Міністерство освіти і науки України

Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук

Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» _____ 2023 р

Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: «**Розробка моделі нейронної мережі для усунення неоднозначності омографів у текстових даних**»

Виконав:

студент групи КУ– 61
Галузь знань: 15 – Автоматизація та приладобудування
за спеціальністю 151 – Автоматизація та комп'ютерно-інтегровані технології
ЄГОРОВ Єгор Сергійович

Керівник:

д.т.н., професор, завідуючий кафедри теоретичної та прикладної системотехніки
ШМАТКОВ Сергій Ігорович_____

Рецензент:

к.ф.-м.н., доцент, завідуючий кафедри електроніки та управляючих систем
ХРУСЛОВ Максим Михайлович____

Захищено на засіданні
Атестаційної комісії № 42
протокол № __ від __.12.2023 р.
Оцінка _____ / _____
Голова Атестаційної комісії
д.т.н., професор
СКОБ Юрій Олексійович

Харків – 2023

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи «Розробка моделі нейронної мережі для усунення неоднозначності омографів у текстових даних» складається із вступу, трьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг роботи складає 84 сторінки із яких 66 сторінок основної частини, 2 таблиці, 27 рисунків, 36 найменувань списку використаних джерел на 4 сторінках і 4 додатків на 10 сторінках.

Дане дослідження присвячене можливості моделі нейронної мережі для усунення неоднозначності омографів у текстових даних. Його метою є підвищення якості обробки тексту, зокрема, поліпшення точності розпізнавання слів, які мають більше одного значення, за допомогою розробки та впровадження нейронної мережі, яка буде мати можливість усувати неоднозначності омографів у текстових даних в реальному часі. Під час виконання даної роботи були розглянуті класичні технології і сучасні методи усунення неоднозначності омографів на основі нейронних мереж. Для досягнення поставленої мети була розроблена модель для усунення лексичної неоднозначності омографів, обрано відповідний набір даних для навчання цієї моделі і розроблена програмна реалізація.

Результатом даного дослідження є натренована нейронна мережа з довгою короткостроковою пам'яттю у поєднанні з ембедінгами і повнозв'язаними шарами, яка здатна визначати і усувати омографи у текстових даних в реальному часі, шляхом інтегрування даної моделі у Telegram бот.

Ключові слова: нейронні мережі, омограф, модель, PYTHON, KERAS.

ABSTRACT

Explanatory note to the qualification paper "Development of a neural network model to eliminate the ambiguity of homographs in textual data" consists of an introduction, three sections, conclusions, a list of used sources and appendices. The total volume of the work is 84 pages, of which 66 pages are the main part, 2 tables, 27 figures, 36 names of the list of used sources on 4 pages, and 4 appendices on 10 pages.

This study is devoted to the possibility of a neural network model to eliminate the ambiguity of homographs in textual data. Its goal is to improve the quality of text processing, in particular, to improve the accuracy of recognition of words that have more than one meaning, through the development and implementation of a neural network that will be able to resolve homograph ambiguities in text data in real time. During the performance of this work, classical technologies and modern methods of eliminating the ambiguity of homographs based on neural networks were considered. To achieve the goal, a model was developed to eliminate the lexical ambiguity of homographs, an appropriate data set was selected for training this model, and a software implementation was developed.

The result of this research is a trained neural network with long short-term memory in combination with embeddings and fully connected layers, which is able to identify and eliminate homographs in text data in real time, by integrating this model into the Telegram bot.

Keywords: neural networks, homograph, model, PYTHON, KERAS.

ЗМІСТ

ВСТУП

У сучасному інформаційному суспільстві, в якому величезні об'єми текстової інформації обробляються щоденно, розробка ефективних інструментів для автоматичної обробки та аналізу текстових даних стає надзвичайно важливою задачею. Однак існують різні лексичні неоднозначності, які ускладнюють цей процес, і однією з основних проблем є неоднозначність омографів — слів, що мають однакову лексичну форму, але різне значення.

Таким чином **актуальність** розробки моделі нейронної мережі для усунення неоднозначності омографів у текстових даних визначається сучасними викликами у галузі обробки природної мови. Данна модель дозволить виправляти помилки, пов'язані з неправильним визначенням значень слів у тексті, що забезпечить більшу точність та якість аналізу тексту. Вона може знайти використання у різних сферах, таких як автоматичне розпізнавання мови, підтримка прийняття рішень на основі текстової інформації, покращення систем пошуку та класифікації даних.

Отже **метою** є підвищення якості обробки тексту, зокрема, поліпшення точності розпізнавання слів, які мають більше одного значення, за допомогою розробки та впровадження нейронної мережі, яка буде мати можливість усувати неоднозначності омографів у текстових даних в реальному часі.

Виходячи з цього **об'єктом дослідження** є процес усунення неоднозначності омографів у текстових даних. Тому **предметом дослідження** є модель нейронної мережі для усунення неоднозначності омографів у текстових даних.

Даний об’єкт дослідження повинен вирішувати **задачу** розробки моделі нейронної мережі, яка буде здатна підвищити якість обробки тексту, зокрема, поліпшити точність розпізнавання слів, які мають більше одного значення.

