1.2.2 Вбудовування слів

При обробці природною мовою (NLP) вбудовування слів - це термін, що використовується для подання слів для аналізу тексту, як правило, у формі дійсного вектора, який кодує значення слова таким чином, що слова, що знаходяться ближче до вектора очікується, що простір буде подібним за значенням. Вбудовування слів можна отримати за допомогою набору мовних моделей та методів навчання особливостей, коли слова або фрази зі словникового запасу відображаються у вектори реальних чисел. Концептуально, це передбачає математичне вбудовування з простору з багатьма вимірами на слово, в неперервний векторний простір з набагато нижчою розмірністю.

Методи генерування цього відображення включають нейронні мережі, зменшення розмірності на матриці спільного виникнення слова, імовірнісні моделі, пояснюваний метод бази знань, та явне представлення в термінах контексту, в якому з'являються слова. Показано, що вбудовані слова та фрази, коли вони використовуються як основне вхідне представлення, підвищують ефективність таких завдань, як синтаксичний розбір та аналіз настроїв та емоційного забарвлення.

Скіп-грам - одна з технік навчання без нагляду, яка використовується для пошуку найбільш споріднених слів для даного слова.

Skip-gram [8] використовується для передбачення контекстного слова для даного цільового слова. Тут цільове слово вводиться, тоді як контекстні слова виводяться. Оскільки передбачається більше одного контекстного слова, це дещо ускладнює проблему.

Як бачимо на рис. 1.3, w (t) - це цільове слово або введене введення. Існує один прихований шар, який виконує точковий добуток між ваговою матрицею та вхідним вектором w (t). У прихованому шарі не використовується функція активації. Тепер результат точкового добутку на прихованому шарі

передається вихідному шару. Вихідний рівень обчислює точковий добуток між вихідним вектором прихованого шару та ваговою матрицею вихідного шару. Потім ми застосовуємо функцію активації softmax для обчислення ймовірності слів, які здаються в контексті w (t) у даному контексті.

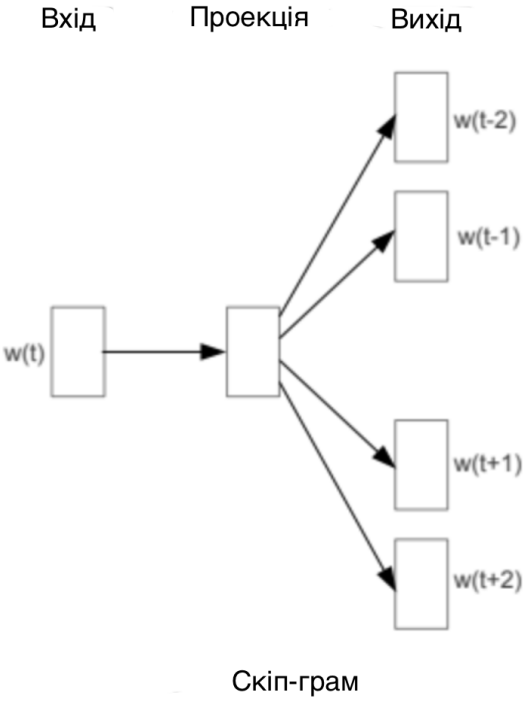

Рисунок 1.3 - Архітектура Скіп-грам моделі

Виконуються наступні кроки:

1. Слова перетворюються у вектор за допомогою одного гарячого кодування. Розмірність цих векторів дорівнює [1, | v |].

2. Слово w (t) передається прихованому шару з | v | нейрони.

3. Прихований шар виконує точковий добуток між ваговим вектором W [| v |, N] та вхідним вектором w (t). З цього можна зробити висновок, що (t)-й рядок W [| v |, N] буде вихідним (H [1, N]).

4. На прихованому рівні не використовується функція активації, тому H [1, k] буде передано безпосередньо вихідному шару.

5. Вихідний рівень застосовуватиме крапковий добуток між H [1, N] і W ’[N, | v |] і дасть нам вектор U.

6. Тепер, щоб знайти ймовірність кожного вектора, ми будемо використовувати функцію softmax. Оскільки кожна ітерація дає вихідний вектор U, який має один тип гарячого кодування.

7. Слово з найбільшою ймовірністю є результатом, і якщо передбачене слово для даної контекстної позиції неправильне, тоді ми будемо використовувати зворотне розповсюдження для модифікації наших вагових векторів W і W ’.

Ці кроки виконуватимуться для кожного слова w (t), що є у словниковому запасі. І кожне слово w (t) буде передано k разів. Отже, ми можемо бачити, що пряме поширення буде оброблятися | v | * k разів у кожну епоху.

Переваги:

- навчання без нагляду, отже може працювати над будь-яким поданим необробленим текстом.
- вимагає менше пам'яті у порівнянні з іншими словами для векторних подань.
- потрібні дві вагові матриці розмірності [N, | v |] кожна замість [| v |, | v |]. І зазвичай N становить близько 300, тоді як | v | складає мільйони. Отже, ми можемо побачити перевагу використання цього алгоритму.

1.2.2.1 GloVe вкладання

GloVe - це керований алгоритм навчання для отримання векторних подань для слів. Навчання виконується на основі агрегованої загальної статистики співіснування слова-слова з корпусу, і отримані подання демонструють цікаві лінійні підструктури векторного простору слів.

Статистика входження слів у текст є основним джерелом інформації, доступною для всіх неконтрольованих методів вивчення подань слів, і хоча

зараз існувало багато таких методів, все ще залишалися питання про те, як значення генерується з цієї статистики та як словесні вектори можуть представляти це значення.

Можна почати з простого прикладу, який демонструє, як певні аспекти значення можуть бути вилучені безпосередньо із ймовірностей співіснування. Розглянемо два слова i та j, які виявляють певний аспект, що цікавить; для конкретності, припустимо, нас цікавить поняття термодинамічної фази, для якої ми можемо взяти i = ice (лід) і j = steam (пар). Взаємозв'язок цих слів можна дослідити, вивчивши співвідношення ймовірностей їх спільної зустрічі з різними словами-зондами, k. Для слів k, що стосуються льоду, але не пари, скажімо k = solid (твердий), ми очікуємо, що співвідношення P_{ik}/P_{jk} буде великим. Аналогічно, для слів k, що стосуються пару, але не льоду, скажімо k = gas (газ), коефіцієнт повинен бути малим. Для слів k, таких як water (вода) або fashion (мода), які відносяться як до льоду, так і до пари, або до жодного, співвідношення повинно бути близьким до одиниці. У порівнянні з вихідними ймовірностями, коефіцієнт краще розрізняє відповідні слова (тверде тіло та газ) від нерелевантних слів (вода та мода), а також краще розрізняє два відповідні слова. Це свідчить про те, що відповідною відправною точкою для вивчення векторів слів має бути співвідношення ймовірностей співіснування, а не самих ймовірностей. Зазначаючи, що відношення P_{ik}/P_{jk} залежить від трьох слів i, j та k, найбільш загальна модель набуває вигляду [9],

$$F(w_i, w_j, w_k) = \frac{P_{ik}}{P_{jk}}.$$

Модель глобальних векторів (GloVe), яка поєднує переваги моделі word2vec skip-gram, які ми описали вище, коли мова йде про завдання аналогії слів, та переваги методів факторизації матриць, які можуть використовувати глобальну статистичну інформацію.

