Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Програмної інженерії
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти – другий (магістерський)

Дослідження методів аналізу емоційності документів про суб'єкта
(тема)

Виконав: студент 2 курсу, групи ІПЗм-18-1

Гладуш І.В.
(прізвище, ініціали)

спеціальності 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Освітньо-наукової програми
(тип програми)

Інженерія програмного забезпечення
(повна назва освітньої програми)

Керівник к.т.н, доц. Турута О.П._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

2020 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет_Комп'ютерних наук_____
Кафедра_Програмної інженерії_____
Рівень вищої освіти – другий (магістерський)

Спеціальність _121 – Інженерія програмного забезпечення_____ (код і повна назва)
Тип програми освітньо-наукова програма_____
Освітня програма_Інженерія програмного забезпечення_____

ЗАТВЕРДЖУЮ:

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

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові _Гладушу Івану Валерійовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи _Дослідження методів аналізу емоційності документів про суб'єкта затверджена наказом університету від "___" _____20 ___ р. № _____ заповнюється вручну після отримання наказу
2. Термін подання студентом роботи до екзаменаційної комісії
14 травня 2020 р.
3. Вихідні дані до роботи розпізнавання іменованих сутностей, аналіз емоційності тексту, пояснювальна записка_____
4. Перелік питань, що потрібно опрацювати в роботі мета роботи, аналіз проблемної галузі і постановка задачі, огляд підходів до створення програмного забезпечення для розпізнавання іменованих сутностей, методи створення алгоритмів для оцінки емоційності тестів, порівняння знайдених методів, вибір оптимальних матодів._____

5 Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецрозділ	к.т.н., доц. Турута О.П.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка *
1	Аналіз предметної галузі	01.03.2020	
2	Методи та алгоритми розпізнавання іменованих сутностей і аналізу емоційності текстів	12.03.2020	
3	Прототипування	20.03.2020	
4	Підготовка пояснювальної записки	26.03.2020	
5	Спецчастина	07.04.2020	
6	Підготовка презентації та доповіді	21.04.2020	
7	Попередній захист	28.04.2020	
8	Нормоконтроль, рецензування	29.04.2020	
9	Занесення диплома в електронний архів	05.05.2020	
10	Допуск до захисту у зав. кафедри	12.05.2020	
* заповнюється вручну після виконання чергового пункту			

Дата видачі завдання _____ _____ 2019 р.

Студент _____
(підпис)

Керівник роботи _____ __к.т.н., доц. Турута О.П._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Атестаційна робота магістра містить: 75 с., 18 рис., 3 табл., 3 форм., 47 джер., 3 дод.

МЕТОДИ РОЗПІЗНАВАННЯ ІМЕНОВАНИХ СУТНОСТЕЙ, ЕМОЦІЙНІСТЬ ДОКУМЕНТА, НЕЙРОННА МЕРЕЖА, ПРОТОТИПУВАННЯ

Метою магістерської атестаційної роботи є дослідження методів аналізу емоційності документів про суб'єкта задля вирішення задачі розпізнавання іменованих сутностей для текстів, написаних українською мовою.

У процесі виконання магістерської атестаційної роботи було проведено аналіз предметної галузі, виділені і виявлені найбільш актуальні методи і алгоритми для розпізнавання іменованих сутностей і оцінки емоційності текстів, знайдені методі оцінки якості алгоритмів, зроблена реалізація та адаптація знайдених методів, а також їх перевірка.

METHODS OF RECOGNITION NAMED ESSENCES, EMOTIONALITY OF DOCUMENT, NEURAL NETWORK, PROTOTYPING

The purpose of the master's diploma work is studying the methods of analyzing the documents emotionality about the subject to solve this problem for texts in the Ukrainian language.

During the process of master's diploma work creation the subject area was analyzed, the most relevant methods and algorithms for recognizing named entities and assessing the emotionality of texts were highlighted and identified, methods for assessing the quality of algorithms were found, implementation, adaptation and verification of founded methods were made.

ЗМІСТ

ВСТУП

З ростом впливу цифрової інформації на наше життя у людей з’явилася можливість отримувати доступ до інформації онлайн, створювати інформацію на просторах інтернету у вигляді коментарів про товар, книгу або подію, а також шляхом оцифровування будь-якої друкованої інформації, такої як рішення судових органів, укази тощо. Зрозуміло, що з’явилася необхідність обробляти таку інформацію, збагачувати її такими даними, як головна особа в тексті, відношення автора до цього об’єкту. Через те, що кількість інформації зростає експоненційно, ми не можемо обробляти все за допомогою людських ресурсів, саме тому з’явився інтерес до автоматичного опрацювання даних. Це стало поштовхом для розвитку такої області комп’ютерних наук, як розпізнавання іменованих сутностей, а також сентиментальний аналіз тексту.

Розпізнавання іменованих сутностей (РІС) – це підзадача видобування інформації, яка намагається знайти і класифікувати іменовані сутності в неструктурованому тексті в заздалегідь визначені категорії, такі як імена людей, організації, місця, медичні коди, час, кількості, грошові значення, відсотки тощо.

Аналіз тональності тексту – клас методів контент-аналізу в комп’ютерній лінгвістиці, призначений для автоматизованого виявлення в текстах емоційно забарвленої лексики і емоційної оцінки авторів (думок) щодо об’єктів, мова про які йде в тексті.

В даній роботі будуть розглянуті та запропоновані рішення для розв’язання проблем аналізу сентименту і розпізнавання іменованих сутностей текстів українською мовою.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Актуальність систем розпізнавання іменованих сутностей

Кількість інформації у світі зростає з кожним днем. Незважаючи на велику кількість інструментів для структурування цієї інформації, більш ніж 80 відсотків інформації залишається не структурованою. Компанія IDC наводить таку статистику розподілу структурованих та неструктурованих даних:

Рисунок 1.1 – Розподіл структурованої та неструктурованої інформації в світі.

Дивлячись на статистику на рисунку 1 стає зрозуміло чому потрібно знаходити та розвивати методи структуризації інформації [1].

Ця тенденція мотивувала людей почати вивчати методи автоматичної обробки інформації природною мовою[2].

Одним з найважливіших методів структуризації інформації, який належить до класу задач обробки природної мови є розпізнавання іменованих сутностей (ріс). Задача РІС – виділити всі спани сутностей в тексті (спан – неперервний фрагмент

тексту) [3]. Наприклад, у нас є текст і в ньому потрібно знайти всі іменовані сутності деякого заздалегідь зафіксованого набору – імена, прізвища, локації, назви організацій, дати і т.д.)

Рисунок 1.2 – Приклад тексту з виділеними сутностями

Мета алгоритмів розпізнавання іменованих сутностей зрозуміти, що у тексті на рисунку 1.2 “20 years” це дата, а “Trump” це персона. Не важко здогадатися, що якщо ми добре навчимося виявляти в тесті персони, локації і організації, це навряд чи викликає великий інтерес у бізнесу. Хоча в класичній постановці задачі є практичне застосування. Один зі сценаріїв, коли дані алгоритми необхідні, це структуризація неструктурованих даних. Припустимо, що у вас є якийсь текст або набір текстів і дані з нього потрібно ввести в базу даних (таблицю). Імена, прізвища, можуть бути значеннями деяких стовпчиків. Відповідно, щоб правильно заповнити таблиці, перед їх заповненням потрібно виділити в тексті ті дані, які будемо в неї вносити і саме для знаходження цієї інформації використовуються алгоритми РІС. Завдяки тому, що розпізнавання іменованих сутностей є кроком в

напрямку розуміння тексту, це може мати як самостійну цінність так і допомагати розв’язувати інші задачі. Якщо ми знаємо, де саме в тексті знаходяться сутності, то можемо легко знайти важливий фрагмент тексту. Наприклад, можемо виділити тільки ті абзаци, де зустрічаються сутності деякого певного типу, а потім працювати тільки з ними. Уявімо, що вам прийшов лист і було б добре зробити сніпит тільки тієї частини, де є щось корисне, просто привітання чи викладення непотрібної інформації. Якщо вміти виділяти іменовані сутності, можна зробити сніпит більш розумним, ніж просто показ першого речення з тексту, як це часто робиться або підсвітити потрібні частини листа для пришвидшення роботи з ним[4]. Крім того, сутності можуть бути послідовністю слів, які потрібно обробляти разом. Наприклад, переводити назву сутності в назву сутності, ми хочемо перекласти назву “Магазин АТБ” на італійську мову єдиним реченням, а не дробити на декілька незв’язних частин. Без розв’язання задачі розпізнавання іменованих сутностей важко уявити собі вирішення багатьох задач обробки природної мови. Наприклад, ми бажаємо проаналізувати текст “Хоробрий лицар Айвенго прискакав на білому коні. Королева вибігала до нього на зустріч і обійняла його”. Якщо ми виділимо слово “Айвенго” як іменовану сутність, то машині буде набагато простіше зрозуміти, що королева обійняла його, а не коня. Також вирішення задачі розпізнавання іменованої сутносі є корисною для актуальної проблеми кластеризації неструктурованих документів[5].

Тепер наведемо приклад, як виділення іменованих сутностей може допомогти при побудові питально-відповідних систем. Якщо задати в пошуковій системі питання “Хто грав головну роль в фільмі Гатака?”, то з великою долею ймовірності ви отримаєте вірну відповідь. Для надання відповіді використовується РІС: виділяються сутності (фільм, роль і т.д.), розуміємо про що нас питають і далі шукаємо відповідь у базі даних. Напевне, це найважніше міркування, завдяки якому алгоритми РІС на стільки популярні: постановка задачі дуже гнучка. Іншими словами ніхто не заставляє нас виділяти тільки імена, прізвища, дати або локації. Ми можемо виділяти будь-які потрібні нам неперервні фрагменти тексту, які відрізняються чимось від іншого тексту [6]. В результаті можна підібрати свій набір

сутностей для конкретної практичної задачі, яка стоїть перед нами, розмітити корпус текстів цим набором і навчити модель. Такий сценарій зустрічається дуже часто і це робить PIC однією із найчастіше вирішуваних задач в індустрії машинного навчання.

Наведемо декілька реальних кейсів. Перший приклад це судові справи. В Україні безліч відцифрованих судових рішень та справ, але для того, щоб зрозуміти хто був суддею, яке рішення було ухвалено, на основі яких статей доводиться читати все діло повністю. Використовуючи алгоритми розпізнавання іменованих сутностей ми можемо вирішити дану проблему і збагатити дані корисною інформацією, що призведе до пришвидшення опрацювання даних. Другий приклад полягає в тому, що в нас є набір грошових переказів. Кожен з них має текстовий опис у якому знаходиться корисна інформація (хто, куди, кому тощо). Даний текст досить структуризований. В банках є спеціальні люди, які читають ці тексти і заносять інформацію до бази даних. Ми можемо вибрати набір сутностей і навчити алгоритм автоматично виділяти їх. Результатом цієї оптимізації буде автоматизування та прискорення роботи переказів.

Третій кейс – це необхідність чат боту в онлайн магазині, щоб відповідати на питання людини. Наприклад, людина питає статус замовлення в довільній формі, бот знаходить необхідні сутності та видає результат. Або людина шукає якийсь товар (наприклад, рожева сукня), бот опрацьовує текст і допомагає людині знайти замовлення. Виходячи з наведених вище прикладів легко зрозуміти користь від алгоритмів розпізнавання іменованих сутностей [7].

1.2 Актуальність алгоритмів сентиментального аналізу текстів.

Однією з важливих характеристик тексту є його емоційне забарвлення. Для вивчення емоційного забарвлення є цілий набір алгоритмів для аналізу тональностей або аналізу сентименту тексту [8]. Sentiment analysis (аналіз

сентименту) – набір алгоритмів призначених для автоматизованого виявлення в текстах емоційно забарвленої лексики і емоційної оцінки авторів (думок) по відношенню до об’єктів, мова про які йде в тексті. Аналіз сентименту інформаційних потоків має великий потенціал використання для моніторингових, аналітичних, документообробчих алгоритмів та рекламних платформ.

Текст природньою мовою, крім інформації може виражати і емоційну оцінку того, про що у ньому йдеться. Наприклад, таке речення містить негативну оцінку події: “Жахлива погода сьогодні.”. Також речення може бути виключно позитивним: “SpaceX отримав дозвіл на запуск нового корабля”.

Виражена в тексті оцінка називається тональністю або сентиментом тексту. Людина оцінює світ одночасно за декількома шкалами (добрий-поганий, сильний-слабкий, великий-маленький, веселий-сумний, швидкий-повільний тощо). Ці шкали по різному емоційно навантажені, але для спрощення можна вважати, що емоційна оцінка зводиться до шкали добрий-поганий, позитивний-негативний. Історично склалося так, що традиційний підхід до аналізу сентименту становить собою задачу класифікації тексту або його частини на дві-три категорії (негативний, позитивний, нейтральний або тільки позитивний, негативний). Саме з такої задачі розпочинали свій розвиток аналіз тональностей: оцінити сентимент оціночних відгуків по будь-якій тематиці. Однако, це не єдиний і не визначальний тип задачі, яку повинен вирішити сентиментальний аналіз тексту. В сучасному світі всіх більше цікавить не загальна емоційна оцінка тексту, а відношення автора до конкретного об’єкту. Об’єкт, відносно якого виражається емоційна оцінка, прийнято називати об’єктом тональності. Носієм вираженої в тексті емоційної оцінки також є певна особа, зазвичай це автор тексту, однак, якщо автор тексту посилається на чиюсь думку або цитує вискакування іншої людини, то носієм емоційної оцінки або як ще говорять “суб’єктом тональності” буде той, на чию думку посилаються. Одним із важливих напрямів аналізу сентименту є виявлення негативних, позитивних атрибутів об’єкта тональностей, а також оцінка тексту на суб’єктивність і об’єктивність. Таким чином тональність вискакування визначається трьома компонентами: суб’єктом тональності (хто дав оцінку),

об’єктом тональності (кому дали оцінку) і власне самою тональною оцінкою (як оцінили).

Аналіз тональності знаходить своє практичне застосування в різних областях. Наприклад в соціології, за допомогою алгоритмів оцінки тональностей можна легко оцінити ставлення людей до того чи іншого явища. Інша сфера застосування – медицина і психологія, а саме аналіз користувачів соціальних мереж може виявити загальну епідемію захворювання або те, що засмучує людей та робить їх нещасними.

Окрема сфера застосування це політологія; багато політиків заплатили великі гроші за можливість моментально оцінити відношення електорату до тієї чи іншої проблеми. Бізнес також може використовувати аналіз сентименту текстів для оцінки відгуків своїх користувачів, а також для розуміння, що не так з їх продуктом та що потрібно покращити [9].

