Міністерство освіти і науки України
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

КУЛИНЯК Роман Володимирович

«Алгоритми фільтрації текстових наборів даних соціальних мереж на основі штучних нейронних мереж / Algorithms of text sets filtration for social networks based on the artificial neural networks»

Студент групи КІм – 21
КУЛИНЯК Роман Володимирович
_____

Науковий керівник
к.т.н., ст. викладач, Ю.М. Батько
_____

Тернопіль – 2020

РЕЗЮМЕ

Кваліфікаціна робота на тему «Алгоритми фільтрації текстових наборів даних соціальних мереж на основі штучних нейронних мереж» зі спеціальності 123 «Комп'ютерна інженерія» освітнього ступеня магістр має обсяг 87 сторінок, та містить 24 рисунки, 2 додатки, та 50 джерел.

Метою випускної кваліфікаційної роботи є створення алгоритму для фільтрації текстових публікацій, які несуть загрозливий вміст використовуючи засоби машинного навчання.

Методи дослідження: методи та алгоритми машинного навчання.

Було досліджено підхід до синтезу та опрацювання текстових публікацій з визначенням коефіцієнтів «токсичності».

Було проаналізовано найпопулярніші алгоритми NLP, та обраний найбільш оптимальний для створення на його основі інформаційної моделі.

Було розроблено додаток, який фільтрує вхідний набір текстових даних відповідно до зазначеного порогового коефіцієнту.

В ході розробки та проектування було проаналізовано аналоги, з метою знаходження оптимального способу реалізації алгоритмів синтезу людської мови.

Можливими напрямками подальших досліджень є розробка ефективних підходів, з використанням машинного навчання, для кращого маркування текствових даних відповідним значенням коефіцієнту «токсичності».

КЛЮЧОВІ СЛОВА: ШТУЧНІ НЕЙРОННІ МЕРЕЖІ, МАШИННЕ НАВЧАННЯ, ОБРОБКА ПРИРОДНОЇ МОВИ, ВИЯВЛЕННЯ ЗАГРОЗЛИВОГО ВМІСТУ, ФІЛЬТРАЦІЯ ВЕБ ПУБЛІКАЦІЙ.

RESUME

Qualification work on "Algorithms for filtering text data sets of social networks based on artificial neural networks" with a specialty 123 "Computer Engineering" master's degree has 87 pages, containing 24 figures, 2 appendices, 50 sources.

The purpose of the final qualification work is to create an algorithm for filtering text publications that do not use threatening content using machine learning tools.

Research methods: methods and algorithms of machine learning.

The approach to the synthesis and processing of text publications with the definition of "toxicity" coefficients was studied.

The most popular NLP algorithms were analyzed and the best information model optimal for its creation was created.

An application has been developed that filters the input text set according to the specified threshold.

In the course of development and design, analogues were analyzed, thanks to the support of finding the optimal way to implement algorithms for the synthesis of human speech.

Possible areas of further research are the development of effective approaches using machine learning for better labeling of textile data with appropriate values of the coefficient of "toxicity".

KEY WORDS: ARTIFICIAL NEURAL NETWORKS, MACHINE LEARNING, NATURAL LANGUAGE PROCESSING, DETECTION OF THREATENING CONTENT, WEB PUB FILTRATION.

ЗМІСТ

ВСТУП

Актуальність теми. При розв’язуванні практичних завдань науки та техніки, виникає проблема знаходження найбільш ефективного підходу до обчислень. Деякі з наявних задач характеризуються відносно великою розмірністю даних, які подаються на вхід і тому потребують значних затрат при опрацюванні інформації. З іншої сторони, базліч задач, тісно пов’язаних, з аналізом наявних ринків, моніторингом роботи провідних промислових підприємств, банківських установ, характеризуються високою частотою надходження нової інформації про стан досліджуваного об’єкта та потребують додатково прийняття оптимальних рішень в режимі реального часу. Зростання розмірності вхідного набору даних та багаторазове використання одного і того ж самого фрагменту обчислень, очевидно призводять до збільшення складності розв’язання подібного роду задач. Тому необхідно вдосконалювати вже існуючі рішення та розробляти нові підходи до організації обчислень на відповідних системах високої продуктивності.

Ще в 40х роках минулого століття визначними стали досягнення нейробіології, які дозволили створити першу штучну нейронну мережу імітуючи цим самим роботу людського мозку. Але тільки через декілька десятиліть, разом з виникненням та розвитком електро-обчислювальних систем (комп’ютерів) і відповідного програмного забезпечення, стала можлива розробка складних додатків в області штучних нейронних мереж. З того моменту теорія нейронних мереж була визнаною один із найбільш перспективних напрямів наукових досліджень. Цьому сприяла сама природа підходу паралельних обчислень і можливість виконання адаптивного навчання штучних нейронних мереж.

Однак, незважаючи на значний розвиток та впровадження паралельних обчислень у практичній науковій діяльності, використання паралельних методів навчання нейронних мереж є відносно новою та мало дослідженою задачею.

Загальне захоплення тематики нейронних мереж, технологіями і глибинним навчанням не оминуло і комп'ютерну лінгвістику - автоматичну обробку текстів на природній (людській) мові комунікації. Підвищена увага лінгвістів до нейронних мереж зумовлена рядом причин.

Застосування штучних нейронних мереж, по-перше, істотно підвищує якість вирішення багатьох тривіальних задач класифікації текстів та послідовностей, по-друге, зменшує трудомісткість при роботі безпосередньо з текстовими даними, по-третє, дозволяє вирішувати завачі фільтрації та блокування загрозливого контенту.

Актуальність використання технології штучних нейронних мереж зумовлена розвитком апаратного забезпечення. Сучасні процесори мають велику потужність та здатні опрацьовувати чималий набір даних. Таким чином найкращий підхід до впровадження фільтрації включає в себе використання засобів Natural Language Processing та моделі машинного навчання.

Мета та завдання дослідження полягає у реалізації паралельного навчання нейронної мережі для визначення коефіцієнту «токсичності» веб публікації та подальша фільтрація відповідно до порогового допустимого значення.

Об’єктом дослідження представлений аналіз текстових публікацій та коментарів на веб-ресурсах.

Предметом досвідження представлений алгоритм фільтрації текстових наборів даних використовуючи засоби Natural Language Processing.

Методами дослідження євляються технологія штучних нейронних мереж та машинного навчання.

Наукова новизна одержаних результатів полягає у впроваджені моделі штучної нейронної мережі здатної проводити ефективний аналіз тексту на визначення смислової нагрузки. Даний підхід представляє сучасне рішення до синтезу та аналізу веб публікацій.

Практичне значення отриманих результатів полягає у викорстанні алгоритму для фільтрації вхідного текстового повідомлення на веб ресурсі.

Дана робота публікувалася на ІІІ Науково-практичній конференції молодих вчених і студентів «Інтелектуальні комп’ютерні системи та мережі» [1,2].

Дана робота буде впроваджена компанією EPAM Systems для використання фільтрації текстових коментарів в межах проведення всеукраїнського AI бот челендж турніру.

У першому розділі магістерської роботи проаналізовано штучні нейронні мережі, класифікація та парадигми навчання, використання семантичних нейронних мереж. Та описано існуючі підходи визначення тональності тексту з використанням лексикону та машинного навчання.

У другому розділі магістерської роботи описано способи реалізації адаптивного рішення задачі класифікації з використанням нейроннох мережі Кохонена та алгоритму класифікації Наївного Байєса.

У третьому розділі магістерської роботи описано інструменти для розробки, дизайн та імплементації, тестування системи визначення тональності (токсичності) текстових повідомлень.

1 ШТУЧНІ НЕЙРОННІ МЕРЕЖІ. АЛГОРИТМИ ТА МЕТОДИ НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ.

1.1 Поняття та класифікація штучних нейронних мереж.

Відразу після свого народження людина не має жодної інформації про навколишній світ. Але вон здатна отримати її за допомогою своїх органів почуттів, таких як зорова система або слуховий апарат. Отримуючи сигнали про світ, людина реагує на них певним чином. Її реакція здатна викликати зміни, які посилают сигнали, що в свою чергу, може викликати іншу реакцію. Деякі сигнали можуть бути однозначно розпізнані як негативні, наприклад біль, інші ж будуть сприйматися як позитивні – приємний запах або почуття ситості [3].

Володіючи від природи здатністю до зіставлення своїх дій і подальшого результату, людина виробляє алгоритми поведінки для мінімізації негативних сигналів і оптимізації позитивних. Подібна здатність до самонавчання є вкрай цікавою, так як її розуміння дозволить вирішити безліч завдань, наприклад, створювати алгоритми обробки інформації, які автоматично підлаштовуються під вхідні дані з метою поліпшення результату. Людина має таку змогу завдяки своєму мозку – біологічній нейронній мережі величезних масштабів.

Інтерес представляє не тільки здатність до самонавчання, а й те, які алгоритми створені людським мозком для вирішення ряду щоденних завдань, такі як розпізнавання образів (людина здатна з високою точністю відрізняти людей один від одного або прийняття рішень, особливо в умовах нестачі інформації). Для таких задач існують різні математичні методи їх вирішення, але часто вони програють людині в швидкості роботи і точності [4]. Отже, можна припустити, що досить точна модель людського мозку покаже подібну ефективність в таких завданнях. Також, дослідження моделі, яка успішно справляється з наведеним класом задач,

дозволить більше зрозуміти як про саму проблематику завдання, і її способи вирішення, так і про алгоритми людського мислення.

Штучна нейронна мережа, як випливає з назви, являє собою спробу побудувати таку модель. Поняття нейронних мереж включає в себе широкий клас об'єктів, відмінності між якими можуть бути досить великі, але всі вони будуються на основі наступних принципів:

1. інформація в нейронній мережі обробляється за допомогою безлічі однотипних елементів. Такий елемент називається "нейроном";

2. кожен нейрон володіє входом, виходом, і деяку функцією, перетворюючу вхід у вихід. Ця функція називається функцією активації нейрона;

3. вихід будь-якого нейрона може бути пов'язаний з входом одного або декількох нейронів. Такий зв'язок називається "синаптичний".

Синаптическая зв'язок при передачі сигналу деяким чином перетворює його. Зазвичай кожний вид синаптичного зв'язку у відповідність проставляється деяким число, і при передачі сигналу він просто перемножується з цим числом [4].

В разі якщо до входу одного нейрона приєднано кілька синаптичних зв'язків, то приходячі по ним сигнали певним чином перетворюються перед подачею в нейрон. Зазвичай вони просто підсумовуються.

Штучні нейронні мережі, також як і біологічні, мають властивість відмовостійкості, яка складається з двох частин. По-перше, нейронна мережа здатна до обробки і розпізнавання сигналів, що відрізняються від тих, що раніше в неї надходили. Наприклад, коли людина бачить нову для себе модель автомобіля, вона розуміє, що це саме автомобіль, а не щось інше. Такий висновок ми здатні зробити завдяки тому, що новий автомобіль схожий, хоч і не повністю, на ті, що ми уже бачили. По-друге, якщо з деякої причини відбувається пошкодження нейронної мережі, то вона часто виявляється здатною продовжити свою роботу, нехай і часто з погіршенням якості результату.

Нейронні мережі знайшли застосування в багатьох галузях, таких як:

– авіаційно-космічна галузь: автопілот, тренажери для пілотів, детектори несправностей для різних підсистем літака;

– автомобільні системи автоматичного управління;

– банківська сфера: системи для оцінки кредитоспроможності клієнтів, для автоматичного визначення використання крадених карт за особливостями скоєних транзакцій;

– армія: системи виявлення, ведення мети, розпізнавання цілі по радарних відгуку;

– електроніка: нелінійне моделювання, комп'ютерний зір, розпізнавання і синтезування мови.

1.1.1 Структура нейрона

Мозок людини складається з мільйонів базових елементів - нейронів. З точки зору моделювання роботи мозку, кожен нейрон можна розділити на три частини: дендрит, аксон, і тіло нейрона. Дендрити представляють собою довгі мережі, завдання яких полягає в поширенні електричних сигналів до тіла нейрона [5]. Тіло, в свою чергу, якимось чином обробляє отриманий сигнал і видає відповідь за допомогою аксона, який поширює його вздовж себе. Нейрони працюють не незалежно один від одного, дендрити кожного нейрона з'єднані з аксонами інших нейронів. Таким чином, в біологічному мозку відбувається поширення сигналів від одного нейрона до інших. На рисунку 1 зображено схематичне представлення нейрона.

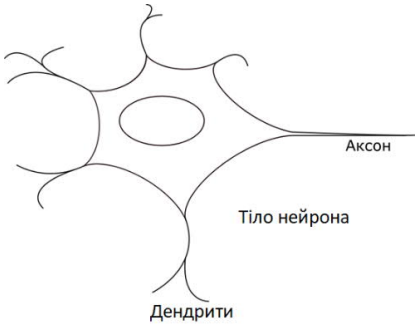

Рисунок 1.1 – Схема біологічного нейрона

Частина структури головного мозку закладається при народженні і зберігається в незмінному вигляді протягом усього життя людини. Інші частини мозку здатні змінюватися з часом. В ході подібних змін відбувається зміна структури зв'язків між нейронами. Нові зв'язки з'являються, а старі зникають. Якісь зв'язку посилюються, в результаті чого сигнал від одного нейрона до іншого доходить майже в незмінному вигляді [6]. Інші зв'язку можуть ослабнути, і сигнал буде доходити погано. Найбільш активно процес формування і зміни зв'язків між нейронами відбувається на початку життя.

При створенні штучної нейронної мережі відбувається моделювання двох особливостей справжньою нейронної мережі. В основі мережі лежить використання простих базових компонентів – нейронів. Нейрони з'єднуються один з одним, і структура цих сполук визначається завданнями, покладеними на мережу [6].

Незважаючи на те, що швидкість роботи біологічних нейронів і швидкість поширення електричних сигналів по нервових волокнах між нейронами на кілька порядків менше швидкості, з якою можуть функціонувати електронні схеми, людський мозок здатний справлятися з об'ємними і обчислювально складними завданнями за вкрай короткий проміжок часу. Подібна продуктивність можлива за рахунок того, що структура мозку являє собою паралельну систему. У такій системі всі нейрони працюють одночасно. Для того, щоб деякий нейрон був здатний зробити обчислення і визначити свій стан в наступний момент часу, йому необхідна інформація про поточний стан тільки тих нейронів, які розташовані поруч з ним і до яких він приєднаний за допомогою дендритів.

Штучні нейронні мережі також мають здатність до паралельної обробки сигналів. І хоча для дослідження і конструювання нейронної мережі зазвичай використовуються програми, що працюють на звичайних комп'ютерах з послідовною системою виконання команд, після завершення проектування мережі її можна реалізувати фізично, отримавши переваги паралельної обробки [7].

Розглянемо структуру окремого нейрона – базового компонента нейронної мережі, яка представлена на рисунку 1.2.

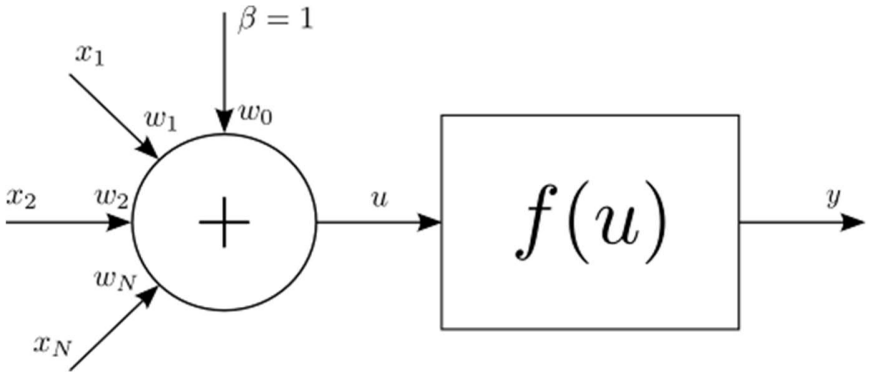

Рисунок 1.2 – Структура нейрона

Символами позначені вхідні значення нейронів. Це можуть бути як вихідні значення інших нейронів, так і вхідні дані - вагові коефіцієнти синаптичних зв'язків. Кожне вхідне значення множиться на відповідний коефіцієнт. Потім, як випливає з схеми, перемножені значення підсумовуються. До отриманого результату додається зсув β. На малюнку, для спрощення структури нейрона, представляється як синаптический зв'язок з ваговим коефіцієнтом w_0, поєднаний з виходом "віртуального" нейрона, вхід якого ні з чим не пов'язаний, а вихідне значення завжди дорівнює 1. Отримана сума подається на вхід функції активації $f(u)$, результат роботи якої і буде вихідним значенням нейрона.

Зсув є важливим поняттям нейронної мережі. Біологічний нейрон реагує на подразник, тільки якщо величина впливу перевищує деякий певний поріг. Для імітації такої поведінки в штучну нейронну мережу і введено поняття зсуву [8]. Для прикладу, припустимо, що функцією активації нейрона є функція Хевісайда, яка визначається наступним чином:

$$f(u) = \begin{cases} 1, & u \geq 0 \\ 0, & u < 0 \end{cases} \quad (1.1)$$

Розглянемо вхідний параметр u:

$$u = \sum_{i=1}^{N} x_i * w_i + x_0 * w_0 \tag{1.2}$$

Причому $x_0 = 1$. Тоді при заданих вагових коефіцієнтах і вхідних значеннях, вихід нейрона можна регулювати зміною величини зсуву, а саме коефіцієнта w_0.

Як правило, конкретна функція активації вибирається дослідником в залежності від особливостей досліджуваної проблеми, а значення ваг підбираються в ході налаштування мережі під конкретну задачу. Подібна настройка зветься процесом навчання нейронної мережі [9].

1.1.2 Структура шару нейронної мережі

Для вирішення багатьох завдань одного нейрона недостатньо. У таких випадках може знадобитися використання групи нейронів, що працюють паралельно. Подібна група нейронів називається шаром нейронної мережі. Приклад мережі з одного шару наведено на рисунку 1.3.

