Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів розуміння природної мови для створення структурованого опису на основі текстів
(тема)

Виконав:
студент (ка) 2 курсу, групи ІПЗм-22-6

Бажанов Д.Г.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-наукова

Керівник доц. Турута О.П.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук

Кафедра програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення

Тип програми освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення

(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Бажанову Дмитру Геннадійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи «Дослідження методів розуміння природної мови для створення структурованого опису на основі текстів»

Затверджена наказом по університету від «29» березня 2024 р. №250 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 12.06.2024 р.

3. Вихідні дані до роботи: моделі глибокого навчання, основи обробки природньої мови

4. Перелік питань, що потрібно опрацювати в роботі: *аналіз існуючих методів розуміння природної мови для створення структурованого опису, вибір оптимальних технологій генерації структурованого опису, використання та розробка алгоритмів розуміння текстової інформації.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі та постановка задачі	23.01 – 14.02.24	*виконано*
2	Аналіз та вибір API для дослідження	15.02 – 24.02.24	*виконано*
3	Аналіз та моделювання предметної області	17.02 – 28.02.24	*виконано*
4	Планування експериментів	25.02 – 28.02.24	*виконано*
5	Програмна реалізація кожного з обраних для дослідження API	25.02 – 01.04.24	*виконано*
6	Експериментальні дослідження	02.04 – 20.04.24	*виконано*
7	Аналіз результатів експериментальних досліджень та розробка рекомендацій	20.04 – 23.04.24	*виконано*
8	Написання та оформлення статті та тез доповіді	17.04 – 23.04.24	*виконано*
9	Підготовка пояснювальної записки	01.04 – 26.04.24	*виконано*
10	Підготовка презентації та доповіді	26.04 – 02.05.24	*виконано*
11	Нормоконтроль	03.05 – 08.05.24	*виконано*
12	Рецензування	08.05 – 14.05.24	*виконано*
13	Занесення диплома в електронний архів	15.05.2024	*виконано*
14	Попередній захист	15.06.2024	*виконано*
15	Допуск до захисту у зав. кафедри	18.06.2024	*виконано*

Дата видачі завдання 20 січня 2024 р.

Студент _____ Бажанов Д.Г.
(підпис)

Керівник кваліфікаційної роботи _____ доц. Турута О.П.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи, 56 с., 21 рис., 8 джерел, 5 додатків.

NLP, ГЛИБОКЕ НАВЧАННЯ, ВЕКТОРИЗАЦІЯ СЛІВ, ГЕНЕРАЦІЯ СТРУКТУРОВАНОГО ОПИСУ.

Об'єктом дослідження в даній роботі є аналіз методів розуміння природної мови (NLP) для створення структурованого опису на основі текстів. Метою даного дослідження є дослідження ефективних алгоритмів та програмного забезпечення для автоматичної генерації структурованого опису з використанням інструментів обробки природної мови. У роботі досліджуються техніки глибокого навчання в області NLP, такі як рекурентні нейронні мережі (RNN), трансформери та векторизація слів (Word Embeddings), такі як Word2Vec, GloVe, FastText. Також враховується аналіз сутностей, розуміння відносин між сутностями та класифікація текстів з метою визначення тем та ключових аспектів. Методи розробки ґрунтуються на теорії математичного моделювання та використанні моделей ШІ для обробки та аналізу текстової інформації. В ході дослідження сплановано комплекс інструментальних засобів для програмної реалізації методів обробки природної мови, спрямованих на генерацію структурованого опису будь-якої складності на основі текстової інформації. Результатом даної роботи є прототип ефективного інструментарію для автоматизованого створення подій на основі текстів, що може бути застосований в різних областях, таких як медицина, фінанси, технічна підтримка та інші. Розглядаються також етичні аспекти використання подібних технологій, а також найновіші тенденції у галузі обробки природної мови.

NLP, DEEP LEARNING, VECTORIZATION OF WORDS, GENERATION OF STRUCTURED DESCRIPTION

The object of research in this work is the analysis of natural language understanding (NLP) methods for creating a structured description based on texts. The purpose of this

research is to investigate effective algorithms and software for the automatic generation of a structured description using natural language processing tools. The work explores deep learning techniques in the field of NLP, such as recurrent neural networks (RNN), transformers and vectorization of words (Word Embeddings), such as Word2Vec, GloVe, FastText. Entity analysis, understanding relationships between entities, and text classification to identify themes and key aspects are also taken into account. Development methods are based on the theory of mathematical modeling and the use of AI models for processing and analyzing textual information. In the course of the research, a set of tools is planned for the software implementation of natural language processing methods aimed at generating a structured description of any complexity based on textual information. The result of this work is a prototype of an effective toolkit for the automated creation of events based on texts, which can be applied in various fields, such as medicine, finance, technical support, and others. Ethical aspects of the use of such technologies are also considered, as well as the latest trends in the field of natural language processing

Я, Бажанов Дмитро Геннадійович, студент гр. ІПЗм-22-6, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя робота на тему «Дослідження методів розуміння природної мови для створення структурованого опису на основі текстів», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

NPL – Natural language processing

RNN – Recurrent neural network

NN – Neural network

WE – Word Embeddings

GPT – Generative Pre-trained transformer

API – Application Programming Interface

LSTM – Long Short-Term Memory

GRU – Gates Recurrent Unit

BERT – Bidirectional Encoder Representations from Transformers

NLU - Natural Language Understanding

ВСТУП

У світі зростаючої обсягом інформації, здатність ефективно розуміти та обробляти природну мову стає визначальною у сфері обчислювальної лінгвістики та інтелектуального аналізу текстів. Дослідження методів розуміння природної мови (NLP) з метою створення структурованого опису на основі текстової інформації стає актуальною задачею в контексті постійного розвитку технологій та зростання важливості обробки мови для великої кількості застосувань.

У віртуальному світі, де інформаційний потік безперервний і непереглядний, ефективні та точні методи розуміння природної мови стають крайньо важливими для розвитку інтелектуальних систем та забезпечення їх взаємодії з користувачами. Здатність автоматично аналізувати та структурувати текстову інформацію має великий потенціал у багатьох сферах, від обробки великих обсягів даних у бізнесі до покращення пошукових систем та підтримки прийняття рішень в науці та медицині.

Це дослідження ставить перед собою завдання розробки програмного забезпечення із використанням існуючих алгоритмів розуміння природної мови та їх вдосконалення із метою генерації структурованого опису. Проаналізувавши сучасні підходи та використовуючи передові методи глибокого навчання, ми спрямовані на створення ефективного інструментарію, який дозволить автоматично витягувати суттєву інформацію з текстових джерел та створювати структурований опис.

Враховуючи розмаїття застосувань, від сфери бізнесу та фінансів до науки та гуманітарних наук, дослідження NLP відкриває нові перспективи в розвитку інтелектуальних систем. Велика увага приділяється також етичним аспектам використання цих технологій та їхньому впливу на суспільство.

Це дослідження сподівається внести вагомий внесок у розвиток області 6 обробки природної мови та створення передових методів для генерації структурованого опису на основі текстів, реагуючи на виклики сучасності та покликане сприяти подальшому розвитку інтелектуальних систем.

1 АНАЛІЗ МЕТОДІВ АНАІЗУ ПРИРОДНЬОЇ МОВИ

1.1 Аналіз базових підходів для розуміння природної мови

Аналіз базових підходів до розуміння природної мови (Natural Language Understanding, NLU) є ключовим етапом у вивченні методів для створення структурованого опису на основі текстів. Розглядаючи різні наукові та прикладні роботи, можна виділити кілька основних напрямків та підходів, які домінують у сучасних дослідженнях з цієї області [1].

Один із популярних підходів використовує семантичні мережі та векторні представлення слів [3]. Моделі, такі як Word2Vec чи GloVe, базуються на ідеї, що слова, які часто зустрічаються разом, мають схожий семантичний зміст. Ці векторні представлення дозволяють враховувати семантичні відносини між словами, що полегшує розуміння текстової інформації.

Застосування глибокого навчання та рекурентних нейронних мереж є іншим важливим напрямком в розумінні природної мови [2]. Використання довгих короткочасних пам'ятей (LSTM) та gated recurrent units (GRU) дозволяє моделям враховувати контекст та залежності в тексті, сприяючи більш точному розумінню семантики [2].

Останнім часом трансформерні моделі, такі як BERT та GPT, здобули велику популярність. Вони використовують механізми уваги для аналізу контексту великих обсягів тексту. Можливість враховувати довгий історійний контекст робить ці моделі особливо ефективними у завданнях розуміння природної мови.

Ще однією перспективною галуззю є інтеграція мовних моделей з аналізом зображень. Методи, які поєднують обробку тексту та візуальні дані, виявляються дуже ефективними для створення повнішого розуміння контексту.

Аналіз базових підходів до розуміння природної мови підсумовується в поглибленні розуміння їхніх переваг та обмежень. Враховуючи це, виробляється 8 фундаментальна основа для подальших досліджень у напрямку створення структурованого опису на основі текстової інформації.

1.2 Аналіз практичних задач, що вирішуються методами розуміння природної мови

Методи розуміння природної мови (NLU) використовується у різноманітних практичних застосуваннях, відзначаючи себе як ключовий компонент в ефективних технологічних рішеннях. Аналізуючи цей аспект, можна виділити кілька сфер, де розуміння природної мови виявляється особливо корисним та впливовим.

- Мовні асистенти та інтелектуальні чат-боти
- Автоматизована обробка текстової інформації
- Ефективний пошук та категоризація інформації
- Персоналізовані системи рекомендацій
- Медичні та наукові дослідження
- Фінансові аналітичні системи

Розуміння природної мови є критичним для функціонування мовних асистентів та чат-ботів. Вони забезпечують можливість взаємодії з користувачем через природну мову, роблячи комунікацію інтуїтивно зрозумілою та ефективною.

Мовні асистенти, такі як Siri від Apple, Google Assistant та Amazon Alexa, відіграють ключову роль у взаємодії з користувачем. Вони здатні розпізнавати голосові команди та створювати відповіді, що базуються на зрозумілому контексті. Це дозволяє користувачам виконувати завдання, такі як пошук 9 інформації, нагадування про події, або навіть керування побутовою технікою, використовуючи природну мову.

У бізнес-середовищі інтелектуальні чат-боти застосовуються для надання клієнтського обслуговування [6]. Вони можуть взаємодіяти з клієнтами через вебсайти, месенджери або мобільні додатки, вирішуючи запитання, надаючи інформацію про продукти та послуги, а також виконуючи інші завдання. За допомогою розуміння природної мови, чат-боти можуть вести природні та зручні діалоги з користувачами.

Мовні асистенти в сфері технічної підтримки вміють розпізнавати проблеми та надавати рекомендації щодо усунення несправностей. Вони можуть аналізувати

запитання користувачів та надавати конкретні відповіді або направляти їх до відповідних розділів техпідтримки.