Ще однією зі сфер застосування алгоритмів аналізу тональностей тексту є соціальні мережі. Дуже часто під час трансляцій у реальному часі багато людей пишуть коментарі, але на жаль часто люди забувають про норми моралі, правил етикету та розмови з іншими людьми і переходять на грубощі. Для того уникнути сутичок у чаті, багато людей використовує послуги модераторів але навіть вони не можуть за всім услідкувати. Системи аналізу тональностей допоможуть пришвидшити і покращити працю адміністраторів, а саме вони зможуть аналізувати кожне повідомлення, на наявність в ньому агресії, негативу закликів до кримінальних вчинків і помічати такі повідомлення, як небезпечні. Адміністраторам залишиться лише продивлятися ті меседжі, де машина сумнівається.

1.3 Постановка задачі

Перша задача даної магістерської роботи полягає в досліджені та розробці метода розпізнавання іменованих сутностей, які знаходять імена, прізвища і по батькові в неструктурованих текстах. Для цього необхідно:

- дослідити існуючі методи розпізнавання іменованих сутностей для іноземних мов і для української мови;
- виконати програмну реалізацію створеного методу;
- підготувати корпус даних, в який входять неструктуровані тексти українською мовою;
- провести тестування та оцінити якість розробленого метода.

Друга задача даної магістерської роботи полягає в досліджені і розробці метода сентиментального аналізу текстів українською мовою. Для цього необхідно:

- дослідити існуючі методи аналізу сентименту в тексті для іноземних мов та для української мови;
- виконати програмну реалізацію метода аналізу сентименту тексту;
- підготувати корпус даних, в який входять неструктуровані тексти українською мовою;
- провести тестування, та оцінити якість методів аналізу сентименту.

2 ОПИС ТЕОРЕТИЧНИХ ДОСЛІДЖЕНЬ

2.1 Аналіз існуючих рішень для розпізнавання іменованих сутностей.

Оскільки проблему розпізнавання іменованих сутностей людство намагається вирішити досить давно, перші алгоритми для вирішення цієї проблеми, засновані на наборі правил, з'явилися також давно. Однією із реалізацій таких алгоритмів є бібліотека Natasha, яка використовує заздалегідь написані правила для розпізнавання іменованих сутностей.

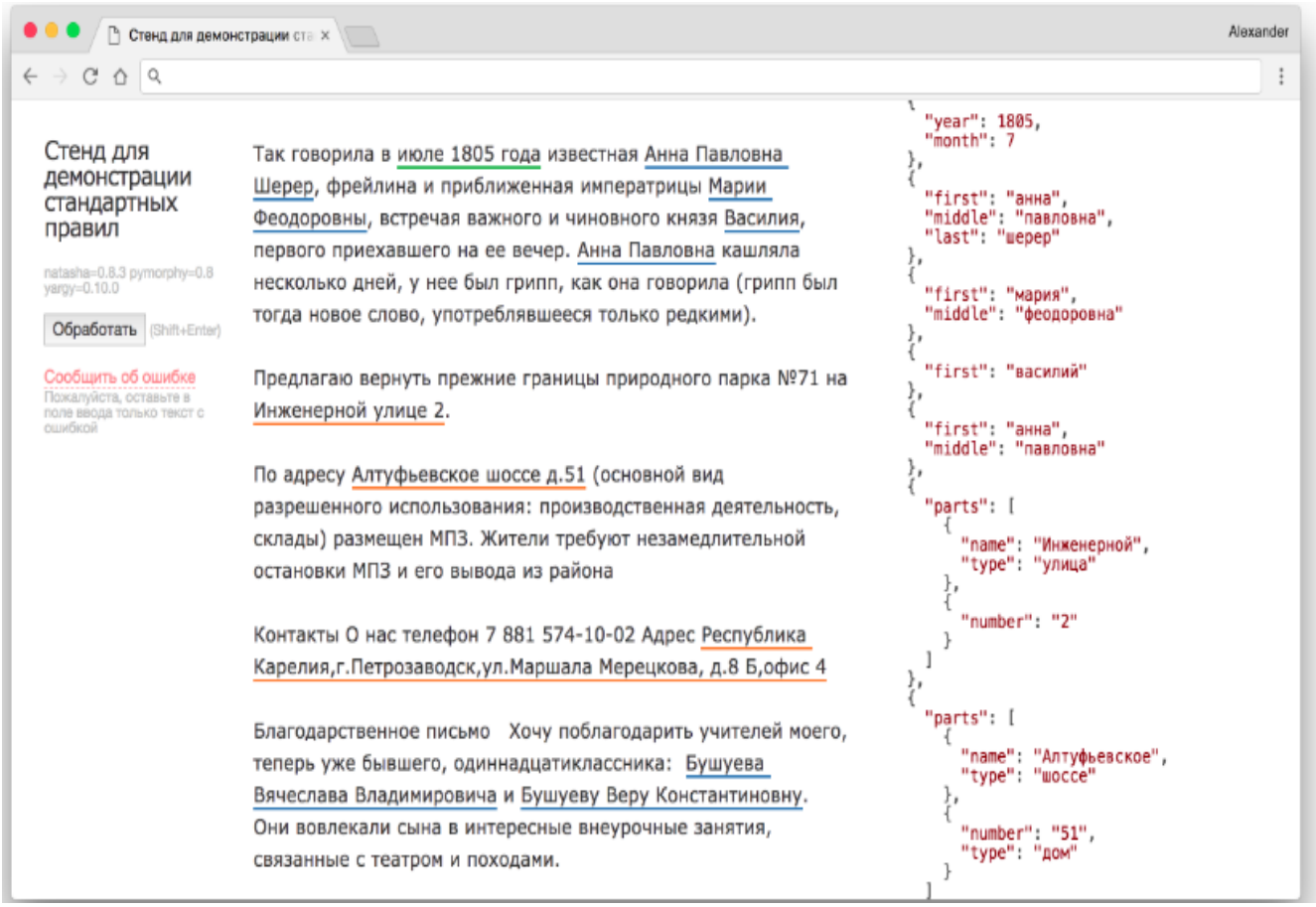

Рисунок 2.1 – Приклад роботи бібліотеки Natasha

Алгоритм роботи даної бібліотеки включає наступні кроки:

– нормалізація тексту;

– токенізація тексту;

– пошук частин тексту, які підходять під набір правил.

Ця бібліотека реалізує багато наборів правил для розпізнавання іменованих сутностей, а саме імена, прізвища, по батькові, назви організацій, гроші, дати, назви вулиць та площ. Інша перевага цієї бібліотеки полягає в тому, що вона реалізує простий інтерфейс для створення власних правил [10]. Також ця бібліотека розповсюджується під ліцензією MIT, що дозволяє усім бажаючим використовувати її безкоштовно. Цей програмний продукт має такі недоліки:

– бібліотека знаходить лише ті іменовані сутності для яких вже є правила;
– стандартні правила містять помилки, та мають невелику точність на деяких наборах даних;
– програмний продукт працює лише з російською мовою.

Друга велика група алгоритмів, які вирішують проблему розпізнавання іменованих сутностей, це алгоритмі побудовані на нейронних мережах. Протягом останніх років особливу популярність набрав алгоритм розпізнавання іменованих сутностей, який базується на технології BERT (Bidirectional Encoder Representations from Transformers). BERT є двонаправленою технологією, це означає, що BERT вивчає інформацію, як зліва так і справа від токена. Основна ідея BERT для вирішення проблеми розпізнавання іменованої сутності полягає в тому, що коли ми не знаємо токен слова, чи це місто, чи ім’я людини, чи ще щось, то ми можемо здогадатися про токен базуючись на контексті [11]. Наприклад, розглянемо речення “Він вилетів із (слово якому ми хочемо дати токен) до Києва”. В даному випадку контекст речення дає нам підказку, що слово на четвертому місці це локація і, скоріше за все, місто. Для ефективної роботи нейронної мережі, а також високої точності її результатів необхідно нормалізувати речення, а саме:

– перевести усі слова в нижній регістр;
– зберегти регістр для перших букв;
– видалити всі розділові знаки.

Невиконання може привести до того, що BERT почне робити наступні помилки, наприклад, речення Ілон Маск це (слово яке потрібно вгадати), та Ілон макс це (слово яке потрібно вгадати), буде мати дуже різний зміст, а саме в першому реченні BERT запропонує підставити такі слова як чоловік, музикант, студент, а

Ілон Маск буде розпізнан як ім’я і прізвище, то в другому реченні ім’я і прізвища не будуть знайдені, а пропущене слово буде бренд або пиво. На базі алгоритму BERT українська спільнота lang-uk вирішувала проблему розпізнавання іменованих сутностей. Lang-uk це відкрита для приєднання спільнота фахівці в галузі комп’ютерної обробки текстів (програмістів, лінгвістів, дослідників), діяльність якої базується на єдиних принципах. Вона займається підтримкою існуючих і розвитком нових проектів по збору українських корпусів та інших текстових даних. Також BERT займається і вирішенням проблеми розпізнавання іменованих сутностей для української мови. Найкращий результат, який наведено на офіційному сайті BERT складає 61.2% точних відповідей.

2.2 Аналіз існуючих рішень для визначення тональності тексту.

Підходи та ідеї, які використовуються для вирішення проблеми аналізу тональності тексту, дуже схожі на підходи і ідеї, які використовуються для вирішення проблеми розпізнавання іменованих сутностей у реченні. Наприклад, перші алгоритми, які вирішували дану проблему були основані на наборі правил, а саме алгоритм мав словник, у якому кожне слово мало вагову характеристику і вхідний текст розбивається на слова, і кожне слово, яке входило до речення, аналізували окремо. Після цього усі значення за допомогою певної формули зводились до одного числа, і базуючись на цьому числі, робився висновок щодо тональності тексту. Очевидно, що перевагою роботи такого алгоритму є швидкість, а також непогана точність на маленьких текстах, але алгоритми з цієї групи дуже погано працювали з великими текстами та недостатньо враховували взаємодію між реченнями у тексті. Саме через недоліки на сьогоднішній день домінуюча більшість алгоритмів базується або використовує нейронні мережі[12].

Людину завжди цікавило своє тіло: те, яким чином воно “працює”, за що відповідає кожен орган, як усе між собою пов’язано, а вершиною всього інтересу було відтворення процесів організму. Одним з таких процесів, який людство почало активно відтворювати з появою перших комп’ютерів у тридцятих роках минулого сторіччя, є процес мислення. Ці роботи почались з моделювання нейрону найменшої частинки мозку, які потім переросли в термін “нейронна мережа”. Перші роботи були зроблені Мак-Каллоком і Піттсом. В 1943 році ними була розроблена комп’ютерна модель нейронної мережі на основі математичних алгоритмів і теорії діяльності головного мозку. Вони висунули ідею, що нейрони можно спрощено розглядати як пристрої, які оперують двійковими числами і назвали цю модель “пороговою логікою”.

В наш час нейронна мережа (також штучна нейронна мережа) – математична модель, а також її програмне або апаратне втілення, що побудовані за принципом організації і функціонування біологічних мереж нервових клітин живого організму.

Приклад простої нейронної мережі наведено на наступному малюнку.

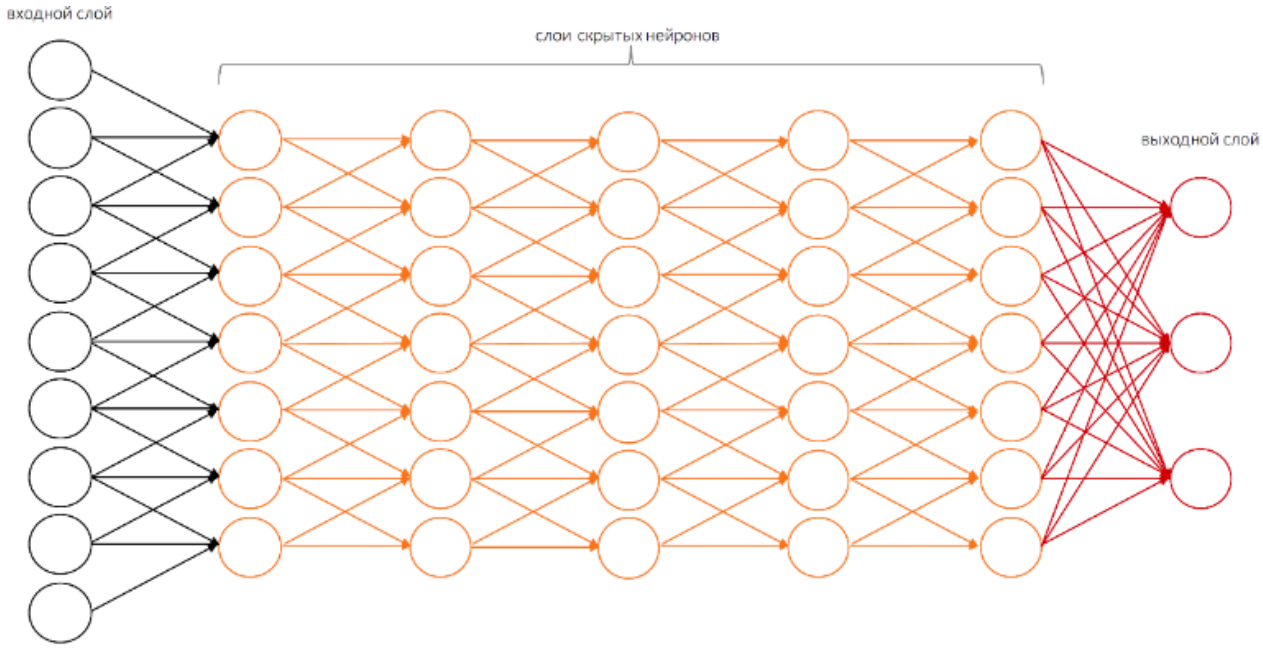

Рисунок 2.2 – Приклад нейронної мережі

В сьогоденні, завдяки розвитку паралельних обчислень та здатності зберігати величезні обсяги інформації в декілька екзобайт, нейронні мережі демонструють свою потужність, точність та швидкість у всіх сферах життя, а саме[13]:

– розпізнавання облич;

– прогнозування епідемій;
– прогнозування фондових ринків;
– розпізнавання спам не спам;
– прогнозування платоспроможності людини;
– розпізнавання іменованих сутностей;
– автоматичний переклад текстів;
– аналіз сентименту тексту.