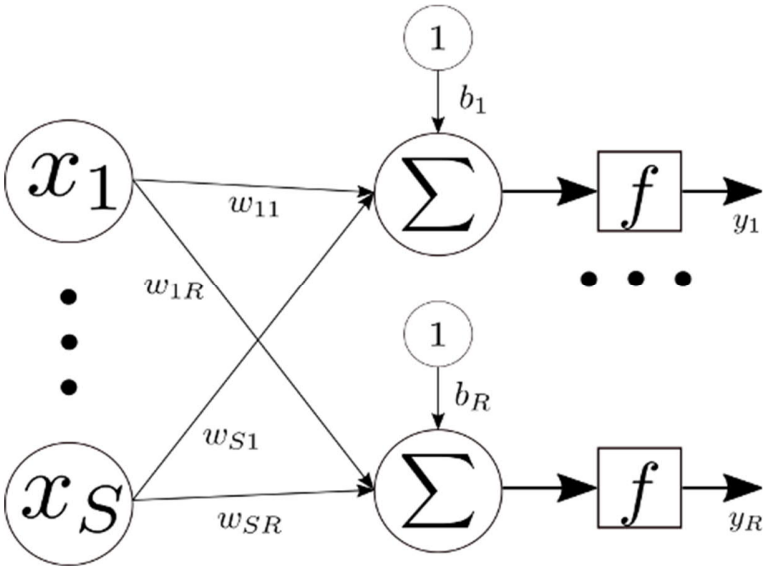

Рисунок 1.3 – Одношарова мережа

В даній мережі міститься R нейронів з однаковою активаційною функцією. В якості входу в мережу виступає вектор розмірності S, виходом є вектор розмірності R. Кожен вхід мережі пов'язаний з кожним нейроном за допомогою синаптичного зв'язку з вагою w_{ij}, де i - номер входу, j - номер нейрона. Також у кожного нейрона є зміщення, представлене у вигляді нейрона з виходом, рівним 1 та відповідною вагою зв'язку, рівною b_i.

У разі, якщо необхідно задати шар мережі, в якому частина нейронів має одну активаційну функцію, а частина - іншу, можна взяти дві мережі, подібних до тієї, що зображена на рисунку 3 та скомбінувати їх в одну. Тоді обидві мережі матимуть одні і ті ж входи, і кожна мережа буде генерувати частину вихідного вектора [10].

1.2 Підходи до навчання штучної нейронної мережі.

Важливою властивістю нейронних мереж є їх здатність до навчання. Грунтуючись на даних, отриманих з навколишнього середовища, нейронна мережа здатна змінити себе з метою підвищення своєї ефективності.

В контексті штучних нейронних мереж, під навчанням можна розуміти як настройку вагових коефіцієнтів мережі, так і зміну її структури з метою підвищення ефективності поставленого завдання [11]. Можливість мережі самій шукати зв'язок між наданими вхідними даними і вихідними цілями робить їх більш привабливими в порівнянні з системами, в яких правила вирішення завдання задаються вручну.

Навчання мережі є ітеративним процесом. В ході однієї ітерації мережі виконуються наступні пункти:

– отримання інформації із зовнішнього середовища за допомогою стимулів;
– зміна параметрів або структури мережі;

– реакція мережі на зовнішні стимули тепер відрізняється від того, що було раніше.

Існує безліч різних алгоритмів навчання нейронних мереж, і кожен алгоритм вимагає виконання певних умов щодо структури нейронної мережі. Так само, в залежності від досліджуваної проблеми, одні алгоритми навчання можуть показати кращий результат, ніж інші.

Різні алгоритми навчання можна умовно розділити на дві групи: навчання з учителем і навчання без учителя [12].

У разі навчання з учителем мається на увазі наявність певної відомої інформації про навколишнє середовище, а саме наявність деякої вибірки, що складається з пар виду, де x – це одиничний набір вхідних даних, а y – інформація, яка описувала, як мережа повинна реагувати на введені дані.

В одному випадку, являє собою опис очікуваного виходу нейронної мережі. Тоді пара (x, y) називається прикладом. Багато алгоритмів навчання в цьому випадку використовують деяку оцінку різниці між необхідним і отриманим відповідями нейронної мережі. Ця оцінка використовується для визначення того, наскільки сильно необхідно змінити структуру мережі [13]. Часто намагаються досягти того, щоб за один процес коригування ваг для оброблюваного в даний момент прикладу зменшити величину помилку до нуля або близько до нього.

В іншому випадку, досліднику для деякого набору вхідних даних відома лише деяка процедура, що повідомляє, чи є відповідь нейронної мережі правильною чи ні. Головною відмінністю є те, що неможливо оцінити, наскільки далеко від правильної відповіді знаходиться надана відповідь нейронної мережі.

Є можливість, що в результаті навчання, мережа просто пристосується до ввідних прикладів, і, отримавши на вхід приклад, який не використовувався в ході навчання, видасть неадекватний результат. Щоб уникнути цього, для самого навчання використовується тільки частина всіх доступних прикладів.

Невикористана частина прикладів застосовується для оцінки якості отриманої нейронної мережі [14].

У разі якщо набору прикладів немає, а відомі тільки вхідні дані, можна використовувати навчання без учителя. В даному випадку, нейронна мережа повинна видавати для кожного прикладу деяке "оптимальне значення", величина якого визначається конкретним алгоритмом навчання. Як приклад можна розглянути задачу кластеризації точок на площині, яка показана на рисунку 1.4.

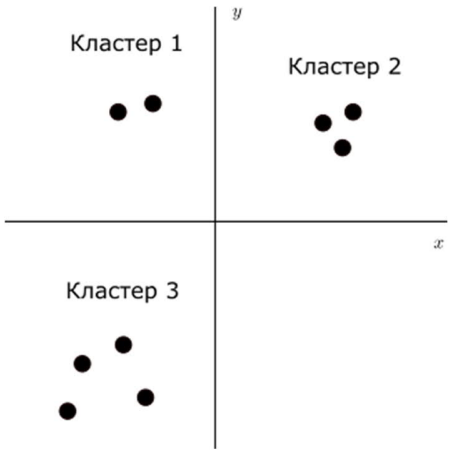

Рисунок 1.4 – Задача кластеризації

Для вирішення даного завдання, необхідно навчити мережу таким чином, щоб вона видавала однакову відповідь для точок, що входять в один і той же кластер. Спочатку, досліднику недоступна інформація про те, яка точка якому кластеру відповідає. У загальному випадку кількість кластерів також невідома. Отже, в процесі навчання, мережа повинна сама якимось чином модифікувати свою структуру для вирішення поставленого завдання [15]. Зазвичай намагаються досягти того, щоб на набори вхідних даних, розташованих досить близько один до одного, мережа видавала однакову відповідь.

В ході навчання, з учителем або без, досліднику необхідно визначити, яким чином він буде модифікувати значення ваг синаптичних зв'язків. До вирішення цього питання є два основних походу: детермінований і стохастичний.

У детермінованому підході величини, на які будуть змінені ваги, визначаються за допомогою детермінованого алгоритму. Зазвичай це алгоритм, заснований на аналізі поточної структури мережі і її параметрів, і на різниці між очікуваною реакцією мережі на якісь дані і реальною [16].

При використанні стохастичного підходу коригування ваг проводиться випадковим чином. Тобто, дослідник пропускає через мережу деякий набір вхідних даних і порівнює вихід мережі з бажаним. Якщо різниця дуже велика, то ваги зв'язків змінюються випадковим чином, потім цей же набір даних знову пропускається через мережу. Якщо величина помилки зменшилася, то нові значення ваг зберігаються, інакше відкидаються.

В ході навчання мережі з учителем може виникнути проблема перенавчання. Перенавчена мережа буде здатна давати точну відповідь при введенні в неї прикладів, що використовувалися при навчанні [17]. Однією з ознак перенавчання мережі є занадто великі вагові коефіцієнти. Для того щоб уникнути цього, можна розділити набір прикладів для тренування на дві частини, і тренувати мережу тільки на одній частині. Іншу частину використовувати при цьому для перевірки ступеня якості роботи мережі, і зупиняти навчання в разі зростання величини помилки.

1.2.1 Алгоритм навчання Хеба

Правило Хеба звучить наступним чином: "Якщо аксон в клітинці А близький до передачі активізації клітинки Б, і постійно і неодноразово бере участь в процесі активізації клітини Б, то в одній або обох клітинах відбувається певний процес зростання або метаболічних змін, в результаті чого ефективність клітини А в процесі активізації клітини Б підвищується".

Даний алгоритм може бути використаний в нейронних мережах з різною конфігурацією. Правило Хеба каже, що якщо два нейрона, пов'язаних синаптичної зв'язком, активуються одночасно, то зв'язок між ними посилюється [18].

$$w_{ij}^{new} = w_{ij}^{old} + l * f(x_i) * g(y_i) \quad (1.3)$$

де, відповідно, $w_{ij}^{new}, w_{ij}^{old}$ – нове та старе значення ваги при зв'язку, що з'єднує і-й елемент вхідного вектора і j-й нейрон, l – деяка константа, що задає швидкість зміни ваг (або, іншими словами, швидкість навчання), $f(x_i)$ деяка функція від і-го значення вхідного вектора, $g(y_j)$ – деяка функція від вихідного значення j-го вектора. Можна помітити, що у формулі ніде немає параметр цільового вектора. Це дозволяє використовувати цей метод для навчання без учителя, коли дані лише вхідні вектора, а цільові невідомі. Функції, а також константа підбираються самим дослідником.

Можна помітити, що у формулі ніде немає параметра цільового вектора. Це дозволяє використовувати цей метод для навчання без учителя, коли дані лише вхідного вектора, а цільові невідомі. Функції f та g, а також константа l підбираються самим дослідником [19].

Модифікуємо метод Хеба для випадку наявності пар, що складаються з вхідного вектора і бажаного результату. Розглянемо одношарову мережу прямого поширення з функцією активації першого шару $F(\bar{a}) = \bar{a}$. Нехай вектор $\bar{x}$ описує вхід мережі, $\bar{y}$ описує реальний вихід мережі, бажаний вихід описує вектор $\bar{t}$, а W – матриця вагових коефіцієнтів. Вихід одного нейрона буде описуватися формулою.

$$F(\bar{x} * W) = \sum_{i \in N} x_i * w_{ij} = y_j \tag{1.4}$$

де N – розмірність вектора $\bar{x}$, j-індекс нейрона. Якщо відомі цільові вектора, то можна модифікувати алгоритм Хеба. Приймемо $l = 1, f(a) = a, g(a) = a$. Тоді формула для обчислення зміни ваг прийме наступний вигляд.

$$w_{ij}^{new} = w_{ij}^{old} + x_i * t_j \tag{1.5}$$

Якщо спочатку значення матриці ваг дорівнюють нулю, то після навчання на n прикладах ми отримаємо такі ваги:

$$w_{ij}^{new} = x_i^1 * t_i^1 + x_i^2 * t_j^2 + \cdots + x_i^n * t_j^n = \sum_{k=1}^{n} x_i^k * t_j^k \quad (1.6)$$

Припустимо, що вхідні вектори ортонормірованs, тобто:

$$\|\bar{x}\| = 1 \quad (1.7)$$

$$\bar{x}_i^T * \bar{x}_j = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases} \quad (1.8)$$

Проведемо навчання мережі і подамо на вхід один з учасників в навчанні вхідних векторів $\bar{x}^q$, q-номер прикладу. Отримаємо наступну формулу:

$$y_j^q = \sum_{i \in N} x_i^q * w_{ij} = \sum_{i \in N} x_i^q * \sum_{k=1}^{n} x_i^k * t_j^k = \sum_{k=1}^{n} \sum_{i \in N} x_i^q * x_i^k * t_j^k \quad (1.9)$$

Так як x_i^q та x_i^k це елементи ортогональних векторів $\bar{x}^q$ та $\bar{x}^k$, то:

$$\sum_i x_i^q * x_i^k = \begin{cases} 1, k = q \\ 0, k \neq q \end{cases} \quad (1.10)$$

Тоді в якості вихідного значення отримаємо:

$$y_i^q = \sum_{k=1}^{n} \sum_i x_i^q * x_i^k * t_j^k = t_j^q \quad (1.11)$$

Тобто вихід мережі в точності дорівнює цільового значення. Розглянемо випадок, коли вхідні вектора не є ортогональними, а тільки нормованими. Такий випадок буде описувати загальний підхід обчислення [20]. Тоді результат роботи мережі буде виглядати наступним чином:

$$y_i^q = \sum_{k=1}^{n} \sum_{i \in N} x_i^q * x_i^k * t_j^k = t_j^q + \sum_{k \neq q} \sum_{i \in N} x_i^q * x_i^k * t_j^k \quad (1.12)$$

Тобто мережа буде видавати результат з помилкою рівній останньому доданку.

1.3 Аналіз існуючих підходів визначення тональності тексту з використанням лексикону та машинного навчання.

Існують різні підходи до визначення тональності тексту. Одним з перших та найбільш ефективних – є метод, оснований на лексиконі. Основна ідея плягає у використанні лексикона, який містить умови з відповідними оцінками почуттів для кожного з термінів. Сутність терміна може бути пов’язана з конкретним словом, фразою або ж ідіомом. Сутність почуттів визначаються виходячи з наявності або відсутності термінів у лексиконі [21]. Підхід, що заснований на лексиконі, включає також в себе підхід, що базується на корпусі, та словниковий підхід.

1.3.1 Словниковий підхід

Основна ідея закладена у словниковий підход полягає у використанні лексичної бази даних думок, для витягнення почуттів з віхдного тексту. Множина маркованиз сентиментальних слів (добре, погано, наприклад) з їх полярністю збирається вручну. Наступним кроком є використання полярності слів, щоб

збільшити опрацьовуваний набір, шукаючи відповідні синоніми на антоніми в лексичній базі даних. Прикладами таких баз даних є WordTwo, TheSentence, Kapra, FCDK та інші.

Алгоритм виконання пошуку є ітеративним. Кожна ітерації приймає оновлений набір слів та робить відповідний пошук знову, доки не буде знайдено нових слів для включення. Зрештою, сукупність сентиментальних слів може бути переглянута з метою видалення помилок. Розробки подібних рішень зосереджені на дослідженні класифікації відгуків користувачів. Вони запрошували характеристику продукту, яка містить набір почуттів, щоб опісля класифікувати пропозиції беручи за основу дані особливості [22]. Результатом опрацювання слугує резюме оглядів продукту користувачами.

Наприклад, якщо такий огляд відноситься до камери, даний підхід дозволив знайти такі особливості, як якість зображення та розмір камери, класифікаціючи позитивні та негативні коментарі. Для призначення позитивного чи негативного тегу для пропозиції, спочатку дослідники запросили полярні слова для кожного з оглядів. В такому випадку, використовувалися прикметники. Даний прогноз базувався на полярності прикметника, котрий містив таку ж саму полярність, як і синоніми та протилежність полярності його антонімів. Полярні слова використовуються для пошуку синонімів та антонімів з відомими орієнтаціями в WordTwo. Це дозволило виявити полярність слів, що з’являються в огляді. Описаний метод показав досить хороші результати, середня точність яких становила 79%. Отже, такий метод можна визначити ефективним для прогнозування семантичних орієнтирів та визначення полярності речення [22].

Інші рішення проводили дослідження полярності тексту та його власника відносно даної теми. Автори дослідження застосовують набір класифікаторів. Перший з яких був застосований для кожного слова в реченні, щоб отримати його полярність. Другий класифікатор дозволяє визначити полярність всього речення, поданого власником думки. Окрім цього, автори запропонували використовувати

малий початковий список вже розмічених слів аналогічним чином (дієслова та іменники). Даинй список слів був розширений, шукаючи відповідні синоніми та антоніми в TheSentence. Автори наголошують, що деякі синоніми/антоніми містять нейтральну або навіть протилежну спрямованість, що робить їх недоступними для використання. Окрім цього, дослідники роблять акцент на необхідності визначення сили позитивного та негативного забарвлення слів, що дозволяє позбутися неоднозначності у словах [23].

Автори дають визначення чотирьох різних регіонів у реченні, які за своєю структурою є спорідненими з власником думки та можуть містити настрій речення. Щоб визначити сентиментальність думки автори запропонували три моделі. Перша приймає припущення, щодо зустрічі одного слова негативного характеру (інші негативні слова не мають впливу). Друга та третя представляють гармонійне та геометричне значення сильних почуттів у конкретній позиції. Проведені експеременти, довзолили отримати найкраші результати за допомогою першої моделі та регіону, який простягається від власника думки до кінця речення.

У роботі автори представили метод, що використовує три різни словника (як правило використовують один) для отримання синонімів та антонімів. Розгорнутий лексикон використовується для класифікації публікацій в мережі твітер. Автори сказали, що запропонована ними методика дозволяє класифіковувати твіти, на що традиційний словниковий метод був нездатний. Попри все, запропонований підхід містить незначні недоліки. Основна проблема полягає у тому, що збір великої кількості синонімів та антонімів вимагає затрати часу [24]. Також, зазвичай словники містять формальні слова, але твіти багаті на неформальну лексику. В основному, головним недоліком словникового підходу визначають нездатність виявляти сентиментальні слова, пов’язані з якоюсь специфічною тематикою.

1.3.2 Корпусний підхід

Корпусний підхід може бути застосований лише у двох випадках. Перший – це ідентифікація оціночних слів та їх популярності в корпусі, з використанням

заданого набору слів. Другий підхід являє собою спосіб побудови нового лексикону в межах заданого домену з іншої лексики, з використанням корпус цього домену. Результати аналізу говорять, що слова які виражають думки, залежать від домену, може зустрітися одне і те ж слово, з протилежною орієнтацію, залежно від наданого на вхід контексту [25].

Дослідження, проведенні Hazivassiloglou та McKeown, відомі в літературі. Автори пропонують метод, який визначає семантичну спрямованість з'єднаних прикметників із корпусу. Дана методика використовує текстуальні слова та прикметники, які виражають конкретні думки. Набір спеціальних лінгвістичних правил застосовується до тексту, для виявлення важливості слів з відповідними орієнтаціями. Автори зазначають, що прикметники мають однакову полярність, якщо вони сполучаються словом «and». В той час як, слово «but» сполучає слова з протилежними семантичними орієнтаціями. Також в аналіз беруться словосполучники «or», «either-or», «neither-nor». Часом ці правила не застосовуються. Тому автори передбачають перевірку полярності об'єднаних одинакових прикметників, з використанням моделі логістичної регресії.

Наступним кроком прогнозування є отримання моделі графа, яка забезпечує зв'язок між спорідненими прикметниками [26]. В такому випадку, кластеризація виконується на графіку, щоб отримати розподіл прикметників на позитивні та негативні підмножини. Такий підхід дозволив досягти точності в 85%. Як згадувалося раніше, одне і те ж сентиментальне слово може часом мати зовсім різну семантичну спрямованість залежно від наданого контексту. Ding та співавтори пропонують метод пошуку орієнтації настроїв, переданих на опрацювання. Автори підкреслюють, що певні прикметники залежать від контексту і можуть мати змінену полярність. Дослідники визначають зміст слів та їх аспектів у реченні, щоб позначити полярність ознаки продукту.