В інтелектуальних чат-ботах можна використовувати для освітніх цілей, забезпечуючи користувачам доступ до навчального матеріалу або відповіді на навчальні запитання. Це може бути особливо корисним у віддалених навчальних середовищах або для самостійного вивчення [6].

Мовні асистенти, обладнані функціями аналізу настроїв, можуть взаємодіяти з користувачами на більш особистому рівні. Вони можуть розпізнавати емоції у голосі чи написаному тексті та відповідати залежно від виявлених настроїв. Мовні асистенти та чат-боти, базуючись на розумінні природної мови, перетворюють взаємодію з технологіями у більш доступний та ефективний процес, прискорюючи вирішення різноманітних завдань та полегшуючи повсякденне життя користувачів.

Автоматизована обробка текстової інформації (Automated Text Processing, ATP) представляє собою обширну галузь, де технології розуміння природної мови знаходять широке застосування. В цьому контексті, алгоритми та системи ATP використовуються для аналізу, обробки та інтерпретації текстового контенту. Автоматизований процес обробки тексту включає в себе кілька ключових етапів. На початку стоїть завдання токенізації, де текст розбивається на окремі 10 слова (токени). Далі відбувається лематизація та визначення частин мови для кожного токена, що допомагає встановити основні форми слів та їхні функції у реченні.

Системи ATP використовують методи видобутку інформації для визначення ключових елементів тексту, таких як іменовані сутності (особи, місця, події) та факти. Алгоритми, побудовані на розумінні природної мови, дозволяють виявляти та структурувати цю інформацію, роблячи текст доступним для подальшого аналізу.

У сучасному світі, де великі обсяги текстової інформації генеруються щоденно, важливо мати можливість визначення сентименту або тону тексту. Системи ATP, використовуючи методи машинного навчання, можуть проводити сентимент-аналіз для визначення емоційного ставлення до конкретних тем або продуктів.

В контексті розуміння природної мови, автоматизована обробка тексту може також включати системи машинного перекладу. Технології, що базуються на розумінні мови, дозволяють створювати більш точні та природні переклади між різними мовами.

Алгоритми розуміння природної мови використовуються для розвитку ефективних пошукових систем та категоризації контенту. Це дозволяє швидко та точно знаходити інформацію великого обсягу текстових даних.

Автоматизована обробка текстової інформації в сфері розуміння природної мови допомагає впоратися з великим обсягом текстових даних, зроблюючи їх доступними для подальшого аналізу та використання. Це має велике значення в сферах від бізнесу та науки до освіти та медіа.

Пошукові системи та методи категоризації в текстовому контексті стають важливим елементом в сучасному інформаційному суспільстві, особливо враховуючи постійний приріст обсягу текстової інформації, доступної в Інтернеті та корпоративних базах даних. Розуміння природної мови дозволяє розробляти ефективні методи пошуку, які враховують не лише ключові слова, але і семантичні зв'язки в 11 тексті. Алгоритми NLU можуть визначати контекст та інтенції користувача, що поліпшує релевантність результатів. Додатково, алгоритми можуть враховувати схожість сенсів слів та контексту визначених понять, що дозволяє знаходити не тільки дослівні відповіді, але й ті, що базуються на схожості сенсів.

Однією з важливих функцій розуміння природної мови є автоматична категоризація текстового контенту. Алгоритми можуть автоматично визначати тематику тексту та призначати йому відповідні категорії. Для цього використовуються техніки, які враховують частоту вживання та контекстуальні зв'язки слів. Наприклад, моделі машинного навчання можуть вивчати специфічні слова та поняття, які характерні для різних тематик.

Сучасні пошукові системи використовують елементи штучного інтелекту та машинного навчання для аналізу користувацьких запитань та надання точних та релевантних результатів. Це включає в себе аналіз контексту, розрізнення схожих

слів та розуміння загальної інтенції. Також, системи можуть надавати розширений аналіз результатів, використовуючи додаткові фільтри, такі як дата, місцезнаходження, або типи контенту.

У бізнесі ці технології використовуються для автоматизації пошуку конкретної інформації в корпоративних базах даних, а також для аналізу ринку та конкурентоспроможності. У науці автоматизована категоризація дозволяє ефективно аналізувати великі обсяги документації, відділяючи ключові поняття та результати. Застосування розуміння природної мови в ефективному пошуку та категоризації інформації значно полегшує взаємодію користувача з текстовим контентом. Це забезпечує швидкий доступ до необхідної інформації та вдосконалює загальний досвід використання пошукових систем та баз даних.

Розуміння семантики та контексту дозволяє більш точно відповідати на запитання користувача та надавати збалансовані та релевантні результати. Персоналізовані системи рекомендацій є ключовим компонентом в сучасних інформаційних технологіях, що використовують розуміння природної 12 мови (Natural Language Understanding, NLU) для забезпечення користувачам індивідуалізованих та релевантних рекомендацій. Основним принципом персоналізованих систем рекомендацій є врахування особистих вподобань, історії взаємодії та контексту користувача. Алгоритми NLU аналізують текстові дані, які користувач надає, враховуючи їхнє семантичне значення та інтенції.

Текстові відгуки, описи виборів, або інші форми користувачів взаємодії з платформою надають інформацію про їхні уподобання. NLU алгоритми можуть виокремлювати ключові слова, теми та емоції, що допомагає у створенні точного профілю користувача. Підходи до персоналізації можуть включати рекомендації для фільмів, книг, товарів, новин, тощо.

За допомогою алгоритмів NLU системи можуть розуміти контекст запитань та зворотного зв'язку, допомагаючи вибирати найбільш релевантні варіанти. Прикладами можуть бути персоналізовані списки відтворення на стрімінгових платформах, індивідуалізовані рекомендації в інтернет-магазинах або підбір новинових матеріалів за інтересами користувача. В основі персоналізованих

систем рекомендацій лежать технології машинного навчання, які використовують алгоритми для аналізу та передбачення поведінки користувачів. Враховуючи текстові дані, моделі можуть навчатися розпізнавати патерни та прогнозувати подальші вибори.

NLU взаємодіє з алгоритмами машинного навчання, розширюючи їхню здатність розуміти складність мовленнєвих конструкцій та інтенцій. Це дозволяє системам рекомендацій адаптуватися до змінних уподобань та контексту користувача. Застосування NLU дозволяє аналізувати не лише конкретні слова, а й їхнє семантичне значення та відношення. Це полегшує розуміння того, що саме користувач шукає або очікує.

Аналіз контексту важливий для розуміння зміни в уподобаннях або потребах користувача в різні часові періоди. Наприклад, зміни у тексті відгуку можуть свідчити про нові інтереси або зміну настрою. Використання NLU в персоналізованих системах рекомендацій дозволяє зробити вибірки більш точними та адаптованими до унікальних потреб кожного користувача. Це підвищує задоволення від взаємодії з платформою та сприяє більш глибокому розумінню користувача, що має ключове значення для покращення якості обслуговування та створення персоналізованого віртуального середовища.

Медичні та наукові дослідження, які використовують розуміння природної мови (Natural Language Understanding, NLU), стають важливим компонентом для розвитку новаторських підходів у медицині та науці. Інтеграція NLU в ці області може значно поліпшити якість досліджень та лікування пацієнтів. NLU може бути застосовано для автоматизованої обробки та аналізу масивів медичних записів.

Алгоритми можуть визначати ключові моменти, діагнози, історію лікування та ризикові фактори в текстових документах. Це дозволяє лікарям ефективніше використовувати інформацію та приймати обґрунтовані рішення. NLU може виявляти симптоми та ознаки захворювань у текстових описах стану пацієнтів. Це особливо важливо для автоматичного виявлення рідкісних або складних захворювань, що може полегшити процес діагностики та покращити результати лікування. Використання NLU може покращити проведення клінічних досліджень.

Алгоритми можуть швидко і точно аналізувати великі обсяги текстової інформації, виявляти тенденції та важливі дані, що сприяє ефективному використанню ресурсів та скороченню строків досліджень. З використанням NLU можливо створити індивідуалізовані підходи до лікування, враховуючи не лише клінічні дані, але і особистість пацієнта та його побажання. Це сприяє покращенню результатів лікування та забезпеченню пацієнтів більш якісною медичною допомогою.

NLU може використовуватися для аналізу великої кількості наукових статей та публікацій, швидко знаходячи і визначаючи ключові ідеї, тенденції та перспективи у медичній та науковій області. Враховуючи важливість конфіденційності медичних даних, дослідники та розробники систем з використанням NLU повинні ретельно дотримуватися норм безпеки та етичних стандартів. Медичні та наукові дослідження, зосереджені на використанні NLU, відкривають шляхи для новаторських рішень у медицині та науці. Використання цих технологій може значно поліпшити ефективність діагностики, підвищити якість лікування та сприяти загальному розвитку наукових досліджень.

Фінансові аналітичні системи в сучасному світі використовують передові технології розуміння природної мови (Natural Language Understanding, NLU), щоб забезпечити більш точний та ефективний аналіз фінансових даних та роботу з інформацією. З'єднання цих двох напрямків визначає нові стандарти в області фінансового аналізу та прийняття рішень. Використання NLU дозволяє автоматизувати аналіз фінансової звітності компаній. Системи можуть швидко і точно виділяти ключові фінансові показники, тенденції та ризики, що дозволяє фінансовим аналітикам та керівникам швидше реагувати на зміни в економічному середовищі.

NLU використовується для аналізу новин, звітів та подій, які можуть впливати на фінансові ринки. Системи можуть автоматично визначати ключові слова та теми, враховуючи їхній можливий вплив на стан ринків. Також NLU може бути використано для прогнозування фінансових тенденцій на основі аналізу текстової інформації. Системи можуть виявляти патерни та взаємозв'язки між

різними факторами, що сприяє покращенню точності прогнозів, або використовуватися для моніторингу соціальних мереж та відгуків в інтернеті.

Аналіз публічної думки може бути важливим фактором в прийнятті рішень щодо інвестицій, а також визначенні ризиків та можливостей. NLU може допомагати фінансовим аналітикам та трейдерам приймати кращі рішення щодо оптимізації портфеля. Системи можуть швидко аналізувати великі обсяги текстової інформації, надаючи зрозумілі рекомендації для оптимального розподілу активів.

Отже застосування розуміння природної мови у фінансових аналітичних системах дозволяє покращити якість прийняття фінансових рішень, зробити їх більш обгрунтованими та ефективними. Розвиток цих технологій відкриває нові перспективи для фінансових аналітиків та спеціалістів у галузі, але оскільки фінансова сфера є чутливою щодо етичних аспектів та регулювання, важливо враховувати конфіденційність даних та дотримання встановлених стандартів при використанні NLU у фінансових аналітичних системах.