РОЗДІЛ ІІ
МОНІТОРИНГ ДАНИХ-РЕЗУЛЬТАТІВ ДІАГНОСТУВАННЯ АФЕКТИВНИХ РОЗЛАДІВ ДЛЯ АНАЛІЗУ

2.1 Пошук даних-результатів діагностування афективних розладів

Перед початком потрібно було зрозуміти, з якими даними нам доведеться працювати. Оскільки ми починали роботу з моделлю для визначення емоційного забарвлення, ми почали з пошуку набору даних (датасету) саме для цієї задачі. На щастя, датасетів для аналізу подібних питань є достатньо у відкритому доступі в інтернеті.

З самого початку використовувався набір даних з відгуками про фільми. З такого датасету було легко почати, тому що там був дуже чіткий та зрозумілий поділ між негативними та позитивними відгуками. Для того, щоб відразу зробити задачу більш подібною на нашу головну, діагностування клінічної депресії, ми ігнорували випадки з нейтральними ревью і використовували для першої моделі лише яскраво негативні та яскрава позитивні відгуки.

Наступним кроком було перейти від емоцій гніву та невдоволення стосовно фільмів до емоційних забарвлень більш властивих депресії. Таких як сум та пригнічення. Іншим важливим нюансом був факт, що все-таки, коли люди пишуть відгуки на фільми вони обирають слова та відфільтровують їх певним чином, аби власне і надати яскравішого забарвлення своєму коментарю. Нас же цікавила ефективність визначення емоції в тих випадках, коли автор не має наміру напряму доносити чи демонструвати цю емоцію засобами тексту. Для цього ми звернулися до датасетів, що мають у собі тексти користувачів із соціальних мереж. Тут вибір випав на соціальну мережу твіттер з декількох причин. По-перше, доступність даних. Оскільки твіттер має чудове відкрите API для розробників існує дуже велика кількість датасетів з

текстами саме з твіттеру. Іншою причиною було обмеження твіттеру на кількість символів.

Це своєю чергою має наступні переваги:

- атомарність емоції. Тобто вірогідність випадку, коли в одному твіті відбудеться перехід від однієї емоції, то іншої, як це може статися з великими текстами в кілька сотень слів, зводиться до мінімум, що дещо спрощує нашу задачу.
- менші тексти потребують менше часу на обробку та аналіз, а отже таким чином теж ведуть до спрощення поставленої задачі.

При роботі з цим датасетом нас цікавило відловлювання випадків текстів з забарвленням суму, печалі, туги в одну групу. Всі інші, включаючи нейтральні ми хотіли класифікувати, як іншу групу. Приклад одного рядку з датасету з твіттеру можна знайти в Додатку А.

Останнім датасетом, до якого ми перейшли була комбінація двох. Один, як і попередні просто був текстовим файлом з постами з твіттеру та усією інформацією про сам пост: його текст, автор, дата публікації, айді. Інший файл містив у собі айді посту та позначку чи є цей текст депресивним. Цей додаток був створений групою терапевтів для досліджень пов’язаних з депресію та соціальними мережами, але дуже добре підходив для наших тренувань.

В цьому датасеті всього було 3574 текстів вигружених з твіттеру за допомогою API. У іншому файлі відповідно було також 3574 входжень з визначенням депресивності відповідного тексту. Ці значенням могли бути наступними: -1, 0, 1. Надалі ми дещо змінили датасет, об’єднавши множини промарковані 0 та 1. Більшість текстів були промарковані, як депресивно забарвлені. Оскільки значна частина була вивантажена з твіттеру використовуючи пошук по тегах і були використані такі теги як “депресія”, “тривога”, і т. д.

Тут варто зазначити два моменти. Усі датасети, що розглядалися нами були англійською мовою. Для цього існує ряд причин. Головною була

доступність даних та кількість. На жаль, не так багато вже існуючих датасетів містять у собі дані українською. Також очевидно, що кількість інформації у датасетах українською мовою буде програвати англійській банально через кількість носіїв мови, особливо на просторах соціальних мереж. Ще однією перевагою використання англійської мови було існування уже готових ембеденгів. Тобто нам не потрібно було генерувати своє перетворення слова у вектор, а можна було скористатися уже існуючим. Це не лише виграш у часі, але й у якості. Оскільки існуючі ембединги, такі як word to vec and glove були треновані на величезних об'ємах даних і містять у собі значно більші словники, ніж той, який би було отримано в результаті аналізу текстів, які було розглянуто. Тобто вірогідність, що слово із тексту, який було перевірено, буде відсутнє у словнику, тому що воно було відсутнє у текстах тренувального набору зводилася до мінімуму.

Другим важливим моментом є відносна неточність. Ці позначки на текстах, що визначали їх депресивність були проставлені базуючись виключно на самих текстах, що не є ідеальним для дослідження. У цьому випадку найкращим був би варіант, якби ця позначка проставлялася після співбесіди психотерапевта з автором цього тексту. Це значно вплинуло б на точність моделі та її ваг і сприяло б отриманню не лише більш точних, але і цікавіших результатів.

2.2 Попередня підготовка даних

Загалом тексти та документи є неструктурованими наборами даних. Однак, ці неструктуровані текстові послідовності повинні бути перетворені у структурований простір об'єктів при використанні математичного моделювання, як частини класифікатора. По-перше, дані потрібно очистити, щоб опустити непотрібні символи та слова.

Більшість наборів даних тексту та документа містять багато непотрібних слів, таких як стоп-слова, орфографічні помилки, сленг тощо. У багатьох

алгоритмах, особливо в статистичних та імовірнісних алгоритмах навчання, шум і непотрібні функції можуть негативно позначитися на роботі системи.

Токенізація - це метод попередньої обробки, який розбиває потік тексту на слова, фрази, символи, або інші значущі елементи, що називаються лексемами. Головною метою цього кроку є дослідження слів у реченні. Як класифікація тексту, так і розробка тексту вимагають синтаксичного аналізатора який обробляє токенізацію документів.

Таким чином із речення “По-перше, дані потрібно очистити, щоб опустити непотрібні символи та СЛОВА.” ми отримаємо наступну структуру: [‘По’, ’-’, ’перше’, ‘,’, ‘дані’, ‘потрібно’, ‘очистити’, ‘,’, ‘щоб’, ‘опустити’, ‘непотрібні’, ‘символи’, ‘та’, ‘СЛОВА’, ‘.’].

Точки даних тексту та документа мають різноманітне використання великих літер для формування речення. Оскільки документи складаються з безлічі речень, різне використання великих літер може бути надзвичайно проблематичним, коли відбувається класифікація великих документів. Найпоширеніший підхід до розв'язання проблем, пов’язаних з непослідовною капіталізацією полягає у зменшенні кожної літери до малих літер. Цей прийом проєктує всі слова в тексті та документі на той самий простір, але це створює значну проблему для інтерпретації деяких слів (наприклад, “USA” (Сполучені Штати Америки) до “us” (займенник)). Перетворювачі сленгу та абревіатури можуть допомогти врахувати ці винятки.

Тобто тепер ми маємо наступне перетворення речення “По-перше, дані потрібно очистити, щоб опустити непотрібні символи та СЛОВА.”: [‘по’, ’-’, ’перше’, ‘,’, ‘дані’, ‘потрібно’, ‘очистити’, ‘,’, ‘щоб’, ‘опустити’, ‘непотрібні’, ‘символи’, ‘та’, ‘слова’, ‘.’]

Більшість наборів даних тексту та документа містять багато непотрібних символів, таких як пунктуація та спеціальні символи. Критичні знаки пунктуації та спеціальні символи важливі для людського розуміння документів, але це може бути згубним для алгоритмів класифікації.