Набір прикметників, прислівників, іменників та дієслів постійно доповнюється автор для роботи з більшим набором даних до опрацювання. Вони

здійснили анотацію більше 1000 ідіом, що містять чітко виражені настрої. Після підготовки лексикона, можна розпочати процес визначення оцінки полярності для кожної ознаки в тексті. Для отримання оцінки по всьому тексту, варто підсумовати всі оцінки, використовуючи запропоновані функції оцінювання, які надають кращі результати, аніж звичайний підсумок [27]. Окрім цього, автори запровадили декілька лінгвістичних правил для опрацювання заперечень та пропозицій, що містять зв'язок «but». Також, в документі представлений цілісний підхід для вирішення проблеми виявлення полярності аналізуючих сентиментальних слів.

1.3.3 Підхід, заснований на машинному навчанні

Іншим способом, який можна використати для аналізу тональності текстового вмісту – це машинне навчання, яке включає в себе методи навчання штучної нейронної мережі з учителем та без. Навчання без вчителя використовує для аналізу не позначені набори вхідних даних, щоб визначити структуру та відзнайти схожі шаблони з отриманого датасету. Навчання без вчителя використовується для збору надійного анотованого набору даних, що виступає складною задачею, але на позначених - простою. Такий підхід не викликає жодних труднощів при завантаженні нових даних, які можуть бути з іншої тематики. Kivo використовує описаний вище метод для класифікації відгуків користувачів. Кожен з таких відгуків містить метадані та позначається як рекомендований абож не рекомендований [28].

Автор опрацьовує фрази, що складаються з двох слів на основі патернів та тегів. Розробка патернів представляється таким чином, що вони захоплюють звичайні сентиментальні набори слів. Кожна з фраз представляє комбінацію прикметника/прислівника та іменника/дієслова (загалом пропонується 5 моделей). Для прийняття рішення, які з фраз варто завантажити, до набору даних застосовується ідентифікатор частини мови POS (part-of-speech).

Важливо відмітити, що фраза витягується, якщо два запропоновані слова підпадають під одну модель. Наступним кроком буде розрахунок семантичної

орієнтації опрацьованих фраз. Автор застосовує алгоритм Pointwise Mutual Information та Information Retrieval algorithm (PMI-IR) для пошуку семантичної орієнтації. PMI вимірює семантичну схожість між двома термінами. Фразу, яка відповідає моделям, вважають першим терміном, в той час як словосполучення вважають другим терміном. Слова «excellent» та «poor» визначаються як довідкові слова, оскільки природньо оцінювати огляд як «poor», коли він отримує одну зірку та «excellent», якщо він отримає п’ять зірок. Семантична орієнтація фрази визначається як різниця між PMI фрази «excellent» та PMI фрази «poor» [29].

У випадку, якщо фраза має сильніший зв’язок з довідковим словом «excellent», семантична орієнтація рахується позитивною. Але якщо такий зв’язок є сильнішим зі словом «poor», тоді відповідно негативною. Для розрахунку PMI потрібно визначати ймовірність співпадіння відповідних термінів. Останній крок полягає у визначенні настрою для всього огляду (рекомендованого чи не рекомендованого). Огляд буде вважатися рекомендованим, якщо його середня семантична орієнтація є позитивною. Та не рекоменованим у іншому випадку. За інформацією аналізу, можна дізнатися середня точність у відсотковому співвідношенні. Codar та Talinc застосувують метод навчання без вчителя для класифікації рецензій на фільмів. Автори змогли адаптувати метод, запропонований користувачами для класифікація китайського набору тексту.

Основна ідея методики полягає у використані позитивних слів, які можна отримати з документа. Такого роду сентиментальні слова (прислівники) є попередніми запереченнями або ж можуть бути й без заперечення. Наявність початкового відгука користувачів дозволило збагатити перелік позитивних слів, застосовуючи ітераційну класифікацію. Натхнені даним підходом автори статті склали початковий набір слів. Текстовий вміст документу, що підлягає класифікації, повинен бути розділений на зони, кожна з яких відповідає певному фрагменту тексту, розташованому між пунктуаційними символами. Потім виконується класифікація кожної окремої зони.

Семантична орієнтація всього вхідного тексту визначається переважанням позитивних та негативних регіонів у документі. Якщо доступні позитивні зони зустрічаються частіше, аніж негативні, то такий документ вважається позитивним, інакше ж негативним. Автори також розширили загальний список початкового набору слів. Вони використовують два-, три-, чотири- грами в якості початкових слів. Проте два- та три- грами не можуть зберегти змісту фрази. Використання чотирьох грамів надало незадовільні результати. Зробивши другу спробу, з використанням семантично значущих прикметників, в якості початкових слів, очікуваного приросту точності все так ж не було досягнуто. Дослідники також намагалися змінити підхід підрахунку на k-середніх кластерах, але отримані результати не збільшили показник [30].

Методи навчання нейронної мережі з учителем передбачають наявність розміченого навчального набору даних, які використовуються для процесу навчання мережі. Зазвичай модель «bag-ofwords» використовується для представлення документа в якості вектору ознак

$$d = (w_1, w_2, \dots, w_i, \dots, w_N) \tag{1.13}$$

де N це набір унікальних термів в тренувальному наборі даних та w_i це вага i-ої ознаки. Щоб перетворити навчальний набір даних у векторну форму, потрібно створити словник з N унікальних слів. Наступним кроком, будь-яка з моделей може бути використана для подубови вектору ознак:

– бінарна модель ознак w_i визначається як 1, якщо ознака присутня в документі, інакше – 0;

– частота ознак (TF) - визначає кількість зустрічаннятерміну в документі;

– TF-IDF (IDF - зворотня частота в документі) вимірює важливість ознаки (TF припускає, що всі ознаки однаково важливі).

Вектор може бути використаний, після того як його дані були представлені, класифікатором для навчання та отримання міток. Значна кількість методів використовується для тренування класифікатора. Найбільш простим та поширеним метод, який використовується для класифікації тексту, визначають метод Наївного Байєсу. Дана модель базується на теоремі Баєйса з припущенням, що всі ознаки є незалежними. Наївний Байєсовий класифікатор визначає ймовірність того, що документ належить до конкретного класу.

До переваг Байєсового класифікатора варто віднести простоту в реалізації, достатньо швидкий процес навчання та хороші результати. Проте «наївне» припущення може стати проблемою, тому як в реальних обставинах ознаки є залежними одна від одної. Згідно з «the idea behind Maximum Entropy models is that one should prefer the most uniform models that satisfy a given constraint».

Ймовірність того, що документ належить до конкретного класу розраховується наступним чином:

$$P(c|d,\lambda) = \frac{\exp\left[\sum_i \lambda_i f_i(c,d)\right]}{\sum_{c'} \exp\left[\sum_i \lambda_i f_i(c,d)\right]} \quad (1.14)$$

де, c - це клас, d - це документ, який потрібно класифікувати, λ - це вага i-ого класифікаційного індикатора f_i.

Класифікатор максимальної ентропії не припускає незалежність ознак. Проте, цей класифікатор теоретично повинен показувати кращі результати, ніж класифікатор Наївного Байєса, але такий алгоритм є більш складним для реалізації та процес навчання є значно довшим [31].

Інший підхід до класифікації - заснований на правилах. Основна ідея полягає у визначенні набору правил, які будуть згенеровані експертами на основі аналізу певної предметної області. Такий метод може показати хороші результати, коли використовують досить широкий набір правил. Однак, створення великої кількості

правил вимагає тривалого часу. Підхід, заснований на правилах, був використаний Chikersal та іншими. Вони представили правила, які напряму залежать від використання емоцій та сентиментальних слів в твітах. Окрім цього, автори використовують Support Vector Machine (SVM) класифікатор. Цей класифіктор містить лінійне ядро та L1-регуляризацію; також були застосовані такі підходи, як n-грами, POS-теги, словесні n-грами на декількох різних лексиконах. Головна ідея підходу полягає у комбінуванні двох різних методів для підвищення точності та зниження коефіцієнту відхилень.

Кожен новий твіт, який містив нейтральну характеристику завдяки SVM класифікатору, був підданий аналізу класифікатором, який використовує правила, для визначення кінцевої оцінки. Аналізуючи результат, варто відмітити, що підхід, який використовує правила, значно покращує прогнози, які були надані SVM класифікатором. SVM класифікатор також використовується в дослідженнях. Інакшим способом розв'язання проблеми з класифікацією текстових наборів даних є підхід з використанням нейронних мереж (NN). Штучна нейронна мережа дотримується принципів біологічної нейронної мережі. Очікується, що нейронна мережа буде вирішувати проблеми таким ж чином, як їх вирішує наш мозок.

Згорткові нейронні мережі (CNN) використовують різновид багатошарових перцептронів, розроблених спеціально з вимогою у використанні мінімального обсягу попередньої обробки. Вони відомі також як інваріативні відносно зсуву або просторово інваріативні штучні нейронні мережі, виходячи з їхньої архітектури спільних ваг та характеристик інваріативності відносно паралельного перенесення. Згорткові мережі було натхнено біологічними процесами, в яких схему з'єднання нейронів натхнено організацією зорової кори тварин [32].

CNN можуть бути використані для класифікації фраз. Підхід полягає у класифікації речення на позитивні та негативні класи, також на більш детальні класи, що визначають чи є пропозиція суб'єктивною або об'єктивною думкою, та класифікують речення на шість категорій. Для такого опрацювання

використовуються різні набори тестових даних. Як і в попередніх дослідженнях, використовувався один шар CNN. Дана модель показалачудові результати і "pre-trained vectors are ‘universal’ feature extractors that can be utilized for various classification tasks".

Рекурентна нейронна мережа (RNN) мережа зі зворотним зв’язком дозволяє зберігати інформацію про попередній момент часу. Даний тип мережі містить вихід, який обчислюється попереднім кроком, використовуючи для обчислення наступний. Далі вихід порівнюється з даними тесту і оцінюється коефіцієнт помилки, визначений на основі того, які ваги були кориговані, що робить процес навчання значно точнішим. RNN корисний для прогнозування наступного слова в реченні. Дана властивість дозволяє краще зрозуміти речення, зафіксувавши контекст кожного слова на основі попередніх. Та визначати речення як атомарну визначену структуру.

Pengfei Liu та інші застосували RNN для класифікації тексту з багатозадачним навчанням. У якості завдання вони обрали класифікацію п’яти класів, бінарну класифікацію, класифікацію речення суб’єктивно та об’єктивно та бінарну класифікацію на рівня текстового документа. У статті автори предсталяють три архітектури обміну інформацією для моделювання текстової послідовності. Перша архітектура використовує загальний шар для всіх завдань. Друга використовує різні шари для різних завдань. Третя передбачає призначення для певного завдання першого рівня, але також містить загальний шар для всіх доступних завдань [33]. Після проведених експериментів автори порівнювали результати та дійшли висновку, що на поставленому завданні вони досягли кращих результатів, протилежних найсучаснішим вихідним рівням.

2 ОПИС СПОСОБІВ РЕАЛІЗАЦІЇ АДАПТИВНОГО РІШЕННЯ ЗАДАЧІ КЛАСИФІКАЦІЇ. ПРОЦЕС НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ

2.1 Інформаційна система аналізу тональності тексту

На сьогоднішній день ми складно собі уявляємо своє життя без доступу до всесвітньої павутини, всі ми користуємося інтернетом для різних цілей, здебільшо шукаючи інформацію деяких відомостей або публікачи щось в мережі. Нова публікаці з легкістю створюється мільярдами користувачів блогів, форумів, соціальних мереж. Існує велика кількість веб-ресурсів, які пропонують огляди продуктів. Наприклад, Amazon – це інтернет-магазин, який надає можливість клієнтам опублікувати свої відгуки про придбані ними продукти, а також проводити пошук відгуків залишених іншими користувачами для прийняття рішення про покупку товару. Іншим прикладом слугує TripAdvisor – це веб-ресурс, який надає десятки виважених відомостей про ресторани, готелі, авіа-рейси, що є надзвичайно корисним для мандрівників. Twitter – це ще один спосіб обміну думками.

Інформація наданим з цих джерез використовується не тільки споживачами, але також є критичною для різних організацій та компаній. Значна кількість наявних текстових даних свідчить про необхідність розробки автоматизованої системи пошуку та класифікації думок. Задача класифікації тексту не є новою областю для вивчення; проте, в основному наводилися дослідження на коротких текстах та оглядах фільмів [34]. Стосовно Twitter, публікації відрізняються від відгуків по їх довжині (210 символів) та наявності спеціальних символів. Крім цього, спосіб спілкування там неформальний, представлений використанням сленгу та ідіом, наявністю помилок в написанні.

2.1.1 Збір та обробка інформації

Дана робота опрацьовує наступний перелік текстової інформації:

1) Заголовки новин з Американського джерела новин ABC (American Broadcasting Corp.) за 20 років. Кількість стрічок у файлі - 4384323. Це файл формату csv, що має дві колонки - дату публікації (у форматі yyyyMMdd) та текст заголовку новини;

2) Набір публікацій повідомлень мережі Твітеру. Немає однієї єдиної тематики, яка б об'єднувала усі наявні твіти. Але значним бонусом цього набору є те, що кожний твіт позначений у відповідності до категорії: empty (не має категорії/пустий твіт, 943 рядків); sadness (сум, 6593 рядків); enthusiasm (ентузіазм, 548 рядків); neutral (нейтральний твіт, 8652 рядків); worry (переживання, 8545 рядків); surprise (сюрприз, 3203 рядків); love (любов, 9653 рядків); fun (веселощі, 9543 рядків); hate (ненависть, 2392 рядків); happiness (щастя, 5494 рядків); boredom (нудьга, 943 рядків); relief (полегшення, 2934 рядків); anger (злість, 965 рядків);

Не кожна з запропонованих вище категорій підходить для обробки тональності. Для використання даного набору даних в оцінці тональності тексту на два класи (позитивний та негативний) було визначено категорії «love», «fun», «happiness», що відносяться до позитивного класу та «hate», «anger», «worry» до негативного. В підсумку, кількість позитивно помічених публікацій склала майже 20000, негативно помічених - майже 15000.

Окрім колонки з категорією тональності, набір також містить три колонки: tweet_id (ідентифікатор твіту), author (автора публікації), content (текст самого твіту). Цей файл також представлений у форматі csv та містить 33463 рядків. Обробка текстових даних з платформи Twitter є критичною в рамках часу роботи структурних моделей та точності класифікаторів, оскільки зашумлені дані можуть сповільнювати процес навчання та погіршувати точність системи.

Наступними кроки обробки твітів твітів визначаються:

– Видалення веб-посилань, часто твіти містять веб-посилання, щоб ділитися деякою додатковою інформацією. Зміст посилань не аналізується, тому адреса не надає ніякої корисної інформації, і її усунення може зменшити розмір об'єкта.

– Видалення імен користувачів, інший користувач може бути згаданий автором повідомлення у твіті, використовуючи символ «@», а потім ім’я користувача. Ці дані також не несуть жодної релевантної інформації.

– Видалення хеш-тегів. Хеш-тег зображується за допомогою символу «#» і використовується перед словом, що представляє назву теми. У даній роботі не стоїть завдання класифікувати теми твітів, тому вони також видаляються.

– Видалення ретвітів і дублікатів, retweet - твіт, який написаний одним користувачем, а потім копіюється та публікується іншим користувачем. Такі дані видаляються, оскільки можуть додати зайві ваги на якісь слова.

– Стиснення «витягнутих» слів. Зазвичай слова подовжують, щоб виразити почуття, наприклад «Hi, maaaannn», радість від того, що зустрів приятеля. Від користувача залежить, скільки букв він буде повторювати, зазвичай, це не є фіксованим числом. Тобто, в одному твіті буде написано «ahh», а в іншому - «ahhh».

– Видалення стоп-слів. Стоп-слова є частинами речень, які вважаються марними для сприйняття їх у якості ознак, тобто «the», «for», «her», «a» тощо;

– Приведення усіх слів до нижнього регістру, це є необхідно тому, що слова «sad» та «SaD» є однаковими.

3) Фрагменти новин зі статей, які були оброблені експертами та помічені, чи були актуальні вони актуальні для економіки США. Файл формату csv, містить наступні колонки: _unit_id, _golden, _unit_state, _trusted_judgments, _last_judgment_at, positiviy, positivity:confidence, relevance, relevance:confidence, articleid, date, headline, positivity_gold, relevance_gold, text. Для тренування та тестування текстом з позитивним окрасом буде вважатися той, у якого positivity більший або рівний 5, відповідно, негативний - менше 5. В підсумку, остаточна кількість рядків для аналізу рівна 2364.

2.1.2 Розробка методів класифікації тексту.

Після того, як обробка даних була завершена, потрібно витягнути словаознаки та використати їх для тренування класифікаторів. У даній роботі було

застосовано декілька методів, які дозволяють отримувати слова-ознаки з тексту. Був досліджений алгоритм вибору ознак на основі моделі Хі-квадрат для моделі Наївного Байєса [35]. x^2 - це статистична перевірка, яка вимірює незалежність між класом та ознакою. Вона оцінює значення кореляції між класом та ознакою. x^2 можна розрахувати, використовуючи наступну формулу:

$$x^2(D,t,c) = \sum_{e_t \in 0,1} \sum_{e_c \in 0,1} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}} \quad (2.1)$$

де D - це тренувальний набір даних,

N - спостережувана частота, а E - очікувана частота,

e_t = 1 якщо документ містить слово t,

e_t = 0 якщо документ не містить слово t,

e_c = 1 якщо документ знаходиться в класі c,

e_c = 0 якщо документ не знаходиться в класі c.

Перепишемо формулу вище:

$$x^2(D,t,c) = \frac{(N_{00} + N_{01} + N_{10} + N_{11}) * (N_{11}N_{00} - N_{10}N_{01})^2}{(N_{11} + N_{01}) * (N_{11} + N_{10}) * (N_{10} + N_{00}) * (N_{01} + N_{00})} \quad (2.2)$$

де N - це кількість даних у тренувальному наборі, де N розраховується за формулою: $N = N_{00} + N_{01} + N_{10} + N_{11}$,

N_{00} - кількість речень, які не містять ознаку (e_t = 0) та не в класі (e_c = 0)

N_{11} - кількість речень, які мають співпадіння ознаки та класу,

N_{10} - кількість речень, які містять ознаку, але знаходяться не в класі,

N_{01} - кількість речень, які знаходяться в класі, але не містять ознаку.

Велике значення x^2 значить, що 2 події не є незалежними (у нашому випадку,

ознака та клас є залежними), це значить, що нуль-гіпотеза про незалежність повинна бути відхилена. Якщо події незалежні - ознака має бути врахована. Однак, значення результату x^2 дозволяє вибрати найбільш інформативну ознаку для тренування класифікатора.