Під час розробки моделі нейронної мережі для усунення омографів у текстових даних використовуються такі **методи дослідження**: системний аналіз, методи глибокого навчання, теорія нейронних мереж, методи обробки та підготовки даних, імітаційне моделювання.

Завдання дослідження:

1. Аналіз технологій і методів для усунення неоднозначності омографів у текстових даних.
2. Визначення типу нейронної мережі.
3. Аналіз додаткових архітектурних компонентів для підвищення якості моделі.
4. Розробка архітектури моделі нейронної мережі з використанням додаткових архітектурних компонентів.
5. Вибір джерел даних для навчання моделі.
6. Аналіз мови програмування і бібліотек для програмної реалізації.
7. Розробка програмної реалізації.
8. Тестування і дослідження даної моделі для визначення ефективних параметрів моделі.
9. Оцінка якості моделі.
10. Визначення сфер в яких модель може використовуватись.
11. Практичне використання моделі.

РОЗДІЛ 1.
АНАЛІЗ ЗАДАЧІ УСУНЕННЯ НЕОДНОЗНАЧНОСТІ ОМОГРАФІВ В ТЕКСТОВИХ ДАНИХ З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ

1.1 Технології і методи вирішення задач лексичної неоднозначності омографів

В сучасному ландшафті обробки природної мови та комп'ютерного зору, проблема неоднозначності омографів виявляється надзвичайно актуальною. З інтенсивним використанням текстової інформації в різних галузях — від пошукових систем та автоматичного перекладу до аналізу соціальних мереж та обробки природної мови — виникає необхідність ефективного усунення лексичної неоднозначності. Відсутність однозначності в розумінні значень слів, що мають різні семантичні відтінки в різних контекстах, створює виклики для точного розпізнавання та аналізу текстових даних.

Розвиток різноманітних технологій для усунення цієї неоднозначності відображається в низці передових підходів та стратегій. Засоби штучного інтелекту, особливо глибоке навчання та машинне навчання, стали ключовими гравцями у подоланні викликів, пов'язаних із лексичними неоднозначностями. Динамічні та інноваційні методи активно впроваджуються для покращення якості розпізнавання слів у різноманітних контекстах, що ставить під сумнів традиційні підходи до роботи з текстовою інформацією.

Важливим є розуміння ключових аспектів цих технологій та їхнього внеску у поліпшення розпізнавання та розуміння текстових даних. Аналізуючи взаємодію різних методів, від статистичних підходів до використання лінгвістичних ресурсів та експертних систем, можна визначити оптимальні стратегії для розв'язання завдань, пов'язаних із неоднозначністю омографів. У цьому контексті важливо розглядати не лише окремі технології, але й їхнє взаємодію та можливість створення гібридних моделей для досягнення кращих результатів.

Таким чином існує низька технологій, які потрібно дослідити і виявити, яка з них буде найліпшої для поставленої задачі:

- машинне навчання та нейронні мережі
- статистичні методи
- використання лінгвістичних ресурсів
- підтримка експертних систем
- синтаксичний та семантичний аналіз

1.1.1 Машинне навчання

Машинне навчання – це галузь штучного інтелекту та інформатики, яка зосереджується на використанні даних і алгоритмів для імітації способу навчання людей, поступово покращуючи його точність [1]. Нейронна мережа — це серія алгоритмів, які намагаються розпізнати базові зв’язки в наборі даних за допомогою процесу, який імітує роботу людського мозку [2]. Таким чином це дає можливість використовувати класифікатори та нейронні мережі для автоматичного визначення значень омографів на основі їх контексту та використання великих корпусів текстових даних для навчання моделей.

Використання потужних глибоких нейронних мереж для автоматичного вивчення складних залежностей між словами та їх контекстом призводить до того, що цей підхід дозволяє моделям адаптуватися до широкого спектру лінгвістичних варіантів. Саме тому існує така технологія, як машинне навчання і нейронні мережі. Ці три ключові концепції взаємодіють між собою в інтегрованому поєданні інтелектуального розвитку (рис. 1.1): глибоке навчання, в якому нейронні мережі використовуються для розв'язання завдань за допомогою багатошарових архітектур; Машинне навчання, що охоплює глибоке навчання в якості важливої складової, а також використовує різноманітні методи та підходи для засвоєння знань на основі даних; і, нарешті, Нейронні мережі, які є фундаментом для глибокого навчання та одночасно можуть допомагати в інших аспектах машинного навчання. Усі ці компоненти відображають

взаємопов'язану еволюцію в інтелектуальному ландшафті, де використання нейромережевих структур стає ключовою точкою зору[3].