Використовуючи цей підхід отримуємо наступну трансформацію: “По-перше, дані потрібно очистити, щоб опустити непотрібні символи та СЛОВА.”: [‘по’, ’перше’, ‘дані’, ‘потрібно’, ‘очистити’, ‘щоб’, ‘опустити’, ‘непотрібні’, ‘символи’, ‘та’, ‘слова’]. Після очищення даних можуть застосовуватися формальні методи вилучення ознак. Поширеними методами вилучення ознак є документ зі зворотною частотою термінів.

Зменшення розмірності: оскільки набори даних тексту або документа часто містять багато унікальних слів, кроки попередньої обробки даних можуть бути повільними через високий час та складність пам'яті. Загальне рішення для розв'язання цієї проблеми - використовувати прості, недорогі алгоритми. Однак у деяких наборах даних ці алгоритми працюють не так добре, як очікується. Щоб уникнути зниження продуктивності, багато дослідників воліють використовувати зменшення розмірності, щоб зменшити час і складність пам'яті для своїх додатків. Використання зменшення розмірності для попередньої обробки може бути більш ефективним, ніж розробка недорогих класифікаторів.

В роботі використовувався вбудований список англійських стоп слів, тобто слів, які не будуть додані в словник, з пакета sklearn. Цей список містить у собі 318 слів за замовчуванням. Але цей список був розширений деякими специфічними слова. Наприклад, “retweet” та скорочення для цього ж слова “rt”, що є специфічним словом для твіттеру, та не несе ніякого емоційного забарвлення.

При формуванні словника для моделі глибинного навчання також потрібно було визначити OOV token (Out of vocabulary token). Це робиться на той випадок, якщо якесь зі слів з речень для тестування буде відсутнє у нашому словнику. В цьому випадку воно буде замінене OOV токеном.

Також для моделі глибинного навчання ми перетворили звичайний словник на індекс слів. Тобто слова не просто зберігаються в списку, але і є відсортованими за частотою появи слова в тренувальних реченнях. Тобто, як

приклад, для тексту “I was there. It was intense. It wasn’t great.” на виході ми отримаємо {'<OOV>': 1, 'was': 2, 'it': 3, 'i': 4, 'there': 5, 'intense': 6, 'wasn’t': 7, 'great': 8}. Ми також створювали обернений індекс, таким чином спрощували пошук слова за його “порядковим номером”.

Після створення словнику та індексу кожен текст кодується і перетворюється у набір чисел. Якщо ми розглянемо приклад з попереднім текстом “I was there. It was intense. It wasn’t great.”, після кодування ми отримаємо [4, 2, 5, 3, 2, 6, 3, 7, 8]. Власне перехід до числових значень і дозволяє нам розрахувати ваги й пізніше зробити припущення на їх основі. Але це ще останні маніпуляції, які нам потрібно зробити перед тим, як ці дані можна буде подати на вхід моделі.

Наступним кроком буде нормалізація даних. Очевидно, що кожен текст має різну кількість слів у ньому. Таким чином, з кожним текстом, що буде поданий на вхід моделі ми не можемо точно знати, яка кількість чисел буде у конкретному списку, що відображає текст. Для цього ми застосуємо операцію padding. Якщо ми мали один текст довжиною 5 слів [123, 5876, 4, 56, 890], та наступний текст довжиною 3 [4389, 704, 51], після операції паддингу отримаємо [123, 5876, 4, 56, 890] та [4389, 704, 51, 0, 0]. Тобто усі представлення тренувальних текстів будуть мати один вимір. Важливо зазначити, що з цією операцією потрібно буде дуже обережним. Оскільки якщо вибрати малу вимірність, частина даних просто зникне, а якщо обрати надто велику, може значно ускладнити обчислення. Ми уже згадували переваги використання текстів з саме твіттеру, але тут можна назвати ще один. Усі пости у твіттері не повинні перевищувати 280 символів. В середньому тренувальні тексти містили 15 слів, тому початкове значення для нормалізації було визначене на рівні 30, але змінювалося протягом процесу поліпшення моделі.

РОЗДІЛ ІІІ
МОДЕЛІ, МЕТОДИ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ДЛЯ ДІАГНОСТУВАННЯ АФЕКТИВНИХ РОЗЛАДІВ З ВИКОРИСТАННЯМ АНАЛІЗУ ПРИРОДНОЇ МОВИ

3.1 Технології, що використовувалися при реалізації

Python є прекрасним вибором для реалізації задач поставлених у проєкті. Існує дві основні причини: його простота, а також повний спектр доступних бібліотек і фреймворків.

Коли справа доходить до великих даних або просто дуже складного шляху прийняття рішень, простота є ключем. Python добре відомий своєю читабельністю, сприятливим для розробника синтаксисом і семантикою.

Великий вибір бібліотек є однією з головних причин, чому Python є найпопулярнішою мовою програмування для AI. Бібліотека є модулем або групою модулів, опублікованих різними джерелами, які включають попередньо написаний фрагмент коду, який дозволяє користувачам досягати певних функцій або виконувати різні дії. Бібліотеки Python надають елементи базового рівня, тому розробники не повинні створювати їх з самого початку.

Keras - це бібліотека нейронних мереж з відкритим кодом, написана на Python. Він здатний працювати поверх TensorFlow, Microsoft Cognitive Toolkit, Theano або PlaidML. Призначений для швидкого експериментування з глибокими нейронними мережами.

Основними перевагами Keras є:

- Зручність для користувачів. Keras - це API, призначений для людей, а не для машин. Вона ставить перед користувачами та центром. Компанія Keras дотримується найкращих практик для зниження когнітивного навантаження: вона пропонує послідовні та прості API, мінімізує кількість дій користувача, необхідних для звичайних випадків використання, і надає чіткі та дієві відгуки про помилку користувача.

- Модульність. Модель розуміється як послідовність або графік автономних, повністю настроюваних модулів, які можна під'єднати разом з якомога меншою кількістю обмежень. Зокрема, нейронні шари, функції витрат, оптимізатори, схеми ініціалізації, функції активації та схеми регуляризації є усіма окремими модулями, які можна об'єднати для створення нових моделей.
- Легка розширюваність. Нові модулі легко додати (як нові класи та функції), а наявні модулі надають широкі приклади. Для того, щоб легко створювати нові модулі, можна отримати повну виразність, зробивши Keras придатною для передових досліджень.
- Робота з Python. Немає окремих файлів конфігурації моделей в декларативному форматі. Моделі описані в коді Python, який є компактним, легшим для налагодження, і дозволяє легко розширювати.

3.2 Реалізація та тестування моделей для діагностування афективних розладів

Обидві моделі були написані та протестовані спочатку виключно для аналізу емоційного забарвлення, а потім адаптовані для аналізу депресивних текстів. Для кожної моделі була розрахована точність та втрата.

Сам код можна умовно розділити на 5 основні модулі:

1. Зчитування на первинне форматування даних
2. Основна обробка даних
3. Створення моделі
4. Тренування моделі
5. Перевірка результатів

Пункти 1 та 2 були описані у розділі 2. Усе стосовно створення на тренування моделей буде описане у розділах 3.2.1 та 3.2.2.