Перш ніж перейти до більш глибокого розгляду нейронних мереж для вирішення задач розпізнавання іменованих сутностей і аналізу сентименту тексту, необхідно вирішити проблему даних для навчання нейронної мережі. Всі найточніші нейронні мережі перед використанням минають етап навчання. Для навчання використовуються великі, заздалегідь розмічені датасети. Для української мови питання розмічених датасетів є відкритим, тому нам потрібної його вирішити. Розглянемо існуючі способи отримання датасету. Перший і найпростіший спосіб отримати датасет - взяти вже існуючий перевірений багатьма науковцями датасет, який був створений декілька років тому, але цей підхід неможливо використовувати для української мови. Другий варіант - це власноруч створити датасет. Цей варіант є дуже затратним, бо якщо для розміщення одного файлу необхідно 3 хвилини, то для отримання датасету з 30000 файлів буде необхідно 1500 годин; перевагою цього варіанту є висока точність. Попередній варіант можна пришвидшити, якщо здійснювати розміщення за допомогою великої кількості людей, але даний варіант має такі недоліки, як менша точність та фінансова високовитратність. Якщо ж зробити краудфандінговий сайт, на який кожен може зайти і добровільно розмітити текст, то в результаті ми отримаємо високий відсоток невірних відповідей, приблизно 10-15%, а час, який необхідно витратити на отримання датасету може налічувати роки. Варіант, який є перспективним, це автоматичний збір інформації, за певними ознаками якої можно сказати чи є вона позитивною або негативною. Цей варіант дуже добре працює на кіносайтах, де люди залишають свої відгуки і оцінку. Базуючись на оцінці, можно зробити висновок стосовно сентименту тексту. Проаналізувавши найпопулярніші

майданчики для перегляду фільмів онлайн, було зроблено висновок, що такий спосіб не є можливим через те, що коментарі дуже короткі, велика кількість коментарів не співпадають з оцінкою, а кількість коментарів українською зовсім незначна і, зазвичай, це одне слово і декілька смайликів. Останній варіант, який є можливим через стрімкий ріст точності і якості перекладу, це переклад існуючого датасету з іноземної мови на українську. Недоліками цього варіанту є те, що машинний переклад великої кількості слів на українську мову буде коштувати грошей, а точність перекладу буде мати свої помилки. В даній роботі для створення першого відкритого датасету українською мовою, для аналізу сентименту тексту було обрано наступні варіанти. За допомогою GoogleAPI було зроблено переклад 28 тисяч текстів з відкритого датасету англійською мовою, який було взято з відкритого датасету кіновідгуків залишених на IMdb. Перевагами даного датасету є те, що кількість файлів у ньому 50 тисяч. Кожен файл містить від 20 до 3000 слів. Виконано переклад 28 тисяч текстів сумарною кількістю символів 24 800 000, через те, що за переклад кожного мільйона символів гугл стягує 540 грн. і після витрачених 12 500 грн.. було прийнято рішення зупинитися. В отриманому датасети 14 тисяч позитивних текстів і 14 тисяч негативних. Розмір кожного тексту був обмежений до 1000 слів, оскільки як показує практика, дана кількість символів достатня, щоб оцінити емоційний окрас тексту. Сто випадкових текстів з отриманого датасету були перевірені власноруч, точність перекладу була задовільна, єдиним знайденим мінусом було те, що перекладач вживав мало поширені слова і словосполучення для перекладу. Для фінального тестування, за допомогою керівника магістерської роботи було зібрано 111 текстів розмічених власноруч. З цих 11 текстів 17 є позитивними, 42 негативними, а решта нейтральними. Отриманий датасет був викладений в мережу з ліцензією MIT, що надає можливість кожній людині скачувати даний датасет та використовувати його для будь- яких власних цілей.

Через те, що нейронні мережі можуть оперувати тільки числами і ніяк не можуть оперувати текстами, зв’явилась необхідність проаналізувати існуючі алгоритми перетворення текстів у числові значення. Дане питання вже вивчене для

текстів, написаних поширеними мовами світу, такими як англійська, іспанська, але зовсім не розглянуто теоретично і, що найголовніше, не перевірено на практиці для текстів, що написані українською мовою. В даному розділі наводиться теоретичний аналіз даної проблематики, а в четвертому розділі наведені приклади практичної перевірки.

В світовій практиці перед початком роботи над текстом, аналізу та здобуття корисної інформації, його нормалізують. Це велика сфера, яка включає в себе наступні основні поняття:

– нормалізація;
– токенізація по реченням або словам;
– лематизація;
– стемінг;
– регулярні вирази.

Нормалізація це процес приведення тексту до єдиного формату. Для цього найчастіше роблять наступні кроки: текст приводиться до одного регістру, виправляються всі помилки, видаляються усі сполучники, частки, скорочення, які не несуть ніякого смислового навантаження[14]. Видаляються знаки пунктуації та інші символи, які не є текстом. Цифри та числа можуть як видалятися, так і замінюватися словами. Токенізація (іноді сегментація) – це процес розділення письмового тексту на речення або слова-компоненти. Ідея виглядає достатньо простою. Для токенізації по реченню можна використовувати крапку, але це не дуже тривіальна задача в українській мові, тому що крапка використовується в скороченнях. Зазвичай для вирішення проблеми скорочень використовується заздалегідь складена таблиця.

Зазвичай тексти містять різні граматичні форми одного і того ж слова, а також у тексті можуть зустрічатися однокореневі слова. Лематизація і стемінг допомогають привести всі словоформи, які містяться у текстах, до однієї нормальної словарної форми. Наприклад, для іменників це є називний відмінок однини чоловічого роду. Лематизація і стемінг – це часні випадки нормалізації і вони відрізняються. Стемінг – це грубий евристичний процес, який відрізає зайве

від корня слова, часто це призводить до втрати словотворчих суфіксів. Лематизація – це більш тонкий процес, який використовує словник і морфологічний аналіз, щоб у підсумку привести слово до його канонічної форми-леми. Різниця полягає у тому, що стемер (конкретна реалізація алгоритму стемінга) діє без знання контексту і відповідно не розуміє різницю між словами, які мають різний сенс в залежності від частини мови. Однак, у стемерів є і свої переваги, їх простіше застосувати і працюють вони краще. Також варто зазначити, що для деяких задач це рішення є дуже ефективним, наприклад, для задач аналізу сентименту.

Регулярний вираз (регулярка, regexp, regex) – це послідовність символів, що визначає шаблон пошуку. В деяких задачах, наприклад, розпізнавання іменованих сутностей, а саме підзадача пошуку імен, прізвищ, по батькові, можна використовувати регулярні тексти для видалення всіх слів, які не починаються з великої літери.

Після вирішення задачі нормалізації тексту постає питання, а як цей текст подати на вхід нейронній мережі, тому що вона може оперувати тільки з числами. Задача конвертації тексту в набір чисел є задачею виділенням ознак. Для вирішення цієї задачі в світовій практиці використовують наступні алгоритми:

– мішок-слів;
– N-грами;
– Tf-idf;
– Word2vec.

Мішок слів – це популярна і проста техніка виділення ознак, яка використовується при роботі з текстом[15]. Вона описує входження кожного слова в текст. Щоб використовувати модель нам потрібно:

– визначити словник відомих слів (токенів);
– вибрати ступінь присутності відомих слів.

Будь-яка інформація про порядок або структуру слів ігнорується, тому ця техніка називається мішком слів. Зазначена модель намагається зрозуміти чи зустрічала вона знайоме їй слово в документі, але не знає де саме вона його

зустріла. Цей метод був побудований і потім показав свою ефективність на практиці завдяки інтуїтивному твердженню, що схожі документи мають подібне наповнення. На наступній картинці наведено приклад кодування речення за допомогою техніки “мішок слів”:

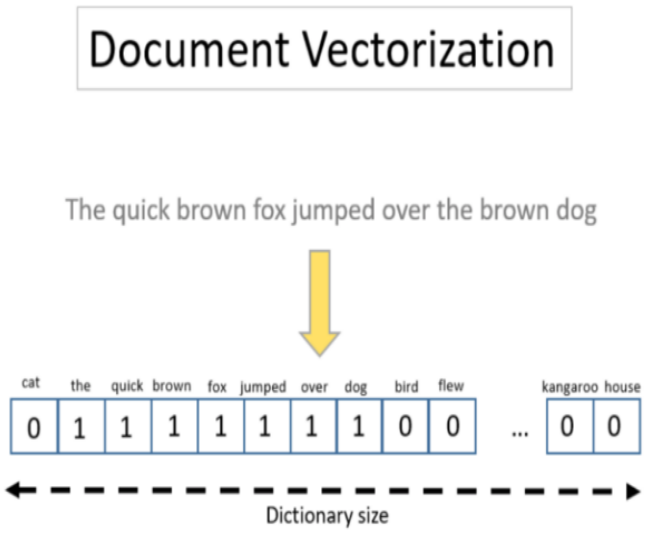

Рисунок 2.3 – Приклад роботи алгоритму мішок слів.

Складність використання цієї моделі полягає в тому, як визначити словник і як підрахувати входження слів. Коли розмір словника збільшується, вектор документу теж росте. На прикладі, що зображений на попередньому рисунку, довжина вектора дорівнює кількості відомих слів. В деяких випадках у нас може бути надзвичайно великий об’єм даних і тоді вектор може складатися із десятків або сотень тисяч елементів. Більш того, кожен документ може містити в собі лише малу кількість слів із словника. Наслідком цього є те, що в векторному вигляді буде багато нулів. Вектори з великою кількістю нулів називаються розрідженими векторами і вони потребують більше пам’яті і обчислювальних ресурсів. Однак ми можемо знизити кількість відомих слів, коли використовуємо дану модель, щоб знизити вимоги до обчислювальних ресурсів. Для цього можна використовувати такі техніки:

– ігнорування регістра слів;
– ігнорування пунктуації;
– видалення стоп-слів;

– приведення слів до базової форми;

– виправлення орфографічних помилок;

– видалення слів, які зустрічаються менш ніж певна кількість разів.

Інший більш складний спосіб складання словника – це використовувати згруповані слова. Це може змінити розмір словника і дасть мішку слів більше деталей про документ. Такий підхід називається N-грама. N-грама – це послідовність будь-яких сутностей (слів, букв, чисел, цифр і т.д.). У контексті мовних корпусів, под N-грамою зазвичай мають на увазі послідовність слів. Юніграма це одне слово, біграма це послідовність із двух слів, триграмма – три слова і т.д. Цифра N позначає скільки згрупованих слів входить в N-граму. В модель потрапляють не всі можливі N-грами, а лише ті, що фігурують в корпусі.

Коли словник створений, потрібно оцінити наявність слів. Вище було описано простий бінарний підхід, у якому 1 – слово є, 0 – слова немає. Є і інші методи:

– Кількісний – підраховує кількість входжень слова у документ;

– Частотність – підраховує, як часто кожне слово зустрічається в тексті (по відношенню до загальної кількості слів).

У частотного скорінга є проблема: слова з найбільшою частотністю мають, відповідно, найбільшу оцінку. В цих словах може бути не так багато інформаційного виграшу для моделі, як в менш частих словах. Одним із способів вирішення цієї проблеми є зменшення оцінки слоів, яке зустрічається часто у всіх схожих документах. Це називається TF-IDF.

TF-IDF (скорочення від term frequency – inverse document frequency) – це статистична міра для оцінки важливості слова в документі, котрий є частиною колекції або корпусу[16].

Скорінг по TF_IDF росте пропорційно частоті появи слова в документі, проте це компенсується кількістю документів, які містять це слово.

TF (term frequency – частота слова) – відношення числа входжень слова до загальної кількості слів документу.

IDF (inverse document frequency – зворотня частота документу) – інверсія частоти з якою деяке слово зустрічається в документах колекції.

Як результат TF-IDF розраховується як добуток TF-IDF.

Word2Vec – це технологія, яка була розроблена компанією гугл, головною метою якої є статистична обробка великих масивів текстової інформації[17]. W2V збирає статистику по сумісній появі слів в словосполученнях або фразах, після чого методами нейронної мережі вирішують задачу зниження розмірності і видають на виході компактні векторі відображення слів, в максимальній ступені відображаючи відношення цих слів в оброблюємих текстах. Даний підхід заснований на дуже важливій гіпотезі, яку в науці прийнято називати "гіпотезою локальності" – слова, які зустрічаються в однаковому оточенні, мають близькі значення. Даний підхід дуже ефективно показує себе через те, що він дає змогу обробляти текст згортковими нейронними мережами.

Проаналізувавши способи нормалізації тексту та способи перетворення тексту в числа, був зроблений аналіз існуючих нейронних мереж для вирішення проблеми аналізу сентименту. Для порівняння у даній роботі використовувалося 4 види нейронних мереж, а саме:

– звичайні нейронні мережі;
– рекурентні мережі LSTM;
– рекурентні мережі GRU;
– звичайні рекурентні мережі.

Згорткові нейронні мережі не були використані через те, що не було знайдено ефективної відкритої реалізації підходу W2V для української мови але варто зазначити, що ефективність згорткових мереж для задачі аналізу сентименту тексту дуже висока в англійській мові.

Основна відмінність рекурентних мереж від традиційних полягає в логіці роботи мережі, при якій кожен нейрон взаємодіє сам з собою[18]. На вхід таким мережам як правило подається сигнал, який є деякою послідовністю. Кожен елемент такої послідовності по черзі передається одним і тим же нейроном, який робить своє передбачення, повертає його собі разом з наступним її елементом до

тих пір поки послідовність не закінчиться. Варто зазначити, що для проблем машинного перекладу тексту використовують рекурентну нейронну мережу у форматі sequence to sequence, це означає що коли на вхід подається послідовність і на виході теж отримується послідовність. На рисунку нижче зображена ця нейронна мережа.

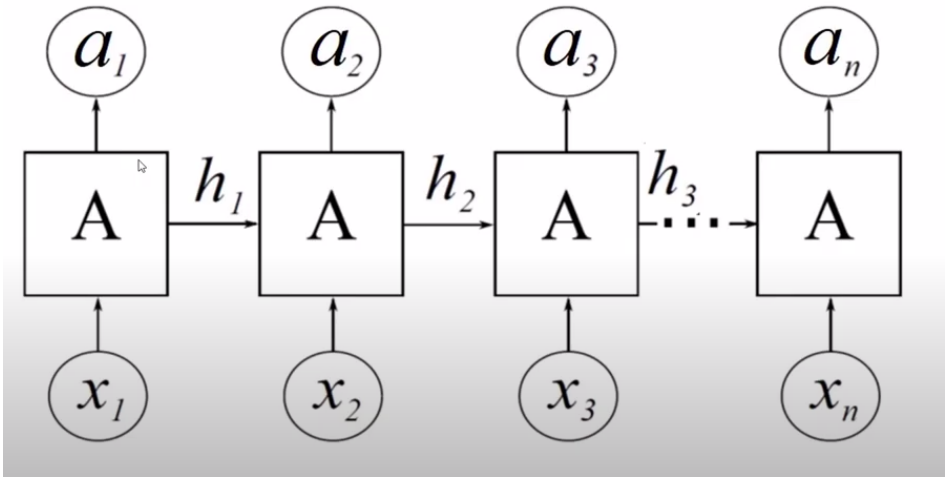

Рисунок 2.4 – Рекурентна нейронна мережа у форматі роботи sequence to sequence.