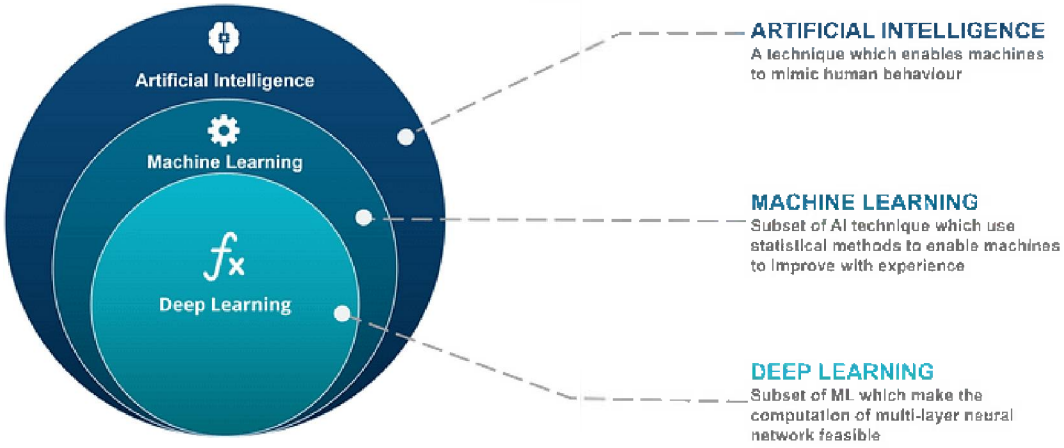

Рисунок 1.1 Графічне відображення поєднання МН, НМ, глибокого навчання.

Данна технологія включає в себе основні методи для вирішення задачі неоднозначності омографів у текстових даних:

- Методи на основі глибокого навчання
- Методи на основі контексту та зв'язків між словами
- Методи на основі великих корпусів тексту

Методи на основі глибокого навчання: дані методи використовують складні математичні моделі, які мають багато шарів для автоматичного вивчення і використання високорівневих абстракцій та представлень з великої кількості даних. Основна ідея полягає в тому, щоб забезпечити цим моделям глибше розуміння взаємозв'язків у вхідних даних шляхом автоматичного витягування значущих ознак та створення складних внутрішніх представлень.

Приклади включають в себе розпізнавання зображень, обробку мовлення або усунення неоднозначності в текстових даних. Згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN) та трансформерні архітектури є типовими структурами для використання в методах глибокого навчання.

Принцип роботи полягає в тому, щоб моделі навчались на великих наборах тренувальних даних, вирізаючи в них закономірності та здатність узагальнювати

знання на нові, раніше не бачені дані. Це досягається шляхом ітеративного коригування параметрів моделі відповідно до визначеного функціоналу втрат[4].

Методи на основі контексту та зв'язків між словами: цей підхід використовує алгоритми, які акцентують увагу на контекстуальних та лінгвістичних зв'язках між словами, розширюючи аналіз не лише на найближчий, але й на віддалений контекст. Принципи цього методу можна розглядати у двох ключових напрямках.

1. **Аналіз віддаленого контексту.** Використання алгоритмів, які охоплюють не лише слова, найближчі до омографу, але й ті, що знаходяться в віддаленому контексті. Це може включати слова, які зустрічаються в інших реченнях або навіть абзацах тексту. Такий підхід дозволяє більш повно враховувати всі можливі семантичні відносини та значення, які можуть впливати на правильне визначення омографів.
2. **Врахування синтаксичних та семантичних зв'язків.** Підвищення точності аналізу шляхом призначення ваги синтаксичним та семантичним зв'язкам між словами. Це включає в себе використання методів аналізу залежностей або семантичного розбору для визначення взаємозв'язків між словами. Врахування синтаксичних та семантичних зв'язків допомагає системі більш точно розуміти семантичний контекст, в якому використовується омограф, покращуючи точність його визначення.

Цей метод націлено на розширення глибини аналізу тексту, дозволяючи більш повно враховувати синтаксичні та семантичні взаємозв'язки слів у різних частинах тексту, що призводить до більш точного визначення значень омографів[5].

Методи на основі великих корпусів тексту: цей підхід базується на використанні обширних корпусів тексту для підтримки процесу усунення неоднозначності омографів. Два ключові елементи цього методу:

1. **Використання статистичних методів та машинного навчання.** В даному випадку використовуються методи, що базуються на

статистичних залежностях та алгоритмах машинного навчання. Ці методи аналізують частоту вживання слів та їхніх різних значень у великих корпусах тексту. Алгоритми машинного навчання можуть автоматично вивчати закономірності вживання слів та їхні контексти, що дозволяє створювати моделі для ефективного визначення значень омографів на основі статистичних взаємозв'язків[6].

2. **Врахування контексту з великих текстових корпусів.** Цей етап включає в себе використання інформації, отриманої з широких текстових корпусів, для створення моделей, які можуть ефективно вирізняти значення омографів. Великі корпуси тексту надають багатий контекст для вивчення та аналізу семантичних взаємозв'язків слів. Інтеграція інформації з великих текстових корпусів дозволяє моделям враховувати різноманітні варіанти вживання слів у різних ситуаціях, що сприяє більш точному визначенню значень омографів[7].

Використання великих корпусів тексту та методів машинного навчання в цьому контексті розширює можливості системи у вирішенні проблеми неоднозначності омографів, створюючи моделі, що добре враховують семантичний контекст слів в різних ситуаціях.