3.2.1 Модель на основі наївного класифікатора Баєса

Для моделі з використанням наївного класифікатора Баєса було використано уже готову модель MultinomialNB з пакета sklearn. Scikit-learn — це безкоштовна програмна бібліотека машинного навчання для мови програмування Python, яка надає функціональність для створення та тренування різноманітних алгоритмів класифікації, регресії та кластеризації, таких як лінійна регресія, random forest, градієнтний бустинг, і працює у зв'язці з бібліотеками NumPy та SciPy

Для визначення точності використовувалися ROC-криві та AUC метрика. Це графік хибнопозитивного коефіцієнта (вісь x) проти справжнього позитивного коефіцієнта (вісь y) для ряду різних порогових значень між 0,0 і 1,0. Іншими словами, графік частоти помилкових припущень в порівнянні з частотою потрапляння. При цьому частоти хибнопозитивних та вірно позитивних припущень розраховуються за формулами:

$$False\ Positive\ Rate\ =\ False\ Positives\ /\ (False\ Positives\ +\ True\ Negatives)$$

$$True\ Positive\ Rate\ =\ True\ Positives\ /\ (True\ Positives\ +\ False\ Negatives)$$

AUC - узагальнений інтеграл або апроксимація площі під ROC кривою. Варто зазначити, що AUC не є явним значенням точності. Використовуючи частоти хибнопозитивних та вірно-позитивних припущень можна перейти до значення точності використавши формулу

$$acc\ =\ \frac{TPR + 1 - FPR}{2},$$

де acc - точність, TPR - частота вірно позитивних припущень і FPR - частота хибнопозитивних припущень. Але оскільки розподіл депресивно забарвлених текстів та інших був нерівномірним в тестовому датасеті, ця метрика була б дуже неточною.

Кінцеве значення AUC становило 0.9628078863891132.

Мінусом моделі була бінарність вихідних результатів та не гнучкість для подальших модифікацій.

3.2.2 Модель з використанням глибинного навчання

Для початку була розроблена базова модель, яка складалася із 3 шарів: одного шару вбудовування та двох щільних шарів.

Шар вбудовування - перший шар моделі, який перетворює слова у відповідні вбудовані слова, загалом перетворює їх на вектори значень.

Keras пропонує рівень вбудовування, який можна використовувати для нейронних мереж на текстових даних. Він вимагає, щоб вхідні дані були кодовані цілими числами, щоб кожне слово було представлене унікальним цілим числом. Цей етап підготовки даних можна виконати за допомогою API Tokenizer, що також надається разом із Keras. Шар Вбудовування ініціалізується випадковими вагами й вивчає вбудовування для всіх слів у навчальному наборі даних.

Шар вбудовування визначається як перший прихований шар мережі. У ньому повинні бути вказані 3 аргументи:

- input_dim: це розмір словникового запасу в текстових даних. Наприклад, якщо ваші дані цілочисельні, закодовані до значень від 0 до 10, тоді розмір словникового запасу складе 11 слів.
- output_dim: це розмір векторного простору, в який будуть вбудовані слова. Він визначає розмір вихідних векторів з цього шару для кожного слова. Наприклад, це може бути 32 або 100 або навіть більше.
- input_length: Це довжина вхідних послідовностей, яку ви визначили б для будь-якого вхідного шару моделі Keras. Наприклад, якщо всі ваші вхідні документи складаються з 1000 слів, це буде 1000.

Після даних отриманих після першого проходження вхідних текстів через цей шар, було отримано дані для візуалізації множини слів, що зустрічалися у тестовому наборі. Використовуючи ці дані було отримано наступні зображення: рис. 3.1.

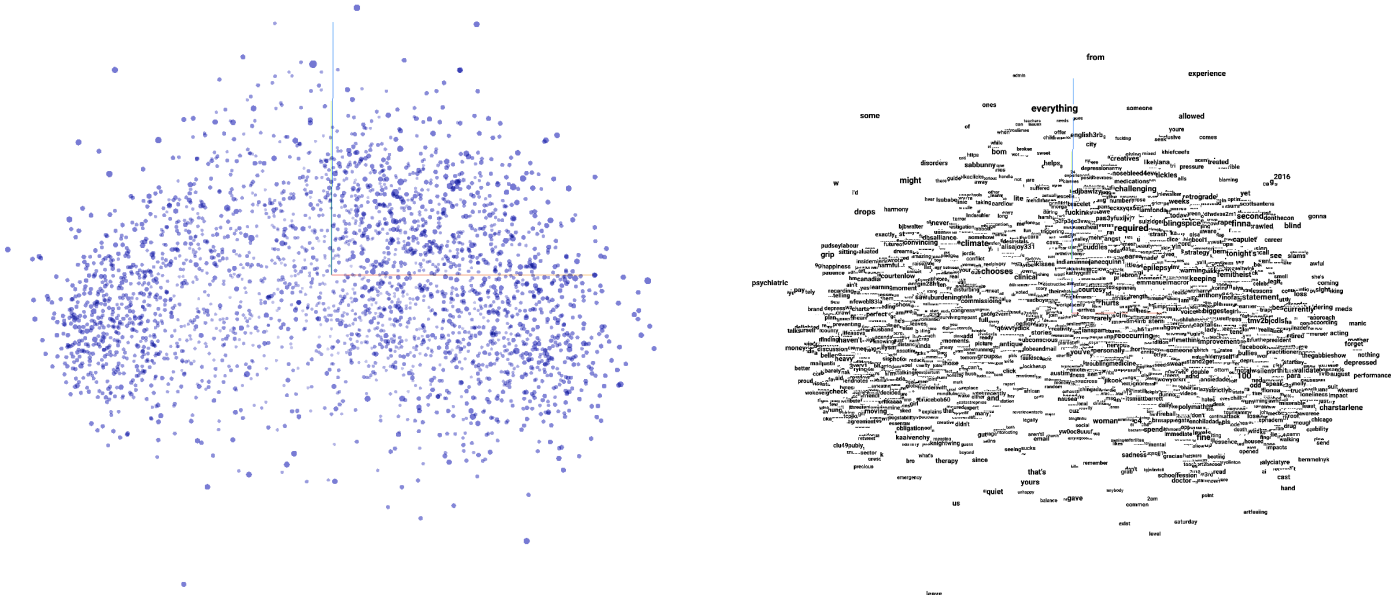

Рисунок 3.1 - Слова зі словника датасету в тривимірному поляризованому просторі

На цих зображеннях можна помітити незначну поляризацію. Тобто немає очевидного розділу на слова, що використовуються у лише у депресивних та не депресивних текстах. Для порівняння, в першому тестовому датасеті з відгуками на кінофільми ця полярність слів зі словника була набагато чіткішою.

Щільний шар - регулярний глибоко пов'язаний шар нейронної мережі. Це найпоширеніший і часто використовуваний шар. Щільний шар виконує нижченаведену операцію на вході та повертає вихід.

На вихідну форму щільного шару впливатиме кількість нейронів / одиниць, зазначених у щільному шарі. Наприклад, якщо вхідною формою є (8,), а число одиниць - 16, то вихідною фігурою є (16,). Весь шар буде мати розмір партії як перший вимір, і тому вхідна фігура буде представлена (None, 8), а вихідна форма як (None, 16). Другий із щільних шарів мав лише один нейрон та функцію активації sigmoid для того, щоб в результаті отримати лише одне значення, що буде лежати в діапазоні [0; 1].

Тренування складалося з 60 епох та проходила у групах по 32, який є розміром груп для тестування за замовчуванням. Тобто тестовий набір даних пройшов через модуль 60 разів.

Але така модель не давала бажаних результатів. Точність не підіймалася вище 0,4, а втрати були аж надто великими. Одним із варіантів причин для такої низької результативності був невеликий розмір датасету та його не різноманітність. Це підвищувало ймовірність перенавчання моделі. Перенавчання трапляється, коли модель дізнається деталі та шум у навчальних даних настільки, що це негативно впливає на результативність моделі на нових даних. Це означає, що шум або випадкові коливання у навчальних даних модель вловлює та вивчає як поняття та характеристики. Проблема полягає в тому, що ці концепції не застосовуються до нових даних і негативно впливають на здатність моделей до узагальнення.