Для задач класифікації використовують рекурентну нейронну мережу у форматі sequence to vector. Ключова різниця цих двох мереж полягає в тому, що в останньому форматі роботи ігноруються виходи всіх проміжних слоїв нейронної мережі і береться лише останній.

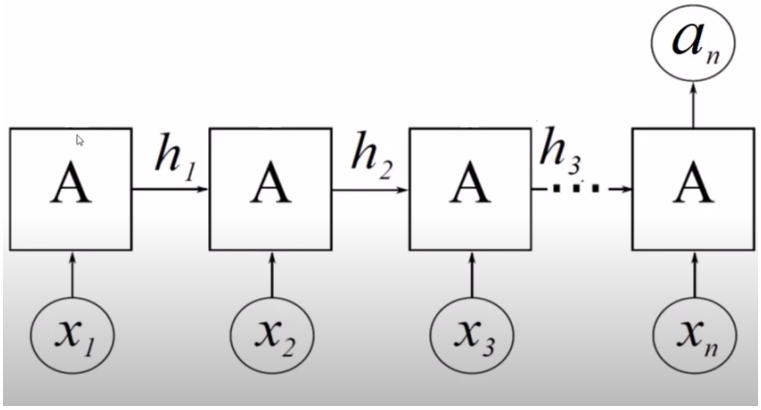

Рисунок 2.5 – Рекурентна нейронна мережа у форматі роботи sequence to vector.

Елементи рекурентної мережі зображуються як звичайні нейрони з додатковою циклічною стрілкою, котра демонструє те, що окрім вхідного сигнала нейрони використовують також і свій додатковий схований стан. Якщо розгорнути таке зображення, то отримаємо послідовність однакових нейронів, кожен з яких отримує на вхід свій елемент послідовності і видає передбачення та передає його далі за ланцюгом, як свого рода осередок пам'яті. Потрібно розуміти, що це абстракція, оскільки це один і той же нейрон, який відпрацьовує декілька разів підряд. Така архітектура нейронної мережі дозволяє вирішувати такі задачі як, передбачення останнього слова в реченні. Моделювання пам'яті нейронної мережі подібним чином вводить нове вимірювання в опис процесу її роботи – час. Нехай нейронна мережа отримує на вхід послідовність даних, наприклад, текст послівно або слово побуквенно. Тоді кожен наступний елемент цієї послідовності поступає на нейрон в новий умовний момент часу. До цього моменту в нейроні вже є накопичений з початку отримання інформації досвід. Основна різниця між типами рекурентних нейронних мереж один від одного полягає в тому, як обробляються осередки пам'яті всередині них. Традиційний підхід передбачає складення двох векторів (сигналу і пам'яті) з наступним розрахунком активації від суми, наприклад, гіперболічним тангенсом. Отримується звичайна мережа з одним прихованим слоєм. На рисунку знизу зображені схеми нейронних мереж, які розглядаються у даній роботі.

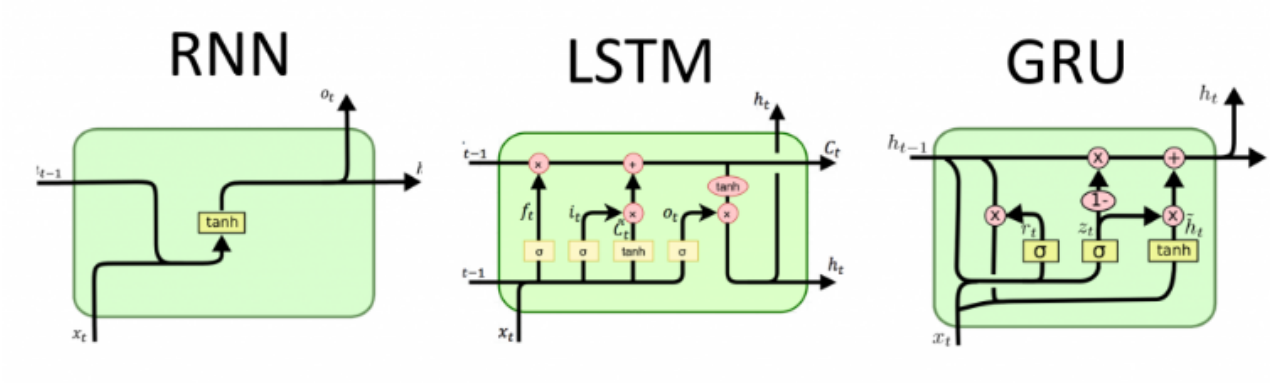

Рисунок 2.6 – Архітектури рекурентних мереж які розглянуті у даній роботі.

Пам’ять, що реалізована першим способом, є досить короткою. Оскільки кожен раз інформація в пам’яті змішується з інформацією у новому сигналі, через 5-7 ітерацій інформація вже повністю перезаписується. Повертаючись до задачі аналізу сентименту, дана мережа буде добре працювати в рамках коротких текстів (20-40 слів, але на більших текстах, закономірності на початку його не будуть вносити якийсь суттєвий вклад в рішення мережі ближче до кінця тексту, також як і помилка на перших елементах послідовності в процесі навчання перестане вносити вклад в загальну помилку мережі. Це дуже умовний опис даного явища на справді є фундаментальною проблемою нейронних мереж, яка називається проблемою “зникаючого градієнта”. Щоб вирішити цю проблему були винайдені дві мережі: LSTM-RNN мережа (Long Short-Term Memory Recurent Neural Network) і GRU (Gated recurrent units), в яких були додані додаткові внутрішні перетворення, які оперують з пам'яттю більш обережно. Мережі з архітектурою LSTM дозволяють запам’ятовувати дані на довше, але для їх навчання необхідно більше обчислювальних потужностей[19]. Мережі GRU були винайдені для недоліку мереж LSTM і вони навчаються швидше, але запам’ятовують інформацію на коротший проміжок часу[20].

Однією з поширених програм аналізу тональності тексту є Twitter Sentiment. Twitter – це одна з найпопулярніших платформ в світі для обміну своїми думками, подіями за допомогою повідомлень довжиною до 140 символів. Число щоденних унікальних користувачів більше ніж 100 мільйонів людей. Так як часто люди пишуть в своїх твітах відгуки про товари, то аналізуючи тональність твітів людей можна зрозуміти, що люди думають про цей продукт або людину. Користувачеві Twitter Sentiment достатньо ввести слово і програма проаналізує останні 100 записів про це слово. Також буде побудовано графік співвідношення позитивних та негативних відгуків [21].

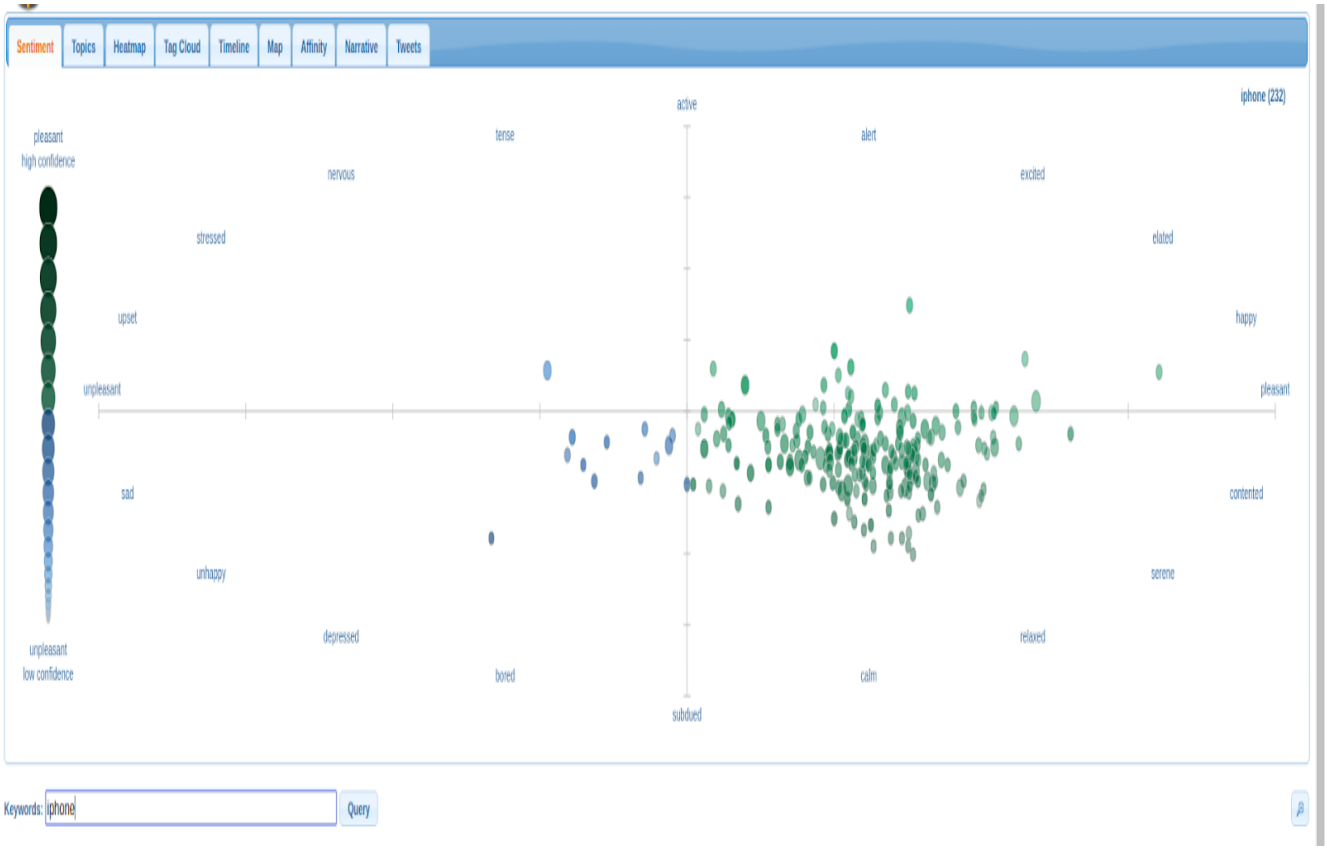

Рисунок 2.7 – Результат роботи программи для слова iphone.

Програма Twitter Sentiment надає легкий спосіб аналізу думок користувачів про продукти. Також Twitter Sentiment використовує метод машинного навчання для аналізу тональностей та має API для того, щоб користувачі могли легко інтегрувати даний продукт до своїх програмних продуктів.

Даний продукт має такі суттєві недоліки, як:

– аналізує лише твіти і тільки маленьку їх частину;

– не може аналізувати довільний набір речень.

Велику ефективність для оцінки емоційності тексту показують себе алгоритми на основі згорткових нейронних мереж. Данні алгоритми добре вивчені і ефективно використовуються в проблемі класифікації зображень[22] Також свою ефективність вони продемонстрували у риноманометричній обробці даних[23]

Прикладом реалізації цього алгоритму для вирішення задачі аналізу тональності тексту є продукт від Microsoft, а саме “Аналіз тексту від Microsoft”. Microsoft – це багатонаціональна компанія, яка є найбільшим виробником програмного забезпечення. Їх продукт “Аналіз тексту” є частиною продуктів, які доступні через

web інтерфейс. Для аналізу тональностей Microsoft використовує заздалегідь навчену нейронну мережу для створення оцінки тональності в діапазоні від 0 до 1. Оцінка, яка ближче до 1, є свідченням позитивної тональності, а оцінки, які ближче до 0 на негативну. Аналіз тональності виконується для всього документу, а не для окремих сутностей в тексті, це означає, що оцінки тональності повертаються на рівні документа або речення.

Рисунок 2.8 – Результат роботи системи аналізу тональності від Microsoft.

Модель, що використовується, заздалегідь натренована на широкому наборі текстів зі зв’язками тональностей. Мережа використовує для аналізу тексту декілька різних методик, які включають обробку тексту, аналіз частин мови, розміщення слів і зв’язків між словами.

Якщо документ складається з декількох речень, а не з одного великого блоку тексту, то точність алгоритму буде вища. Алгоритм на етапі оцінки об’єктивності моделі виявляє, чи є документ в цілому об’єктивним і чи містить він тональність. Документ, який здебільшого є об’єктивним не піддається фазі знаходження тональності, що призводить до оцінки 0.5, без подальшої обробки. Для документів,

які залишаються, наступна фаза генерує оцінку вище або нижче 0.5. Ця реалізація алгоритму оцінки тональності тексту містить наступні недоліки:

– відсутня можливість навчання нейронної мережі на своїх даних;
– алгоритм знаходить лише дві тональності або показує їх відсутність;
– алгоритм не працює з українською мовою;
– для використовування в промислових продуктах необхідно платити.

Крім продуктів великих корпорацій є продукти, які були зроблені спільнотою програмістів. Одним з таких продуктів є Sentiment Analysis with Python NLTK Text Classification. Цей продукт має веб-інтерфейс.

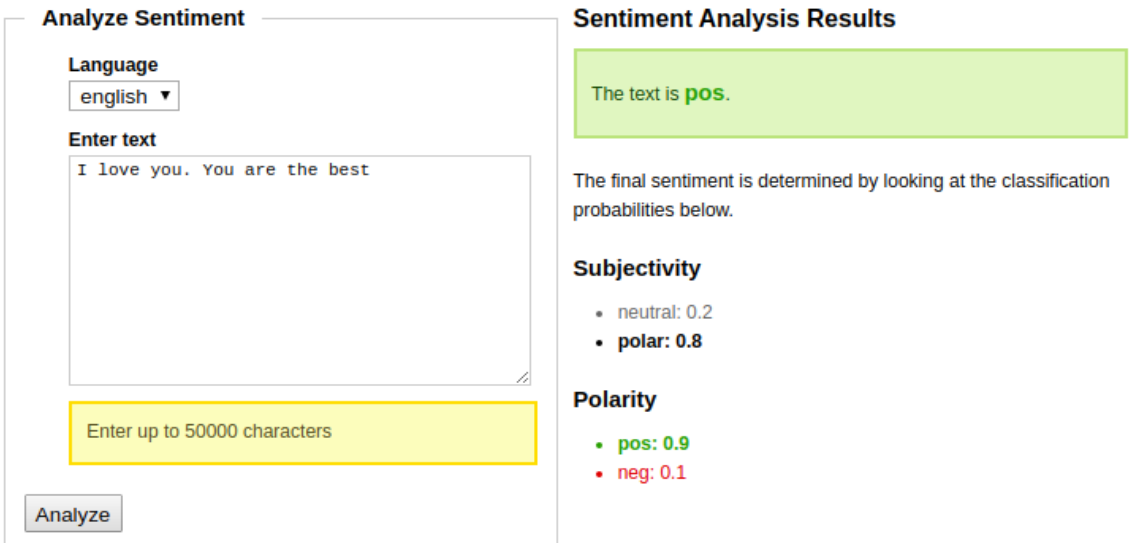

Рисунок 2.9 – Результат роботи Sentiment Analysis with Python NLTK Text Classification

Використовує методи машинного навчання, але якість даної нейронної мережі гірша ніж якість у аналогічного рішення від корпорацій гігантів. Це відбувається через те, що зазвичай продукти з відкритим кодом не мають достатню кількість ресурсів для навчання нейронних мереж, але значною перевагою даного продукту є те, що він дозволяє навчати нейронну мережу додатково, на своїх даних та на своєму обладнанні [24]. До недоліків Sentiment Analysis with Python NLTK Text Classification можна віднести те, що вона:

– видає лише бінарну класифікацію текстів.
– має слабо треновану нейронну мережу.
– немає підтримки української мови.