1.1.2 Статистичні методи

Статистичні методи — це підхід до обробки даних та прийняття рішень, який використовує принципи статистики та ймовірності для аналізу та інтерпретації великих масивів інформації. Ці методи враховують розподіл даних, тенденції та зв'язки між великою кількістю об'єктів або явищ для виведення статистичних закономірностей та зроблення висновків[8].

В контексті обробки природної мови, статистичні методи використовуються для аналізу текстових даних, визначення ймовірностей та моделювання лінгвістичних явищ. Наприклад, статистичні методи можуть бути

застосовані для визначення ймовірності вживання певного слова в конкретному контексті чи для аналізу частоти вживання слів у текстових корпусах.

1.1.3 Використання лінгвістичних ресурсів

Використання лінгвістичних ресурсів відноситься до процесу використання різноманітних мовних інструментів та даних для розв'язання завдань в області обробки природної мови (Natural Language Processing, NLP) або в лінгвістиці. Ці ресурси можуть бути створені вручну або автоматично, та включають в себе різні види лінгвістичних даних, словники, корпуси текстів, граматичні правила, еталонні тексти, тезауруси, мовні моделі тощо.

Використання лінгвістичних ресурсів в NLP може включати в себе створення та покращення моделей машинного навчання, навчання комп'ютерів розуміти та генерувати природну мову, аналіз текстів для витягування інформації, автоматичний переклад, визначення емоцій у тексті, тощо.

Цей підхід сприяє розробці більш ефективних та точних систем обробки природної мови, оскільки лінгвістичні ресурси надають комп'ютерам доступ до різноманітної та структурованої інформації, яка допомагає їм краще розуміти та взаємодіяти з мовними даними.

До цієї технології відносять один з методів, який теоретично може вирішувати поставлену задачу[9].

Методи, що використовують зовнішні ресурси: цей підхід базується на інтеграції лінгвістичних ресурсів, таких як словники, тезауруси та онтології, для вдосконалення визначення семантичного спектру омографів та їхнього контексту. Даний метод використовує зовнішні знання для більш точного розуміння значень слів.

Цей підхід передбачає використання різноманітних лінгвістичних ресурсів, таких як словники, тезауруси та онтології. Ці ресурси містять інформацію про семантику слів, їхні синтаксичні та семантичні відносини, що дозволяє краще розуміти контекст вживання омографів. Використання лінгвістичних ресурсів дозволяє системі не тільки визначати основні значення

слів, але і враховувати їхню семантичну ширину. Такий підхід корисний у випадках, коли контекст тексту не надає достатньої інформації для однозначного розуміння омографів.

Цей метод надає системі доступ до лінгвістичних знань, що допомагає вирішувати проблеми неоднозначності, особливо там, де інформація з корпусів тексту може бути неповна або недостатньо репрезентативна. Інтеграція зовнішніх ресурсів покращує точність визначення значень омографів, особливо у випадках, коли доступ до багатофакторного лінгвістичного контексту є ключовим[10].

1.1.4 Підтримка експертних систем

У контексті розробки моделі нейронної мережі для усунення неоднозначності омографів у текстових даних, підтримка експертних систем означає використання технологій, які поєднують лінгвістичні та мовознавчі знання з метою ефективного розуміння та вирішення випадків неоднозначності мови.

- Експертні системи: використання програмних систем, які моделюють експертне знання в області лінгвістики та мовознавства. Ці системи можуть включати в себе правила, логіку, тезауруси та інші компоненти, які враховують мовні варіанти та їх семантичні відмінності.
- Лінгвістичні та Мовознавчі Знання: інтеграція експертних знань у галузі лінгвістики та мовознавства для розробки моделі, яка може визначати різні значення омографів на основі їх контексту та синтаксичних особливостей.
- Розуміння та Вирішення Неоднозначності: застосування експертних систем для виявлення та розуміння ситуацій неоднозначності в текстах, зокрема, коли слова мають багато значень, а контекст не визначає однозначності вибору.

– Системи Інференції: використання механізмів інференції для виведення логічних висновків та визначення належного значення омографів на основі зібраних даних та експертних знань[11].

Такий підхід дозволяє використовувати лінгвістичні та мовознавчі аспекти для покращення роботи моделі нейронної мережі, забезпечуючи їй здатність більш ефективно вирішувати проблему неоднозначності омографів у текстових даних.

1.1.5 Синтаксичний та семантичний аналіз

Синтаксичний аналіз — це етап обробки природної мови, на якому вивчається синтаксична структура речення чи фрази. Основна мета полягає в визначенні граматичної структури та взаємозв'язків між словами у тексті. Цей аналіз допомагає визначити правильний порядок слів, структуру речень та визначити граматичні відношення між словами. Синтаксичний аналіз є важливою частиною обробки природної мови та машинного навчання, оскільки він допомагає розуміти граматичні особливості текстів.