Отже, з першим варіантом моделі ми отримали дещо незадовільні графіки для точності та втрат:

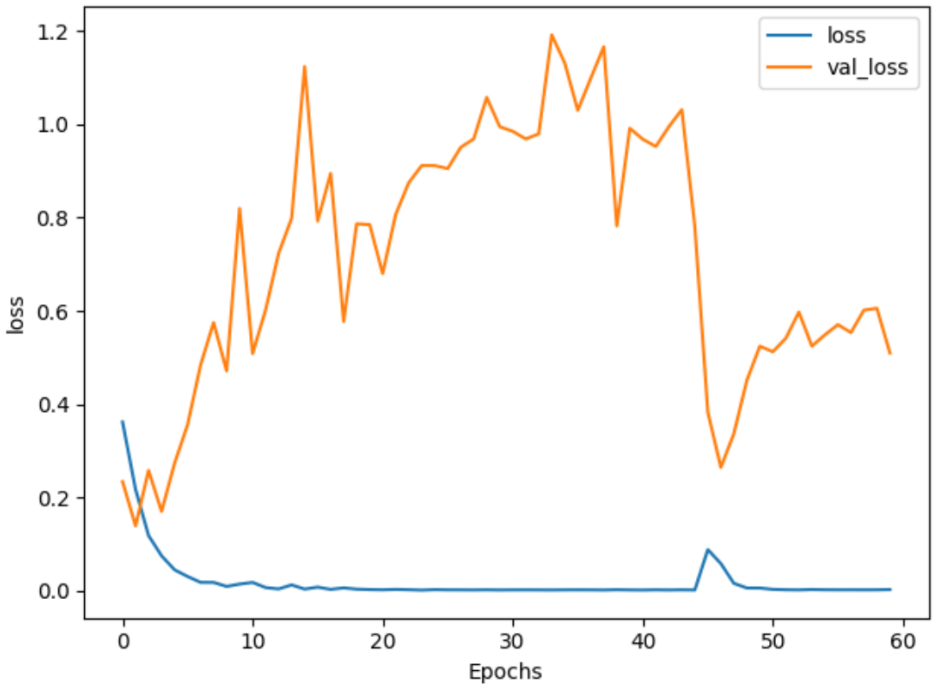

Рисунок 3.2 - Графік зміни точності залежно від часу навчання першої моделі

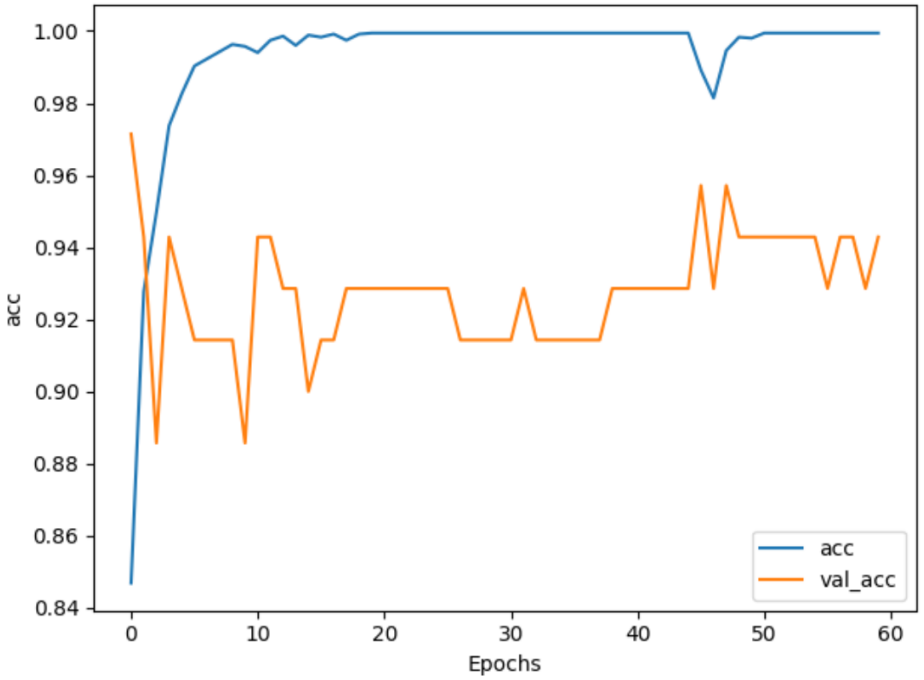

Рисунок 3.3 - Графік зміни втрат залежно від часу навчання першої моделі

Очевидно, що ці результати нас не влаштовували, тому було розроблено другу, трішки складнішу модель.

Друга модель складалася з 11 рівнів:

- шар вбудовування
- згортковий 1D шар
- субдескритизуючий шар
- згортковий 1D шар
- субдескритизуючий шар
- згортковий 1D шар
- max субдескритизуючий шар
- шар згладжування
- шар скидання
- щільний шар
- щільний шар

За допомогою вбудованих функцій keras можна отримати повний опис моделі, після того, як вона скомпільована. Повний опис другої моделі наведений у табл. 3.1.

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 30, 50)	400050
conv1d_1 (Conv1D)	(None, 30, 512)	77312
average_pooling1d_1 (AveragePooling1D)	(None, 10, 512)	0
conv1d_2 (Conv1D)	(None, 10, 256)	393472
average_pooling1d_2 (AveragePooling1D)	(None, 3, 256)	0
conv1d_3 (Conv1D)	(None, 3, 128)	98432
max_pooling1d_1 (MaxPooling1D)	(None, 1, 128)	0
flatten_1 (Flatten)	(None, 128)	0
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 128)	16512
dense_2 (Dense)	(None, 2)	258

Таблиця 3.1 Опис збудованої моделі

В другому варіанті моделі було введено 5 нових типів шарів.

Згортковий шар створює ядро згортки, яке згортається із введенням шару через один просторовий (або часовий) вимір, щоб отримати тензор вихідних даних.

Об’єднання шарів забезпечує підхід до зменшення вибірки карт об’єктів шляхом узагальнення присутності об’єктів у патчах карти об’єктів. Два поширені методи об’єднання - це середнє об’єднання та максимальне об’єднання, які узагальнюють середню присутність ознаки та найбільш активовану присутність ознаки відповідно.

Згладжування використовується для згладжування вхідних даних. Наприклад, якщо згладжування застосовується до шару, який має вхідну форму (розмірність) як (batch_size, 2,2), то вихідною формою (розмірністю) шару буде (batch_size, 4).

Рівень скидування випадково встановлює 0 з певною частотою на кожному кроці під час навчання, що допомагає запобігти перенавчанню. Важливо, що шар "Скидування" застосовується лише тоді, коли для навчання встановлено значення "True", так що під час реальної роботи модулі для аналізу та передбачення це скидування відбуватися не буде.

Останній щільний шар в цій моделі має уже не один нейрон, а два. Тобто на виході ми отримаємо не одне число з проміжку [0; 1], а два. Кожне відповідно буде позначати ймовірність, що цей текст має характеристики депресивно забарвленого тексту. Очевидно, що в сумі ці ймовірності будуть давати 1. Конкретно в цьому прикладі, зі штучно прибраними з датасету нейтральними маркерами, це великої ролі не грає. Але такий підхід до побудови моделі є дещо кращим, тому що дозволяє розширити список характеристик, наявність яких перевіряється у текстах.