1.3 Аналіз різновидів структурованих текстів

Головною перепоною у комунікації між людиною і комп'ютером є особливості розуміння інформації. Людині зручніше сприймати звичайний текст з емоційним забарвленням та різноманітними словами. Порядок слів, обдруківки, контекст між рядками – усе це людський мозок легко опрацьовує. Комп’ютер у свою чергу легко обробляє великі об'єми даних, йому не потрібне емоційне забарвлення текстів, а подвійний сенс, обдруківки, порядок слів є проблемою. Тому всі програмні рішення розпочинаються з формулювання плану роботи з даними. Це забезпечує обмін інформацією між програмним забезпеченням, модулями, комп'ютерами і роботами. Так існують інструменти для зберігання інформації і запитів та інструменти для обміну і представлення.

MySQL - одна з найпопулярніших систем управління базами даних, що використовує мову SQL для взаємодії з даними. Вона дозволяє ефективно зберігати, змінювати та отримувати великі об'єми інформації. MySQL часто використовується завдяки своїй продуктивності та надійності.

Oracle - потужна реляційна система управління базами даних, яка відома своєю масштабованістю та можливістю працювати з великими об'ємами даних. Вона широко використовується у великих корпораціях для забезпечення високої продуктивності, надійності та безпеки даних.

PostgreSQL - це об'єктно-реляційна система управління базами даних, яка відрізняється своєю розширюваністю та підтримкою стандарту SQL. Вона забезпечує надійність та ефективність у роботі з даними, часто використовується для складних додатків, де необхідна висока ступінь налаштовуваності.

XML (Extensible Markup Language) - мова розмітки, що використовується для зберігання та передачі даних у структурованому форматі. Вона забезпечує читабельність даних як для людей, так і для машин, що робить її зручною для обміну інформацією між різними системами.

HTML (Hyper Text Markup Language) - основна мова розмітки для створення веб-сторінок. HTML використовується для структурування вмісту в Інтернеті, дозволяючи браузерам правильно відображати текст, зображення та інші медіа-елементи.

JSON (JavaScript Object Notation) - легка текстова мова обміну даними, що є зручною для читання і написання людиною, а також легкою для аналізу і генерації машинами. JSON широко використовується для передачі даних у веб-додатках завдяки своїй простоті та ефективності.

Загалом задача розуміння текстів і створення структурованих описів є актуальною для забезпечення ефективної комунікації між людиною і комп'ютером. Структуровані дані забезпечують чітке і зрозуміле зберігання, обмін та представлення інформації, що сприяє кращій взаємодії між різними програмними компонентами та системами.

2 АНАЛІЗ ПІДХОДІВ ДЛЯ СТВОРЕННЯ СТРУКТУРОВАНИХ ДАНИХ

2.1 – Постановка задачі дослідження

Головною метою магістерської роботи є аналіз та порівняння методів аналізу, обробки і розуміння текстів з метою підготовки структурованих даних, які будуть використовуватись у різних сферах діяльності. Ці сфери включають структуровані дані забезпечують ефективний пошук, зберігання і управління інформацією в інформаційних системах, що підвищує продуктивність та зручність використання таких систем. Також у сучасному світі великі об'єми даних потребують ефективних методів аналізу та обробки. Структуровані дані допомагають виявляти кореляції, тренди та патерни, що можуть бути корисними для бізнесу, науки та технологій. Машинне навчання і штучний інтелект, для них структуровані дані є основою для навчання моделей машинного навчання та алгоритмів штучного інтелекту. Високоякісні дані сприяють точнішому та швидшому навчанню моделей, що підвищує їхню ефективність. Веб-технології та інтернет додатки також використовують структуровані дані для побудови веб-сайтів та додатків, що забезпечують зручний і швидкий доступ до необхідної інформації для користувачів.

Дослідження охоплюватиме огляд сучасних методів обробки та аналізу текстових даних, включаючи методи природної мови (NLP), техніки машинного навчання, алгоритми кластеризації та класифікації текстів. Окрему увагу буде приділено порівнянню різних підходів до створення структурованих даних, їх переваг та недоліків у різних контекстах.

Очікується, що результати цього дослідження дозволять створити ефективні методи та інструменти для автоматизації процесів аналізу та обробки текстової інформації, що значно спростить роботу з даними у зазначених сферах і підвищить загальну ефективність використання інформації.

2.2 – Моделі обробки тексті основані на ймовірносних підходах

Генерація текстового опису може бути виконана різними методами, в залежності від контексту і завдань. Ми подивимося на такі методи генерації як Марковські моделі, машинне навчання, GPT (generative pre-trained transformer), та генерація на основі шаблонів. Почнемо з Марковських моделей - В лінгвістиці та інформаційних технологіях є поняття «ланцюжки Маркова», вони використовуються для моделювання випадкових процесів, таких як послідовності подій або послідовності слів у тексті. Ланцюжок Маркова - це математична модель, що описує послідовність станів, де ймовірність переходу в новий стан залежить від поточного стану. Марковські ланцюжки або моделі базуються на математичній концепції ймовірності. Ідея полягає в тому, що ми можемо прогнозувати ймовірність наступного слова або символу на основі поточного слова або символу. Ланцюжки Маркова використовуються для оцінки процесів та моделювання випадкових змін в часі. У випадку текстового аналізу, ланцюжок Маркова може бути використаний для генерації тексту або прогнозування слів в залежності від попереднього контексту. Основна ідея ланцюжків Маркова це те, що вони вказують на залежність майбутніх станів від поточного і тільки поточного стану і не залежать від попередніх станів, які вже були враховані. Це дозволяє ефективно моделювати імовірності переходів у послідовностях, таких як простий текст.

Приклади застосування моделі Маркова. Розглянемо простий приклад генерації тексту за допомогою Марковської моделі. Припустимо, у нас є текст:

«Кіт сидить на підвіконні. Кіт дивиться на вулицю. Кіт спить.»

Для побудови Марковської моделі, ми можемо розрахувати ймовірності переходів між наступними словами:

- Після "кіт" найчастіше зустрічаються "сидить", "дивиться", "спить".
- Після "сидить" найчастіше зустрічається "на".
- Після "на" найчастіше зустрічається "підвіконні", "вулицю".

Наведемо реалізацію простої Марковської моделі, її можна виконати за допомогою Python:

```python
import random

text = "кіт сидить на підвіконні. кіт дивиться на вулицю. кіт спить."

words = text.split()
word_dict = {}

for i in range(len(words) - 1):
    if words[i] not in word_dict:
        word_dict[words[i]] = []
    word_dict[words[i]].append(words[i + 1])

def generate_text(word_dict, start_word, length=10):
    current_word = start_word
    result = [current_word]
    for _ in range(length - 1):
        if current_word not in word_dict:
            break
        next_word = random.choice(word_dict[current_word])
        result.append(next_word)
        current_word = next_word
    return ' '.join(result)

generated_text = generate_text(word_dict, 'кіт')
print(generated_text)

```

Рис 1 – Приклад примітивної реалізації моделі Маркова

```
кіт спить.


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Рис 2 – Результат роботи примітивної моделі Маркова

Тепер розглянемо переваги та недоліки моделей маркова. Очевидною перевагою моделей Маркова є їх зрозумілість та простота у реалізації, як ми бачили на прикладі вище, простіший варіант моделі Маркова можна створити

використовуючи Python у кілька десятків строк коду. Також слід зазначити що моделі Маркова є дуже ефективними при роботі з малими обсягами тексту.

Розглянемо недоліки моделей Маркова. Основними нелодіками таких моделей є те, що вони розроблені для застосування на малих обсягах інформації і не враховують контексту попередніх текстів. Якщо ми запустимо таку модель для генерації або аналізу великих об’ємів тексту, вона з великою долею імовірності згенерує нелогічні, неузгоджені тексти.

Для покращення якості тексту, згенерованого Марковською моделлю, можна використовувати н-грами, де н-грамою є послідовність з n слів. Наприклад, біграма враховує два попередніх слова, що значно покращує контекстну узгодженість тексту. Слід зазначити що n-грамми не є базовою моделю Маркова а її модифікацією. Якщо базова Марковська модель використовує лише один попередній стан для передбачення наступного, то n-грамні моделі враховують послідовність із n попередніх слів. Це покращує якість тексту, роблячи його більш логічним та узгодженим.

Н-грамні моделі працюють так само, як і базові Марковські моделі, але з тією різницею, що вони враховують більше попередніх слів. Наприклад, біграми (2-грами) враховують два попередніх слова, тріграми (3-грами) – три слова і т.д. Розглянемо приклад тексту і побудови моделі біграм. Візьмемо вже знайомий нам текст:

«Кіт сидить на підвіконні. Кіт дивиться на вулицю. Кіт спить.»

Для побудови біграмної моделі, ми розраховуємо ймовірності переходів між парами слів:

- Після "кіт сидить" зустрічається "на".
- Після "сидить на" зустрічається "підвіконні".
- Після "кіт дивиться" зустрічається "на".
- Після "дивиться на" зустрічається "вулицю".

```python
import random
from collections import defaultdict

text = "кіт сидить на підвіконні. кіт дивиться на вулицю. кіт спить."

words = text.split()
bigram_dict = defaultdict(list)

for i in range(len(words) - 2):
    bigram = (words[i], words[i + 1])
    next_word = words[i + 2]
    bigram_dict[bigram].append(next_word)

def generate_text(bigram_dict, start_bigram, length=10):
    current_bigram = start_bigram
    result = list(current_bigram)
    for _ in range(length - 2):
        if current_bigram not in bigram_dict:
            break
        next_word = random.choice(bigram_dict[current_bigram])
        result.append(next_word)
        current_bigram = (current_bigram[1], next_word)
    return ' '.join(result)

start_bigram = ('кіт', 'дивиться')
generated_text = generate_text(bigram_dict, start_bigram)
print(generated_text)
```

Рис 3 – Приклад примітивної біграмної моделі

```
кіт дивиться на вулицю. кіт спить.


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Рис 4 – Результат виконання коду

Як можна побачити біграмна модель може генерувати більш осмислений текст, що робить її використання більш продуктивним.

Загалом Марковські моделі є потужним інструментом для моделювання випадкових процесів, включаючи текстову генерацію. Вони використовуються в різних галузях, таких як лінгвістика, інформатика, фінанси та багато інших.

N-грамні моделі, такі як біграми, дозволяють покращити якість генерації тексту, враховуючи більше контексту. Це робить тексти більш узгодженими та природними. Використання n-грам є наступним кроком у розвитку Марковських моделей, що допомагає у вирішенні складніших завдань обробки природної мови.