Семантичний аналіз — це етап обробки природної мови, спрямований на вивчення значень слів та їхніх взаємозв'язків для розуміння смислу тексту. Основна мета полягає в визначенні семантичних відтінків, взаємозв'язків між словами та висновків про загальний смисл висловлювання чи тексту. Семантичний аналіз включає в себе розробку методів та моделей, які дозволяють комп'ютерам розуміти не лише граматичну правильність тексту, але й його смисловий зміст. В обробці природної мови, семантичний аналіз є ключовим для досягнення більш глибокого розуміння текстової інформації, зокрема в контексті завдань таких як розпізнавання інтенцій, витягування інформації та автоматичний переклад[12].

1.1.6 Технології гібридних систем

Гібридні системи в області обробки природної мови та розробки моделей нейронних мереж представляють собою інноваційний підхід, який поєднує

різноманітні технології та методи з метою досягнення ефективності у вирішенні конкретних завдань.

Основна ідея гібридних систем полягає в тому, щоб комбінувати переваги різних підходів, спрямованих на розв'язання конкретних аспектів завдання. Наприклад, можна поєднати потужність нейронних мереж, зокрема методів глибокого навчання, з класичними статистичними алгоритмами, що дозволяє отримати комплексний підхід до усунення неоднозначності омографів. Також можна використовувати лінгвістичні ресурси та експертні системи для покращення розуміння текстових даних та підвищення якості обробки.

Однією з переваг гібридних систем є їхній високий рівень гнучкості та адаптивності. Комбінування різних підходів дозволяє створювати більш адаптовані рішення, які можуть ефективно працювати в різних умовах та вирішувати завдання різного характеру. Також гібридні системи можуть бути менш чутливими до обмежень і недоліків окремих методів, оскільки різні компоненти можуть компенсувати один одного.

З іншого боку, недоліками гібридних систем можуть бути складність в їхній реалізації та конфігурації. Інтеграція різних технологій може вимагати додаткового зусилля з сторони розробників. Також, не завжди очевидно, які саме компоненти системи краще комбінувати для досягнення найкращих результатів у конкретному завданні.

Архітектура технології гібридних систем представляє собою інтеграцію різних технологій та методів з метою створення більш комплексного та адаптивного рішення (рис. 1.2). В загальному розумінні, гібридні системи можуть включати в себе різноманітні компоненти, такі як нейронні мережі, статистичні методи, лінгвістичні ресурси, та експертні системи[13].

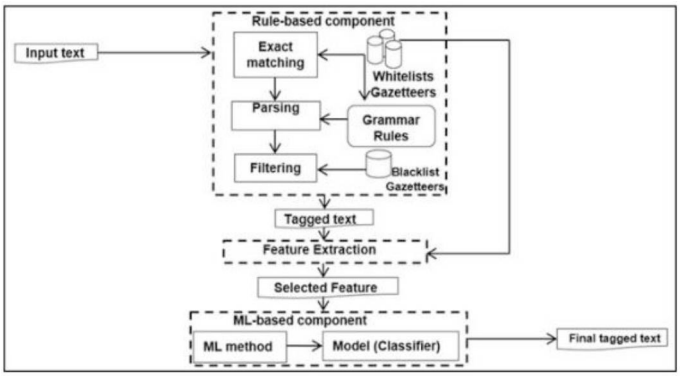

Рисунок 1.2 Архітектура гібридної системи

Данна технологія включає в себе методи, засновані на гібридних підходах, що теоретично має можливість вирішувати задачу лексичної неоднозначності омографів.

Методи, засновані на гібридних підходах: ці підходи використовують комбінацію різних методів з метою отримання оптимальних результатів у вирішенні завдань неоднозначності омографів.

1. **Комбінація різних методів.** Цей підхід передбачає поєднання різних методів, таких як глибоке навчання та статистичні підходи, для досягнення оптимальних результатів у розв'язанні задачі неоднозначності омографів. Це може включати в себе використання різних моделей та алгоритмів для вирішення різних аспектів проблеми. Поєднання різних методів дозволяє використовувати переваги кожного з них, зменшуючи обмеження та недоліки окремих підходів.
2. **Використання гібридних моделей.** Використання моделей, які комбінують переваги різних підходів, з метою досягнення кращих результатів у вирішенні проблем неоднозначності омографів. Гібридні моделі можуть бути побудовані на основі об'єднання архітектур, функцій та характеристик різних методів. Використання гібридних моделей дозволяє вирішувати складні завдання, об'єднуючи потужності різних підходів. Це може бути особливо

ефективним у випадках, коли жоден окремий метод не може забезпечити задовільні результати через складність проблеми.

Гібридні підходи намагаються злагодити переваги та недоліки різних методів, створюючи комплексні моделі, спроможні ефективно розглядати проблему неоднозначності омографів з різних сторін[14].

1.2 Методи на основі градієнтного спуску

1.2.1 Градієнтний спуск

Метою навчання нейронної мережі є зниження значення функції вартості, що фактично означає мінімізацію функції з багатьма змінними. Оскільки ця функція зазвичай складна та має велику кількість параметрів, стандартні методи пошуку екстремуму стають непридатними. Перш ніж визначити локальні екстремуми, необхідно обчислити похідні першого порядку, розв'язати систему рівнянь і визначити точки, які можуть бути потенційними екстремумами. Далі, для визначення, яка з цих точок є локальним мінімумом, потрібно обчислити похідні другого порядку. Проте цей метод дуже складний та витратний за рахунок обчислень. Таким чином, алгоритми мінімізації функцій, використовувані в машинному навчанні, переважно ґрунтуються на градієнтному спуску (рис. 1.3) [15].