3 ОПИС ПРАКТИЧНИХ ДОСЛІДЖЕНЬ

3.1 Опис інструментів програмної реалізації

Перш ніж почати писати реалізацію алгоритмів та перевірку своїх ідей потрібно визначитись з технологіями, які будемо використовувати. Найважливіше питання, на якій мові ми будемо реалізовувати програмний продукт. За основну мову програмування було обрано Python. Мова програмування Python – це високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності програміста і читаємость коду. Дана мова підтримує декілька парадигм програмування, в тому числі і структурне, об’єктно-орієнтоване, функціональне, імперативне і так далі[25]. Еталонною реалізацією Python являється інтерпретатор CPython, він підтримує більшість активно використовуємих платформ. Він поширюється під вільною ліцензією Python Software Foundation License, яка дозволяє використовувати його будь в яких додатках, без обмежень. Типізація у мові програмування Python динамічна, тобто тип змінної визначається лише під час виконання. Тому замість «присвоювання 42 значення змінної» краще говорити про «зв’язуванні значення з деяким ім’ям». Однією з головних причин, через яку було обрано Python, як основну мову програмування, є те, що він ідеально підходить для задач машинного навчання, математичних задач і простих наукових розрахунків. Python є однією з головних мов сучасних науковців, які використовують його для реалізації і перевірки своїх теоретичних результатів. Іншою причиною вибору даної мови програмування є те, що на сьогодні існує дуже багато реалізації різних методів машинного навчання, відповідних бібліотек і інших допоміжних інструментів таких як, наприклад, пакет для наукових обчислень NumPy, пакет для роботи з даними Pandas і т.д.

Однією з задач, яку необхідно вирішити, до початку роботи над програмою це вибір середовища розробки. Мій вибір пав на середовище розроблення від компанії JetBrains Pycharm версії 2019.1. Це крос платформне інтегроване середовище розробки програмного забезпечення на мові програмування Python.

Дане середовище працює, як з мовою програмування Python версії 2 так і з Python 3 версії. Переваги даного продукту наступні:

– зручна навігація по проекту;
– зручний пошук необхідних фрагментів коду, як в усьому проекті так і в файлі;
– кросплатформеність;
– неявна перевірка на наявність помилок;
– допомога у виправлені помилок;
– пришвидшує рефакторінг;
– професійна версія даного продукту є безкоштовною для студента;
– велике ком'юніті та гарна підтримка проекту;
– легкий менеджмент бібліотек проекту.

До недоліків даного програмного продукту потрібно віднести:

– вимоглива до ресурсів комп'ютера;
– може повільно працювати на великих розмірах проекту.

Зважаючи на те, що середовище розробки PyCharm ідеально відповідає усім вимогам поставленим до даного продукту, то вибір був зроблений на його користь.

Pandas – програмна бібліотека на мові Python для обробки і аналізу даних. Робота Pandas з даними будується поверх бібліотеки NumPy, яка є інструментом більш низького рівня. Pandas дає нам змогу користуватися спеціальними структурами даних для маніпуляції числовими таблицями і часовими рядами [26]. Назва бібліотеки походить від економетричного терміну "персональні дані", що використовується для опису багатовимірних структурованих наборів інформації. Pandas розповсюджується під новою ліцензією BDS. Не дивлячись на те, що Python на протязі довгого часу успішно використовується для очищення і підготовки даних, його використовування під час проведення аналізу і моделювання ускладнено. Дуже часто необхідно виконувати частину задач за допомогою інструментів, що відображають специфіку предметної області, такі як мова Java. Можливості Pandas компенсують перелічені недоліки і дозволяють повністю

проводити усі етапи аналізу інформації на Python. Бібліотека має такі корисні властивості:

– об’єкт DataFrame для маніпулювання індексованими масивами двовимірних даних;
– інструменти для обміну даними між структурами в пам’яті і файлами різних форматів;
– вбудовані засоби поєднання даних і способи їх обробки.

Ця бібліотека оптимізована для високої продуктивності, найбільш важливі частини кода написані на мові програмування C.

Так як однією з вирішуваних задач є задача розпізнавання іменованих сутностей, то нам знадобиться досить часто вирішувати такі проблеми, як:

– приводити слова до безособової форми (наприклад гуляв -> гуляти);
– ставити слово в потрібну форму;
– повертати граматичну інформацію про слово (число, рід, відмінок, частина мови і т.д.).

Усі ці властивості має бібліотека pymorphy2. Ця бібліотека ефективно упаковує слова на основі алгоритму побудованому на кінцевих автоматах (Deterministic Acyclic Finite State Automaton) з використанням бібліотеки DAWG. У структурі даних DAFSA деякі частини слів не дублюються і через це потребують менше пам’яті для зберігання, крім того, в DAWF можна швидко виконувати не тільки точний пошук слова, але і інші операції, наприклад, пошук по префіксу або пошук з замінами. Для використовування цієї бібліотеки потрібен словник в форматі OpenCorpora, на жаль словника української мови в цьому форматі не має. Щоб створити словник української мови в форматі OpenCorpora, потрібно використовувати бібліотеку, LT2OpenCorpora. Ця бібліотека перетворює відкритий словник українських слів, який потрібно окремо завантажувати і компілювати, в словник формату OpenCorpora, який в свою чергу використовується бібліотекою pymorphy2. Під час виконання цієї роботи бібліотека LT2OpenCorpora була розширена, а саме була додана підтримка трьох тегів, що слово це ім’я, прізвище або по батькові.

Для написання, навчання і перевірки отриманих нейронних мереж було вирішено використовувати GC(Google Colaboratory). GC – це хмарний сервіс, який направлений на спрощення дослідів в області машинного і глубокого навчання. Використовуючи Coloboratory можно отримати доступ до віддаленої машини з підключенною відеокартою, а головна перевага полягає в тому, що використання відеокарти є зовсім безкоштовне, що значно спрощує життя. В цьому сервісі вже є установлений Tensorflow і майже всі необхідні для роботи Python-бібліотеки. Якщо якийсь пакет відсутній, то він з легкістю встановлюється на ходу за допомогою pip.

3.2 Методи перевірки результатів досліджень.

Виходячи з того, що потрібно оцінити отримані результати, вводимо критерій оцінки. При виборі критерію необхідно враховувати те, що і задача розпізнавання іменованих сутностей і задача аналізу тональності тексту є задачею класифікації.
Для початку введемо важливу концепцію опису метрики в термінах помилок класифікації. Допустимо, що у нас є два класи і алгоритм, який показує до якого класу належить об’єкт.
Тоді у нас можливі 4 ситуації:

- об’єкт належить до класу 1 і алгоритм показав, що це клас 1. Така ситуація називається True Positive(TP);
- об’єкт належить до класу 1, а алгоритм показав, що це клас 2. Така ситуація називається False Positive(FP);
- об’єкт належить до класу 2, а алгоритм каже, що це клас 1. Така ситуація називається False Negative(FN);
- об’єкт належить до класу 2 і алгоритм каже, що це клас 2. Така ситуація називається True Negative(TN).

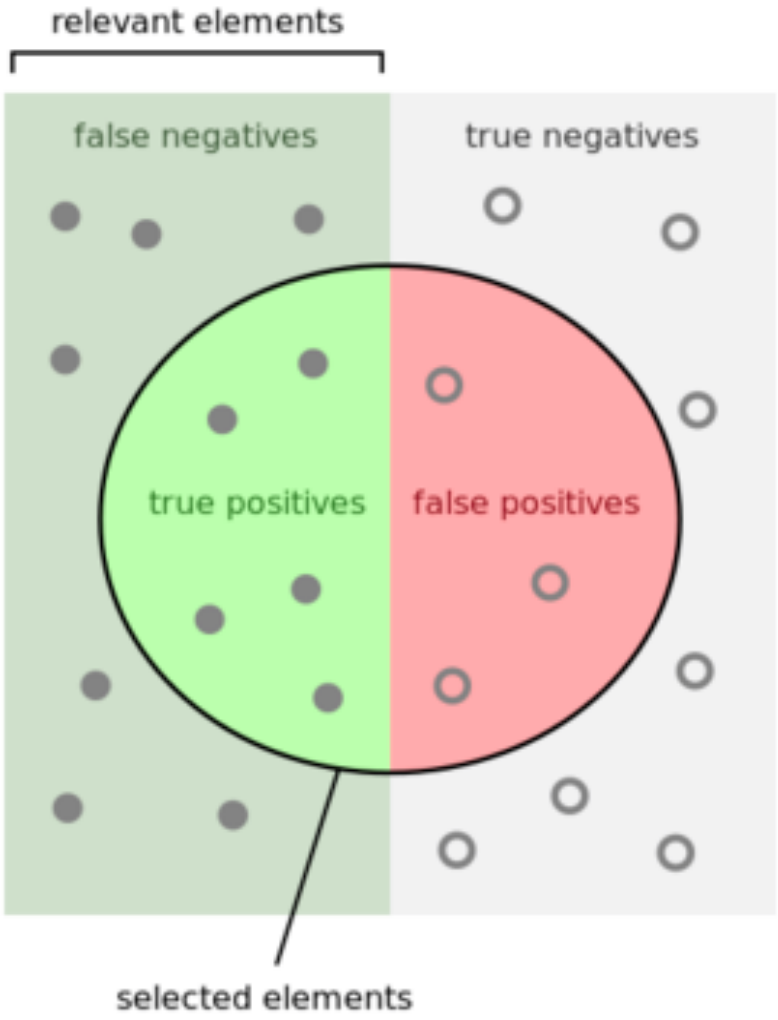

Рисунок 3.1 – Схематичне зображення результатів алгоритму класифікації

Для розрахунку точності використовується формула:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Для розрахунку повноти використовується наступна формула:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Точність можна інтерпретувати як долю об’єктів, які названі класифікатором позитивними і які при цьому є позитивними, а повнота показує, яку долю об’єктів

позитивного класу з усіх об’єктів позитивного класу знайшов наш алгоритм. Саме введення точності не дозволяє нам вписувати усі об’єкти в один клас, так як в цьому випадку ми отримаємо ріст рівня FP. Повнота демонструє здатність алгоритму знаходити даний клас в цілому, а точність – здатність алгоритму відрізняти цей клас від інших класів.

Є безліч способів об’єднати точність і повноту в агрегований критерій якості. В цій роботі обрано F-критерій, тому що цей критерій задовольняє всі вимоги до критерію, а також це є один із стандартних критеріїв для оцінки якості роботи алгоритмів класифікації. F-критерій – це середнє гармонічне точності і повноти. У статистичному аналізі ця бінарна оцінка є мірою точності тесту. Цей критерій враховує як точність так і повноту які були отримані під час тестування. Формула розрахунку F-критерія має наступний вигляд:

$$\mathrm{F}_\beta = (1 + \beta^2)\frac{\mathrm{precision} * \mathrm{recall}}{(\beta^2 * \mathrm{precision}) + \mathrm{recall}}$$

В цій роботі у якості β було взято 1. F-критерій досягає максимуму при повноті і точності рівним одиниці і близький до нуля, якщо якийсь із аргументів близький до нуля [27].

Також для перевірки досліджень нам необхідні дані. Для цього ми візьмемо відкритий набір даних, який вже був розміщений до нас. Цей набір складається з більш ніж 200 текстів, в кожному тексті знаходиться декілька іменованих сутностей. Ми будемо шукати імена, прізвища та по батькові. Також, щоб перевірити якість роботи різних алгоритмів на задачах, які необхідно вирішувати при створенні чат ботів, ми створили власний датасет з речень від 5-10 слів кожне. У кожному реченні є одна або більше іменованих сутностей. Частина іменованих сутностей це тільки імена або прізвища, частина це імена, прізвища та по батькові.

3.3 Програмна реалізація теоретичних досліджень

Реалізацію було розпочато з алгоритмів оцінки тональностей неструктурованих текстів, які написані українською мовою. Для перевірки було обрано дві групи тональності, а саме:

– позитивні;

– негативні.

Нейтральна тональність не розглядалась через те, що для неї не було розміченого датасету для навчання нейронної мережі.

Серед розглянутих рішень для цієї проблеми було обрано два основні підходи: перши базується на словниках, а другий базується на нейронних мережах[28].

Перший підхід, який базується на словниках, був реалізований і показав свою неефективність через те, що цей метод потрібно доповнювати правилами під кожну конкретну ситуацію. Алгоритм на основі словників, наприклад, не міг відрізнити окрас слова “сильний” у двох контекстах: сильно вдарив та сильний_вплив. Таких прикладів було знайдено безліч, базуючись на цьому зроблено висновок, що даний підхід не може вирішити проблему оцінки тональності тексту.

Була спроба використовувати нейронні мережі, які були навчанні на датасетах інших мов, але вони не завершилися успіхом[29]. У другому підході перевірялися чотири нейронні мережі запропоновані у розділі два. Для перевірки даних нейронних мереж, отриманий датасет був поділений на дві частини. Перші 14 тисяч текстів використовувались для навчання нейронної мережі, а інші 14 тисяч для перевірки її якості. Для перевірки здатності нейронних мереж знаходити сентимент текстів не пов'язаних з тематикою датасету була зроблена перевірка на датасеті, який був власноруч розмічений. Детальні результати можна побачити в наступному розділі. Також під час цієї перевірки було зроблено перевірку не тільки якості нейронних мереж але і алгоритмів перетворення текстів на вектори чисел.