2.3 – Моделі обробки текстів основані на базових методах машинного навчання

Машинне навчання - це галузь штучного інтелекту, яка вивчає розробку алгоритмів, які можуть самостійно навчатися та вдосконалювати свою продуктивність на основі даних. У контексті генерації тексту машинне навчання використовується для створення моделей, які можуть генерувати текст, що відповідає певним задачам або структурам. Машинне навчання має кілька основних методів «навчання», це навчання з учителем, без учителя, а також використання нейронних мереж. При навчанні з учителем використовується великий набір розмічених даних з відповідями. Моделі навчаються по цих дата сетах, єфективність навчання залежить від точності розмітки даних у дата сеті. Навчання без учителя схоже на навчання із вчителем, модель отримує дата сет, але на цей раз він не має відповідей чи поміток. Також для навчання без вчителя використовуються алгоритми, такі як тематичне моделювання (наприклад, Latent Dirichlet Allocation) або автокодування (наприклад, autoencoders), можуть використовуватися для виявлення патернів та структури в нерозмічених даних. Використання нейронних мереж для генерації тексту є популярним напрямком у сфері машинного навчання. Існує кілька архітектур нейронних мереж, які успішно використовуються для генерації тексту. Декілька з них включають рекурентні нейронні мережі, довготривалу короткочасну пам'ять, та трансформери. Розглянемо рекурентні нейронні. RNN призначені для обробки послідовностей, що робить їх дуже єфективними для обробки тексту. Також рекурентні нейронні мережі мають внутрішню пам’ять, що дозводяє враховувати попередній контекст при обробці наступного входу. Принцип роботи RNN – на кожному кроці послідовності (наприклад тексту) нейрон мережі отримує вхідні дані з поточного

кроку аналізу і з усіх попередніх кроків також. Таким чином мережа має змогу зберегати попередні значення у пам’яті, що дозволяє їй більш ефективно передбачати наступні ланцюжки слів.

Однією з відмінностей RNN від звичайної нейронної мережі є те шо вихідні дані генеруються за допомогою прямого розповсюдження, а вагові параметри оновлюються за допомогою зворотного розповсюдження, подібно до того, як це робиться зі звичайними нейронними мережами, RNN спільно використовують параметри між рівнями, а не ініціалізують різні вагові коефіцієнти, що відповідають кожному вузлу мережі.

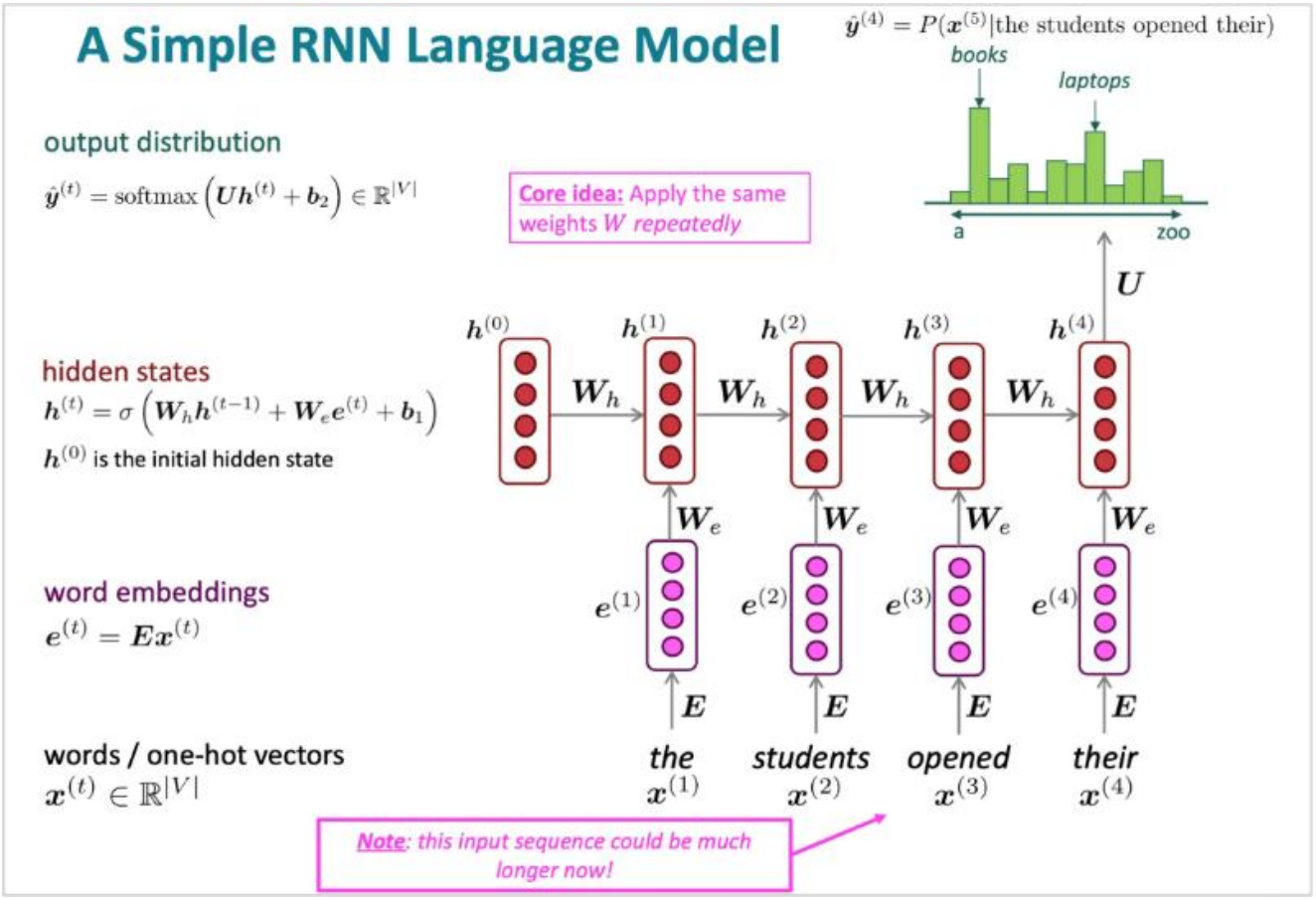

Рис 5 – Приклад простої рекурентної мережі

Основними частинами RNN є:

- Вхідний рівень - вхідний рівень відповідає за отримання послідовних даних. У завданнях обробки природної мови, наприклад, кожен елемент послідовності може відповідати слову або символу

- Прихований рівень - прихований рівень є основним компонентом RNN. Він підтримує внутрішній або прихований стан, який розвивається, коли мережа обробляє кожен елемент послідовності. Прихований стан фіксує інформацію з попередніх часових кроків, отже, відповідає за збереження контексту та моделювання залежностей у послідовності
- Повторне з'єднання - це важлива функція RNN. Повторюване з'єднання представляє собою набір ваг і зв'язків, які повертаються до прихованого шару від самого себе на попередньому часовому кроці. Цей цикл дозволяє RNN підтримувати та оновлювати свій прихований стан під час обробки кожного елемента в послідовності. Періодичне з'єднання дозволяє мережі запам'ятовувати та використовувати інформацію з минулого.
- Вихідний рівень - вихідний рівень відповідає за створення прогнозів або виходів на основі інформації в прихованому стані. Кількість нейронів у цьому шарі залежить від конкретного завдання. Наприклад, у мовній моделі це може бути рівень softmax для передбачення наступного слова в послідовності.

Хоча рекурентні нейронні мережі є потужними інструментами для аналізу, обробки та генерації тексту, але вони мають недолік, а саме зникання градієнтів при занадто довгому навчанні.

Градієнт - це вектор, який показує напрямок та швидкість найшвидшого зростання функції. У контексті машинного навчання і глибокого навчання градієнт використовується для оптимізації функції втрати або цільової функції, щоб змінити ваги моделі так, щоб мінімізувати цю функцію.

Зникаючий градієнт відноситься до проблеми, яка може виникнути під час навчання глибоких нейронних мереж, коли градієнти функції втрат щодо параметрів моделі стають надзвичайно малими (близькими до нуля), оскільки вони поширюються у зворотному напрямку через шари мережі під час навчання. Це призводить до погіршення навчання в глибоких нейронних мережах (DNN). Коли градієнти стають занадто малими, це означає, що ваги моделі не оновлюються

ефективно. У результаті навчання мережі може зупинитися або стати надзвичайно повільним, що ускладнить для мережі вивчення складних шаблонів у даних.

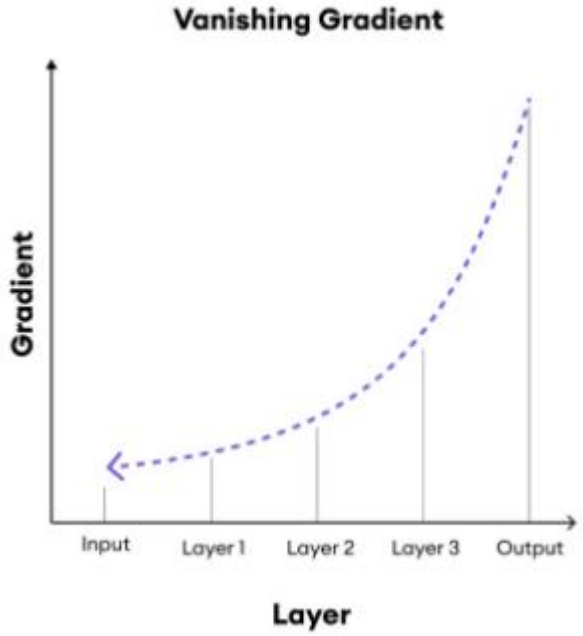

Рис 6 – Іллюстрація зникаючого градієнту

Причиною зникання градієнтів є множинні вбудовані у функції активації, які використовуються у шарах нейронних мереж. Наприклад, у функції сигмоїди або тангенса гіперболічного (tanh) градієнти близькі до нуля для великих або малих значень входу. При зворотньому розповсюдженні помилки це означає, що градієнти шару зближуються до нуля, і ваги цього шару практично не оновлюються.

Також існує зворотня проблема, проблема розривного градієнта. Подібно до зникнення градієнта, проблема вибухового градієнта виникає частіше, коли функція активації використовується в прихованих шарах, оскільки вихід цих активацій має тенденцію зосереджуватися на крайніх кінцях кривої (0 або 1 для sigmoid, або -1 і 1 для tanh). Проблема вибухового градієнта особливо виражена в глибоких мережах з багатьма шарами, де градієнти обчислюються за правилом ланцюга і можуть накопичуватися мультиплікативно.

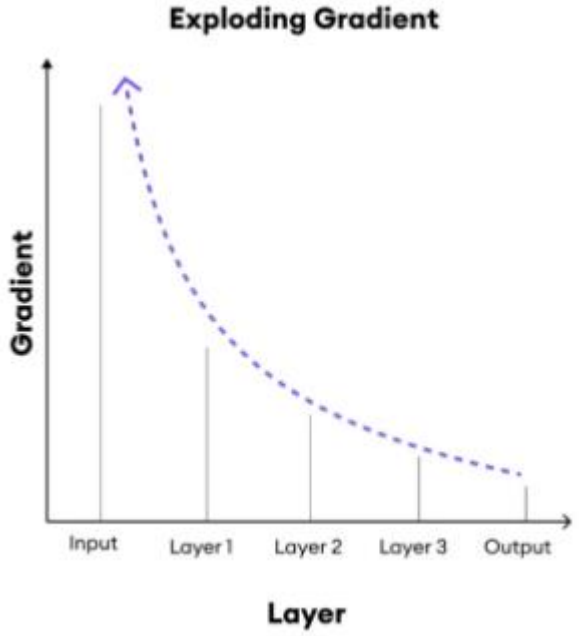

Рис 7 – Ілюстрація розривного градієнта

Для вирішення цих проблем було створено LSTM модель. LSTM є спеціальним типом RNN (Рекурентні нейронні мережі), призначеним для уникнення проблеми пов'язаних з довгостроковими залежностями, зникання або вибухання градієнтів. Вони мають вбудовану пам'ять, яка дозволяє враховувати довгострокові залежності в тексті.