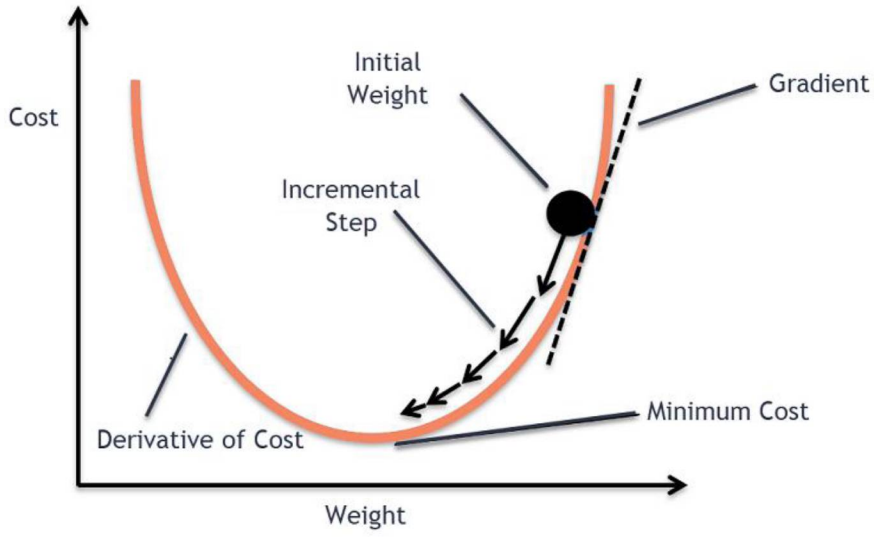

Рисунок 1.3 Алгоритми мінімізації функцій на градієнтному спуску.

Розглянемо алгоритм градієнтного спуску. Для простоти обчислень оберемо функцію вартості (1.3). Наше завдання полягає в мінімізації функції багатьох змінних, ваги і зсуви.

Визначення 1. Глобальний мінімум функції $f: X \to \mathbb{R}, x_0 \in min f <=>$

$\forall x \in X$ *виконується* $f(x) \geq f(x_0)$

Визначення 2. Локальний мінімум функції $f: X \to \mathbb{R}, x_0 \in loc\, min f <=>$

$\exists O\epsilon(x_0)$,що $\forall x \in O\epsilon(x_0)$ виконається $f(x) \geq f(x_0)$

Визначення 3. Градієнтом функції $f: \mathbb{R}n \to \mathbb{R}$ називається вектор приватних похідних.

Визначення 4. Похідна функції $f: \mathbb{R}n \to \mathbb{R}$ у напрямку e називається проекцією градієнта цей напрям.

Нехай завдання полягає у мінімізації функції $f: \mathbb{R}n \to \mathbb{R}$.

Для мінімізації функції f необхідно знайти напрямок, у якому f зменшується найшвидше. Тобто необхідно знайти $\min e \nabla f = min \|e\| \nabla f \| cos\theta$, вважаючи довжину вектору постійною, вираз зведеться до $min cos\theta$, $\theta = \pi$ тобто. значення похідної у напрямку мінімально якщо її напрямок протилежний напрямку вектору градієнта.

Таким чином можна зменшувати функцію у напрямку зворотного градієнту за таким правилом:

$$x_j' = x_j - \eta \frac{\delta f}{\delta x_j}, \tag{1.3}$$

де η – learning rate, швидкість навчання, позитивне число, що задає розмір кроку, з яким задана функція зменшується. Зазвичай за η вибирається невелике позитивне число. Так можна зменшувати значення функції, застосовуючи це правило кілька разів поспіль. Перенесемо (1.3) на функцію вартості, отримаємо:

$$w_i' = w_i - \eta \frac{\partial C}{\partial} \tag{1.4}$$

$$b_i' = b_i - \eta \frac{\partial C}{\partial b_i}^{w_i}$$

Так як функція вартості (1.3) усереднена по всіх входах, то для обчислення нових значень доведеться обчислювати градієнт ∇C для кожного входу, а потім усереднювати його. Іноді кількість таких входів може бути більшою. І таке обчислення займе багато часу. Тому зручніше проводити обчислення градієнта на випадкових невеликих наборах даних (нам не потрібно знати точне зменшення функції, нам головне рухатися в потрібному напрямку). Такі входи називають mini batch. При цьому вважають, що значення градієнта на таких партіях приблизно таке саме, як і на всіх даних. Такий алгоритм навчання називається стохастичним градієнтним спуском.