Другий експеримент проводився за допомогою згорткової нейронної мережі, CNN використовує фільтри (ядра), які відіграють роль детекторів ознак. Використовуючи початковий набір даних, необхідно сформувати словниковий запас, де кожне слово індексується. Після створення словника, перші шари CNN представлені як малорозмірні вектори. А саме, кожен документ обробляється як послідовність слів $s = [s_1, s_2, \dots, s_n]$, де кожне слово має свій індекс, який вказує на позицію цього слова в словнику v. Речення різної довжини були пронормовані, підбиванням їх до максимальної довжини речення.

Для вибору інформативних ознак з початкового набору даних і переходу до вищого рівня, необхідно використовувати операції згортки та об'єднання [36].

2.1.3 Алгоритм класифікації Наївного Байєса

Підхід з використанням класифікатора Наївного Байєса показав свою ефективність та простоту у класифікації тональності. Це ймовірнісний підхід, який дозволяє обраховувати ймовірність того, що ознака належить до якогось класу

$$P(label|features) = \frac{P(label) * P(features|label)}{P(features)} \quad (2.3)$$

де $P(label|features)$ це апостеріорна ймовіність того, що ознака належить до класу (позитивного чи негативного),

$P(label)$ - це ймовірність обраного класу,

$P(features|label)$ - це умовна ймовірність того, що конкретна ознака з'явиться у даному класі,

$P(features)$ - це ймовірність обраної ознаки.

Зробимо «наївне» припущення про те, що ознаки є незалежними одна від одної. Це дає можливість написати наступне:

$$P(label|features) \approx P(f_1|label) * \dots * P(f_n|label) = \prod_{i=1}^{n} P(f_i|label) \quad (2.4)$$

де f_i - це конкретна ознака.

Не дивлячись на те, що модель Наївного Байєса робить припущення про незалежність відносно позицій слів, вона показує досить непогані результати. Основна ціль класифікатора це визначити клас, до якого належить ознака [37]. Класифікатор Наївного Байєса використовує оцінку максимальної апостеріорної ймовірності для визначення найбільш відповідного класу $label_{map}$:

$$label_{map} = argmax_{label \in L}[\frac{\hat{P}(label) * \prod_{i=1}^{n} \hat{P}\,(f_i|label)}{\hat{P}(features)}] \quad (2.5)$$

Знаменник можна не включати, тому що для позитивного класу він дорівнює тому самому, що й для негативного. Отже, вираз можна переписати як:

$$label_{map} = argmax_{label \in L}[\hat{P}(label) * \prod_{i=1}^{n} \hat{P}\,(f_i|label)] \quad (2.6)$$

P помічено як $\hat{P}$, тому що істинні значення відповідних параметрів будуть оцінюватися з навчального набору даних. На підставі останній вираз може призвести до проблеми зникнення порядку, яку можна уникнути, якщо використовувати логарифмічну властивість, а саме $log(xy) = log(x) + log(y)$.Тепер множення імовірностей представлено як сума логарифмів, тому рівняння вище можна переписати наступним чином:

$$label_{map} = argmax_{label \in L}[log\hat{P}(label) + \prod_{i=1}^{n} log\hat{P}\,(f_i|label)] \qquad (2.7)$$

Як було згадано, підрахуно $P(label)$ та $P(f_i|label)$ проводиться на тренувальному наборі даних. Ймовірність $\hat{P}(label)$ може бути обчислена як

$$\hat{P} = \frac{N_{label}}{N} \qquad (2.8)$$

де N_{label} - кількість ознак, яка належить до конкретного класу, та N - загальна кількість ознак. Умовна ймовірність $\hat{P}\,(f_i|label)$ обчислюється:

$$\hat{P}(f_i|label) = \frac{F_{ilabel}}{\sum_{i\prime \in V} F_{i\prime label}} \qquad (2.9)$$

де F_{ilabel} - кількість разів, скільки i − та ознака зустрілася у тренувальному наборі даних у конкретному класі, включаючи повторення ознак, та v - це словник усіх унікальних ознак у конкретному класі.

Важливо зауважити, що на етапі класифікації може статися так, що класифікатор зустрів нову ознаку, якої не було у тренувальній вибірці, отже, є невідомою для класифікатора. Кінцева формула виглядатиме насутпним чином:

$$label_{map} = argmax_{label \in L}[log\frac{N_{label}}{N} + \sum_{i=1}^{n} log\frac{F_{ilabel}}{\sum_{i\prime \in V} F_{i\prime label}}]\ . \qquad (2.10)$$

2.1.4 Згорткова нейронна мережа

Припускаємо, що перетворення вхідного тексту було виконано і тепер він представлений у вигляді високорозмірного вектора. Наступним кроком є

застосування згорткових, нелінійних операцій та максимізаційного агрегування для вилучення ознак із вхідних даних [38].

Згорткові шари. Операція згортки представлена на рисунку 2.1. Згортковий шар дозволяє отримати шаблони, які часто використовуються в даних. Більш конкретно, для того, щоб отримати нову ознаку, необхідно застосувати операцію згортки. З цією метою, n слів з речення згортаються з ваговими фільтрами v для отримання карти ознак. Ваги фільтру ініціалізуються спочатку випадково, а потім регулюються під час навчання.

$$c_i = f(w * s_{i:i+n-1} + b), \tag{2.11}$$

де w - це вектор вагів, $s_{i:i+n-1}$ - ковзне вікно, $b \in R$ - вектор упередження, f - нелінійна функція.

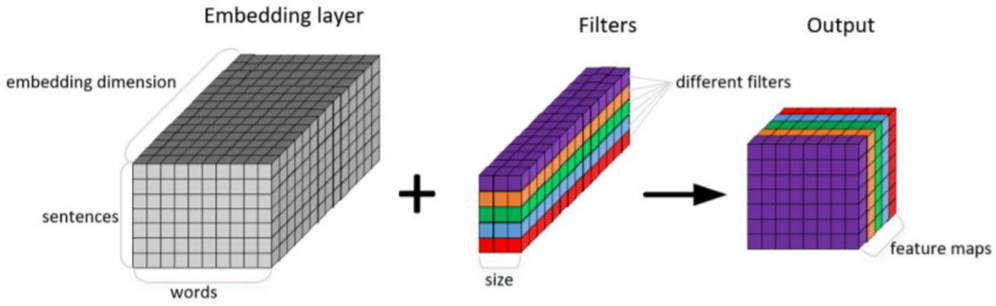

Рисунок 2.1 – Операція згортки

Фільтр застосовується до кожної послідовності слів у реченні, що відповідає розміру фільтра $\{s_1{:}\,n, s_2{:}\,n+1, \dots, s_{s-n+1}{:}\,s\}$ для створення карти ознак:

$$c(w) = [c_1, c_2, \dots, c_{s-n+1}] \tag{2.12}$$

ReLU (Rectified Linear Unit) - береться у якості нелінійної функції та дуже інтенсивно використовується дослідниками, і застосовується після шару згортки.

Всі негативні значення на карті ознак перетворені на 0, щоб гарантувати, що кари ознак є позитивними. Приведемо наступну формулу:

$$f(w) = \max(0, x). \tag{2.13}$$

Агрегувальні шари. Після того, як ReLU був використаний на шарі згортки, він генерує вхідну інформацію для агрегувального шару. Графік функції активації ReLU наведено на рисунку 2.2.

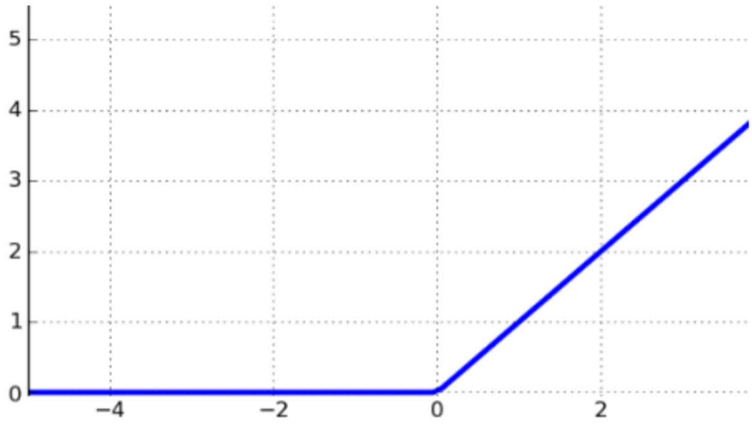

Рисунок 2.2 – Фунція активації ReLU

У даній роботі була застосована операція максимізаційного агрегування, яка дозволила зменшити розмір карти ознак та одночасно зберегти найбільш релевантні ознаки:

$$\hat{c} = \max(c(w)). \tag{2.14}$$

У цій роботі було використано 128 фільтрів, що породжувало 128 карт ознак. Після максимізаційного агрегування виходи передаються повністю підключеному шару, де вони об’єднуються в один вектор ознак. Використовуючи останній рівень

softmax, виводиться розподіл ймовірності по двох класах (позитивному чи негативном):

$$p(y = j|x) = \frac{e^{x^T w_j + b_j}}{\sum_{k=1}^{K} e^{x^T w_j + b_j}} \quad (2.15)$$

де x - вихід передостанніх згорток та агрегування, представлених у вигляді щільного вектора, w_k - вектор вагів k-ого класу, та b_k – зміщення k-ого класу.

Функція відключення - це спосіб запобігання переповнення мережі. Навчання, як правило, виконується за допомогою стохастичного градієнтного спуску шляхом випадкового вибору деяких зразків з набору даних [39]. Відключення передбачає, що лише на етапі навчання вилучається деяка частина нейронів (показник відсіву встановлений до 0.5), що запобігає коадаптації нейронів і призводить до навчання більш надійних ознак та робить модель більш узагальнюючою. Вихід після застосування відсіву відображається у вигляді:

$$y = w(zr) + b, \quad (2.16)$$

де передостанній шар $z = [\hat{c}_1, \dots, \hat{c}_m]$, r - вектор, який містить 0 та 1.

У загальному випадку, відсів пришвидшує процес навчання. Модель CNN навчається, щоб мінімізувати функцію перехресної ентропії, яка може бути записана наступним чином:

$$H(p, q) = -\sum_{x} p(x) log q(x) \quad (2.17)$$

де $p(x)$ - ймовірність істинної ймовірності (правильної відповіді), $q(x)$ - оцінена ймовірність.

Тренування CNN передбачає коригування параметрів мережі. Цей процес налаштування називається методом зворотного поширення помилки. Для розрахунку градієнта функції помилки відносно ваги фільтра застосовується зворотне поширення. Алгоритм Адама – це стохастичний градієнтний алгоритм спуску, використовується для оптимізації параметрів CNN (оновлення ваг).

Модель згорткової нейронної мережі зображена на рисунку 2.3. Вихідні розмірі, вироблені після кожного шару, наведені на малюнку нижче, де партія відповідає розміру пакету і дорівнює 64. Параметр «len» відповідає максимальній довжині послідовності в наборі даних; затемнення позначає розмірність і становить 128; параметр «filter_size» становить 3, 4, 5 відповідно (зображені 3-ома кольорами); параметр «num_filters» – 128 та відповідає кількості фільтрів. Розмір кроку дорівнює 1 (зміна фільтра на крок) Архітектура згорткової нейронної мережі наведена на рисунку 2.3.

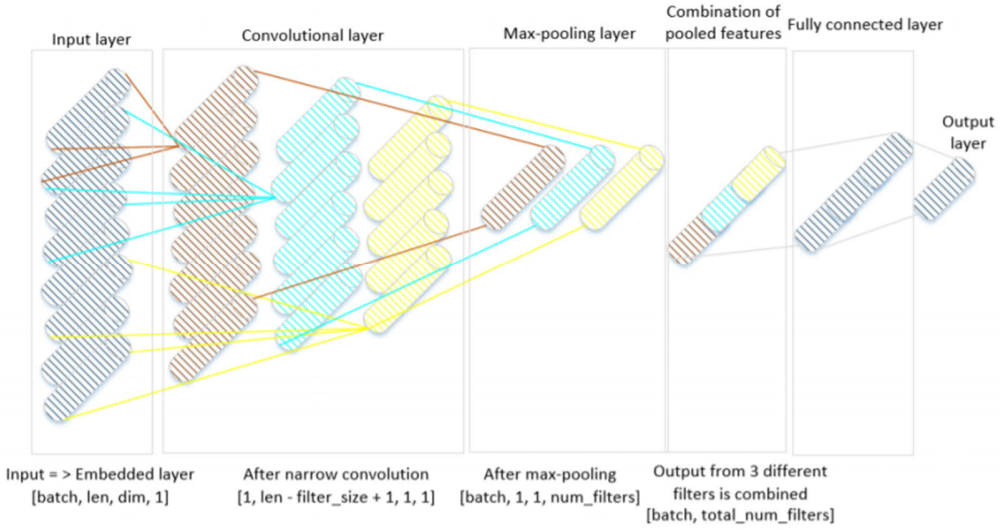

Рисунок 2.3 – Архітектура CNN

Підхід Наївного Байєса досить часто використовується дослідниками завдяки своїй простоті та високій продуктивності, незважаючи на припущення про незалежність ознак. Крім того, в останні десятиліття нейронними мережами стали

цікавитися високопоставлені вчені, особливо після того, як Krizhevsky та інші представили результати, отримані ними для завдання розпізнавання зображень та відео [40]. Подальші дослідження показали, що CNN також застосовується до завдань природньої мови та може ефективно класифікувати текст. Основна частина цього розділу ілюструє деталі алгоритмів, що використовуються для класифікації тексту, а саме Наївний Байєс на згорткова нейронна мережа.

2.2 Використання нейронної мережі Кохонена у проектах розпізнававння рекламних текстів.

Карта Кохонена представляє нейронну мережу за типом навчанням без вчителя, яка виконує завдання візуалізації та кластеризації. Підхід до проектування багатовимірного простору в простір нижчого порядку також знайшов застосування у вирішенні завдань моделювання, прогнозування абощо. Карта або мережа Кохонена складається з групи компонентів, які називаються вузлами або нейронами. Представлення карти зображено на рисунку 2.4. Кількість вузлів задається програмістом або ж аналітиком.

Кожен із вузлів описується двома векторами. Перший – вектор ваги, що містить такий самий розмір, як і отримані вхідні дані, другий – координати вузла на карті. Опис карти формується з вищого вхідного вузла, до найнижчого. Відповідно до відомого розміру вхідного набору даних, будується початковий варіант карти. В процесі навчання мережі, вектори ваги вузлів наближаються до вхідних даних. Для кожного спостереження обирається найбільш схожий по вектору ваги вузол, і значення його наближається до спостереження. Також до спостереження наближаються вектори ваги декількох вузлів, розташованих поряд [41]. В результаті, якщо в масиві вхідних даних два елементи являються подібними, на

карті їм будуть відповідати близькі нейрони активності. Приклад нейронної мережі Кохонена наведено на рисунку 2.4.

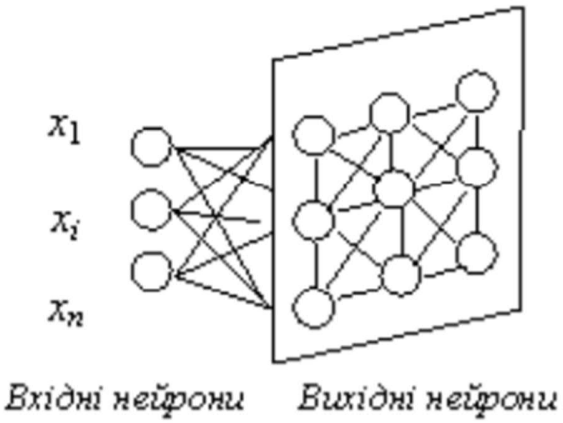

Рисунок 2.4 – Нейронна мережа Кохонена

Проводячи навчання в мережі Кохонена, ми стараємося досягти результату, щоб на виході різні частини мережі однаково реагували на певні вхідні дані. Навчання розпочинається із задавання випадкових значень матриці зв'язків W_n^m. Наступним кроком відбувається процес самоорганізації, що виконується для модифікації ваги при пред'явленні на вхід векторів навчальної вибірки. Кожному з нейронів, можна визначити відстань до вектора входу:

$$d_m = \sum_{i=1}^{N} (x_i(t) - W_i^m(t))^2 \qquad (2.18)$$

Наступним кроком обирається нейрон $m = m*$, для якого ця відстань мінімальна. Даний етап навчання t будуть модифікувати тільки ваги нейронів в найближчі до нейрона $m*$:

$$W_n^m(t+1) = W_n^m(t) + \eta(x_n(t) - W_n^m) \qquad (2.19)$$

На початку найближчими до будь-якого з нейронів знаходяться всі нейрони мережі, надалі це наближення звужується. В кінці етапу навчання підстроюються тільки ваги найближчого нейрона. Темп навчання $h(t) < 1$ з часом також зменшується. Ітеративно відбувається розпізнання образів вибірки. При кожному розпізнаванні відбувається зважування.

Будь-який нейрон несе інформацію про свій кластер – згусток в просторі вхідних образів, формуючи для даної групи збірний образ. Саме тому, нейронна мережа Кохонена має здатність до узагальнення [42]. Певному вибраному кластеру може відповідати декілька нейронів з близькими значеннями векторів ваги, отже похибки у роботі одного нейрона не представляються критичними для функціонування всієї карти Кохонена.

Розглянемо алгоритм побудови нейронної мережі Кохонена для аналізу рекламних текстів.

Першим підготовчим етапом є процес верифікації, нормалізації речень, як структурних одиниць моделі тексту реклами. Для цього ми використовуємо синтаксичний аналіз речень наданого рекламного тексту на основі граматики зв'язків. За допомогою граматики зв'язків можливим є синтаксичний аналіз речення, на підставі якого формується дерево залежностей між парами синтаксично значущих слів, пунктуація та написання слів при цьому відкидуються. Окрім цього, паралельно відбувається морфологічний аналіз. Приклад синтаксичного розбору речення представлено на рисунку 2.5.

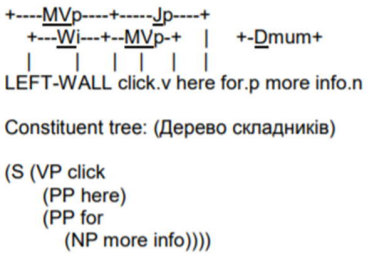

Рисунок 2.5 – Синтаксичний розбір речення парсером Link Grammar Parser

MVp – MV-зв'язок поєднує дієслова (та прикметники) із означальними фразами;

Jp – J- зв'язок поєднує прийменники та їх об’єктами;

WI – W-зв'язок використовується для поєднання головної частини речення із початком речення;

Dmum - D-зв’язок визначає зв'язок іменників та їх означень, означальними конструкціями.

Після цього, наступним етапом попередньої підготовки тексту є його переведення у цифровий формат, який буде зрозумілим для нейронної мережі Кохонена. Таку задачу можна вирішити двома способами: використання ресурсу WordNet або використання хеш-функції.