Архітектура LSTM включає в себе спеціальні компоненти, які дозволяють зберігати та маніпулювати інформацією протягом тривалого часу. Основними компонентами LSTM є осередок (cell), входи (input), виходи (output) та шлюзи (gates).

Архітектура LSTM - Кожен осередок складається з кількох важливих компонентів: комірки стану (cell state), шлюзів введення, забування та виходу.

1. **Комірка стану (Cell State)** Комірка стану є ключовим компонентом LSTM, який відповідає за зберігання інформації. Вона дозволяє передавати інформацію через численні часові кроки, зберігаючи необхідний контекст.
2. **Шлюз забування (Forget Gate)** Шлюз забування визначає, яку частину інформації з попереднього стану потрібно забути. Він отримує на вхід

поточний вхід і попередній стан прихованого шару і вирішує, яку частину інформації передати далі.

3. **Шлюз введення (Input Gate)** Шлюз введення визначає, яку нову інформацію потрібно додати до стану комірки. Він складається з двох частин: сигмоїдної функції, яка вирішує, які значення оновити, і тангенса гіперболічного (tanh), який створює нові кандидатські значення.
4. **Шлюз виходу (Output Gate)** Шлюз виходу визначає, яку частину інформації з комірки стану передати як вихід. Він використовує сигмоїдну функцію для визначення частини стану комірки, яка буде використана для створення виходу.

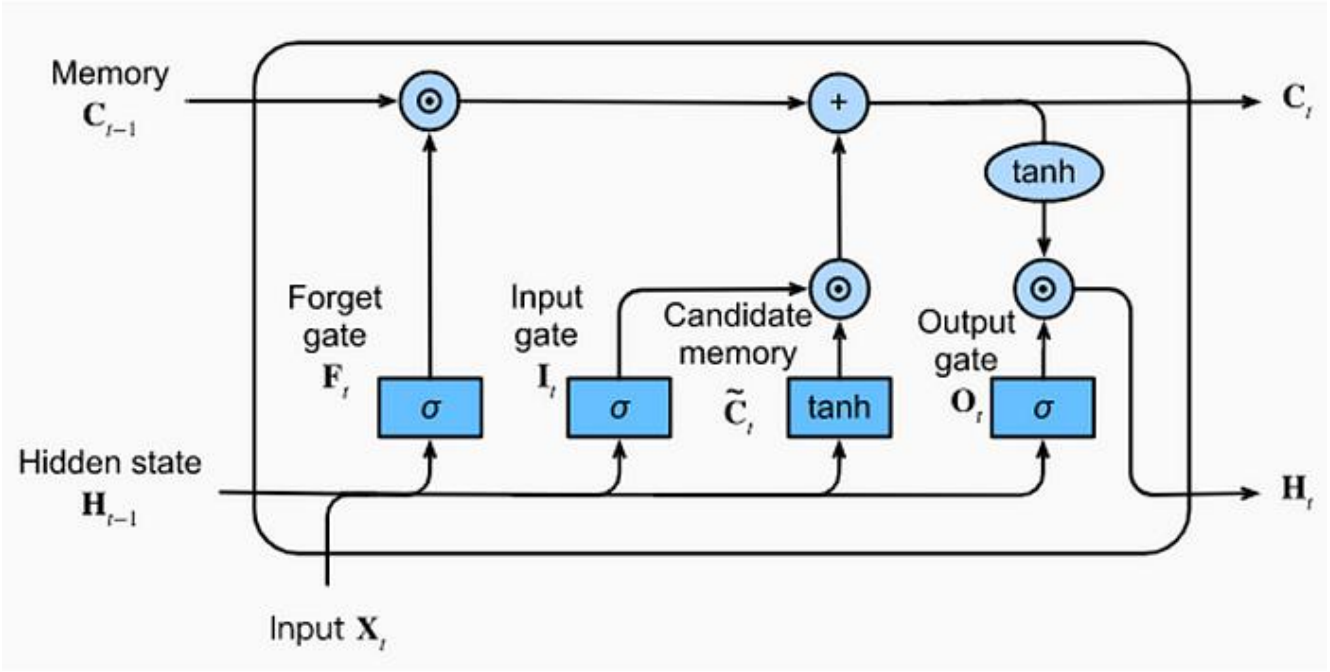

Рис 8 – Ілюстрація базової архітектури LSTM

Принцип проботи – input output:

Блок LSTM отримує три вектори (три списки чисел) як вхідні дані. Два вектори надходять із самого LSTM і були згенеровані LSTM у попередній момент (момент $t-1$). Це стан комірки (C) і прихований стан (H). Третій вектор йде ззовні. Це вектор X (званий вхідним вектором), надісланий до LSTM у момент t . За наявності трьох вхідних векторів (C , H , X) LSTM регулює через вентилі внутрішній потік інформації та перетворює значення векторів стану комірки та

прихованого стану. Вектори, які будуть частиною вхідного набору LSTM у наступний момент (момент $t + 1$). Контроль потоку інформації здійснюється таким чином, що стан комірки діє як довготривала пам'ять, тоді як прихований стан діє як короткочасна пам'ять.

На практиці блок LSTM використовує нещодавно минулу інформацію (короткочасну пам'ять, H) і нову інформацію, що надходить ззовні (вхідний вектор, X), щоб оновити довготривалу пам'ять (стан комірки, C). Нарешті, він використовує довготривалу пам'ять (стан комірки, C), щоб оновити короткочасну пам'ять (прихований стан, H). Прихований стан, визначений у момент t, також є виходом блоку LSTM у момент t . Це те, що LSTM надає назовні для виконання конкретного завдання. Іншими словами, це поведінка, за якою оцінюється продуктивність LSTM.

Принцип роботи гейтів (gates):

Три гейти (forget gate, input gate, output gate) є інформаційними селекторами. Їхнє завдання — створювати вектори-селектори. Вектор-селектор — це вектор із значеннями між нулем і одиницею та близькими до цих двох крайніх значень.

Вектор-селектор створюється для множення елемент за елементом на інший вектор такого ж розміру. Це означає, що позиція, де вектор-селектор має значення, що дорівнює нулю, повністю усуває (у множенні поелементно) інформацію, включену в ту саму позицію в іншому векторі. Позиція, де вектор-селектор має значення, що дорівнює одиниці, залишає незмінною (у множенні елемент за елементом) інформацію, включену в ту саму позицію в іншому векторі.

Усі три гейти є нейронними мережами, які використовують сигмоподібну функцію як функцію активації на вихідному рівні. Сигмоїдна функція використовується для отримання на виході вектора, що складається із значень між нулем і одиницею та поблизу цих двох крайніх значень.

Усі три вентилі використовують вхідний вектор (X) і прихований вектор стану, що надходить із попереднього моменту ($H\_[t-1]$), об'єднані в один вектор. Цей вектор є входом усіх трьох воріт.

Загалом LSTMs дозволяють ефективніше моделювати взаємозв'язки в послідовностях, що робить їх популярними для генерації тексту. Процес використання нейронних мереж для генерації тексту зазвичай включає наступні етапи:

- Підготовка даних
- Побудова моделі
- Навчання моделі
- Використання навченої моделі (Генерація тексту)

2.4 – Моделі обробки текстів основані на великих лігвістичних моделях

GPT (Generative Pre-trained Transformer) — це серія моделей глибокого навчання, розроблених компанією OpenAI. GPT використовує трансформерну архітектуру і став відомий завдяки здатності генерувати високоякісний текст та розв'язувати різноманітні завдання мовного розуміння. Модель базується на трансформері, який представляє собою архітектуру засновану на механізмі внимання. Цей механізм дозволяє моделі ефективно обробляти довгі залежності в тексті та взаємодіяти з контекстом.

Модель GPT проходить через етап "перед-тренування" на величезних обсягах текстових даних, використовуючи модель для передбачення наступного слова в послідовності. Наприклад GPT-3 Модель глибокого навчання зі 175 мільярдами параметрів здатна створювати текст, схожий на людину, і була навчена на великих текстових наборах із сотнями мільярдів слів.

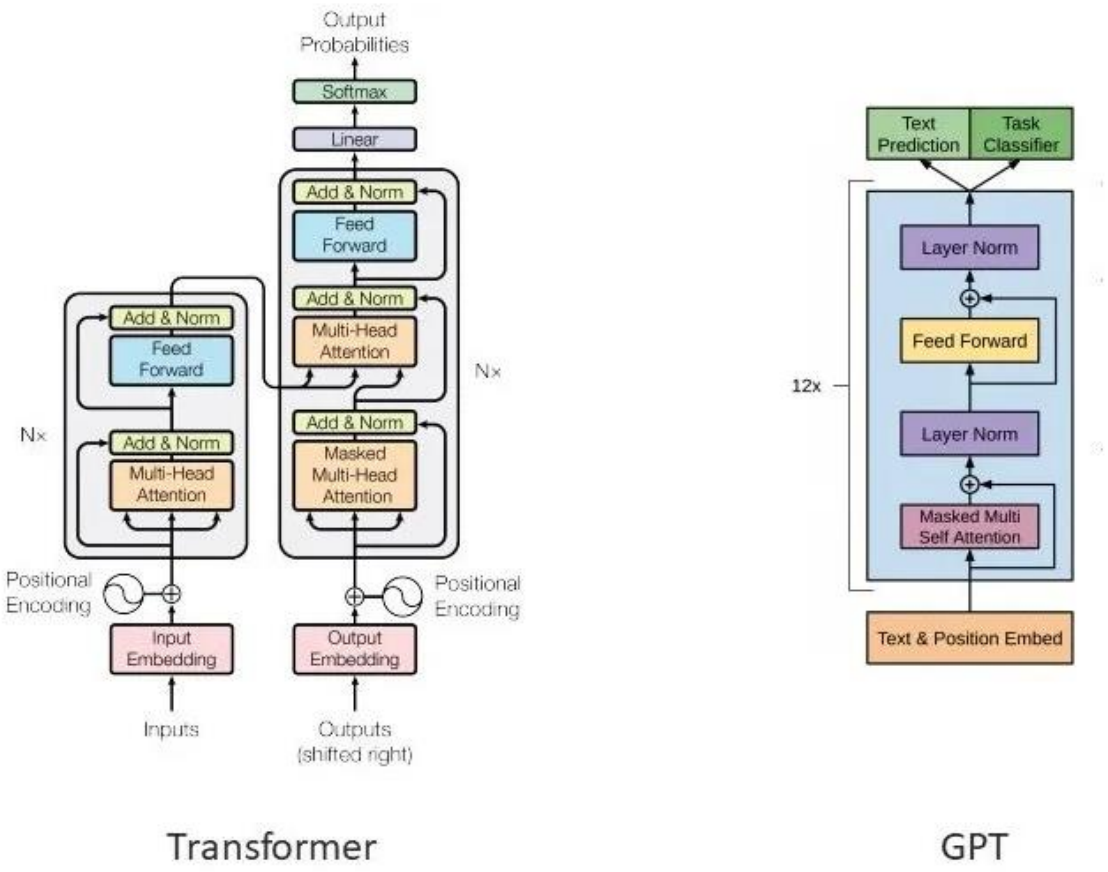

Рис 9 – Ілюстрація GPT та трансформеру

GPT-3 навчався з використанням величезних наборів текстових даних Інтернету — загалом 570 ГБ. На момент випуску це була найбільша нейронна мережа зі 175 мільярдами параметрів (100x GPT-2). GPT-3 має 96 блоків уваги, кожен з яких містить 96 головок уваги.