Наприклад, вся множина вхідних даних X випадково розбивається на кілька підмножин: $X1$,$X2$,… Xk, нехай при цьому розмір кожного підмножини m. Далі алгоритм тренується на кожній з цих підмножин, тобто.

$$w_i' = w_i - \eta \frac{\partial C_{X_j}}{\partial} \qquad (1.5)$$

$$b_i' = b_i - \eta \frac{\partial C_{X_j}}{\partial b_i}^{w_i}$$

де C_{Xj} – функція усереднена за набором Xj. Значення вагів та зсувів будуть змінюватися після кожного тренування на наборах Xj. Набір тренувань з усіх партій даних Xj називається епохою. Коли одна епоха закінчилася дані знову поділяються на підмножини і відбувається навчання кожному з підмножин, тобто починається наступна епоха (рис 1.4) [16].

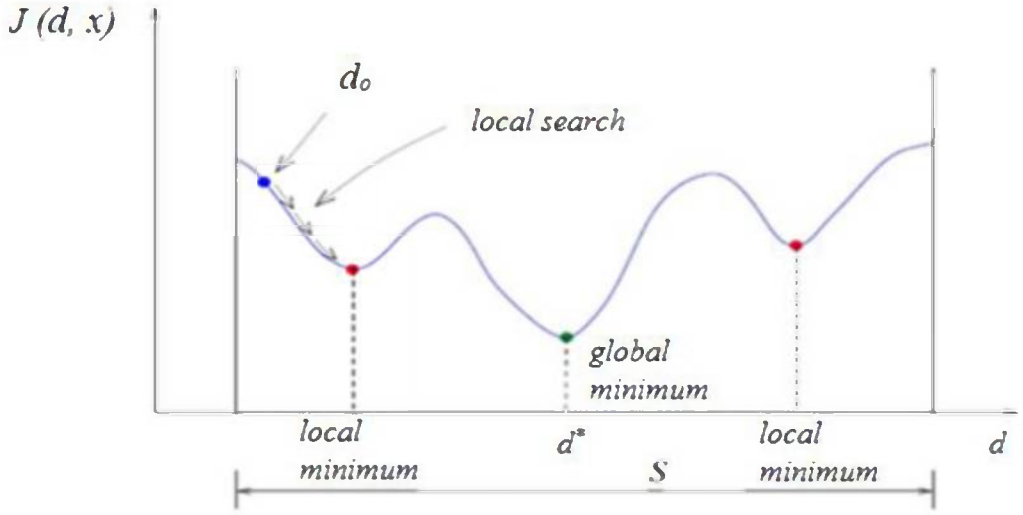

Рисунок 1.4 Локальний мінімум

У випадку, коли під час виконання алгоритму градієнтного спуску початкове положення ваги та зсуву вибрано близько до одного з локальних мінімумів, існує ризик застрягнути в цьому локальному мінімумі. Якщо значення цього локального мінімуму суттєво відрізняється від глобального мінімуму, то результат функціонування нейронної мережі може значно відхилятися від бажаного.

У глибокому навчанні ми маємо справу з мінімізацією функції багатьох змінних, яка має безліч локальних мінімумів, сідлових точок, оточених плоскою областю. Ця складність робить оптимізацію вельми витратною. Таким чином, часто доводиться приймати компроміс і шукати значення функції, яке є дуже низьким, але не обов'язково є глобальним мінімумом.

1.2.2 Алгоритм зворотного розповсюдження помилки на основі градієнтного спуску

Якщо алгоритм градієнтного спуску служить для мінімізації функції, то алгоритм зворотного поширення помилки призначений для швидкого обчислення градієнта. Для обчислення градієнта необхідно обчислювати окремі похідні: $\frac{\partial C}{\partial w_j^i}, \frac{\partial C}{\partial b_j^i}$. Обчислювати їх щоразу вручну важке завдання, оскільки нейронна мережа може бути глибокою і мати безліч параметрів. При цьому необхідно робити це швидко. Для швидкого обчислення градієнта функції

вартості придумали алгоритм зворотного поширення помилки (рис. 1.5) [17]. Розглянемо далі.

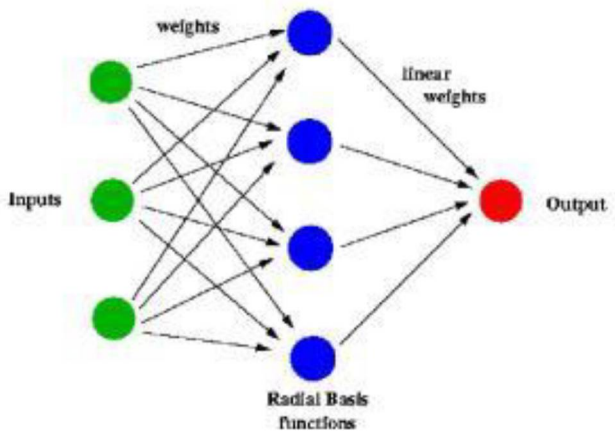

Рисунок 1.5 Алгоритм зворотного розповсюдження помилки

Символом ⊙ позначатимемо операцію Адамара.

Визначення 5. Операція Адамара – операція над двома матрицями однакової розмірності, результатом якої є матриця такої ж розмірності. Значення елемента з індексом ij виходить, як добуток елементів з таким самим індексом вихідних матриць.