Хешування (англ. hashing) – процес перетворення вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини. Такі перетворення також називаються хеш-функціями або функціями згортки, а їх результати називають хешем, хеш-кодом. Існує велика кількість ефективних алгоритмів хешування з різними характеристиками (обчислювальна складність, розрядність тощо). Вибір конкретної хеш-функції визначається специфікою задачі, яку треба вирішувати [43].

Ідея використання ресурсу WordNet полягає у побудові цифрового коду для певної лексичної одиниці в залежності від відстані слів у WordNet. Штучна нейронна мережа обов’язковим чином вимагає виконання умови про можливість порівняння значень, в іншому випадку групування нейронів для вхідного сигналу певного типу позбавлене сенсу. Теоретично, якщо присвоїти вузлу графа певний випадковий номер, тоді спорідненим йому словам можна назначити номери, відмінні, наприклад на +1 для синонімів, та на +10 для антонімів. Словникові статті містять синонімічні рядки для кожної лексеми та пропонують як посилання на статтю, при чому визначаючи значення, яке і буде синонімічним. Прикладом синоніму до лексеми "skill" є лексема "accomplishment" у трьох або шести

значеннях, таким чином, можна зробити висновок, що номер словникової статті співвідноситься із віддаленістю синоніму від «кореневого» значення.

Наступним крок, базуючись на проведеній нормалізації речень, із подальшим їх синтаксичним розбором і оцифруванням, подаємо масив речень на виконання програмою Кохонена з метою використання навченої нейронної мережі для категоризації речення як рекламного чи нерекламного змісту [44].

У активному вікні Source Ads в графі Train and Visualize Neural Network ми можемо ввести речення, натискаючи Train and Visualize Neural Network та переходячи до вкладки Digitized Ads ми можемо відстежити результат перевірки речення на його рекламний характер. Візуалізація нейронної мережа являє собою зображення вузлів, кольорове рішення (діапазон від чорного до білого із відтінками сірого) для наочного представлення результатів. Таким чином, отримуємо область найінтенсивнішого кольору (чорний колір), яка вказує на наявність рекламного характеру в повідомленні, показник шкали насиченості кольору зменшується відповідно до виявлення нерекламного тону речення. Знімки екрану роботи програми зображено на рисунку 2.6, 2.7, 2.8, 2.9.

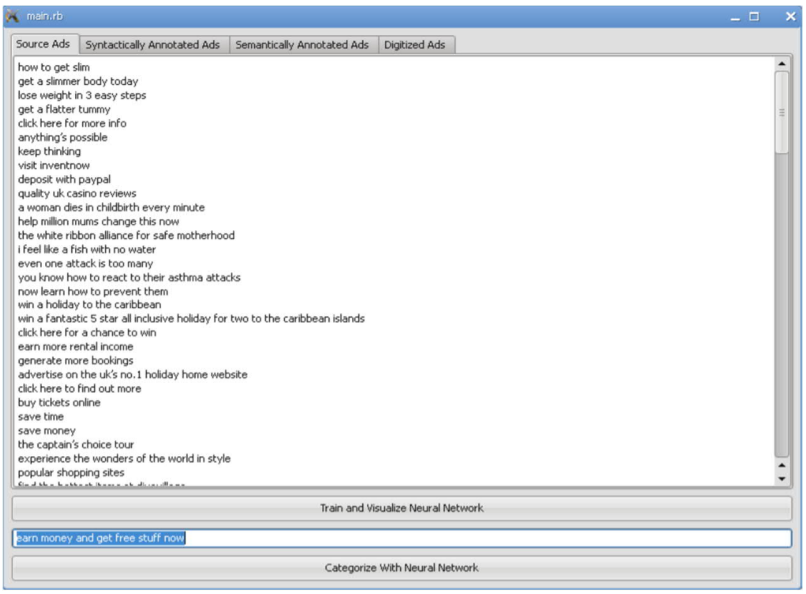

Рисунок 2.6 – Вікно вводу. Використання нейронної мережі Кохонена для категоризації речення «earn money and get free stuff now»

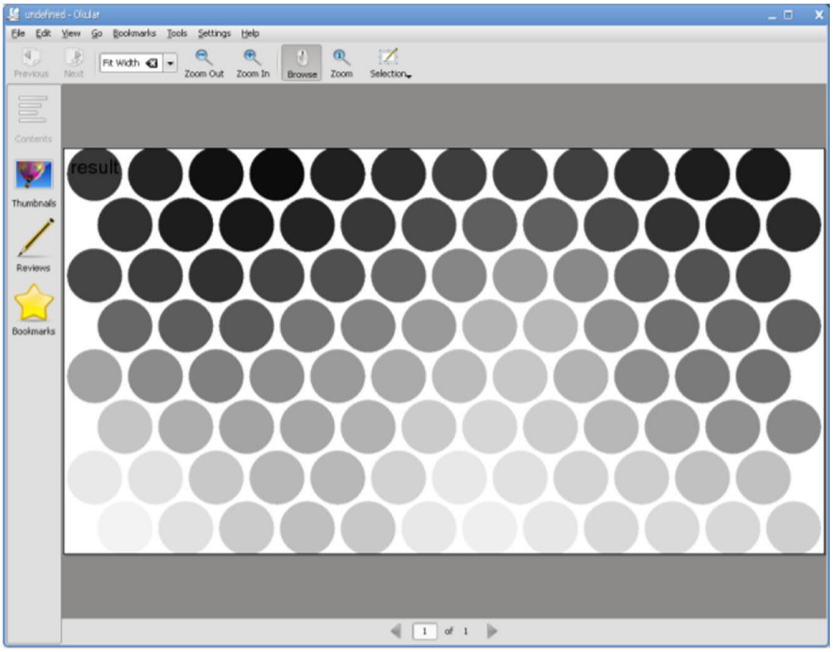

Рисунок 2.7 – Вікно виводу. Використання нейронної мережі Кохонена для категоризації речення «earn money and get free stuff now result ≈ an ad»

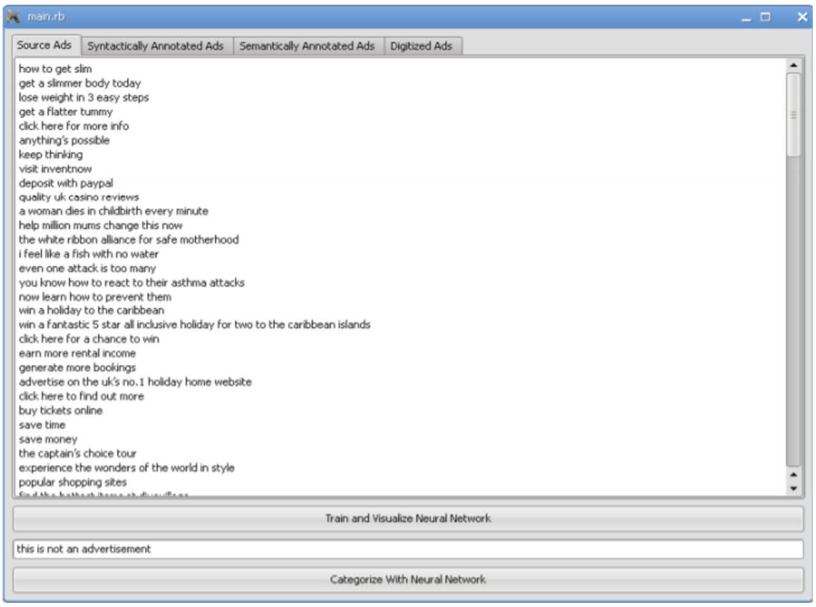

Рисунок 2.8 – Вікно вводу. Використання нейронної мережі Кохонена для категоризації речення «this is not an advertisement»

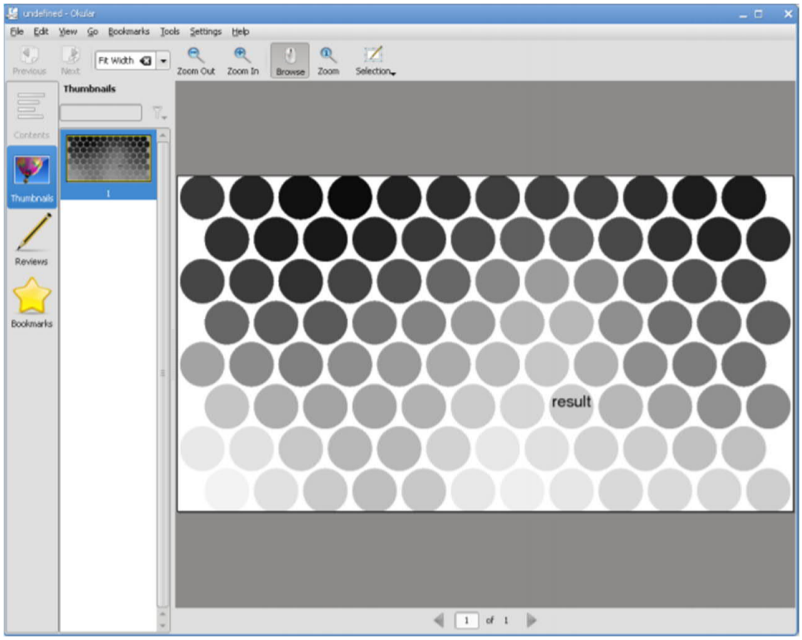

Рисунок 2.9 – Вікно виводу. Використання нейронної мережі Кохонена для категоризації речення «this is not an advertisment result = not an ad»

У випадку потреби до ідентифікації не лише окремих речень як рекламних, а загалом тексту, можна розглянути два варіанти до вирішення цього питання. Враховуючи достатньо обмежений структурний потенціал рекламних текстів, ми повинні безперечно враховувати коефіцієнти насиченості кольору для всіх речень рекламного тексту [45]. Тому визначаємо середній показник коефіцієнту насиченості кольору:

$$x = \sum_{i-1}^{n} \frac{x_i}{n} \tag{2.20}$$

де x_i – значення змінної X із номером, n – об’єм вибірки.

Виходячи з того, що повинно бути представлено у рекламному повідомленні, відокремлюються певні структурні елементи, функціональне та змістовне навантаження яких варіюється та на підставі когнітивних особливостей сприйняття

інформації людиною, ставимо за мету визначення підходу до введення ваги для показників насиченості кольору:

$$x = w_1 * a + w_2 * b + w_3 * c + w_4 * d \quad (2.21)$$

де w_1= 0,6 , w_2= 0,4 , w_3= 0,4 , w_4= 0,8 – ваги слів у реченні, a, b, c, d – показники насиченості кольору для речень 1,2,3,4 відповідно.

2.3 Алгоритм розпізнавання загрозливого вмісту у текстових повідомленнях

Галузь машинного навчання відноситься до підрозділу ком'ютерних наук, яка надає комп'ютерам здатність до навчання, не будучи програмованими, у загальному розумінні цього слова. Розвиток машинного навчання сфокусований на вивченя та побудову алгоритмів, які зможуть «навчатися» з набору даних певного наданого датасету та виконувати аналіз, щоб надати прогнози [46].

Одним із основних алгоритмів машинного навчання являється алгоритм класифікації. Такий алгоритм здатен розділяти дані по різним групам. Важливо розуміти універсальність алгортитмів машинного навчання. Прикладом можна навести ситуація, коли один алгоритм, який використовується для розпізнавання символів на зображені, може також успішно використовуватись для класифікації листів електронної пошти з метою поділу повідомлень на групи «нормальних» повідомлень та згенерований «спам», або ж класифікувати повідомлення за їх емоційним забарвленням на «позитивні», «нейтральні» та «негативні».

У відомих методах оперативного моніторингу загрозливі текстові повідомлення відбирають шляхом аналізу їх словарного складу та пошуку в них слів з «терористичного» словника. Проте точність такого відбору є надзвичайно

низькою внаслідок лексичної омонімії та контекстного характеру вживання будь-якого слова [46]. Це призводить до численних помилок 1-го роду – нейтральне повідомлення отримує статус загрозливого, а черга текстів, що потребують модерації, починає перевищувати можливості відповідної системи моніторингу загроз. З іншого боку, запобігання цьому шляхом збільшення нижньої межі для кількості різних слів з «терористичного» словника або визначення окремих з них як ключових слів усього тексту може мати гірший наслідок у вигляді помилок 2-го роду – загрозливе повідомлення не розпізнається. Зрозуміло, що алгоритм розпізнавання має максимально «розуміти» зміст повідомлення, у т.ч. прихований, інакше застосування зловмисниками синонімічних та метафоричних конструкцій унеможливить визначення загроз через порівняння з «терористичним» словником.

Для підвищення точності класифікації та аналізу прихованого змісту терористичних загроз пропонується метод на основі аналізу синтаксичних зв'язків між словоформами в реченнях текстового повідомлення. Метод базується на застосуванні моделі образного мислення людини та нечіткого відношення сенсу і забезпечує самовдосконалення бази знань системи моніторингу загроз. Згідно з парадигмою машинного навчання класифікаційне правило будується поступово за допомогою тренувальної колекції. Такий підхід забезпечує якість класифікації, яка може порівнюватися с якістю класифікації, що здійснюється людиною [47].

Розглянемо множину документів $D = \{d1,...,d|D|\}$, інформацію про кожний з яких можна отримати тільки з самого документа без залучення зовнішніх джерел. Відомою також є $C = \{c1,...,c|C|\}$ – множина категорій або абстрактних міток, кожна з яких може помічати один з документів множини D. Нехай $\Phi : D \times C \rightarrow \{0,1\}$ невідома цільова функція, що за парою (di , $c\ j$) визначає, чи належить документ i d категорії $c\ j$ (*1* або *T*) або ні (0 або *F*). Задача полягає у побудові (класифікаційної) функції Φ^{k} максимально близької до Φ.

Для отримання формальних параметрів тексту та побудови відповідного класифікатора текстові документи індексуються, як правило, на основі частотних

характеристик термів (слів) – такий підхід вважають базовим. Головною проблемою при цьому є зменшення розмірності векторів, якими представлені тексти та які пропорційні кількості різних слів, що зустрілися в колекції документів. Для зменшення розмірності векторів використовують як евристичні (фільтрація рідких та найчастотніших слів (прийменники, артиклі), визначення коефіцієнтів корисності окремих термів, їх групування тощо), так і математичні методи (сингулярний розклад матриць). Проте маємо додаткову проблему – уникнення авторами повідомлень слів загрозливого характеру заздалегідь нівелює ефективність зазначених методів та знижує якість класифікатору.

У роботі було отримано формальні переваги складних залежностей на противагу традиційному частотному словнику окремого тексту з огляду на визначення семантично-залежних параметрів цього тексту. З метою розв’язання задачі побудови функції класифікатора загроз k пропонується застосувати можливості сучасних лінгвістичних пакетів для визначення складних залежностей між реченнями текстового документу та покласти отримані чисельні параметри в основу процедури індексації текстової інформації [48]. У першому наближенні спосіб визначення порога для лінійного класифікатора та, власне, алгоритм лінійної on-line класифікації можна обрати стандартними.

У базовому підході текст розглядається як мультимножина термів (слів). Кожному слову ставиться у відповідність певне число (вага) – узагальнена характеристика частоти знаходження цього слова у тексті. Порядок слів, як правило, не враховується, а окрім частоти знаходження (головна ознака) враховується, зазвичай, додаткові ознаки, такі як «слово зустрілося в заголовку», «слово позначено іншим кольором», «слово позначено напівжирним або курсивом або іншими відмінними параметрами шрифту» тощо. На основі всіх цих ознак для кожного слова в тексті розраховується його вага.

Одним з найбільш популярних способів представлення ваги слова вважається TF·IDF (TF = term frequency, IDF = inversed document frequency). TF показує

наскільки часто слово зустрічається в конкретному документі, а IDF навпаки – наскільки рідко слово зустрічається в усіх документах деякої колекції. Іноді проводиться нормалізація по окремому документу для того, щоб сума квадратів усіх ваг для нього дорівнювала 1. Власне кожний текстовий документ за таким підходом можна вважати вектором у багатовимірному просторі, координати якого – це номери слів, а значення координат – значення відповідних ваг. Тоді розмірність вектора дорівнює кількості слів, що зустрічаються в колекції документів. Але з-за того, що враховуються всі слова, які коли-небудь зустрічалися в усій колекції документів, отримані вектори є розрідженими, тобто такими, що мають надвелику кількість координат, причому значення більшості з них – нулі. На відміну від базового у підході, що пропонується, замість слів (термів) координатами вектора вважаються зв'язки між словами. Таке, на перший погляд, парадоксальне рішення у квадратичній залежності лише посилює проблему розмірності векторів. Але, відповідно до моделі образного мислення людини, найбільш інформативними змістовними ознаками тексту можна вважати асоціативні зв'язки між мовними образами, які об'єднують леми та словоформи споріднених за ознакою спільного кореня, але, можливо, навіть відмінних за частинами мови слів. Тому розмірність більш інформативного вектору текстового документу на основі складних залежностей між словоформами не перевищує розмірність відповідного вектору за базовим підходом. Технологічно визначення складних залежностей забезпечують сучасні лінгвістичні пакети, зокрема DKPro Core (на основі фреймворку Apache UIMA та з програмною підтримкою Java / Maven / Eclipse) для англомовного тексту – цю платформу було обрано для створення програмного прототипу системи моніторингу загроз.

Формально ранжувальним класифікатором будемо вважати визначення функції $CSVi : D \rightarrow [0,1]$, що для кожного документу d_j повертає значення приналежності (*categorization status value*) d_j до c_i. Побудувати точний класифікатор можна таким чином: або відразу будувати функцію $CSVi : D \rightarrow \{T,$

F} або обчислити аналогічну ранжувальній функцію $CSVi : D \rightarrow [0,1]$, а потім визначити поріг (*threshold*) τi такий, що $CSVi \geq \tau i$ інтерпретується як *T*, а $CSVi < \tau i$ інтерпретується як *F*.

Зазвичай класифікатори за методами машинного навчання будуються на основі колекцій $\Omega = \{d1,...,d|\Omega|\}$ заздалегідь класифікованих експертами документів, тобто таких, для яких значення цільової функції точно відомо. Отже, для визначення якості, наприклад, бінарної класифікації загрозливих текстових повідомлень потрібно колекцію $\Omega = \{d1,...,d|\Omega|\}$ відповідних документів розбити на множини, що не перетинаються – *Tr* навчальної (*training*), V_a перевірочної (*validation*) та *Te* тестової (*test*) колекцій. Якщо перші дві з них використовуються для побудови (*Tr*) та оптимізації параметрів *(Va)* класифікатора, то множину незалежних з точки зору створення класифікатора документів *Te* потрібно застосувати для визначення метрик повноти π (доля вірно знайдених загрозливих документів серед усіх загрозливих документів) та точності ρ (доля справді загрозливих документів серед усіх загрозливих документів, що визначені так класифікатором). Відповідна формула навенеда нище:

$$\pi = \frac{TP}{TP + FN}, \tag{2.22}$$

$$\rho = \frac{TP}{TP + FN}, \tag{2.23}$$

де TP – число вірно знайдених загрозливих документів (true positive).