Перший метод, який буде реалізовано та перевірено це буде метод розпізнавання іменованих сутностей за допомогою правил. В рамках реалізації цієї

програми буде зроблений додаток, який дозволить будь-якій людині писати правила для розпізнавання іменованих сутностей на українській мові у декларативному стилі. Як ми знаємо для того, щоб набір правил давав високий результат, нам потрібен гарний словник, який дозволить перетворювати слова в нормальну форму. Вимоги до словника були виставлені наступні: він повинен мати змогу розпізнавати частини мови такі як іменник, прикметник, займенник, числівник. Найважливішою вимогою до словника є те, щоб підтримувався таг, який вказував на те чи є слово іменем, чи слово є прізвищем або по батькові, чи є слово одухотвореним чи ні. Наявність проставлених тегів роду таких, як чоловічій, жіночій та середній є теж вимогою до словника. Також важливою умовою є наявність тегу числа, чи є слово у множині, чи слово вживається тільки у однині. Приємним бонусом може бути наявність тегу відмінку у словнику, щоб даний тег вказував чи є дане слово у орудному відмінку, а може слово у родовому відмінку. Однією з найголовніших вимог до словника є те, що він повинен бути сумісним з бібліотекою, яка його використовує, це дасть змогу зменшити час на інтеграцію частини програми і словника. Серед усіх існуючих словників, які є відкритими та містять велику кількість слів, словником, який найкраще підпадає під опис заданих критеріїв є словник OpenCorpora[30]. OpenCorpora це проект мета якого створити морфологічний, синтаксичний і семантичний розмічений корпус текстів на російській мові, який в повному обсязі доступний для дослідження і редагуємий користувачами. Як результат такої роботи вони створили словник, але, нажаль, для української мови такого словника не має. В українській мові є теж не поганий і постійно оновлюваний словник з відкритим доступом, який підтримується проектом LanguageTool[31]. Нажаль, цей словник не структурований за форматом OpenCorpora, через це перед використанням словника необхідно його перетворити у формат OpenCorpora. Для цього на github була знайдена бібліотека, яка займається таким мапінгом. Нанаступному малюнку кремовим кольором позначені таги, які є тільки у словнику в форматі OpenCorpora, блакитні кружечки це теги, які є тільки у словнику LanguageTool, а зелені ті, які є в обох вищезазначених словниках

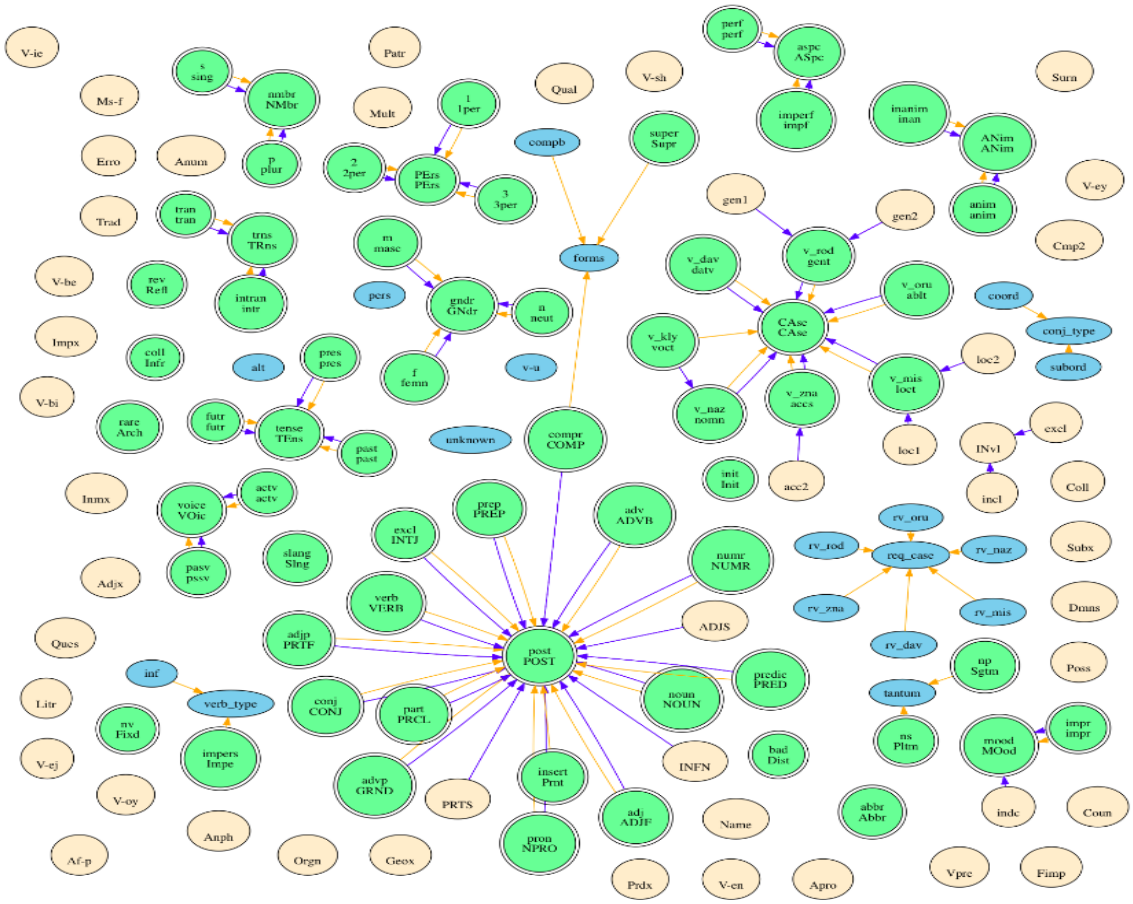

Рисунок 3.2 – приклад мапінгу тегів у форматі LanguageTool до формату словника OpenCorpora

Даний мапер не підтримував мапінг найважливіших тегів, а ще тих, які відповідають за те чи є слово ім’ям, прізвищем та по батькові. Щоб вирішити ці проблеми у рамках роботи бібліотека була доопрацьована, а необхідні зміни були збережені в гілку мастер у головному репозиторію цієї бібліотеки.

Після того, як усі головні етапи були зроблені, усі бібліотеки знайдені і проаналізовані, було розпочато розробку програмного забезпечення для перевірки усіх теоретичних досліджень. Перший підхід, який був опрацьований, це підхід на основі регулярних виразів[32]. Цей підхід простий у реалізації та може бути швидко перевірений. Після перевірки даного алгоритму на тестових даних виявилось, що він погано підходить для вирішення цієї задачі і має такі недоліки:

– повільно працює для великих текстів;

– не може зібрати докупи прізвище ім’я та по батькові однієї особи;

– низька точність;

– важко виправити код не ламаючи чогось.

Перед реалізацією наступного підходу необхідно було вирішити проблему, а саме – знайти великий, всеохоплюючий словник українських імен, прізвищ та по батькові. Цей словник повинен не тільки охоплювати найпопулярніші імена але імена, які є рідковживаними. Для вирішення цієї проблеми було взято відкриті датасети українських імен, прізвищ та по батькові . Цей датасет містив 8987 імен, 191 000 прізвищ та 16 000 по батькові, але мав наступні проблеми, які не дозволяли його використовувати одразу:

– слова містили не українські літери, а російські та англійські;

– у словах були символи, які не використовуються при написанні імен;

– слова були у різних регістрах.

Вирішивши дані проблеми за допомогою програм для автоматичного редагування текстів великого об’єму, було розпочато роботу над наступним підходом розпізнавання іменованих сутностей.

Наступні підходи, які були реалізовані, це були підходи на основі правил.

```
FIRST_LAST = rule(FIRST, LAST)

LAST_FIRST = rule(LAST, FIRST)

FIRST_DICT = set(load_dict('first.txt'))
MAYBE_FIRST_DICT = set(load_dict('maybe_first.txt'))
LAST_DICT = set(load_dict('ukr_last.txt'))

IN_FIRST = dictionary(FIRST_DICT)
IN_MAYBE_FIRST = dictionary(MAYBE_FIRST_DICT)
IN_LAST = dictionary(LAST_DICT)

FIRST_MIDDLE = rule(FIRST, MIDDLE)

FIRST_MIDDLE_LAST = rule(FIRST, MIDDLE, LAST)

LAST_FIRST_MIDDLE = rule(LAST, FIRST, MIDDLE)

NAME = or_(
    FIRST_LAST, LAST_FIRST,

    ABBR_FIRST_LAST,
    LAST_ABBR_FIRST,
    ABBR_FIRST_MIDDLE_LAST,
    LAST_ABBR_FIRST_MIDDLE,

    FIRST_MIDDLE,
    FIRST_MIDDLE_LAST,
    LAST_FIRST_MIDDLE,

    JUST_FIRST,
    JUST_LAST,
).interpretation(
```

Рисунок 3.3 – Приклад правил для знаходження іменованих сутностей

Перший із них це підхід який на основі правил. Алгоритм працює наступним чином:

– загружає у пам’ять словник українських імен, прізвищ та по батькові;
– приводить усі слова в тексті в нижній регістр;
– на основі правил знаходить усі цікавлячи нас сутності.

Даний алгоритм має такі переваги перед попереднім:

– має можливість легко адаптуватися до нових вимог;
– більш швидке знаходження сутностей;
– ім’я та прізвище для однієї особи зразу об’єднуються.

Також у такого підходу є великий недолік, а саме те, що він погано працює із закінченнями слів. Наприклад алгоритм легко знаходить імена Марина, Вікторія і Наталка, але не знаходить такі відмінки імен, як Маринин, Вікторії та Наталкою. Для вирішення цієї проблеми було вирішено застосувати наступний підхід, який включав би та усував би даний недолік, але не сильно зменшував швидкість роботи алгоритму. Алгоритм включає наступні кроки:

– загружає у пам’ять словник українських імен, прізвищ та по батькові;
– приводить текст у нижній регістр;
– нормалізує слова;
– на основі правил знаходить усі цікавлячи нас сутності.

Нормалізатор, який використовувався в цьому алгоритмі, був pymoprphy, який описувався у попередньому розділі і словник, на якому базувався pymorphy, також описувався у попередньому розділі. Новий алгоритм усуває всі недоліки попереднього крім одного – точність, вона залишилась не задовільною для великих художніх текстів. Детальніше про точність буде описано у наступному розділі. Також помічено, що алгоритм дуже часто знаходить слова, які не є іменами в тексті Одним із кроків, який мав би збільшити точність алгоритму була нормалізація тексту до його опрацювання [33], а саме:

– видалити із тексту усі прийменники(без, в, від);
– видалити із тексту усі сполучники (і, та, але, й);

– видалити із тексту усі частки (ні, не, де) ;
– розбити, текст на окремі речення;
– видалити всі розділові знаки;
– видалити усі слова, які починаються не з великої літери.

Дані кроки нормалізації були перевірені, як окремо кожний так і усі разом. Несподівано крок 4 і 5 знизили точність існуючого алгоритму. У свою чергу перші три кроки не дали значних приростів у точності алгоритму. Ця нормалізація проводилась і з іншими варіаціями алгоритму, але до зростання точності вона ніколи не приводила. Крок 6 приводив до збільшення точності для деяких даних, але він також приводив до значного падіння точності алгоритму на великій кількості текстів.

Наступним етапом, який збільшує точність алгоритму був етап додавання тегів, отриманих з нормалізатора. Як ми пам'ятаємо з попереднього розділу, наш словник у форматі openCorpora має додаткові теги, що показують чи є слово ім'ям прізвищем або по батькові. Додавши обробку цих тегів у набір правил ми отримали наступний вигляд правил.

Цей варіант алгоритму має такі сильні сторони, як швидкість роботи, стійкість до різних форм відмінку ім'я, по батькові, або прізвища, але також у даного алгоритму є проблеми з точністю, а саме велика кількість позитивно негативних випадків спостерігається коли алгоритм вирішує, що дане слово є іменною сутністю, але воно таким не вважається.

Проаналізувавши проблеми і прочитавши про успіхи сучасних технологій у інших мовах, було зроблено висновок, що потрібно спробувати поєднати існуюче рішення з методом умовно імовірних полів (Conditional Random Fields CRF)[34]. Метод CRF, яке відноситься до статистично лінгвістичних методів[35]. Даний метод є однією із можливих реалізацій Марковських випадкових полів. Марківським імовірним полем або Марківською імовірною мережею називають графову модель, котра використовується для сумісних розподілів даних випадкових перемінних. Формально Марківське випадкове поле складається з наступних компонентів:

– неорентований граф або фактор-граф G=(V,E), де кожна вершина є випадковою змінною X і кожне ребро являє собою залежність між двома випадковими величинами;
– набір потенціальних функцій або факторів.

Вершини, які не є суміжні, повинні відповідати умовно незалежними випадковими величинами[36].

Підсумовуючи все вищесказане отримуємо що, метод умовно ймовірних полів є модель, яка по значенню елемента, та попереднім елементам дає передбачення, щодо наступного.

Для створення гарної CRF моделі необхідно велика кількість розмічених даних, також потрібна значна кількість обчислювальних машин для навчання моделі і якщо останню проблему дуже легко вирішити, то перша проблема не була розв'язана[37]. Замість CRF моделі, яка натреована на українських розмічених корпусах даних, у роботі використовується модель, що була навчана на корпусі російських слів. Російська мова була вибрана не випадково, багато імен, прізвищ пишуться однаково як на російській мові так і на українській, наприклад, Тарас, Адам, Олена, Антон, Богдан, Вадим та багато інших. Потрібно зазначити, що модель CRF використовує формат IOB(скорочення від inside, outside, beginning). Формат IOB (короткий для внутрішньої, зовнішньої, початкової). Це є звичайний формат тегів для позначення лексем у задачі обчислювальної лінгвістики (наприклад, розпізнавання названих сутностей). Її представили Рамшо та Маркус у своїй роботі "Текстовий фрагмент за допомогою навчання, заснованого на трансформації", 1995 р. Префікс B перед тегом вказує, що тег – це початок фрагменту, а префікс I перед тегом вказує, що тег знаходиться всередині шматка. Тег B використовується лише тоді, коли за тегом слідує тег того самого типу без O лексем між ними. Тег O вказує, що маркер належить жодному відриву[38].

Після даних змін, якість роботи алгоритму збільшилась через те, що він перестав виявляти імена, що не є іменами. Також додавання даної моделі до набору правил знизило швидкість обробки інформації, але не суттєво, також аналогічні

зміни відбулися і в споживанні потужностей центрального процесору комп'ютера, а також збільшилась кількість оперативної пам'яті, яка споживається програмою. Подальші вдосконалення алгоритму потребували або нових більш якісних словників українських слів, щоб більше словоформ мали тег ім'я, прізвище або по батькові, або створення великої кількості розмічених даних для навчання crf моделі. Обидва ці шляхи неможливо вирішити за короткий час. Тому було вирішено змінити підхід. Перш ніж обговорювати новий підхід до пошуку іменованих сутностей в текстах, потрібно показати, які складні і актуальні задачі були вирішені за допомогою інфраструктури, створеної під час перевірки першого підходу. Перше, що було зроблено, це створено набір правил, які допомогають легко знайти усі грошові суми у тексті.