Після побудови та навчання було отримано значно кращі результати. Потім після деяких змін внесених в розмір словника, максимальну довжину тексту, що тестується, кількості епох та розміру груп для навчання було отримано точність d2 = 0.9272 і наступні графіки: рис. 3.4 та рис. 3.5

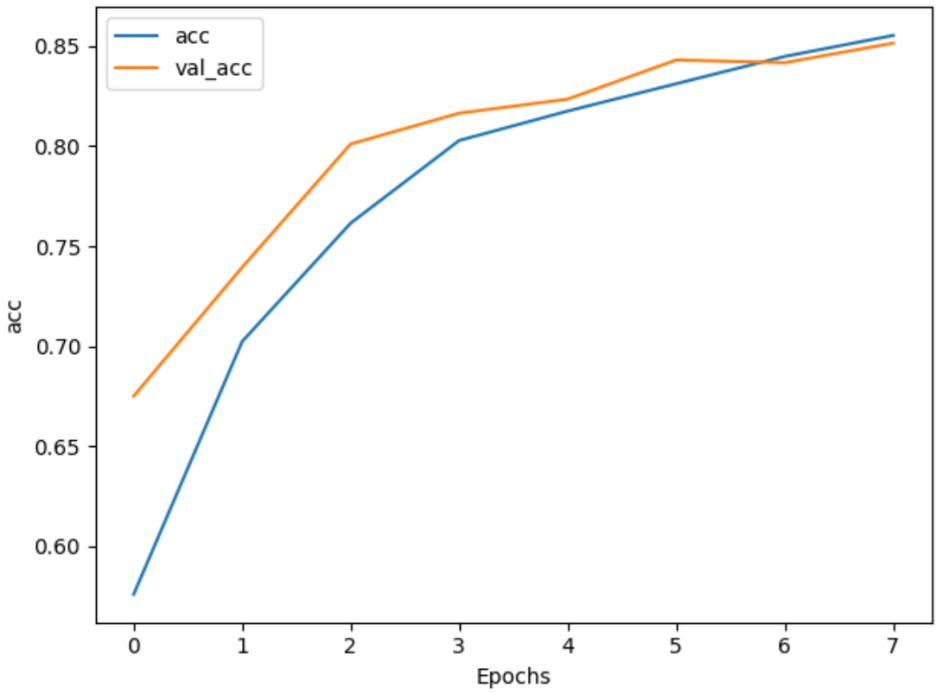

Рисунок 3.4 - Графік зміни точності залежно від часу навчання другої моделі

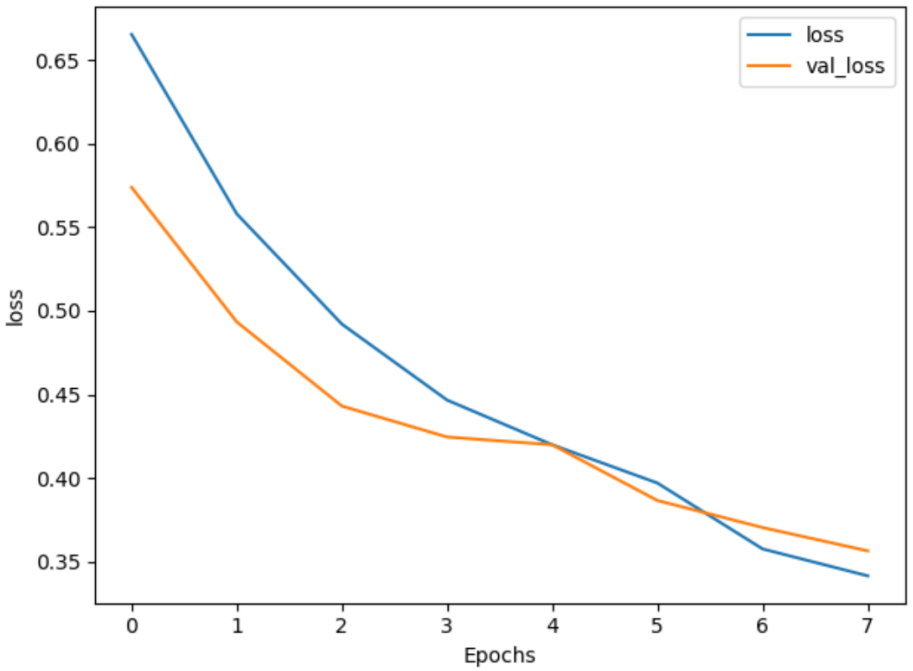

Рисунок 3.5 - Графік зміни втрат залежно від часу навчання другої моделі

Лістинг коду даної моделі можна знайти в додатку Б. Там представлені наступні функції:

- train - функція, в якій власне і подаються вхідні тренувальні дані на вхід моделі та проходить процес навчання та отримання ваг.
- make_model - функція, в якій відбувається визначення та компіляція моделі, тобто визначаються усі шари та їх параметри.
- make_embedding_layer - окрема функція для створення першого шару моделі, шару вбудовування.
- get_training_and_validation_sets - функція, яка використовує split_the_data та load_data_set для того, щоб підготувати тренувальний сет та сет для перевірки моделі.
- split_the_data - розділяє вхідні дані на дві частини. В останній версії, тренувальний сет був розміром 3550 входження, всі інші використовувалися для перевірки.
- load_data_set - вивантажує інформацію про твіти та забарвлення з файлів та приводить до формату (текст, забарвлення).
- tokenize_data - очищає та розбиває вхідні тексти на слова (токени) та вносить їх у словник.
- get_embeddings - вивантажує вбудовування слів. Було використано готові GloVe файли на 6 мільярдів слів з розмірністю векторів 50.
- plot_graphs - допоміжна функція для побудови графіків візуалізації навчання. Візуалізувалися графіки функції точності та функції втрат.
- auroc - допоміжна функція, яка використовувалася, як метрика при компіляції моделі. Для цієї метрики було використано roc_auc_score з бібліотеки sklearn.

Для цієї моделі ми також використали AUC як основну метрику. На рис.3.6 зображений графік зміни AUC метрики в залежності від часу.

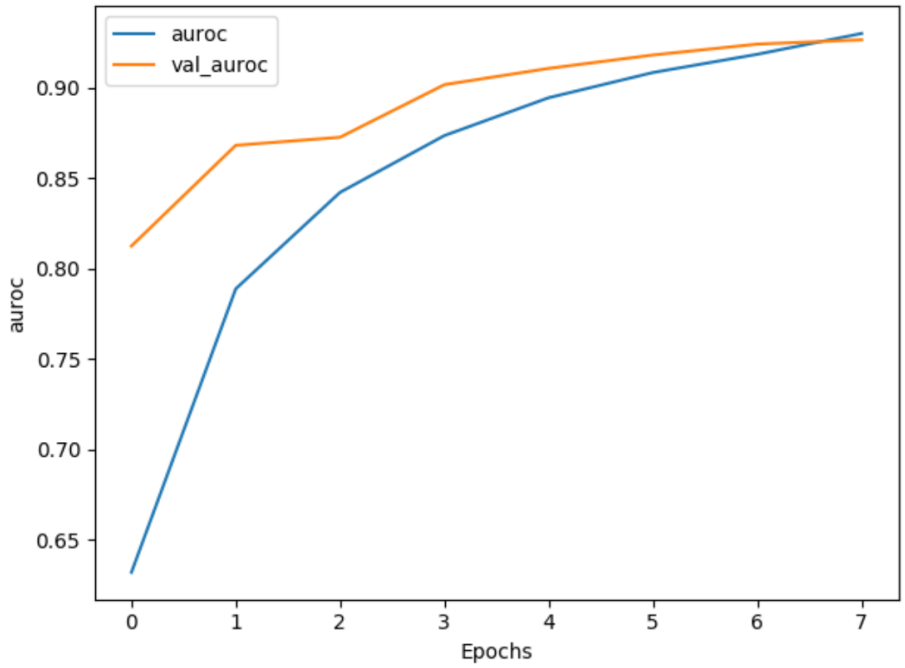

Рисунок 3.6 - Графік зміни AUC залежно від часу навчання другої моделі

Після тренування було перевірено модель на реченнях не з датасету. В табл. 3.2 наведено приклади результатів натренованої моделі.