Отриманий програмний прототип системи моніторингу загроз будує мережу мовних образів (значимих словоформ) повідомлення шляхом накопичення складних синтаксичних зв’язків між лемами. Формальний аналіз отриманого графу лексичної онтології дозволяє визначити ступінь його близькості до класифікованого простору аналогічних графів, що відповідають відомим типам загроз.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ФІЛЬТРАЦІЇ ТЕКСТОВИХ НАБОРІВ ДАНИХ СОЦІАЛЬНИХ МЕРЕЖ НА ОСНОВІ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

3.1 Інструменти розробки штучної нейронної мережі для фільтрації текстових наборів даних соціальних мереж.

Keras — відкрита нейромережна бібліотека, написана мовою Python. Вона здатна працювати поверх TensorFlow, Microsoft Cognitive Toolkit, R, Theano та PlaidML. Спроєктовану для уможливлення швидких експериментів з мережами глибинного навчання, її зосереджено на тому, щоби вона була зручною в користуванні, модульною та розширюваною. Її було створено як частину дослідницьких зусиль проєкту ONEIROS (англ. Open-ended Neuro-Electronic Intelligent Robot Operating System), а її основним автором та підтримувачем є Франсуа Шоллє (фр. François Chollet), інженер Google.

Типові робочі процеси Keras виглядають так:

– визначення тренувальні даних: вхідний тензор і цільовий тензор;

– визначення мережі шарів (або моделі), яка відображає вхідні дані для наших цілей;

– налаштування процесу навчання, вибравши функцію втрат, оптимізатор і деякі показники для моніторингу;

– повторення даних тренування, викликаючи метод fit () наявної моделі.

Глибоке навчання відноситься до нейронних мереж з декількома прихованими шарами, які можуть вивчати все більш абстрактні уявлення вхідних даних.

У Keras є два основних типи моделей: послідовна модель і клас Model, який використовується з функціональним API.

Ці моделі мають ряд загальних методів і атрибутів:

– model.layers це плоский список шарів, складових моделей;

– model.inputs список вхідних тензорів моделі;

– model.outputs список вихідних тензорів моделі;

– model.summary () друкує короткий уявлення моделі;

– model.get_config () повертає словник, що містить конфігурацію моделі;

– model.get_weights () повертає список всіх вагових тензорів в моделі у вигляді масивів Numpy;

– model.set_weights (weights) встановлює значення ваг моделі зі списку масивів Numpy. Масиви в списку повинні мати ту ж форму, що і повертаються get_weights ();

– model.save_weights (filepath) зберігає вагу моделі у вигляді файлу HDF5.

– model.load_weights (filepath, by_name = False) завантажує вага моделі з файлу HDF5 (створеного save_weights). За замовчуванням очікується, що архітектура не зміниться.

Методи API послідовної моделі (Sequential model API) включають в себе наступний ряд:

Компіляція послідовної моделі налаштовує модель для вивчення

```
compile(
        optimizer,
        loss=None,
        metrics=None,
        loss_weights=None,
        sample_weight_mode=None,
        weighted_metrics=None,
        target_tensors=None
)
```

Список аргументів:

– optimizer: рядок (ім'я оптимізатора) або екземпляр оптимізатора.

– loss (втрата): рядок (ім'я цільової функції) або цільова функція або Lossекземпляр. Якщо модель має кілька виходів, ви можете використовувати різні

втрати на кожному виході, передавши словник або список втрат. Значення втрат, яке буде мінімізовано моделлю, буде тоді сумою всіх індивідуальних втрат.

– metrics: список метрик, які будуть оцінюватися моделлю під час навчання і тестування. Як правило, використовується metrics = ['accuracy']. Щоб вказати різні метрики для різних виходів моделі з декількома виходами, ви також можете передати словник, наприклад metrics = { 'output_a': 'accuracy', 'output_b': ['accuracy', 'mse']}. Також можна передати список (len = len (висновки)) списків метрик, таких як metrics = [['accuracy'], ['accuracy', 'mse']] або metrics = ['accuracy', ['accuracy', 'mse']].

– loss_weights: необов'язковий список або словник, що задає скалярні коефіцієнти (числа Python) для зважування вкладів втрат в різні вихідні дані моделі. Значення втрат, яке буде мінімізовано моделлю, буде потім зваженою сумою всіх індивідуальних втрат, зважених по loss_weightskоеффіцієнтам. Якщо список, очікується, що він буде мати співвідношення 1: 1 до виходів моделі. Якщо це диктат, очікується, що вихідні імена (рядки) будуть співставлені скалярним коефіцієнтами.

– sample_weight_mode: Якщо вам потрібно зробити зважування вибірки по тимчасовим кроків (2D ваги), встановіть це значення "temporal". Nonepo замовчуванням використовуються ваги вибірки (1D). Якщо модель має кілька виходів, ви можете використовувати різні sample_weight_modeна кожному виході, передавши словник або список режимів.

– weighted_metrics: список метрик, які будуть оцінюватися і зважуватися по sample_weight або class_weight під час навчання і тестування.

– target_tensors: за замовчуванням Keras створить наповнювачі для мети моделі, які будуть забезпечені цільовими даними під час навчання. Якщо замість цього ви хочете використовувати свої власні цільові тензори (в свою чергу, Keras не чекатиме зовнішніх даних Numpy для цих цілей під час навчання), ви можете вказати їх за допомогою target_tensorsаргумента. Це може бути один тензор (для

моделі з одним виходом), список тензорів або точні зіставлення вихідних імен з цільовими тензорами.

– kwargs: при використанні бекенда Theano / CNTK ці аргументи передаються в K.function. При використанні бекенда TensorFlow ці аргументи передаються в tf.Session.run.

Навчання моделі для фіксованого числа епох (ітерацій в наборі даних).

```
fit(
        x=None,
        y=None,
        batch_size=None,
        epochs=1,
        verbose=1,
        callbacks=None,
        validation_split=0.0,
        validation_data=None,
        shuffle=True,
        class_weight=None,
        sample_weight=None,
        initial_epoch=0,
        steps_per_epoch=None,
        validation_steps=None,
        validation_freq=1,
        max_queue_size=10,
        workers=1,
        use_multiprocessing=False
)
```

Список аргументів:

– x: вхідні дані. Це може бути: Масив Numpy (або схожий на масив) або список масивів (в разі, якщо модель має кілька входів). Діктовое відображення (dict mapping) вхідних імен в відповідний масив / тензори, якщо модель має іменовані входи. Генератор або keras.utils. Sequence повернення (inputs, targets) або (inputs, targets, sample weights). None (default) - Ні (за замовчуванням) при подачі з тензорів, вбудованих в каркас (наприклад, тензори даних TensorFlow).

– y: цільові дані. Як і вхідні дані x, це можуть бути або масив (и) Numpy, тензор (и), вбудовані в платформу, список масивів Numpy (якщо модель має кілька

вихідних даних), або None (за замовчуванням), якщо вони надходять з тензорів, вбудованих в платформу (наприклад, TensorFlow) тензори даних). Якщо вихідним верствам в моделі присвоєні імена, ви також можете передати словник, що відображає вихідні імена в масиви Numpy. If хявляется генератором або keras.utils.Sequence екземпляром, у вказувати не слід (оскільки цілі будуть отримані з x).

– batch_size: ціле число або None. Кількість зразків на оновлення градієнта. Якщо batch_sizene вказано, за замовчуванням буде 32. Не вказуйте, batch_sizeеслі ваші дані представлені в вигляді символічних тензорів, генераторів або Sequenceекземпляров (так як вони генерують пакети).

– epochs: цілочисельні. Кількість епох для навчання моделі. Епоха - це ітерація по всьому xі у наданими даними. Зверніть увагу, що в поєднанні з initial_epoch, epochsследует розуміти як «кінцеву епоху». Модель не навчається для ряду ітерацій, заданих epochs, а просто до тих пір, поки не epochsбудет досягнута епоха індексу.

– verbose: Integer. 0, 1 або 2. Режим багатослів'я. 0 = тихий, 1 = індикатор виконання, 2 = один рядок за епоху.

– callbacks: список keras.callbacks.Callbackекземпляров. Список зворотних викликів, які застосовуються під час навчання і перевірки (якщо). Дивіться зворотні виклики.

– validation_split: з плаваючою точкою від 0 до 1. Частка систему адаптації, які будуть використовуватися в якості даних перевірки. Модель виділить цю частину навчальних даних, що не буде навчатися їм і буде оцінювати втрати і будь-які метрики моделі на цих даних в кінці кожної епохи. Дані перевірки вибираються з останніх вибірок xі упредоставленних даних перед перетасуванням. Цей аргумент не підтримується, коли він хявляется генератором або Sequenceекземпляром.

– validation_data: дані для оцінки втрат і будь-які метрики моделі в кінці кожної епохи. Модель не буде навчатися на цих даних. validation_dataперекроет

validation_split. validation_dataможет бути: - кортеж (x_val, y_val) масивів або тензорів (x_val, y_val, val_sample_weights) Numpy - кортеж масивів Numpy - набір даних або итератор набору даних. Для перших двох випадків, batch_sizeдолжни бути надані. Для останнього випадку, validation_stepsдолжни бути надані.

– shuffle: Boolean (чи слід перемішувати дані тренування перед кожною епохою) або str (для «партії»). «Пакетна» - це спеціальна опція для роботи з обмеженнями даних HDF5; він тасується шматками розміром з партію. Не має ефекту, коли steps_per_epochнет None.

– class_weight: необов'язковий словник, що відображає індекси класу (цілі числа) на значення ваги (з плаваючою комою), що використовується для зважування функції втрат (тільки під час навчання). Це може бути корисно для того, щоб сказати моделі «приділяти більше уваги» вибірках з недопредставлених класу.

– sample_weight: необов'язковий масив ваг Numpy для навчальних вибірок, що використовується для зважування функції втрат (тільки під час навчання). Ви можете або передати плоский (1D) масив Numpy такої ж довжини, що і вхідні вибірки (відображення ваг і вибірок 1: 1), або в разі тимчасових даних ви можете передати двовимірний масив з формою (samples, sequence_length), щоб застосувати різну вагу для кожного тимчасового кроку кожного зразка. В цьому випадку ви повинні обов'язково вказати sample_weight_mode = "temporal" в compile (). Цей аргумент не підтримується, коли хгенератор Sequenceекземпляр натомість надають sample_weights в якості третьої елемента x.

– initial_epoch: ціле число. Епоха, з якої починається тренування (корисно для відновлення попереднього тренувального заїзду).

– steps_per_epoch: ціле число або None. Загальна кількість кроків (партій зразків) до оголошення однієї епохи закінченою і початку наступної епохи. При навчанні з використанням вхідних тензорів, таких як тензори даних TensorFlow, значення за замовчуванням None рівно числу вибірок в вашому наборі даних,

поділеній на розмір пакета, або 1, якщо це неможливо визначити, або визначення не дало результатів.

– validation_steps: тільки релевантне, якщо steps_per_epoch вказано. Загальна кількість кроків (партій зразків) для перевірки перед зупинкою.

– validation_steps: релевантно, тільки якщо validation_datapредоставлено і є генератором. Загальна кількість кроків (партій зразків), які потрібно намалювати перед зупинкою при виконанні перевірки в кінці кожної епохи.

– validation_freq: доречно, тільки якщо надані дані перевірки. Ціле число або список / кортеж / набір. Якщо ціле число, вказує, скільки тренувальних епох має бути виконано до того, як буде виконано новий прогін перевірки, наприклад, validation_freq = 2выполняет перевірку кожні 2 доби. Якщо в списку, кортежі або наборі вказуються епохи, в яких потрібно виконувати перевірку.

– max_queue_size: ціле число. Використовується тільки для генератора або keras.utils.Sequence входу. Максимальний розмір черги генератора. Якщо не вказано, за max_queue_sizeумолчанію буде 10.

– workers: ціле число. Використовується тільки для генератора або keras.utils.Sequencевхода. Максимальна кількість процесів, які можуть прискорюватися при використанні потоків на основі процесів. Якщо не вказано, за workers замовчуванням буде 1. Якщо 0, буде запускати генератор в основному потоці.

– use_multiprocessing: Boolean. Використовується тільки для генератора або keras.utils.Sequencевхода. Якщо True, використовуйте процесні потоки. Якщо не вказано, за use_multiprocessingумолчанію False. Зверніть увагу, що, оскільки ця реалізація спирається на многопроцессорность, ви не повинні передавати невивантажуваного аргументи генератору, так як вони не можуть бути легко передані дочірнім процесам.

3.2 Дизайн та імплементація штучної нейронної мережі для фільтрації текстових наборів даних соціальних мереж.

Дані, якими я користувався, були надані відкритою базою даних Kaggle і складались із приблизно 1,8 мільйона коментарів на форумах, представлених користувачами. Кожен коментар був позначений з імовірністю змусити іншу людину відмовитися від розмови. Я визначив токсичний як будь-яку ймовірність більше 0,5.

Перш ніж мої дані були переведені у формат, який може використовуватися алгоритмом класифікації, я попередньо обробив текстові дані. Сюди входило:

Видалення небажаних символів.

```
def clean_text(df, text):
    """
    Cleans text by replacing unwanted characters with blanks
    Replaces @ signs with word at
    Makes all text lowercase
    """
    df[text]           =           df[text].str.replace(r'[^A-Za-z0-
9()!?@\s\'\`\*\"\_\n]', '')
    df[text] = df[text].str.replace(r'@', 'at')
    df[text] = df[text].str.lower()

    return df
```

Токенізація тексту.

Наразі ми будемо токенізувати 1 грам. Ми можемо легко повернутися назад і токенізувати N-грами, якщо це необхідно.

```
tokenizer = RegexpTokenizer(r'\w+\'*[a-zA-Z]+')
df['tokens'] = df['comment_text'].apply(tokenizer.tokenize)
df.head()
```

Видалення стоп-слів та скорочень.

```
stopwords = ["ain't", "aren't", "can't", "can't've", "'cause",
"could've", "couldn't", "couldn't've", "didn't", "doesn't", "don't",
"hadn't", "hadn't've", "hasn't", "haven't", "he'd", "he'd've",
"he'll", "he'll've", "he's", "how'd", "how'd'y", "how'll", "how's",
"it'd've", "it'll", "it'll've", "it's", "let's", "ma'am", "mayn't",
"mustn't've", "needn't", "needn't've", "o'clock", "oughtn't",
"she'll", "she'll've", "she's", "should've", "shouldn't",
"shouldn't've", "so've", "so's", "that'd", "that'd've", "that's",
"there'd", "there'd've", "there's", "they'd", "they'd've", "they'll",
"we'd've", "we'll", "we'll've", "we're", "we've", "weren't",
"what'll", "what'll've", "what're", "what's", "what've", "when's",
"when've", "where'd", "where's", "where've", "who'll", "who'll've",
"would've", "wouldn't", "wouldn't've", "y'all", "y'all'd",
"y'all'd've", "y'all're", "y'all've", "you'd", "you'd've", "you'll",
"you'll've"]
df['no_stopwords'] = df['tokens'].apply(lambda x: [(if item not in
stopwords)])
df.head()
```

Позначення частин мови.

Ми тегуємо частини мови за допомогою nltk, а потім перетворюємо мовні теги в wordnet теги, оскільки лематизація сумісна лише з wordnet. Ми досягаємо цього за допомогою функції перетворення мовних тегів у wordnet.

```
df['speech_tags'] = df['no_stopwords'].progress_apply(pos_tag)
def wordnet_pos(tag):
    from nltk.corpus import wordnet
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN

df['wordnet_pos'] = df['speech_tags'].progress_apply(lambda x:
                                [(word, wordnet_pos(pos_tag)) for
(word, pos_tag) in x])
df.head()
```

Лемматизація.

Ми лематизуємо, щоб злити такі слова, як "having" та "have", в одне слово. Це можна зробити за допомогою однієї функції застосування і лямбда-функції, і ми збережемо наш прогрес тут за допомогою соління.

```
lemmatizer = WordNetLemmatizer()
df['lemmatized'] = df['wordnet_pos'].progress_apply(lambda x:
                                        [lemmatizer.lemmatize(word,  tag)
for (word, tag) in x])
df.head()
```

Ми збираємось обчислити кількість слів у нашій лематизованій колонці та нашому словниковому запасі (унікальні слова). Крім того, може бути корисно знати, яке за довжиною наше найдовше речення.

```
words = [word for row in df['lemmatized'] for word in row]
vocab = list(set(words))
sentence_lengths = [len(sentence) for sentence in df['lemmatized']]
print('Total Words: ', len(words))
print('Unique Words: ', len(vocab))
print('Longest Sentence: ', max(sentence_lengths), 'words')
```

Total Words: 42157140

Unique Words: 588800

Longest Sentence: 304 words

Для створення нашого minimal valuable product ми зробимо наступне, складемо результати та розглянемо найважливіші слова для кожної категорії. Ми будемо використовувати перехресну перевірку на різних показниках, щоб виміряти нашу ефективність

3.2.1 Вбудовування (CountVectorizer)

CountVectorizer використовує One-Hot Enconding, щоб створити набір вбудованих слів для всіх слів у вашому тексті. Він вбудував параметри попередньої обробки, але ми вже зробили попередню обробку, тому ми збираємось викликати

фіктивну функцію в нашій функції CountVectorizer, щоб пропустити її самоініціацію попередньої обробки.

```
X_train_cv, cv = count_vectorize(X_train)
X_val_cv = cv.transform(X_val)
y_train_cv, cv = count_vectorize(y_train)
y_val_cv = cv.transform(y_val)
```

Опис функції count_vectorize виглядає наступним чином

```
def count_vectorize(text_tokens):
    """
    Gives text data in tokenized form to create bag of words
    with CountVectorizer.  Returns bag of words and vectorizer
    fitted to tokened data
    """
    from sklearn.feature_extraction.text import CountVectorizer

    def dummy(doc):
        """
        Used since text data is already tokenized and preprocessed
        """
        return doc

    cv = CountVectorizer(tokenizer=dummy, preprocessor=dummy,
token_pattern=None)

    bag = cv.fit_transform(text_tokens)

    return bag, cv
```

3.2.2 Зменшення розмірності (LSA)

Ми використовуємо LSA для векторизованих даних. LSA виконує SVD і є еквівалентом PCA для текстових даних. Ми використаємо функцію для виконання цього завдання для нас. Ми побудуємо наші дані в 2-х вимірах, щоб перевірити, чи є розділення між нашими класами.

```
plot_lsa(X_train_cv, y_train)
```

Опис функції plot_lsa виглядає наступним чином.

```
def plot_lsa(vectorized_words, class_labels):
    """
    Plots embedded words in two dimensions with color based on class
    """
    from sklearn.decomposition import TruncatedSVD

    # We are using 2 dimensions for visualization
    lsa = TruncatedSVD(n_components=2)
    lsa.fit(vectorized_words)
    lsa_scores = lsa.transform(vectorized_words)

    plt.figure(figsize=(8, 6))
    plt.scatter(lsa_scores[:,0],        lsa_scores[:,1],        s=4,
c=class_labels, alpha=0.5)

    plt.show()
```

Всі наші пункти відносно близькі, і наші класи не дуже добре розділені. Представлення відмінності в класифікації даних представлено на рисунку 3.1.