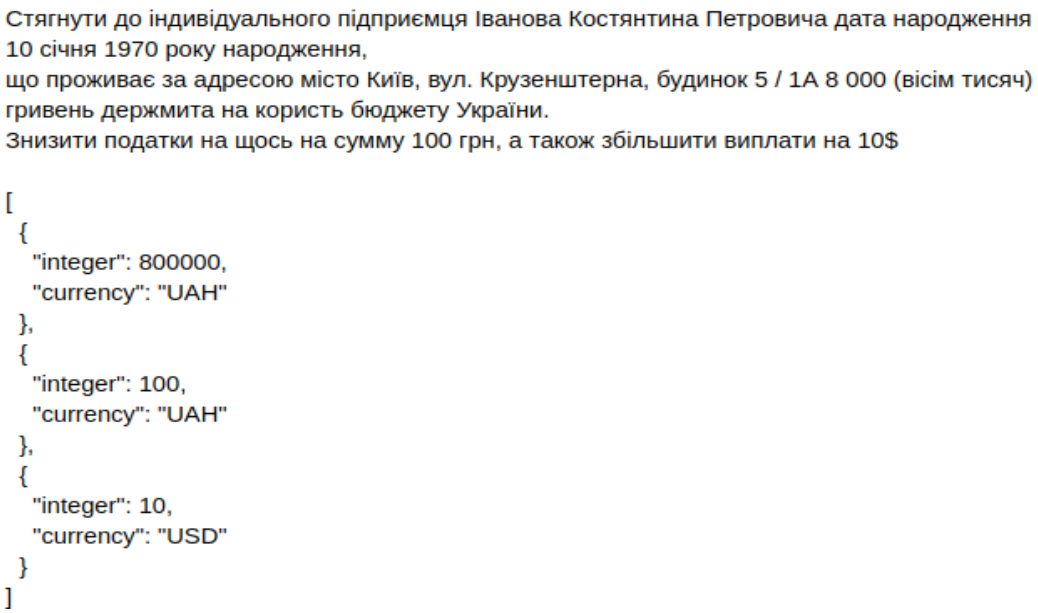

Рисунок 3.4 – Результат роботи набору правил для пошуку грошових одиниць.

Також було зроблено набір правил для пошуку дат у тексті[39]. Цей набір дат працює з з наступними форматами дат:

– з датами у форматі день.місяць число;

– з датами у форматі рррр рік/року;

– з датами у форматі день місяць;

– з датами у форматі місяць рік.

Слід зазначити, що ці правила не знаходять текст у форматі перше вересня, або третє травня, через те, що правила не налаштовані на це.

```
text = ""1 вересня найбільше свято для школярів. А 15 травня 2020 року відбувається
захистю. це просто дата яку знаходить парсер 18-02-2020.
Ще який дивний рік 1234 р.
 Грудень 1941 був дуже суворим""
[
  {
    "month": 9,
    "day": 1,
    "current_era": true
  },
  {
    "year": 2020,
    "month": 5,
    "day": 15,
    "current_era": true
  },
  {
    "year": 2020,
    "month": 2,
    "day": 18,
    "current_era": true
  },
  {
    "year": 1234,
    "current_era": true
  },
  {
    "year": 1941,
    "month": 12,
    "current_era": true
  }
]
```

Рисунок 3.5 – Приклад роботи правил по знаходженню дат.

Однією з потужних можливостей алгоритму є розпізнавання адреси у текстах[40]. Набір правил, який можна покращувати і змінювати, на теперішній час може легко знаходити такі сутності як:

– номера будинків;

– назви вулиць;

– назви міст.

```
text = ""я мешкаю за адресою місто Київ вулиця  Рокосовського будинок 25 квартира 17, а
моя бабуся у селі Нова-Гусарівка вул. Центральна б. 7""
 [
    {
      "name": "Київ",
      "type": "місто"
    },
    {
      "name": "Рокосовського",
      "type": "вулиця"
    },
    {
      "number": "25",
      "type": "будинок"
    },
    {
      "number": "17",
      "type": "квартира"
    }
  },
  {
    {
      "name": "Нова-Гусарівка",
      "type": "село"
    },
    {
      "name": "Центральна",
      "type": "вулиця"
    },
    {
      "number": "7",
      "type": "будинок"
    }
  }
]
```

Рисунок 3.6 – Результат роботи правил з розпізнавання адрес

Останню ідею, яку було перевірено під час цієї роботи з розпізнавання іменованих сутностей, була ідея використання алгоритму BERT на українських текстах[41]. Основна ідея цього алгоритму дуже проста, вчені запропонували подавати на вхід нейронній мережі фрази, в котрих 15% слів замінили на [MASK] і навчили цю нейронну мережу передбачати ці вилучені слова. Наприклад, якщо подаємо на вхід фразу “Я прийшов до [MASK] і ліг на свій улюблений [MASK]”, вона повинна на виході показати слова “дому” і “диван”. Це спрощений приклад, на довших текстах кількість варіантів менша, а нейронна мережа відповідає більш однозначно[42].

Так як для навчання BERT необхідно мати велику кількість текстів, а це було неможливо зробити за короткий час, то було прийнято рішення використовувати BERT, який натренований компанію Google на усіх текстах в вікіпедії. В українській мові вже були спроби перевірити якість роботи BERT на українських текстах, але тодішні спроби не показали гарних результатів. Під час теперішньої перевірки BERT продемонстрував кращі результати, які можна побачити в наступному розділі

3.4 Результати практичних досліджень

В практичній частині даної роботи було перевірено декілька варіантів нейронних мереж для вирішення задачі аналізу сентиментую. Навчання нейроннох мережі було зроблено на 24 тисячах текстів 12 тисяч позитивних і 12 тисяч негативних. Перевірка бура зроблена на 4 тисячах текстів з яких 2 тисячі були позитивними і 2 тисячі негативними. У кожному тексті було не більше 1000 символів. Додатково була зроблена перевірка на маленькому датасеті, якій був створений власними руками але ніяких високих результатів ми не очікували, через те, що датасет сильно відрізнявся за нобором слів від датасету, який використовувався для навчання. Нейронні мережі порівнювались за одним

основним параметром, таким як точність, та двома додатковими, такими як швидкість навчання та точність на маленькому датасеті. В рамках практичнох перевірки були підтверджені теоретичні міркування стосовно швидкості навчання нейронних мереж, а саме що найповільнішею нейронною мережою була LSTM-RNN, потім йде GRU-RNN, потім звичайна RNN і найшвидшою виявилась просто нейронна мережа. Також під час навчання усіх нейронних мереж була виявлена проблема перенавчання нейронних мереж, а саме те що точність на навчальному наборі даних росте, а на тестовому падає. Під час навчання рекурентних нейроних мереж текст перетворювався у цифрові значення задопомогою статистичного підходу.

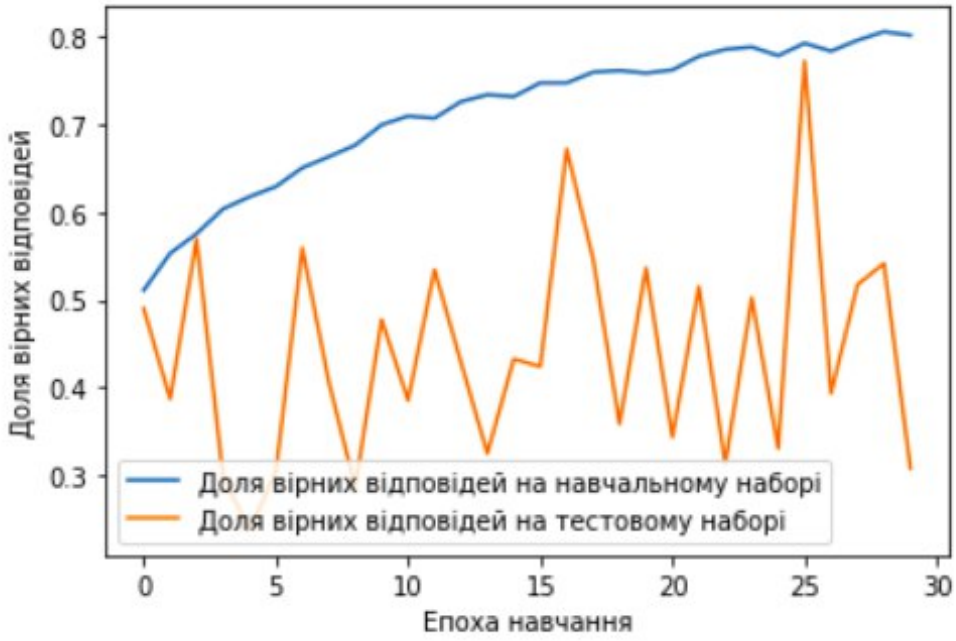

Рисунок 3.7 – Приклад перенавчання нейронної мережі

Варто зазначити що в таблицю результатів занесені лише найкращі конфігурації нейронних мереж. Також для звичайних нейронних мереж високу точність показав алгоритм one hot encoding. Варто зазначити що все навчання проходило на путужностях компанії Google, а саме безкоштовним потужностям відеокарти Tesla 80K, яка є доступною всім користувачам чого сервісу. Детальні результаті зображені на таблиці нижче:

Таблиця 1 – Результати перевірки нейронних мереж для задачі аналізу сентименту

	Точність на великому наборі(%)	Точність на малому наборі(%)	Швидкість навчання однієї епохи(сек)
Simple NN	50	42	3
Simple NN(One hot encoding)	80	35.5	**3**
RNN	60	39	9
LSTM-RNN	**83**	40	105
GRU-RNN	**81**	42	25

В цій роботі перевірялися декілька теоретичних підходів, які існують у світі для вирішення проблеми розпізнавання іменованих сутностей для інших мов. Перший такий підхід був побудований на групі правил. До переваг даного підходу можна віднести легкість змінювання та швидкість роботи. Зазначений підхід спочатку було перевірено на простих реченнях, які складаються з 3-10 слів і кожне речення містить від 1 до 3 іменованих сутностей. Речення у вхідному датасеті містили як українські імена, так і іноземні. У результаті точність даного підходу була досить високою – 89%. Варто окремо відмітити те, що алгоритм працював дуже швидко: для обробки кожного з речень йому було потрібно менше ніж 100мс. Після того, як ми підтвердили працездатність такого підходу на простому датасеті, було здійснено його перевірку на більшому датасеті, який складається більше ніж з 250 текстів, розмір кожного тексту від 100 до 1000 слів. Кожен текст містив від однієї до двадцяти іменованих сутностей. На цьому експерименті було виявлено, що алгоритм показує не дуже велику точність, а саме 65%. Алгоритм іноді не виділяв всі частини ім’я. Наприклад, алгоритм виділив прізвище і ім’я, але ігнорував по батькові. Також була виявлена проблема з пошуком іноземних імен, більшість незнайдених імен були іноземні імена. Інша частина помилок була пов’язана з тим, що алгоритм не правильно знаходив прізвища. Позитивним

моментом є те, що швидкість роботи алгоритму зросла лінійно, і на обробку кожного тексту, алгоритм, який будувався на правилах, витрачав від однієї до десяти секунд.

Подібні маніпуляції були проведені і з алгоритмом, який базується на нейронній мережі BERT. Для цього було взято приклад моделі і способу її навчання від компанії DeepPavlov. Після апробації данного підходу на першій групі тестових даних було виявлено, що цей підхід потребує більше комп'ютерних ресурсів, а саме оперативної пам'яті. Перший алгоритм використовував 100 мб оперативною пам'яті, а нейронні мережі використовували 1Гб. Також було помітно, що даний алгоритм більш повільний, для пошуку іменованих сутностей у тексті йому було необхідно від однієї до трьох секунд. Нейронні мережі показали дуже високу якість на обох наборах даних, на першому наборі якість була 89%, на другому 76%. Усі тестові експерименти проводились на комп'ютері MacBook Pro 2019 року випуски з розміром оперативної пам'яті 32Гб з 8-ядерним процесором Intel Core i9.

Дані отримані під час тестування наведені у таблиці 2.

Таблиця 2 – Результати тестування алгоритмів на текстах великого розміру

	Точність(F1 критерій)	Швидкість	Навантаження на процессор	Кількість оперативної пам'яті
Підхід на основі регулярних виразів	33%	12000 слів/хв	15%	165 МБ
Підхід на основі правил	65%	8000 слів/хв	12%	300 МБ
Підхід на основі BERT	76%	4300 слів/хв	30%	1000 МБ
Алгоритм з lang-uk	62%	3900 слів/хв	35%	800МБ
Існуючий алгоритм на основі регулярних виразів з гітхабу	48%	11600 слів/хв	12%	200 МБ

Дуже цікавий результат був зазначений для текстів меншої довжини, а саме те, що точність алгоритму, який базується на наборі правил значно зросла і стало дорівнювати точності алгоритму Bert. Варто зазначити, що усі алгоритми збільшили свою точність на даному наборі даних. Значних змін у показниках швидкості та використання ресурсів процесору не було помічено. Дані отримані у ході практичної перевірки наведені у таблиці 3.

Таблиця 3 – Результати тестування алгоритмів на текстах малого розміру

	Точність(F1 критерій)	Навантаження на процессор	Кількість оперативної пам’яті
Підхід на основі регулярних виразів	43%	15%	155 МБ
Підхід на основі правил	89%	12%	280 МБ
Підхід на основі BERT	88%	30%	950 МБ
Алгоритм з lang-uk	71%	33%	770МБ
Існуючий алгоритм на основі регулярних виразів з гітхабу	58%	12%	190 МБ

Варто зазначити, що проблема часткового розпізнавання іменованих сутностей не є актуальною для даного алгоритму. Також алгоритм однаково добре знаходив, як іноземні імена так і українські

4 ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ У НАУКОВУ ТА ПРАКТИЧНУ ДІЯЛЬНІСТЬ

4.1. Опис можливості використання отриманих результатів у науковій і практичній діяльності.

Однією з частин цієї роботи була підготовка бази, а саме оцінка існуючих рішень, підходів та ідей, які існують для вирішення проблеми оцінки тональності текстів. На базі цього аналізу були зроблені практичні реалізації алгоритмів, які показали високу ефективність алгоритмів на основі нейронних мережах і не ефективність алгоритмів на основі правил та словників. Основними критеріями для оцінки алгоритмів на основі нейронних мереж була точність на датасеті схожого змісту, а також на датасеті, який не описує тематику навчального датасету. Додатковим критерієм була швидкість навчання нейронних мереж. Створений датасет для навчання нейронних мереж є першим відкритим датасетом текстів на українській мові розміром більше ніж 24 мільйони символів.