Текст	Переклад	Ймовірність, що він депресивний	Ймовірність, що він не депресивний
Depression sucks. I want to die	Депресія відстій. Я хочу померти	0.7780547	0.22194528
Good vibes only. What a lovely nice day	Тільки хороші вібрації. Який прекрасний приємний день	0.00998493	0.99001515
we should educate more on the topic of depression	нам слід більше вивчати тему депресії	0.4093594	0.5906406

I want to buy some food	Я хочу купити трохи їжі	0.6432694	0.35673058
I feel really lonely today and I don't know what to do	Сьогодні я почуваюся дійсно самотньо і не знаю, що робити	0.8266745	0.17332545

Таблиця 3.2 – Приклади отриманих результатів

Як можна бачити в табл. 3.2, можуть виникати деякі сумніви з більш нейтральними реченнями. Оскільки, як зазначалося в розділі 1, симптомами великого депресивного розладу є не лише пригнічений настрій (як в прикладі “Депресія відстій. Я хочу померти”), але і втрата мотивацій, недосипання, переїдання, тощо. Тому речення як “Я хочу купити трохи їжі” важко визначити однозначно. Також модель погано справляється з реченнями зі словами, які не потрапили до словника, а зрозуміло, що таких слів є значна кількість.

ВИСНОВКИ

В результаті роботи було досліджено класифікацію та симптоматику афективних розладів та був обраний великий депресивний розлад для подальшого розгляду. Було досліджено специфіку цього розладу та розглянули задачі, максимально наближені до задачі діагностування великого депресивного розладу. Розглянули різні підходи для класифікації текстів та обрали два широких методи для побудови моделі: наївний класифікатор Баєса та глибинне навчання. Засобами мови Python та фреймворку keras та tensor flow було реалізовано, навчено та протестовано відповідні моделі.

Обидві моделі показали достатньо високу точність на тестових даних, а саме d1 = 0.9628078863891132 та d2 = 0.9272 відповідно. Хоча друга модель показала нижчу точність, вона є більш гнучкою для подальших досліджень.

Аналіз настрою та обробка природних мов дають змогу практикуючим психічного здоров’я мати можливість видобувати текстові дані з Інтернету та виявляти симптоми, які можуть бути передвісником різних проблем психічного здоров’я.

Важливо зауважити, що кількість даних для тренування була недостатньою. Тому результати можуть бути значно покращенні за використання більших об’ємів даних, а також даних кращої якості. Як на приклад, даних з сервісів для текстової комунікації для терапевтів. Оскільки для діагностування великого депресивного розладу недостатньо аналізу тексту в лише один момент часу, потрібно розуміти, що подібні моделі повинні застосовуватися протягом тривалого проміжку часу, від двох тижнів, для спостереження реальних симптомів.

Дані моделі можуть бути використані в додатках для спілкування з психотерапевтами для контролю депресивних настроїв. Таких додатків стає дедалі більше і більшість країн регулює на законодавчому рівні протоколи комунікації з пацієнтами які мають серйозні депресивні чи суїцидальні

настрої. Тому моніторинг, який може бути здійснений за допомогою таких моделей є важливою частиною подібних додатків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Depression / World Health Organisation/ 30.01.20, https://www.who.int/news-room/fact-sheets/detail/depression .
2. МКХ-10-АМ: Міжнародна статистична класифікація хвороб та споріднених проблем охорони здоров'я Десятий перегляд, Австралійська модифікація / 10-те видання , 1.08.2017.
3. Text classification and classifiers: a survey. International Journal of Artificial Intelligence & Applications / [Korde, V., & Mahender, N. C]. 2012., 3(2), 85-99.
4. Text classification techniques: a literature review / M. Thangara - Interdisciplinary Journal on Information, knowledge and management, 2018.
5. Bayesian Reasoning and Machine Learning 1st Edition / David Barber , 2012.
6. Structure extended multinomial Naive Bayes. / [Jiang, L., Wang, S., Li, C., & Zhang, L.], Information Sciences, 2016. 329, 346–356.
7. Hierarchical Deep Learning for Text Classification. Machine Learning and Applications (ICMLA) / [Kowsari, K.; Brown, D.E.; Heidarysafa, M.; Jafari Meimandi, K.; Gerber, M.S.; Barnes], 2017.
8. Distributed Representations of Words and Phrases and their Compositionality / Tomas Mikolov, Google Inc. Mountain View, 2013.
9. GloVe: Global Vectors for Word Representation / [Jeffrey Pennington, Richard Socher, Christopher D. Manning], Computer Science Department, Stanford University, Stanford.
10. Understanding Machine Learning: From Theory to Algorithms / [Shai Shalev-Shwartz, Shai Ben-David], 2014.
11. A review of machine learning algorithms for text documents classification. / [Khan, A., Baharudin, B., Lee, L. H., & Khan, K.] - Journal of Advances in Information Technology, 2010, 4-20.

12. Text classification based on deep belief network and softmax regression / [Jiang, M.; Liang, Y.; Feng, X.; Fan, X.; Pei, Z.; Xue, Y.; Guan, R.] - Neural Comput. Appl. 2018, 29, 61–70.
13. A comparison of event models for naive bayes text classification. / [McCallum, A.; Nigam, K.] , 1998; Volume 752,pp. 41–48.
14. Comparative study of five text classification algorithms with their improvements. International Journal of Applied Engineering Research / [Aliwy, A. H., & Ameer, E.], 2017., 12, 4309-4319.
15. A feature dependent Naive Bayes approach and its application to the software defect prediction problem. / [Arar, O. F., & Ayan, K.], 2017.
16. Using differential evolution for fine tuning Naive Bayesian classifiers and its application for text classification / [Diab, M., & El Hindi, K.], 2017.
17. Efficient Estimation of Word Representations in Vector Space / [Tomas Mikolov , Kai Chen, Greg Corrado, Jeffrey Dean].
18. Graphical Models, Lecture2: Bayesian Network Representation / McCallum, Andrew, 2019.