GPT-3 використовує не модифіковану архітектуру декодера зазначену на ілюстрації. GPT-3 було навчено за допомогою прогнозування наступного слова, завдання типу неконтрольованого навчання, у якому — воно передбачає наступне слово в реченні. Фактично вхідна послідовність становить 2048 слів (для GPT-3). Ми все ще можемо передавати короткі послідовності як вхідні дані: ми просто заповнюємо всі додаткові позиції «порожніми» значеннями. Вихідні дані GPT — це не просто одне припущення, це послідовність (довжиною 2048) припущень (ймовірність для кожного ймовірного слова). По одному для кожної «наступної» позиції в послідовності. Але під час генерування тексту ми зазвичай дивимось лише на припущення для останнього слова послідовності.

Також GPT має словник, для третьої версії кількість слів у ньому є більшою за 50 000. Таким чином GPT може давати кожному слову значення і «розуміти» їх. Якщо слова нема у словнику, GPT використовує алгоритм підслів для створення слова.

Ця модель продемонструвала вражаючі досягнення в генерації тексту та розумінні мови, відкриваючи нові можливості для різноманітних застосувань у сфері машинного навчання та обробки природної мови.

Останній пункт зі списку — це генерація тексту на основі шаблонів. Це метод створення тексту визначеного заздалегідь. Генерація тексту відбувається на основі визначених раніше шаблонів або структур. Шаблони можуть містити статичний текст, змінні, що заповнюються даними, та логіку управління потоком текстової генерації наприклад, умовні конструкції. Вони визначають структуру та взаємодію між елетентами тексту. Це підходить для випадків, коли потрібно генерувати тексти з фіксованою структурою, наприклад, шаблонні відповіді на запитання, автоматизовані звіти чи стандартизовані повідомлення. Генерація на основі шаблонів може бути використана в різноманітних сферах, таких як відповіді на користувацькі запитання, автоматизовані звіти, електронна кореспонденція, тощо. Вона є ефективним інструментом для створення текстового контенту, який повторюється або має стандартизований формат. Однак, цей метод має серйозні недоліки. Дуже важливо слідкувати за якістю та актуальністю змінних, що використовуються для заповнення шаблонів, і вибір самих шаблонів, які дозволяють достатню гнучкість у генерації різних текстових варіантів. Якщо цього не робити то вихідний текст може мати низьку якість.

3 ПОРІВНЯННЯ МОДЕЛЕЙ ГЕНЕРАЦІЇ ТЕКСТІВ

3.1 – Постановка експерименту дослідження

Метою експерименту є порівняння ефективності різних методів розуміння природної мови (NLP) для створення структурованого опису на основі текстів. Ми будемо аналізувати тексти з електронних листів, трансформувати тексти або їх частини для ефективного використання вибраної моделі і застосовувати різні моделі для виявлення шаблонів та структури.

Етапами експеременту буде збір даних, попередня їх обробка, трансформація тексту, застосування моделей, аналіз результатів.

Збір даних - Зібрати корпус текстів з електронних листів, які містять різні типи повідомлень (запрошення, нагадування, підтвердження тощо). Розмір корпусу близько 1000 листів для забезпечення репрезентативності вибірки.

Попередня обробка даних включатиме в себе токенізацію тексту – розбивка тексту на слова і речення, видалення стоп-слів, або слів які не несуть смислового навантаження такі як «і» або «але». Останнім етапом попередньої обробки даних буде нормалізація тексту, тобто приведення усіх слів до нижнього регістру та лемматизація. Також при необхідності текст буде розділятися на частини якщо цього потребують особливості роботи з моделю. Після усіх етапів підготовки буде застосовано обрані моделі для аналізу, а саме моделі Маркова, рекурентні нейронні мережі і моделі основані на великих лінгвістичних моделях, після чого буде проведено оцінку результатів шляхом порівняння F1 score - метрика оцінки, яка вимірює точність моделі. Вона поєднує в собі точність і показники запам’ятовування моделі.

В результаті експерименту очікується отримати: Порівняльний аналіз ефективності різних моделей для задачі структурованого опису текстів. Визначення найбільш підходящих моделей для різних типів текстів та шаблонів. Рекомендації щодо вибору моделей та методів обробки текстів для подальшого використання в інформаційних системах, машинному навчанні та інших сферах.

Цей експеримент дозволить визначити, які методи аналізу та обробки текстів є найбільш ефективними для створення структурованих описів на основі текстів з електронних листів. Отримані результати сприятимуть покращенню якості автоматизованих систем аналізу текстів та підвищать ефективність використання інформації в різних сферах діяльності.

Вибір моделі для аналізу та обробки текстів залежить від багатьох аспектів. Щоб вибрати найбільш підходячий інструмент для аналізу та генерації текстів потрібно враховувати наступні аспекти задачі:

- Тип завдання
- Довжина контексту опису
- Точність та глибина аналізу
- Обчислювальні ресурси у наявності
- Складність реалізації при використанні вибраного інструменту

Проаналізуємо чи підходять нам існуючі інструменти аналізу тексту і виберемо найкращій з варіантів для реалізації завдання. Аналіз буде базуватися на основі аспектів зазначених вище. Для аналізу обрано наступні інструменти:

- Нейронні мережі
- Рекурентні нейронні мережі (RNN)
- Моделі типу GPT (Generative Pre-trained Transformers)
- Моделі Маркова

Оскільки вони найбільше підходять для аналізу текстів і мають можливості для налаштування.

Почнемо аналіз з типу завдання для якого буде потрібно рішення. Наша задача – створити інструмент-помічник для електронної почти. В задачі якого входить аналіз листів що надходять користувачу і подальше створення нотатки у календарі.

Для якісного виконання такої задачі нам потрібен інструмент що зможе якісно аналізувати великі тексти, це потрібно виходячі з можливостей gmail відправляти листи розміро до 25 мегабайт, що дорівнює приблизно від 500 000 до 1 000 000 слів у одному листі без вкладень. Також треба очікувати що деякі листи

не потрібно аналізувати повністю, оскільки для нас інтерес мають лише деякі, організаційні аспекти листа, а саме запрошення або інші подібні контексти.

Також треба враховувати можливість надсилання у письмі файлів і вкладень, наприклад документів чи зображень, у яких може міститися інформація про подію на яку запрошують користувача. Тому важливо оцінювати не тільки можливість інструменту аналізувати текстове наповнення, але і можливість проводити аналіз вкладень.

Персоналізація налаштувань під конкретного користувача є одним із важливіших пріорітетів подібних інструментів, оскільки вони використовуються саме для вирішення повсякденних задач користувача. Тому треба зазначити що ми також повинні мати можливість налаштування інструменту під забаганки користувача.

Таким чином можна констатувати важливість наступних аспектів інструментарію:

- Здатність якісно аналізувати довгі тексти.
- Можливість обробки та подальшого аналізу аналізу тексту з документів та зображень.
- Здатність проводити аналіз чанків інформації у контексті листування.
- Можливість налаштування під конкретного користувача.

Для визначення оптимальної довжини контексту для аналізу листів у Gmail (чи будь-якого іншого листування) необхідно враховувати кілька ключових факторів. Довжина контексту безпосередньо впливає на здатність моделі розуміти зміст та контекст листування, що особливо важливо для аналізу та генерації зв'язного опису.

Основним фактором для визначення контексту є саме характеристика листів. Листування може включати короткі повідомлення або довгі дискусії. Важливо проаналізувати середню довжину листів та діалогів у нашому випадку.

Для нашої програми помічника важливо саме запрошення користувача на якусь подію, для подальшої обробки цієї інформації і створення запису у календарі

користувача. Аналізуючі різні типи листів із запрошеннями можна дійти висновку що у більшостів випадків запрошення у листах знаходяться на самому початку листування (у перших 3-5 рядках), або у кінці листа. Виходячи з цієї інформації буде логічно розділяти листа на декілька чанків з інформацією перед тим як починати аналіз. Першою частиною на аналіз буде вважатися початок листа, а саме перші 3-5 строк. Якщо при аналізі не було виявлено ніяких запрошень, то наступною частиною на аналіз стає остання частина листа, а саме останні 5-10 строк. Якщо у перших двох чанках не було виявлено ніякого запрошення то буде проведено аналіз останнього, середнього чанку з інформацією, враховуючі контекст першої та останньої частини листа.

Таким чином можна констатувати що нам потрібен інструмент який зможе орієнтуватися як у доволі малих текстах починаючі від кількох строк, так і у великих текстах у разі листів які потрібно буде проаналізувати повністю.

Наступним кроком аналізу є розуміння того яка точність аналізу нам потрібна. Наше завдання полягає у тому, що нам потрібно знайти запрошення користувача на подію і записати це до розпорядку дня користувача. Слід зазначити що у такій системі потрібна максимальна точність від інструменту аналізу, оскільки створювати не існуючі події у календарі користувача не є припустимим.

Оскільки тренування ідеальної моделі займе дуже багато часу і буде коштувати дуже дорого, ми надамо користувачу можливість перевіряти чи правильно був проаналізований текст, і коректувати зміни до внесення їх у календар. Хоча так рішення можливо і не буде оптимальним з точки зору користувача, але буде таким якщо розраховувати вартість розробки ідеальної моделі. Таким чином можна зазначити що ми прагнемо отримати модель на яку користувач зможе розраховувати у більшості випадків, бажана точність складає від 90-95 відсотків, такої точності буде можливо досягти оскількі більшість листів не є дуже великими, або складними для аналізу.

Складність реалізації і обчислювальні потужності. У цьому розділі ми зосередимося на плюсах і мінусах різних підходів до аналізу текстів. Визначему складність їх реалізації і вартість використання у нашій програмі-помічнику.