У цій роботі на практиці було реалізовано декілька теоретичних і практичних підходів, які існують у світі для вирішення проблеми розпізнавання іменованих сутностей на різних мовах. Дані результати були порівняні з реальними алгоритмами, які вже написані для української мови[43].

Основними трьома критеріями, обраними для оцінки результатів роботи алгоритмів, були точність (якість) розпізнавання іменованих сутностей, легкість зміни об’єктів для розпізнавання та швидкість роботи алгоритму[44]. Було виявлено, що коли розпізнаються сутності в коротких реченнях до 10 слів, то якість алгоритмів заснованих на правилах та нейронних мережах майже однакова, але зі збільшенням розмірів тексту якість нейронних мереж підвищується. Таким чином було виявлено, що для написання чат боту або аналізу твітів підходить і алгоритм заснований на правилах, так як він значно швидший ніж нейронна мережа. Особливо важливим артефактом, який був отриманий під час цієї роботи, є програма для створення правил, щоб розпізнавати іменовані сутності в напів

структурованих текстах[45]. Ця програма дозволить значно пришвидшити збагачення даними такі оцифровані тексти, як нотаріальні рішення, судові позови, вироки і т.д. Потрібно відмітити, що якість реалізованих алгоритмів не гірша, а на деяких випадках і краща ніж алгоритмів, які вже були реалізовані для української мови.

4.2. Питання подальшого дослідження

Для надання відповідей на важливі питання, та перевірки всіх підходів, які існують для вирішення проблеми розпізнавання іменованих сутностей необхідно проводити додаткові роботи: аналіз рішень для виявлених проблем та пошук існуючих рішень для менш поширених мов[46].

Одним із подальших питань, які потрібно вирішувати, це точність роботи системи. Як було зазначено вище, точність для речень, які не дуже великі, досить висока але точність алгоритмів падає коли об’єм тексту збільшується. Особливо важливою є проблема знаходження іменованих сутностей, які насправді не є такими. Ця проблема актуальна для обох підходів, які було розглянуто і реалізовано у даній роботі.

Проблема, що є дуже актуальною для рішення, яке базується на нейронній мережі і менш актуальною для рішення, яке базується на основі правил, це проблема швидкості роботи рішення. На даний момент швидкість роботи нейронної мережі невелика, це призводить до необхідності розгортання додаткових потужностей для пришвидшення роботи, а також робить систему.

Окремою проблемою, яка актуальна тільки для нейронних мереж і ніяк не пов’язана з алгоритмами, побудованими на базі правил, є швидкість навчання та донавчання. Наразі швидкість навчання нейронної мережі дуже низька, як і швидкість її донавчання. Ця проблема робить неможливим або ускладненим

застосування алгоритмів побудованих на нейронній мережі в системах, які потребують донавчання.

Дуже важливим напрямком покращення алгоритму, який базується на наборі правил, є напрям збільшення слів у словнику. Також збільшення якості словнику, а саме додаткова розмітка слів. Такі кроки дозволять значно покращити якість роботи алгоритму[47].

Для алгоритмів аналізу сентименту неструктурованих текстів на українській мові необхідно, перш за все потрібно збільшити датасет текстами довільної тематики. Окремою проблемою, яку необхідно дослідити і вирішення якої може призвести до покращення отриманого результату, це реалізація алгоритму word2vec, для українських слів. Вирішення цієї проблеми дозволить перевірити якість згорткових нейронних мереж. Цікавим напрямом є використання алгоритму BERT для аналізу сентименту текстів але щоб його перевірити необхідно зібрати велечезний датасет украхнських текстів та знайти обладнення для навчання мережі.

ВИСНОВКИ

У процесі виконання теоретичних досліджень та їх практичної перевірки в рамках цієї магістерської роботи з питання розпізнавання іменованих сутностей і аналізу сентименту в неструктурованих текстах, що написані українською мовою, було проведено аналіз актуальності теми, вивчення поширених теоретичних підходів і існуючих практичних рішень визначеної проблеми для інших мов, розглянуто методи оцінки точності алгоритмів, а також виконано практичну реалізацію теоретичних досліджень.

В ході роботи для аналізу сентименту було теоретично запропоновано два методи: перший базується на нейронних мережах, інший на наборі правил. Практично показано не ефективність методу на наборі правил. Було виявлено проблему навчання мережі для вирішення даної задачі, а саме відсутність розмічених датасетів. Для даної проблеми були запропоновані і реалізовані методи вирішення цієї задачі. У результаті роботи був створений первший відкритий розмічений датасет текстів українською мовою для аналізу сентиментів. В ході практичної перевірки алгоритмів на основі нейронних мереж була показана висока ефективність рекурентних мереж, а також запропоноване подальше дослідження згорткових нейронних мереж та мережі для аналізу сентименту на основі BERT.

Під час аналізу алгоритмів і базових ідей для вирішення проблеми розпізнавання іменованих сутностей з’ясувалося, що ця задача є недостатньо дослідженою для текстів на українській мові. Як результат цієї роботи було запропоновано і реалізовано два алгоритми для пошуку іменованих сутностей у структурованих і неструктурованих текстах.

Перший алгоритм базується на побудові правил для пошуку сутностей і має високу точність в структурованих текстах, також його можна легко адаптувати під потреби конкретної задачі. Під час практичної перевірки теоретичних досліджень продемонстровано ефективну реалізацію цього методу для такої інформації, як

адреси, гроші та дати. Недоліком даного підходу для виявлення імен є середня точність, що складає 65%.

Другий алгоритм, побудований на технології BERT показав точність 76%, що на 14% більше ніж точність у найкращого алгоритму з lang-uk, який базується на цій самій технології. Недолік другого алгоритму полягає в тому, що процес його адаптації під іншу задачу потребує значних витрат для перенавчання нейронної мережі, також швидкість роботи цього алгоритму менша за швидкість роботи алгоритму з правилами.

Теоретичні та практичні дослідження, проведені під час цієї магістерської роботи показали що вивчені, покращені та адаптовані алгоритми для знаходження іменованих сутностей в текстах на українській мові виявились кращими ніж усі існуючі аналоги, які є у відкритому доступі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) О. Вешняковська Компьютеры учатся добывать из текста смысл. Журнал Наука и жизнь 2014. Вип. 12. С.49-58

2) Бондаренко М. Ф., Осыка, А. Ф. Автоматическая обработка информации на естественном языке: навч. посіб – Киев: УМК ВО, 1991 – 364 С.

3) Ермаков А. Е., Киселев С. Л. Лингвистическая модель для компьютерного анализа тональности публикаций СМИ. Компьютерная лингвистика и интеллектуальные технологии: навч. посіб – Киев: Наука, 2005 – С. 282-285.

4) Шатовская Т., Каменева, И. Интегрированный подход текстовой кластеризации для неструктурированных документов: навч. посіб. – Винница: AMT, 2008. – 231 С.

5) Ritter A., Clark S., Etzioni O. Named entity recognition in tweets: an experimental study. In Proceedings of the conference on empirical methods in natural language processing: France: WSX, 2011. – pp. 1524-1534.

6) Gudivada V. N. Rao D. Raghavan V. V. Big data driven natural language processing research and applications. In Handbook of Statistics: Amsterdam: Elsevier, 2015. – pp. 203-238.

7) Zhao M., Masino A. J., Yang C. C. A framework for developing and evaluating word embeddings of drug-named entity. In Proceedings of the BioNLP: Amsterdam: Elsevier, 2018. – pp. 156-160.

8) Филиппова Е. Анализ тональности текста: концепция, методы, области применения. URL: http://datareview.info/article/analiz-tonalnosti-teksta-kontseptsiya-metodyi-oblasti-primeneniya/ (дата звернення 03.03.2020)

9) Gerani S., Carman M. J., Crestani F. Proximity-based opinion retrieval. Michigan: AIN, 2015. – pp. 403-410.

10) Кукушкин А. Наташа – библиотека для извлечения структурированной информации из текстов на русском языке. URL: https://habr.com/ru/post/349864/ (дата звернення 07.04.2020)

11) R. Devlin J., Chang M. W., Lee K., Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding: SF: AI, 2018. – 392 p.

12) Валенда Н. А. Методы анализа естественного языка на основе функциональной модели семантики / Н. А. Валенда, Г. Ф. Дюбко // Бионика интеллекта : науч.-техн. журн. – Х. : Изд-во ХНУРЭ, 2005. – № 2(63). – С. 48-52.

13) Chen S., Billings S. A., Grant, P. M. Non-linear system identification using neural networks. International journal of control – NY: AIN, 2013. – pp. 1191-1214.

14) Мокроусов, М. Н. Автоматизированная система нормализации естественно-языковых текстов. Интеллектуальные системы в производстве – Москва: Наука и жизнь, 2015. – С. 93-96.

15) Lopez M. M., Kalita J. Deep Learning applied to NLP: Michigan: AIN, 2017. – 542 p.

16) Xia T., Chai, Y. An improvement to TF-IDF: Term Distribution based Term Weight Algorithm – SF: JSW, 2011 – pp. 413-420.

17) Turner C. A., Jacobs A. D., Marques C. K., Oates J. C., Kamen D. L., Anderson P. E., Obeid J. S. Word2Vec inversion and traditional text classifiers for phenotyping lupus. BMC medical informatics and decision making– NY: AIN, 2017. – pp. 126.

18) Cho K., Van Merriënboer B., Gulcehre C., Bahdanau D., Bougares F., Schwenk H., Bengio Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation – Berlin: Connel, 2014. – pp. 126.

19) Ghosh S., Vinyals O., Strope B., Roy S., Dean, T., Heck L. Contextual lstm (clstm) models for large scale nlp tasks. – Osteraich: AMK, 2016. – pp. 286.

20) Yin W., Kann K., Yu M., Schütze H. Comparative study of cnn and rnn for natural language processing. – Berlin: Connel, 2017. – pp. 126.

21) Nakov P., Ritter A., Rosenthal S., Sebastiani F., Stoyanov V. Sentiment analysis in Twitter. – NY: Connel, 2016. – pp. 126-143.

22) Chupryna A., Ruban I., Smelyakov K., Arsenov A. Evolution of Convolutional Neural Network Architecture in Image Classification Problems. In Selected Papers of the XVIII International Scientific and Practical Conference on IT and Security ITS 2018. – CEUR Workshop Processing. pp. 35-45.

23) Yerokhin A., Nechyporenko, A., Babii A., Turuta O. A new intelligence-based approach for rhinomanometric data processing. In 2016 IEEE 36th International Conference on Electronics and Nanotechnology. 2016. pp. 198-201

24) Perkins J. Python 3 text processing with NLTK 3 cookbook. Packt Publishing Ltd – Munich: CBA, 2014. – pp. 126.

25) Любанович Б. Простой Python. Современный стиль программирования – Питер: Издательский дом, 2016. – С.34-123

26) McKinney W. Pandas: a foundational Python library for data analysis and statistics. Python for High Performance and Scientific Computing. – Berlin: Connel, 2011. – pp. 43-78.

27) Лябинцев Е. Метрики в задачах машинного обучения. URL: https://habr.com/ru/company/ods/blog/328372/ (дата звернення 15.04.2020).

28) Hutto C. J., Gilbert, E. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In Eighth international AAAI conference on weblogs and social media. – Berlin: Connel, 2014. – pp. 126.

29) Barbosa L., Feng J. Robust sentiment detection on twitter from biased and noisy data. – German: BCV, 2010. – pp. 36-44.

30) Бочаров В. В., Грановский Д. В., Суриков А. В. Вероятностная модель токенизации в проекте Открытый корпус. Новые информационные технологии в автоматизированных системах. – Киев: УМК ВО, 2012 – 286 С.

31) Pelle D. Grammar checker for English, French, German in Vim 2016 URL: https://www.vim.org/scripts/script.php?script_id=3223 (дата звернення 29.04.2020).

32) Глибовець, А. М. Автоматизований пошук іменованих сутностей у нерозмічених текстах українською мовою. Штучний інтелект. – Киев: Знання, 2017 – 208 С.

33) Slanna I. Нормализация текста в задачах распознавания речи. URL: https://habr.com/ru/post/491260/ (дата звернення 15.04.2020)

34) Марченко О. О. Машинно-навчальні методи розпізнавання іменованих сутностей тексту. Проблеми програмування. – Днепр: Наука, 2016 – 259 С.

35) Sutton C., McCallum A. An introduction to conditional random fields. Foundations and Trends in Machine Learning, – German: BCV, 2014. – pp.267-373.

36) Zhang C. Automatic keyword extraction from documents using conditional random fields. Journal of Computational Information Systems. – Munich: JBC, 2008. – pp. 1169-1180.

37) Jie Z., Lu W. Dependency-Guided LSTM-CRF for Named Entity Recognition. – SF: Science, 2019. – pp. 158-197.

38) Carpenter B. Coding chunkers as taggers: – NY: Scientific, 2010. – pp. 23-43.

39) Bai, S., Wu, H. J. P., Li, H., & Loudon, G. U.S. Patent No. 6,311,152. Washington, DC: U.S. Patent and Trademark Office.

40) Whitelaw C., Kehlenbeck A., Petrovic N., Ungar L. Web-scale named entity recognition. – SF: Science, 2008. – pp. 123-132.

41) Emelyanov A. A., Artemova E. Multilingual named entity recognition using pretrained embeddings, attention mechanism and NCRF. – Berlin: Heidelberg, 2019. – pp 256.

42) Akbik A., Bergmann T., Vollgraf R. Pooled contextualized embeddings for named entity recognition. – Munich: BCV, 2019. – pp. 724-728.

43) Романюк А., Карпінська О. Розпізнавання іменованих сутностей при видобуванні інформації з медичних текстів. посіб – Киев: Наука, 2015 – 364 С.

44) Марченко О. О. Розробка системи розпізнавання іменованих сутностей тексту. посіб – Киев: УМК ВО, 2015 – С.152-159.

45) Nadeau D., Sekine S. A survey of named entity recognition and classification. Lingvisticae Investigationes, – Berlin: Springer, 2007. – pp. 3-26.

46) Isozaki H. Japanese named entity recognition based on a simple rule generator and decision tree learning. – SF: Heidelberg, 2012. – pp. 314-321.

47) Abdallah S., Shaalan K., Shoaib M. Integrating rule-based system with classification for arabic named entity recognition. – Berlin: Springer, 2012 – pp. 311-322.