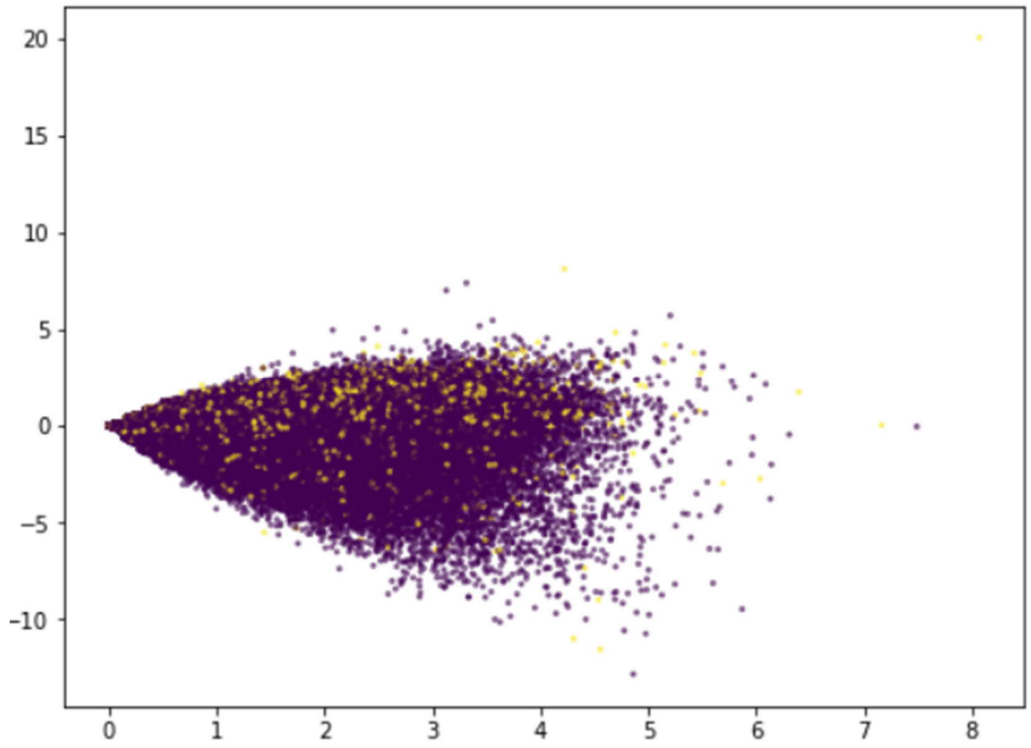

Рисунок 3.1 – Графічне представлення відмнності в класифікації даних

3.2.3 Логістична регресія

Почнемо з простої логістичної моделі регресії, оскільки її обчислювальна вартість низька, і її порівняно легко пояснити. Ми будемо судити про нашу модель за принаками акуратність, F1_Score, точність, повторення.

```
evaluate_model(y_val, clf.predict(X_val_cv))
```

Опис функції evaluate_model виглядає наступним чином.

```
def evaluate_model(y_true, y_predicted):
    accuracy = accuracy_score(y_true, y_predicted)
    precision = precision_score(y_true, y_predicted)
    recall = recall_score(y_true, y_predicted)
    f1 = f1_score(y_true, y_predicted)

    print('Accuracy: ', accuracy)
    print('Precision: ', precision)
    print('Recall: ', recall)
    print('f1_score: ', f1)
```

На основі отриманих даних, можемо представити графік моделі логістичної регресії на рисунку 3.2.

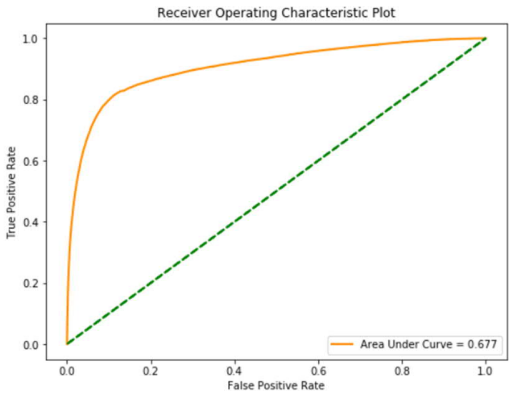

Рисунок 3.2 – Представлення графіку моделі логістичної регресії

Точність точності нашої моделі досить гарна, але оскільки ми маємо дисбаланс класів, це може бути не найкращим показником.

3.2.4. Інспекція. Огляд результатів опрацювання.

Тепер давайте розглянемо confusion матрицю та значення функцій, щоб краще зрозуміти нашу модель. Ми збираємося скласти найважливіші слова, які наша модель використовує для класифікації позитивного класу, щоб побачити, чи приймає наша модель інформативні рішення.

```
make_confusion_matrix(y_val, clf.predict(X_val_cv))
```

Опис функції make_confusion_matrix виглядає наступним чином.

```
def make_confusion_matrix(y_true, y_predicted):

    confusion = confusion_matrix(y_true, y_predicted)
    plt.figure(dpi=80)
    sns.heatmap(confusion, cmap=plt.cm.winter, annot=True,
square=True, fmt='d',
           xticklabels=['Irrelavant', 'Toxic'],
           yticklabels=['Irrelavant', 'Toxic']);
    plt.xlabel('Prediction')
    plt.ylabel('Actual')
    plt.title('Confusion Matrix')
    print(confusion)
```

На основі отриманих даних можемо зобразити представлення confusion матриці на рисунок 3.3.

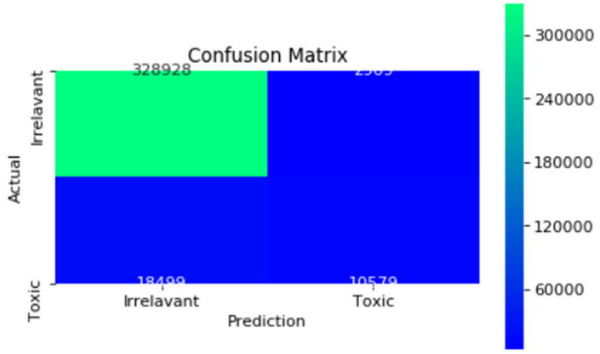

Рисунок 3.3 – Графічне представлення confusion матриці

Визначимо набір найбільш важливих слів, які використовуються для класифікації класу. Визначемо їхній коефіцієнт.

```
most_important_words(cv, clf, 15)
```

Опис функції most_important_words виглядає наступним чином.

```
def most_important_words(vectorizer, model, number):
    word_indices = {word:index for index,word in
vectorizer.vocabulary_.items()}

    word_importance = [[importance, word_indices[index]] for index,
                       importance in enumerate(model.coef_[0])]
    most_important_words = sorted(word_importance, key = lambda x:
x[0], reverse=True)

    values = np.array(most_important_words)[:number, 0]
    words = np.array(most_important_words)[:number, 1]

    values_list = []
    for value in values:
        values_list.append(value)
    values_list.reverse()
    words_list = []
    for word in words:
        words_list.append(word)
    words_list.reverse()

    plt.figure(figsize=(8,6))
    plt.barh(words_list, values_list, align='center', alpha=0.5)
    plt.title('Important Words')
    plt.xticks(rotation=65)
    plt.show()
```

В результаті ми отримали 15 слів, які наша модель використовує найбільше для класифікації токсичних коментарів. Більшість із цих слів мають сенс. Крім того, всі ваги, які вона надає словам, відносно низькі, тому наша модель може не підбирати, які слова гірші за інші. Графік найважливіших слів із зазначиними коефіцієнтами токсичності наведено на рисунку 3.4.

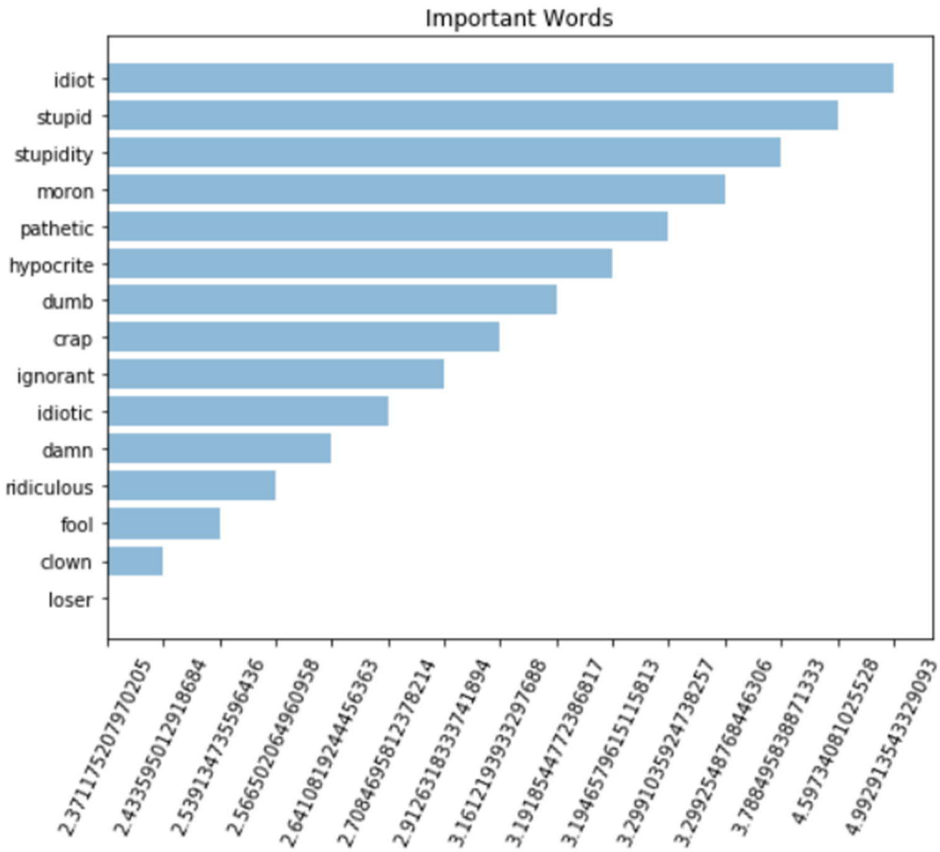

Рисунок 3.4 – Графічне представлення моделі найважливіших слів з зазначеними коефіцієнтами токсичності

Моя найкраща модель була отримана завдяки глибокому навчанню. Я використовував двонаправлений LSTM із вбудовуванням Word2Vec. Це означає, що я вклав усі свої слова у словник Google Word2Vec за допомогою попередньо навчених векторів слів, а решта дозволив моїй моделі навчитися. У моєму словнику було 200 мільйонів слів, які були в Google Word2Vec, що залишило моїй моделі лише 200 000 вкладених матеріалів для навчання. Використання попередньо навчених векторів слів значно скорочує обчислювальний час та значно покращує продуктивність.

Двонаправлена довготривала короткострокова пам’ять (LSTM) - це архітектура періодичної нейронної мережі (RNN) з додатковою складністю навколо стратегії оновлення пам’яті, що дозволяє нам фіксувати просторові зразки

вживання слів, які надзвичайно важливі для спілкування людей. Обробляючи текст як послідовність слів та обробляючи ці слова явно послідовно, ми можемо набути багато прогнозуючої сили в наших моделях. Архітектура періодичної нейронної мережі наведена на рисунку 3.5.

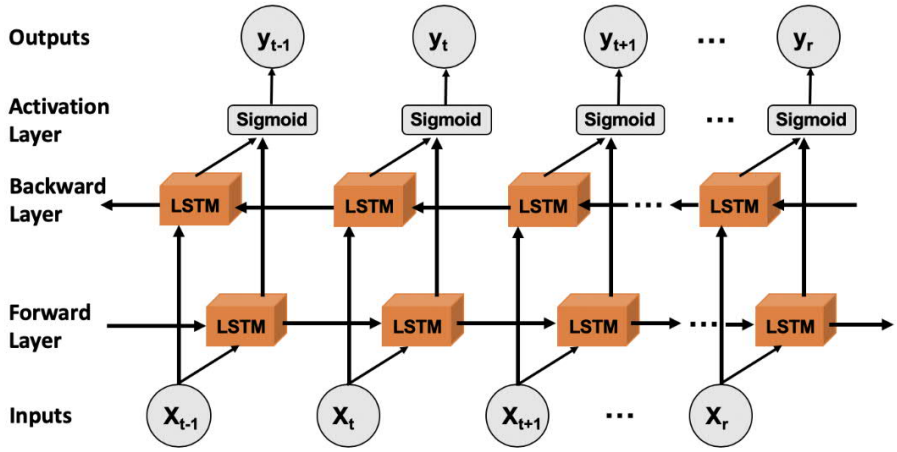

Рисунок 3.5 – Графічне представлення архітектури періодичної нейронної мережі

Двонаправлений просто означає, що ми обробляємо текст спереду назад і назад спереду, що дозволяє нам охопити багатий набір контексту.

3.3. Тестування штучної мережі для фільтрації текстових наборів даних соціальних мереж.

Єдина проблема використання глибокого навчання полягає в тому, що ми втрачаємо пояснюваність у наших моделях. Я не зміг зрозуміти важливість особливостей, щоб перевірити, чи приймає моя модель обґрунтовані рішення.

Натомість я зайшов на форуми з коментарями і попросив свою модель передбачити коментарі в реальному часі, щоб я їх спостерігав.

Тож повертаючись до публікації Коліна Каперніка з попередньої моєї моделі, ми бачимо, що токсичний коментар позначений токсичністю 99,5%. Тоді як коментар “Top Fan” має токсичність 0,2%. Приклад роботи фільтра наведено на рисунку 3.6.

Рисунок 3.6 – Представлення роботи фільтра на зразках коментарів

Повертаючись до початкової проблеми, як можна використовувати цю модель для підвищення ввічливості в наших онлайн-розмовах? Ну, це залежить від платформи. Наприклад, в іграх в Інтернеті ви можете виявити токсичних споживачів, а потім застосувати певні обмеження на основі кількості правопорушень або тяжкості правопорушень.

Однак для таких платформ соціальних медіа, як Facebook та Twitter, можливо, ви не захочете обмежувати свободу слова людей. Якщо ми розмістимо відстежувачі токсичності в режимі реального часу над коментарями, як люди, вони можуть бути більш самосвідомими того, що вони говорять, і, сподіваємось, вдруге здогадаються про токсичний коментар, який вони збиралися розмістити.

Інструмент відстеження токсичності потрібно буде розміщувати лише на тих користувачах, котрі в історії мали грубість в Інтернеті. Подібно до ігор, ми можемо

використовувати ШІ для виявлення цих користувачів, а потім використовувати на них відстежувач токсичності. Було б цікаво провести тематичне дослідження, щоб побачити, чи користувачі, у яких в минулому була токсичність, змінюють свою поведінку, беручи участь в онлайн-розмовах з увімкненим режимом.

Для повноцінного тестування фільтрації коментарів, використовуємо обрану нами базу даних Kaggle, яка складались із приблизно 1,8 мільйона коментарів на форумах, представлених користувачами. Кожен коментар був позначений з імовірністю змусити іншу людину відмовитися від розмови. Я визначив токсичний як будь-яку ймовірність більше 0,5. Представлення фрагменту датасету коментарів зображено на рисунку 3.7.

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
1	0000997932d777bf	Explanation Why the edits made under my username Ha...	0	0	0	0	0	0
2	000103f0d9cfb60f	D'aww! He matches this background colour I'm seeming...	0	0	0	0	0	0
3	000113f07ec002fd	Hey man, I'm really not trying to edit war. It's just that th...	0	0	0	0	0	0
4	0001b41b1c6bb37e	" More I can't make any real suggestions on improvemen...	0	0	0	0	0	0
5	0001d958c54c6e35	You, sir, are my hero. Any chance you remember what pa...	0	0	0	0	0	0
6	00025465d4725e87	" Congratulations from me as well, use the tools well. · t...	0	0	0	0	0	0
7	0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0
8	00031b1e95af7921	Your vandalism to the Matt Shirvington article has been ...	0	0	0	0	0	0
9	00037261f536c51d	Sorry if the word 'nonsense' was offensive to you. Anyw...	0	0	0	0	0	0
10	00040093b2687caa	alignment on this subject and which are contrary to tho...	0	0	0	0	0	0
11	0005300084f90edc	" Fair use rationale for Image:Wonju.jpg Thanks for uplo...	0	0	0	0	0	0
12	00054a5e18b50dd4	bbq be a man and lets discuss it-maybe over the phone?	0	0	0	0	0	0
13	0005c987bdfc9d4b	Hey... what is it.. @ | talk . What is it... an exclusive grou...	1	0	0	0	0	0
14	0006f16e4e9f292e	Before you start throwing accusations and warnings at ...	0	0	0	0	0	0
15	00070ef96486d6f9	Oh, and the girl above started her arguments with me. S...	0	0	0	0	0	0

Рисунок 3.7 – Представлення фрагменту датасету коментарів

Ці 6 змінних є нашими цільовими змінними і мають двійкове значення. Підбір моделей для прогнозування 6 змінних одночасно буде використано в ході тестування.

Опрацювання датасету з коментарями надає нам рейтинг слів з найбільшими коефіцієнтами токсичності. При аналізу коментаря, дані слова сигналізують систему, що збільшується ймовірність закладеного негадивного змісту, що суттєво

впливає на вирахування коефіцієнту. Також коефіцієнт токсичності суттєво відрізняється відповідно до конксту, в якому його було використано.

В такому випадку цитати або ж слова які на пряму не відносяться до адресата матимуть меншу токсичність. Рейтинг слів з найбільшими коефіцієнтами токсичності наведено на рисунку 3.8.