3.2 – Аналіз і використанням великих лінгвістичних моделей

Почнемо з моделі типу GPT. Моделі типу Generative Pre-trained Transformers підходять для завдань генерації тексту, перекладу, створення чат-ботів, резюмування текстів, тощо. Іншими словами будь яка задача де потрібен контекстуальний аналіз і генерація інформації. GPT та подібні йому трансформери краще обробляють довгі контексти та складні залежності у текстах завдяки механізму самоуваги (self-attention), але також мають здатність аналізувати і короткі тексти. Це можливо теж завдяки механізму самоуваги який дозволяє кожному токену у вхідній послідовності враховувати інформацію з усіх інших токенів цієї ж послідовності. Це означає, що навіть при короткому контексті, кожен токен може взаємодіяти з іншими токенами безпосередньо і ефективно. Також незалежно від довжини контексту, кожен токен отримує ваги, що вказують на його важливість щодо інших токенів у послідовності.

GPT та подібні сучасні трансформерні моделі забезпечують високу точність та глибокий контекстуальний аналіз завдяки великій кількості параметрів та потужній архітектурі.

Загалом GPT має багато плюсів і ідеально підходить для використання у нашій задачі, але він має деякі недоліки. Такі моделі як GPT потребують значних обчислювальних ресурсів, як для навчання так і використання. Наприклад тренування ChatGPT-3.5 обійшлося у 4.6 мільйонів USD. Також треба враховувати те що нам потрібні величезні дата сети для тренування моделі, при цьому треба зазначити що значну частину цих даних повинен контролювати інженер, який буде стежити за якістю даних які використовуються для навчання GPT, що тягне за собою додаткові витрати на кваліфікований персонал.

Тож окрім великої вартості тренування моделі є складним процесом, який потребує багато ресурсів і кваліфікований функціонал.

Результати експеременту з використанням великих лінгвістичних моделей дає нам наступні результати f1 score:

Модель	Точність	Запам’ятовування	F1 Score
GPT-3.5	0.85	0.80	0.82

3.3 – Аналіз і використанням Recurrent Neural Networks

Нейронні мережі, окрім моделей типу Generative Pre-trained Transformers (GPT), також включають інші архітектури, які можуть бути ефективно використані для аналізу текстів. Ми будемо досліджувати саме RNN (рекурентні нейронні мережі).

Рекурентні нейронні мережі (RNN) підходять для роботи з послідовними даними, такими як текст. Основна ідея RNN полягає в тому, що вони мають цикли всередині мережі, що дозволяє їм зберігати інформацію про попередні елементи послідовності. Це дозволяє RNN враховувати контекст при аналізі текстів. Перевагами RNN є можливість аналізувати тексти змінної довжини із достатньою точністю, завдяки можливості зберігати данні про контекст. Недоліками RNN є проблеми коли мережа працює із довгими текстами, якщо текст листа буде занадто довгий, RNN модель може втратити зв’язок між початком і кінцем тексту, що призведе до незадовільних результатів аналізу. Але завдяки особливості нашої задачі є можливість достатньо ефективно використовувати навчену модель RNN, якщо розбивати тексти на чанки з інформацією і проводити послідовний аналіз доки не буде виявлено запрошення. Тож використання RNN моделі може бути більш доцільним у разі якщо ми плануємо тренувати власну модель для обробки текстів, оскільки тренування подібної моделі дешевше за тренування моделі GPT, але теоретично може бути не менш ефективним у заданій задачі.

Результати експеременту з використанням рекурентних нейронних мереж дає нам наступні результати f1 score:

Модель	Точність	Запам’ятовування	F1 Score
Тренована RNN	0.80	0.75	0.77

3.4 – Аналіз і використанням моделей Маркова

Моделі маркова. Моделі Маркова можуть бути використані для аналізу текстів електронних листів і виявлення запрошень на івенти, але вони мають свої недоліки, а саме проблеми з довгими залежностями і обмеженим контекстом. Моделі Маркова оперують на припущенні про те, що майбутній стан залежить тільки від поточного стану. Це може бути обмеженням для аналізу текстів, де контекст може бути складним або залежати від більш широкого контексту. Також моделі Маркова можуть втратити інформацію про попередні стани при довших послідовностях. Це може призвести до втрати важливої інформації або зниження точності передбачень.

Що важливіше, моделі Маркова відносно прості для навчання і реалізації, що робить їх створення значно дешевшим за інші нейромережі. Також їх ефективність при роботі із невеликими об’ємами тексту є значним плюсом, оскільки утримання і використання цієї моделі буде дешевшим за альтернативи, а із поступовим підходом до аналізу тексту чанками, можна забезпечити більшу точність при аналізі запрошення користувача у листі. Але слід також зазначити що моделі Маркова потребують значний обсяг якісних данних для навчання.

Результати експеременту з використанням рекурентних нейронних мереж дає нам наступні результати f1 score:

Модель	Точність	Запам’ятовування	F1 Score
Модель Маркова	0.50	0.35	0.43

4 ОПИС РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАСТОСУНКУ

4.1 – Налаштування параметрів проекту у google console

В умовах сучасного ритму життя та швидкого розвитку інформаційних технологій, ефективне управління часом та комунікацією стає все більш важливим завданням. Програмний застосунок, який допомагає аналізувати електронні листи та планувати події у календарі, значно спростить ці процеси. Метою створення такого застосунку є автоматизація рутинних завдань, що дозволяє користувачам зосередитися на більш важливих аспектах їх роботи та особистого життя. Наведемо схему взаємодії плагіна з клієнтом та сервером.

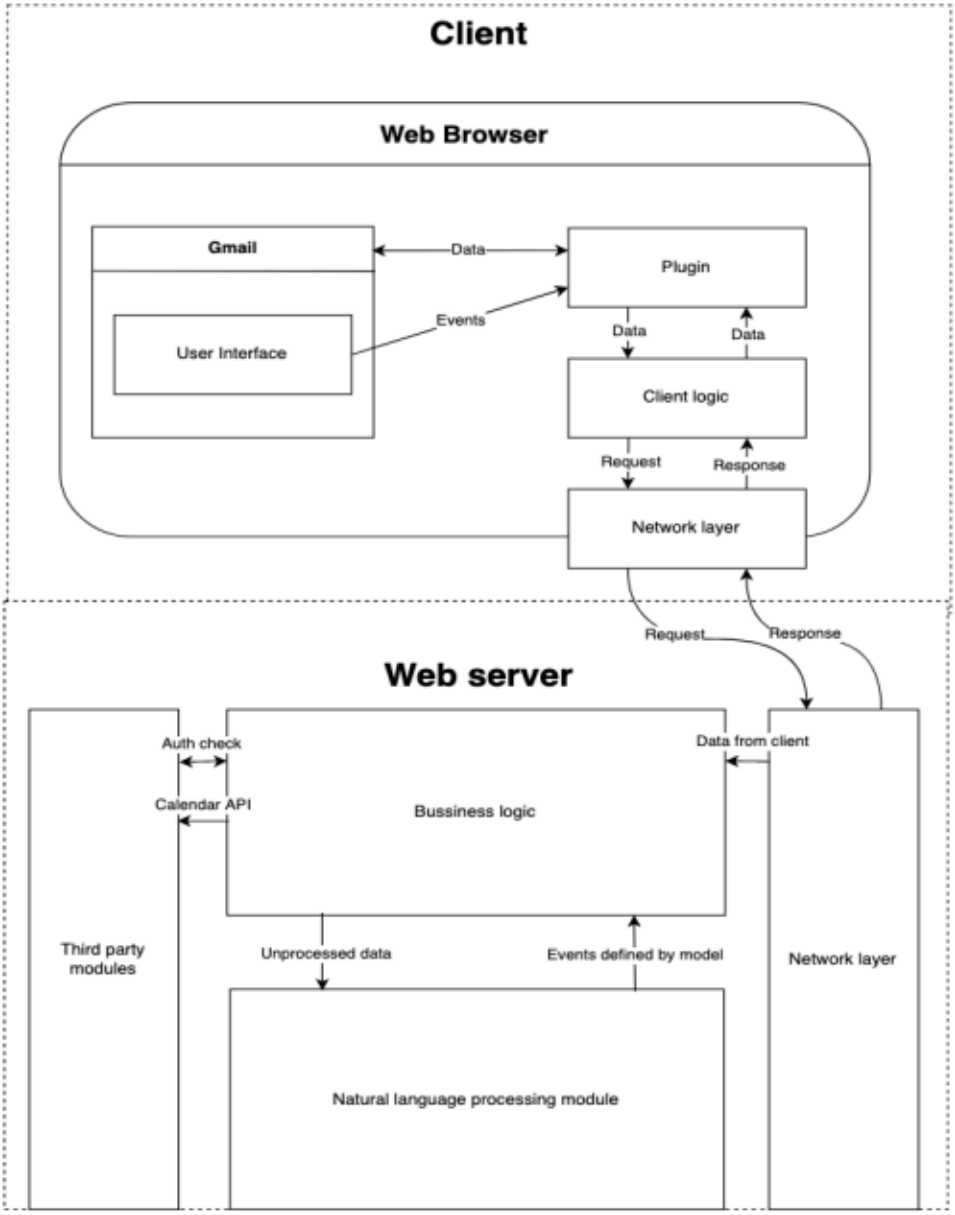

Рис 10 — схема взаємодії плагіна з клієнтом і сервером

Першим кроком при створенні плагіна буде реєстрація у google console і створення проекту у google workspace [4]:

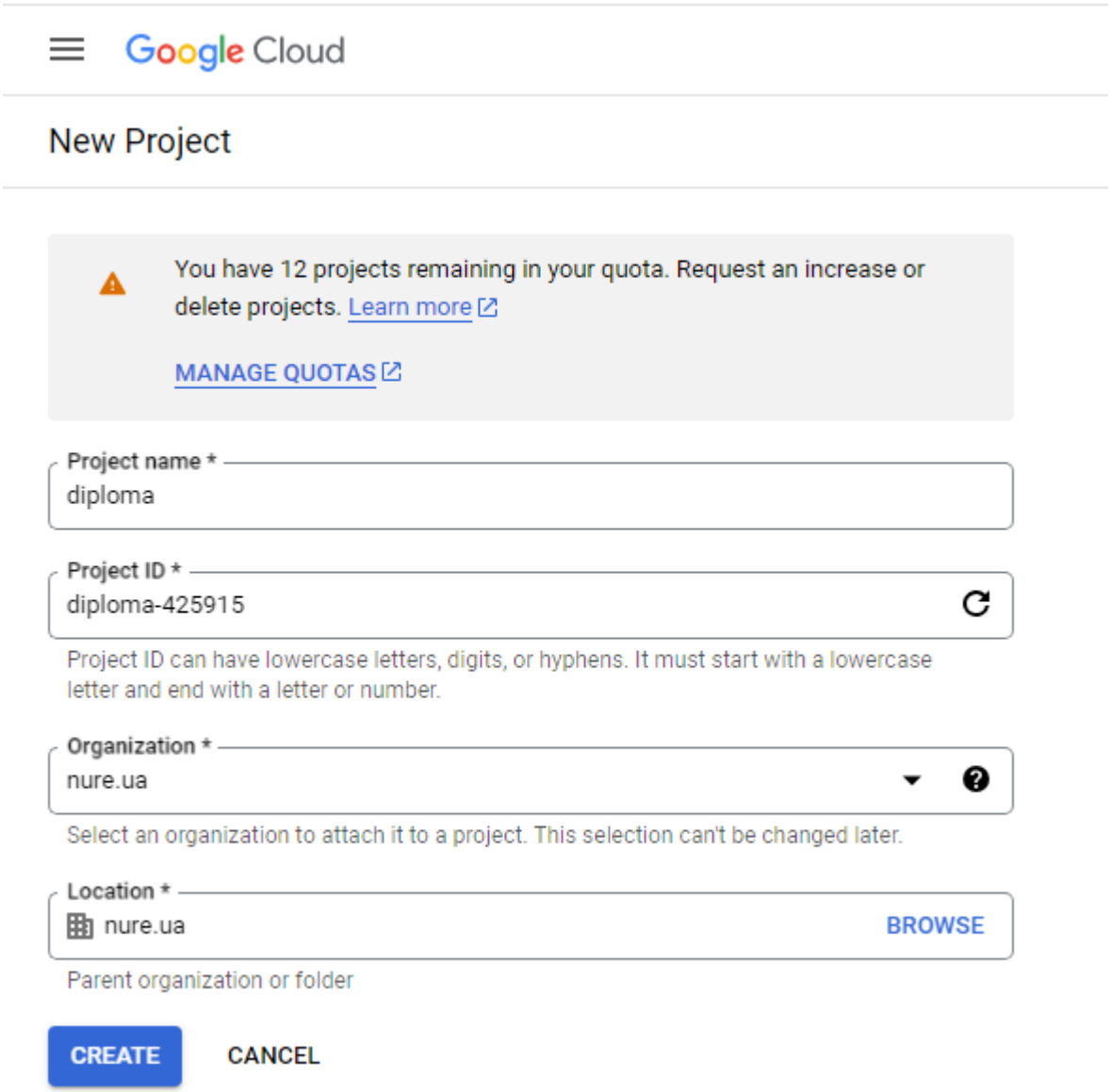

Рис 11 – створення проекту у google workspace

Після цього нам потрібно додати gmail і google calendar API до нашого проекту [4]:

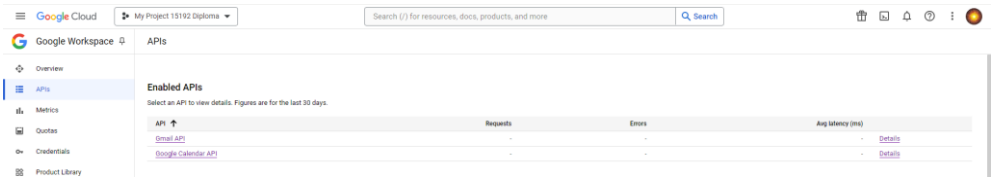

Рис 12 – Додавання API до проекту

4.2 – Реалізація програмного модуля (плагіну)

Для створення плагіну нам знадобляться такі інструменти як HTML, CSS, JavaScript. Для роботи з цими інструментами буде використано Visual Studio Code. Створимо index.html для нашого плагіну.

```
<!DOCTYPE html>
<html>

<head>
    <title>GPT Planner</title>
    <meta charset="utf-8">
</head>

<body>
</body>

</html>
```

Рис 13 – index.html

Далі ми створимо файли із інтерфейсом для подій у нашій програмі. Спочатку створимо інтерфейс на випадок коли користувач побажає змінити деталі запланованого івенту [8].

```
<div>
    <h1>Notification</h1>
    <p>Please check if the event is scheduled right</p>

    <div>
        <button>Edit the event details</button>
        <button>Schedule an event</button>
    </div>
</div>
```

Рис 14 – Повідомлення для користувача

Тепер створимо форму для редагування події.

```
<div>
    <h1>Specify the event details</h1>

    <div>
        <label>Choose the date</label>
        <input type="date" />
    </div>

    <div>
        <label>Specify time</label>
        <input />
    </div>
</div>
```

Рис 15 – Форма редагування події

Останнім що нам потрібно створити, стане повідомлення про успішне застосування змін.

```
<div>
    <h1>Scheduled successfully!</h1>
</div>
```

Рис 16 – Повідомлення про успішне планування

4.3 – Опис запитів програмного модуля.

Додамо до нашого index.html скрипт щоб отримати доступ до використання google APIs [4].

```
<head>
    <title>GPT Planner</title>
    <script src="https://apis.google.com/js/api.js"></script>
    <meta charset="utf-8">
</head>
```

Рис 17 – Скрипт для використання гугл APIs

Створимо функцію для авторизації для google API.

```
gapi.client
  .init({
    apiKey: API_KEY,
    clientId: CLIENT_ID,
    discoveryDocs: DISCOVERY_DOCS,
    scope: SCOPES,
  })
  .then(
    () => {
      gapi.auth2.getAuthInstance().isSignedIn.listen(updateSigninStatus);

      updateSigninStatus(gapi.auth2.getAuthInstance().isSignedIn.get());
      authorizeButton.onclick = handleAuthClick;
      signoutButton.onclick = handleSignoutClick;
    },
    (error) => {
      appendPre(JSON.stringify(error, null, 2));
    }
  );
```

Рис 18 – Авторизація у гугл API

Створимо функції для роботи із гугл API, а саме scheduleEvent для планування івентів у календарі і getMessage для отримання листа, його аналізу і створення івенту у календарі користувача [7].

```
function scheduleEvent(description) {
  const event = {
    summary: 'Scheduled from Gmail',
    description: description,
    start: {
      dateTime: new Date().toISOString(),
      timeZone: 'Ukraine/Kharkiv',
    },
    end: {
      dateTime: new Date(new Date().getTime() + 3600000).toISOString(),
      timeZone: 'Ukraine/Kharkiv',
    },
  };

  const request = gapi.client.calendar.events.insert({
    calendarId: CURRENT_USER_CALENDAR_ID,
    resource: event,
  });

  request.execute((event) => {
    appendPre('Event created: ' + event.htmlLink);
  });
}
```

Рис 19 – Функція для роботи з кадендарем

Створимо функцію для аналізу листа з використанням ChatGPT API.

```
async function analyzeWithChatGPT(message) {
  const response = await fetch('https://api.openai.com/v1/chat/completions', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      Authorization: `Bearer ${OPENAI_API_KEY}`,
    },
    body: JSON.stringify({
      model: 'gpt-3.5-turbo',
      messages: [
        {
          role: 'system',
          content: `Analyze the following email content and find the
          invitation on the event. Do not make up any information if
          thereis no invite. Return data in JSON format
          { date: <date time of the event>, title: <event title> }`,
        },
        { role: 'user', content: message },
      ],
    }),
  });

  const data = await response.json();
  const eventDate = data.choises[0].date;
  const eventTitle = data.choises[0].title;

  return [eventDate, eventTitle];
}
```

Рис 20 – функція для аналізу листа

```
function getMessage(messageId) {
  gapi.client.gmail.users.messages
    .get({
      userId: CURRENT_USER_ID,
      id: messageId,
    })
    .then(async (response) => {
      const snippet = response.result.snippet;
      const analysisResult = await analyzeWithChatGPT(snippet);
      scheduleEvent(analysisResult);
    });
}
```

Рис 21 – Фінальний вид функції getMessage

ВИСНОВКИ

У даній роботі було створено рішення для актуальної проблеми створення системи інтеграції користувача із календарем та поштовою скринькою, проведений аналіз проблеми, на підставі якого сформульовані й вирішені основні завдання дослідження. Показано, що для рішення завдань наведених у роботі необхідне використання сервісів для безпечної автентифікації користувачів, маханізму синхронізації, модуль аналізу текстового контенту на основі нейронних мереж, інтеграція з календарем та Gmail. На цій основі запропонований експериментальний варіант плагіну і відповідної інфраструктури, які використовуються для виконання поставленого завдання. Цей плагін допомагає підвищити продуктивність, раціоналізувати процес планування та внести автоматизацію у календарний контекст, що робить його корисним інструментом для осіб, які активно використовують Gmail та Google Calendar.

У контексті сучасного швидкого темпу життя такий підхід дозволяє ефективно керувати часом та ресурсами. Розвиток підтримки більшої кількості мов та розширення аналізу контекстів може зробити плагін більш універсальним та придатним для користувачів із різноманітними лінгвістичними потребами.

Застосування більш складних алгоритмів обробки природної мови та машинного навчання може покращити точність розпізнавання подій та підвищити ефективність аналізу тексту.

Додавання функцій персоналізації дозволить користувачам вибирати критерії визначення подій, налаштовувати алгоритми аналізу та визначати власні ключові слова чи контексти. Розробка мобільних застосунків для Android та iOS дозволить користувачам зручно взаємодіяти з плагіном та контролювати свій календар у русі, а розширення можливостей інтеграції плагіна з іншими сервісами, такими як замітки, завдання чи проекти, розширить функціональність та забезпечить більш повне керування завданнями користувача. Узагальнюючи, цей плагін є потужним інструментом для автоматизації календарного планування на основі електронної пошти. Його розвиток в напрямку більшої гнучкості та точності може зробити його ще більш привабливим для широкого кола користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Manning, C. D., & Schütze, H. (1999). Foundations of Statistical Natural Language Processing. MIT Press
2. Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing (3rd ed.). Pearson
3. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space
4. Google API documentation
5. Вихідців Е.І. Математичні моделі й методи рішення завдань прогнозування рівнів забруднення прикордонного шару атмосфери.
6. Клименко Е.Г. Програмно-алгоритмічні засоби інтелектуального аналізу даних // Радіоелектроніка й інформатика. - 2001. - № 3. - С. 64-67
7. K. Guntupally, R. Devarakonda and K. Kehoe, "Spring Boot based REST API to Improve Data Quality Report Generation for Big Scientific Data: ARM Data Center Example," 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 5328-5329, doi: 10.1109/BigData.2018.8621924.
8. Aggarwal S. Modern web-development using reactjs //International Journal of Recent Research Aspects. – 2018. – Т. 5. – №. 1. – С. 2349-7688.
9. Публікація огляд методів перекладу тексту в зображення URL: https://openarchive.nure.ua/server/api/core/bitstreams/faf70de0-710c-4c8a-8908-59e5d135d67e/content (дата звернення: 18.04.2024).
10. Falatiuk H., Shirokopetleva M., Dudar Z. Investigation of Architecture and Technology Stack for e-Archive System. *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)*, Kyiv, Ukraine, 8–11 October 2019. 2019. URL: https://doi.org/10.1109/picst47496.2019.9061407 (дата звернення: 05.04.2024).
11. Мулєсап Диплом, URL: http://openarchive.nure.ua/bitstream/document/1168/1/MulesapPP.pdf (дата звернення: 22.06.2024).