ДОДАТОК А
ПРИКЛАД ВХІДНИХ ДАНИХ

{"created_at":"Fri Jun 02 00:04:00 +0000 2017","id":870430762255953920,"id_str":"870430762255953920","text":"Hey, look - I found my social anxiety again. Was wondering where that went.","source":"\u003ca href=\"http:\/\/tapbots.com\/software\/tweetbot\/mac\" rel=\"nofollow\"\u003eTweetbot for Mac\u003c\/a\u003e","truncated":false,"in_reply_to_status_id":null,"in_reply_to_status_id_str":null,"in_reply_to_user_id":null,"in_reply_to_user_id_str":null,"in_reply_to_screen_name":null,"user":{"id":682583,"id_str":"682583","name":"Ryan Markel","screen_name":"ryanmarkel","location":"St. Louis, MO","url":"http:\/\/ryanmarkel.com","description":"aka Backlogathon - I think WordPress, OSS, the FGC, speedrunning, streaming, and compassion are super-rad - 2x CB garbage game medalist - biz: ryan@26mag.com","protected":false,"verified":false,"followers_count":1282,"friends_count":321,"listed_count":115,"favourites_count":3766,"statuses_count":25513,"created_at":"Mon Jan 22 21:26:57 +0000 2007","utc_offset":-18000,"time_zone":"Central Time (US & Canada)","geo_enabled":true,"lang":"en","contributors_enabled":false,"is_translator":false,"profile_background_color":"00012A","profile_background_image_url":"http:\/\/pbs.twimg.com\/profile_background_images\/212535735\/galaga-bg1.png","profile_background_image_url_https":"https:\/\/pbs.twimg.com\/profile_background_images\/212535735\/galaga-bg1.png","profile_background_tile":true,"profile_link_color":"396393","profile_sidebar_border_color":"8CAAD0","profile_sidebar_fill_color":"CEE5FF","profile_text_color":"092F60","profile_use_background_image":true,"profile_image_url":"http:\/\/pbs.twimg.com\/profile_images\/869990970728656899\/Odmn1VSf_normal.jpg","profile_image_url_https":"https:\/\/pbs.twimg.com\/profile_images\/869990970728656899\/Odmn1VSf_normal.jpg","profile_banner_url":"https:\/\/pbs.twimg.com\/profile_banners\/682583\/1464801377","default_profile":false,"default_profile_image":false,"following":null,"follow_request_sent":null,"notifications":null},"geo":null,"coordinates":null,"place":null,"contributors":null,"is_quote_status":false,"retweet_count":0,"favorite_count":0,"entities":{"hashtags":[],"urls":[],"user_mentions":[],"symbols":[]},"favorited":false,"retweeted":false,"filter_level":"low","lang":"en","timestamp_ms":"1496361840200"}

ДОДАТОК Б
ЛІСТИНГ КОДУ

```
labels_index = { 'Negative': 0, 'Positive': 1}

word_index, x_train, x_val, y_train, y_val = get_training_and_validation_sets()
model = make_model(labels_index, word_index)
train(model, x_train, x_val, y_train, y_val)

valid_predicted_out = model.predict(x=x_val, batch_size=256)
reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])
e = model.layers[0]
weights = e.get_weights()[0]
import io

out_v = io.open('vecs.tsv', 'w', encoding='utf-8')
out_m = io.open('meta.tsv', 'w', encoding='utf-8')
for word_num in range(len(word_index)):
    word = reverse_word_index[word_num]
    embeddings = weights[word_num]
    out_m.write(word + "\n")
    out_v.write('\t'.join([str(x) for x in embeddings]) + "\n")
out_v.close()
out_m.close()
sentence, y, w = tokenize_data(["Depression sucks. I want to die", "Good vibes only. What a lovely nice day", "we should educate more on the topic of depression", "I want to buy some food"], [1, 0])
print(sentence)
padded = pad_sequences(sentence, maxlen=MAX_SEQUENCE_LENGTH)
print(model.predict(x=padded))
evaluate(y_val, valid_predicted_out)
```

```
def get_training_and_validation_sets():
    X_raw, Y_raw = load_data_set()
    X_processed, Y_processed, word_index = tokenize_data(X_raw, Y_raw)
    x_train, x_val, y_train, y_val = split_the_data(X_processed, Y_processed)
    return word_index, x_train, x_val, y_train, y_val


def train(model, x_train, x_val, y_train, y_val):
    print(y_train)
    print("Train")
    cb = [ModelCheckpoint("weights.h5", save_best_only=True,
save_weights_only=False)]
    history = model.fit(x_train, y_train, validation_data=(x_val, y_val), nb_epoch=8,
batch_size=512, callbacks=cb)
    plot_graphs(history, "acc")
    plot_graphs(history, "loss")

    try:
        os.remove("model.h5")
    except OSError:
        pass
    model.save("model.h5")

def evaluate(expected_out, predicted_out):
    expected_categories = [np.argmax(x) for x in expected_out]
    predicted_categories = [np.argmax(x) for x in predicted_out]
    print(expected_categories, predicted_categories)
    cm = confusion_matrix(expected_categories, predicted_categories)
    print(cm)
```

```
    def auroc(y_true, y_pred):
        return tf.py_func(roc_auc_score, (y_true, y_pred), tf.double)

    def make_model(labels_index, word_index):
        embedded_sequences = make_embedding_layer(word_index)
        model = Sequential([
            embedded_sequences,
            Conv1D(512, 3, activation='relu', padding='same'),
            AveragePooling1D(3),
            Conv1D(256, 3, activation='relu', padding='same'),
            AveragePooling1D(3),
            Conv1D(128, 3, activation='relu', padding='same'),
            MaxPooling1D(3),
            Flatten(),
            Dropout(0.4),
            Dense(128, activation='relu'),
            Dense(len(labels_index), activation='softmax')
        ])

        model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc',
auroc])
        model.summary()
        return model

    def make_embedding_layer(word_index):
        embeddings = get_embeddings()
        nb_words = MAX_NB_WORDS+1
        embedding_matrix = np.zeros((nb_words, EMBEDDING_DIM))

        for word, i in word_index.items():
            if i >= MAX_NB_WORDS:
```

```
                continue
            embedding_vector = embeddings.get(word)
            if embedding_vector is not None:
                embedding_matrix[i] = embedding_vector

        embedding_layer = Embedding(nb_words, EMBEDDING_DIM,
weights=[embedding_matrix], input_length=MAX_SEQUENCE_LENGTH,
trainable=False)
        return embedding_layer

    def split_the_data(X_processed, Y_processed):
        indices = np.arange(X_processed.shape[0])
        prng.shuffle(indices)
        X_processed = X_processed[indices]
        Y_processed = Y_processed[indices]
        nb_validation_samples = int(VALIDATION_SPLIT * X_processed.shape[0])
        x_train = X_processed[:-nb_validation_samples]
        y_train = Y_processed[:-nb_validation_samples]
        x_val = X_processed[-nb_validation_samples:]
        y_val = Y_processed[-nb_validation_samples:]

        return x_train, x_val, y_train, y_val

    def tokenize_data(X_raw, Y_raw):
        tokenizer = Tokenizer(num_words=MAX_NB_WORDS)
        tokenizer.fit_on_texts(X_raw)
        sequences = tokenizer.texts_to_sequences(X_raw)
        word_index = tokenizer.word_index
        X_processed = pad_sequences(sequences,maxlen =
    MAX_SEQUENCE_LENGTH)
        Y_processed = to_categorical(np.asarray(Y_raw), 2)
```

```
    return X_processed, Y_processed, word_index

def load_data_set():
    tweets_data_path = 'tweetdata.txt'

    tweets_data = []
    tweets_file = open(tweets_data_path, "r")
    for line in tweets_file:
        try:
            tweet = json.loads(line)
            tweets_data.append(tweet)
        except:
            continue

    sent = pd.read_excel(
        'sentiment5.xlsx',
        engine='openpyxl',
    )
    print(sent.head())
    print(sent['id'])
    print(len(sent))
    x = []
    y = []
    for i in range(len(tweets_data)):
        if tweets_data[i]['id'] == sent['id'][i]:
            x.append(tweets_data[i]['text'])
            y.append(sent['sentiment'][i])
    return x, y


def get_embeddings():
```

```
    embeddings = {}
    with open(GLOVE_FILE, 'r') as f:
        for line in f:
            values = line.split()
            word = values[0]
            coefs = np.asarray(values[1:], dtype='float32')
            embeddings[word] = coefs
    return embeddings

def plot_graphs(history, string):
    plt.plot(history.history[string])
    plt.plot(history.history['val_' + string])
    plt.xlabel("Epochs")
    plt.ylabel(string)
    plt.legend([string, 'val_' + string])
    plt.show()
```