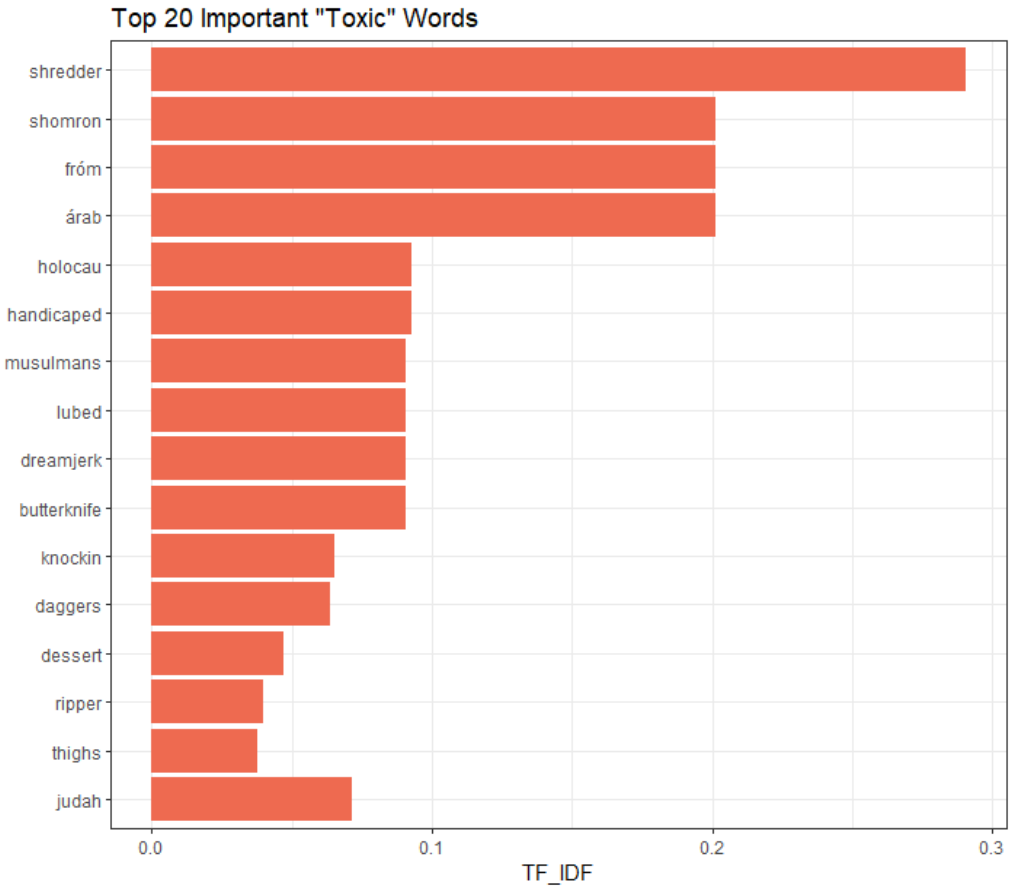

Рисунок 3.8 – Представлення рейтингу слів з найбільшими коефіцієнтами токсичності

На основі датасету та аналізу рейтингу слів з коефіцієнтами токсичності проводимо подальше опрацювання. Даний підхід дозволив отримати результуючу таблицю, яка представлена на рисунку 3.9. Відповідно до конкретного ідентифікатора текстового повідомлення, дізнаємося коефіцієнти та значення

відношення до класифікаторів: toxic, severe_toxic, obscene, threat, insult, identity_hate.

Подальше адміністрування може визначити набір порогових значень коефіцієнтів, які можуть допускаються до публікації. Результат опрацювання фільтру на вхідному наборі тестового датасету наведено на рисунку 3.9.

	id	toxic	severe_toxic	obscene	threat	insult	identity_hate	word	n	document	tf	idf	tf_idf
1	40a358e41dcc21f4	1	1	0	1	1	0	shredder	9	34	0.111111111	2.6149598	0.290551086
2	cf012f8122791d7e	1	1	0	1	0	1	shomron	30	33	0.033333333	3.0204249	0.100680830
3	cf012f8122791d7e	1	1	0	1	0	1	fróm	30	33	0.033333333	3.0204249	0.100680830
4	cf012f8122791d7e	1	1	0	1	0	1	árab	30	33	0.033333333	3.0204249	0.100680830
5	cf012f8122791d7e	1	1	0	1	0	1	shomron	30	33	0.033333333	3.0204249	0.100680830
6	cf012f8122791d7e	1	1	0	1	0	1	fróm	30	33	0.033333333	3.0204249	0.100680830

Рисунок 3.9 – Представлення результату опрацювання фільтру на вхідному наборі тестового датасету

Консольне представлення роботи фільтра наведено на рисунку 3.10.

```
Comment: What on earth does anti - semitism have to do with the
notion that the US military shot down the plane, Cberlet? I am
quite mystified.

toxic : 0.055252425
severe_toxic : 0.002406106
obscene : 0.04549172
threat : 0.0027631551
insult : 0.024192393
identity_hate : 0.0069899107
---------------------------------
Comment: WikiAwards Hi there. Just to thank you for being the 2
nd participant in the WikiAwards Project. Unfortunately the pro
ject is causing some controversy and I need to know if it is wo
rth going on.

toxic : 0.0039544455
severe_toxic : 9.456437e-05
obscene : 0.009531345
threat : 0.00067216606
insult : 0.0016024308
identity_hate : 0.0007145586
---------------------------------
Comment: I'm afraid not. I think you need to ask somebody who's
an administrator on Commons. ( I'm an admin on en. wiki, not on
Commons. )

toxic : 0.017199917
severe_toxic : 0.0008454597
obscene : 0.025758682
threat : 0.002444282
insult : 0.008249491
identity_hate : 0.0033726753
---------------------------------┼
```

Рисунок 3.10 – Консольне представлення роботи фільтру

Фінальна діграма роботи системи, з описаними взаємодіями компонентів, представлена на рисунку 3.11.

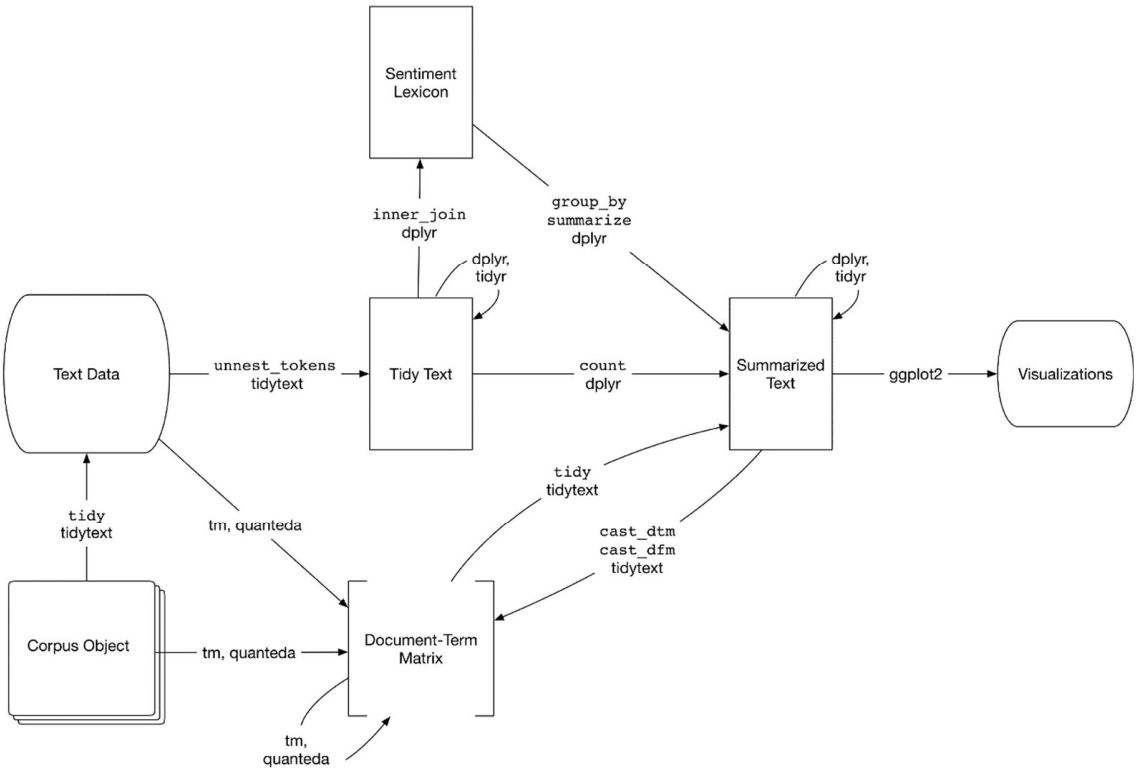

Рисунок 3.11 – Діаграма роботи системи фільтрації

Спочатку ми пройшли кроки в середині картини. А потім ми також спробували різні кроки внизу. За допомогою пакету tm ми створили корпус, відлили їх у dtm та підігнали модель. Нарешті, ми дозволили детектору класифікувати коментарі з токисчність, яка перевищує допустимий поріг.

ВИСНОВКИ

Результатом виконання магістерської роботи є програмне забезпечення, яке дозволяє визначати тональтість текстової публікації та фільтрувати відповідно до певного порогового значення. Фільтр такого зразку надає адміністрації можливість слідкувати за емоційною складовою комунікації, запобігаючи розпалу конфлікту та базі протиріч переданих у грубій формі.

Використовуючи засоби NLP на основі нейронних мереж, було розроблено алгоритм визначення токсичності повідомлення для класифікації текстового вмісту. Кожне повідомлення опрацьовується та маркується відповідним коефіцієнтом токсичності.

Використання підходу до опрацювання та фільтрації текстових наборів даних перед публікацією, зумовлює дотримання користувачами норм політкоректонсті у спілкуванні. Таким чином, запобігаючи безпідставного цькування та приниження гідності особи адресата.

Рекомендація для подальших досліджень полягає у покращенні підходу до тренування нейронної мережі. Зменшуючи час та необхідні ресурси для навчання, та збільшуючи якість результату.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кулиняк Р.В., Каштелян І.В. Алгоритм визнаачення «токсичності» повідомлення з використанням засобів NLP. III Науково-практична конференція молодих вчених і студентів «Інтелектуальні комп'ютерні системи та мережі». 26 листопада 2020, Тернопіль, Україна. Тернопіль: ЗУНУ, 2020. с.23

2. Каштелян І.В., Кулиняк Р.В. Використання систем штучного інтелекту для покращення засобів планування задач. III Науково-практична конференція молодих вчених і студентів «Інтелектуальні комп'ютерні системи та мережі». 26 листопада 2020, Тернопіль, Україна. Тернопіль: ЗУНУ, 2020. с.13

3. Алексєєв А. Алгоритм розпізнавання символів на основі структурного підходу / А. Алексєєв, В. Заяць, Д. Іванов // Вісник Нац. ун-ту „Львівська політехніка‖ „Комп'ютерна інженерія та інформаційні технології". – 2002. – № 468. – С.129 - 133.

4. Горелик А.Л. Методы распознавания / А.Л. Горелик, В.А. Скрипник. – М. : Высшая школа, 1989. – 232 с.

5. Фукунага К. Введение в статистическую теорию распознавания/ К.Фукунага. – М.: Наука, 1979. – 512 С.

6. Дуда Р. Распознавание образов и анализ сцен / Р. Дуда, П. Харт. – М. : Мир, 1976. – 512с.

7. Математические основы нейронных сетей [Електронний ресурс] // Режим доступу: http://5ka.ru/67/26159/1.html

8. Калан Роберт Основные концепции нейронных сетей / Роберт Каллан; – М. : Вильямс, 2001. – 288 с.

9. Хайкин Саймон Нейронные сети: полный курс / Саймон Хайкин; – М.: Вильямс, 2006. – 1104 с.

10. Заяць В.М. Архітектура подіє – орієнтованих систем на прикладі системи розпізнавання рукописного тексту / В.М. Заяць, Д.О. Іванов // Вісник Нац. ун-ту „Львівська політехніка‖ „Комп‘ютерна інженерія та інформаційні технології". – 2004. – № 530. – С. 78 - 83.

11. Ротштейн А.П. Интеллектуальные технологии идентификации: нечеткие множества, генетические алгоритмы, нейронные сети. – Винница: «УНІВЕРСУМ-Вінниця», 1999

12. Perelson A. S. Immunology for physicists / A. S. Perelson, G. Weisbuch // Review of Modern Physics. — 1997. — Т. 69, № 4. — С. 1219—1267.

13. Хайкин С. Нейронные сети –М. : Издательский дом «Вильямс»,2006 – 1104 с.

14. Bishop C. M. Neural network for pattern recognition–New York: Oxford University Press, 1995 –482 pp.

15. BroomnheadD.S. and LoweD.Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Nctworks –London, 1988 URL: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA196234

16. Fausett L. Fundamentals of neural network –New Jersey: Prentice-Hall, 1994–XVII pp. +492 pp.

17. G. Thimm, E.FieslerHigh Order and Multilayer Perceptron Initialization//IEEE Transactions on Neural Networks. 1996, No94-07

18. Hagan M. T., Demuth H. B., Beale M.Neural network design –USA: PWS Publishing Company, 1996 –733 pp.

19. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning –Springer Verlag, 2009–764 pp.

20. N. H. Beltran, M. A. Duarte-Mermoud, M. A. Bustos, S. A. Salah, E. A. Loyola, A. I. Pena-Neira, J. W. Jalocha. Feature extraction and classification of Chilean wines // Journal of Food Engineering. 2006, Т. 75 вип. 1, с. 1–10

21. PauloCortez, AntonioCerdeira, FernandoAlmeida, TelmoMatos, JoseReis. Modeling wine preferences by data mining from physicochemical properties– Portugal, University ofMinho, 2008

22. RojasR. Neural Networks.A Systematic Introduction –Berlin: Springer,1996–509 pp.

23. S.R. Gunn.Support Vector Machines for Classification and Regression – University of Southampton, 1998

24. Vlassides, J. Ferrier, and D. Block. Using Historical Data for Bioprocess Optimization: Modeling Wine Characteristics Using Artificial Neural Networks and Archived Process Information//Biotechnology and Bioengineering. 2001, No73(1), c. 55-68

25. Tim JonesM.Al Application Programming –Massachusetts: CHARLES RIVER MEDIA, INC, 2003 –311pp.

26. Yann LeCun, Leon Bottou, Genevieve B. Orr, Klaus-Robert Muller.Efficient BackProp. URL: http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf

27. Cohen I. R. The cognitive paradigm and the immunological homunculus / I. R. Cohen // Immunol Today. — 1992. — V. 13, № 12. — P. 490—494.

28. Bersini H. The immune recruitment mechanism: A selective evolutionary strategy : праці міжн. наук. конф. Fourth international conference on genetic algorithms, 13-16 лип. 1991, San Diego. — С. 520—526.

29. Jerne N. K. Towards a network theory of the immune system / N. K. Jerne // Annales d'immunologie. — 1974. — V. 125, № 1—2. — P. 373—389.

30. Hunt J. E. An adaptive, distributed learning system, based on the immune system : праці. наук. конф IEEE international conference on systems, man and cybernetics, 22—25 жовт. 1995. — V. 3. — P. 2494—2499.

31. Hunt J. E. Learning using an artificial immune system / John E. Hunt, Denise E. Cooke // Journal of Network and Computer Applications. — 1996. — V. 19, № 2. — P. 189—212.

32. Gilbert C. J. Associative memory in an immune-based system : праці наук. конф. 12th national conference on artificial intelligence (AAAI-94), 31 лип. — 4 квіт. 1994, Seattle. — С. 852—857. — ISBN 0-262-61102-3.

33. Faraoun K. M. Artificial Immune Systems for text-dependent speaker recognition / K. M. Faraoun, A. Boukelif // INFOCOMP — Journal of Computer Science. — 2006. — V. 5, № 4. — P. 19—26.

34. Астафьева Н. М. Вейвлет-анализ: основы теории и примеры применения / Н. М. Астафьева // Успехи физических наук. — 1996. — Т. 166, № 11. — С. 1145—1170.

35. Рабинер Л. Теория и применение цифровой обработки сигналов / Л. Рабинер, Б. Гоулд. — М. : Мир, 1978. — 848 с.

36. Walker J. Fourier Analysis and Wavelet Analysis / James S. Walker // Notices of the AMS. — 1997. — V. 44, № 6. — P. 658—670.

37. Дремин И. М. Вейвлеты и их использование / И. М. Дремин, О. В. Иванов, В. А. Нечитайло // Успехи физических наук. — 2002. — Т. 171, № 5. — С. 465—501.

38. Фант Г. Акустическая теория речеобразования / Гуннар Фант. — М. : Наука, 1964. — 284 с.

39. Ермоленко Т. В. Применение вейвлет-преобразования для обработки и распознавания речевых сигналов / Т. В. Ермоленко // Искусственный интеллект. — 2002. — № 4. — С. 200—208. — ISSN 1561-5367.

40. Schwenk H. Neural Network Language Models for Conversational Speech Recognition : праці. міжн. наук. конф. International Conference on Speech and Language, 4—8 жовт. 2004, Чеджу, Корея. — С. 1215—1218.

41. Schwenk H. Training Neural Network Language Models On Very Large Corpora : праці наук. конф. Joint Conference HLT/EMNLP, 6—8 жовт. 2005, Ванкувер, Канада. — С. 201—208.

42. Moonasar V. Speaker Identification using a Combination of Different Parameters as Feature Inputs to an Artificial Neural Network Classifier : праці наук. конф. AFRICON, 28 вер. — 1 жовт. 1999, Дурбан, ПАР. — Т. 1. — С. 189—194.

43. Васенко, Д. В. Методи навчання штучних нейронних мереж. Інформатизація освіти, 2007. - С. 20-29.

44. Васильєв, В. І. розпізнати системи. Довідник. 2-е видання - Київ: Наукова думка, 1983. - 424 с.

45. Воронцов, К. В. Комбінаторні оцінки якості навчання по прецедентах // Доповіді РАН. - 2004. - Т. 394. - №2. - С. 175-178.

46. Заєнцев, І. В. Нейронні мережі. Основні моделі / І. В. Заєнцев. - Воронеж: ВДУ, 1999. - 76 с.

47. Комарцова, Л. Г. Нейрокомп'ютери: навчальний посібник / Л. Г. Комарцова, А. В. Максимов. - М .: МГТУ, 2002. - 318 с.

48. Корн, Г. Довідник з математики для науковців та інженерів / Г. Корн, Т. Корн. - М .: Наука, 1970. - 246 с.

49. Березький О.М., Дубчак Л.О., Мельник Г.М. Методичні рекомендації до виконання кваліфікаційної роботи з освітнього ступеня “Магістр”. Спеціальність: 123 - Комп’ютерна інженерія. Магістерська програма - Комп’ютерна інженерія"/ Під ред. О.М. Березького. Тернопіль:ЗУНУ,2020.32 с.

50. Гураль І.В., Дубчак Л.О. Методичні вказівки до оформлення курсових проектів, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності «Комп’ютерна інженерія»/Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 33 с.