Скористаємося позначеннями (1.2) даними вище. Як функцію вартості використовуємо середню квадратичну помилку (1.3), як функцію активації сигмоїд. Введемо помилку нейрона j шару l:

$$\delta_j^l = \frac{\delta C}{\delta z_j^l}. \tag{1.6}$$

Тоді загальний алгоритм зворотного розповсюдження помилки буде таким:

1. Розрахуємо активацію для кожного з шарів нейронної мережі:

$$a^l = \sigma(w^l a^{l-1} + b^l). \tag{1.7}$$

2. Обчислюємо вектор помилки останнього шару:

$$\delta^L = \nabla C \odot \sigma'(w^L a^{L-1} + b^L). \quad (1.8)$$

3. Розповсюджуємо помилку назад, для кожного шару $l=L, L-1,..2$

$$\delta l = ((wl+1)T\delta l+1) \odot \sigma'(wl-1al-1 + bl) \quad (1.9)$$

4. Обчислюємо градієнт функції вартості за формулами:

$$\frac{\delta C}{\delta b_j^l} = \delta_j^l, \frac{\delta C}{\delta w_{ij}^l} = a_j^l \delta_j^l. \quad (1.10)$$

Алгоритм називається зворотним поширенням помилки, оскільки помилка поширюється тому, перші верстви мережі. Обчислюють помилку з останнього шару і закінчуючи першим. І обчислюють всі необхідні параметри нейронної мережі за два проходи [17].

1.3 Штучні нейронні мережі

Розробка штучних нейронних мереж виходить із бажання відтворити біологічні принципи. Дослідники аналізують мережеві структури та алгоритми, використовуючи термінологію, характерну для опису організації мозкової діяльності. Проте ця аналогія має свої межі, оскільки наше розуміння роботи мозку дуже обмежене, що ускладнює пошук орієнтирів для моделювання. Таким чином, розробники нейронних мереж вимушені виходити за рамки поточних біологічних знань у пошуках структур, які можуть виконувати корисні функції.

Основним компонентом людського мозку є нейрони - спеціальні клітини, що можуть запам'ятовувати та застосовувати попередній досвід до наступних дій, що робить їх унікальними серед інших клітин організму.

Кора головного мозку людини є тонкою поверхнею товщиною від 2 до 3 мм та площею близько 2200 см2, що значно перевищує площу стандартної

клавіатури. У цій корі приблизно 1011 нейронів, що майже дорівнює кількості зірок в Чумацькому шляху. Кожен нейрон пов'язаний з 103 - 104 іншими нейронами. Загалом у мозку людини близько від 1014 до 1015 зв'язків.

Потужність людського розуму залежить від числа базових компонентів, різноманітності їх з'єднань та генетичного програмування та навчання.

Кожен окремий нейрон складний, має свої компоненти, підсистеми та механізми управління, і передає інформацію через велику кількість електрохімічних зв'язків. Існують навколо сотні різних класів нейронів. Усі разом вони та їх зв'язки формують недвійковий, нестабільний та несинхронний процес, що відрізняється від обчислень традиційних комп'ютерів. Штучні нейронні мережі моделюють лише основні елементи природного мозку, але це викликає інтерес дослідників та розробників у пошуку нових шляхів вирішення цієї проблеми.

Основні кроки в розвитку та застосуванні штучних нейронних мереж:

1. У 1943 році У. Маккалок і У. Піттс сформалізували поняття нейронних мереж у фундаментальній статті про логічні обчислення та нервову активність.
2. У 1948 році Н. Вінер та його команда опублікували роботу про кібернетику, де основною ідеєю було представлення складних біологічних процесів математичними моделями.
3. У 1949 році Д. Хебб запропонував перший алгоритм навчання.
4. У 1958 році Ф. Розенблатт винає одношаровий перцептрон і продемонстрував його здатність розв'язувати задачі класифікації.
5. У 1960 році Уідроу та його студент Хофф розробили Адалін на основі дельта правила, який став стандартним елементом багатьох систем обробки сигналів.
6. У 1963 році в Інституті проблем передачі інформації АН СРСР А.П. Петровим було проведено докладне дослідження завдань "важких" перцептронів.

7. У 1969 році М. Мінський опублікував формальний доказ обмеженості перцептрону та показав, що він нездатний вирішувати деякі завдання, пов'язані з інваріантністю уявлень, що призвело до спаду інтересу до нейронних мереж.
8. У 1972 році Т. Кохонен та Дж. Андерсон незалежно пропонують новий тип нейронних мереж, здатних функціонувати як пам'ять.
9. У 1973 році Б. В. Хакімов запропонував нелінійну модель із синапсами на основі сплайнів та впровадив її для вирішення завдань у медицині, геології, екології.
10. У 1974 році Пол Дж. Вербос та А. І. Галушкін одночасно винаходять алгоритм зворотного розповсюдження помилки для навчання багатошарових перцептронів.
11. У 1975 році Фукусіма представляє когнітрон – мережу, що самоорганізується, призначену для інваріантного розпізнавання образів, проте досягається це за допомогою запам'ятовування практично всіх станів образу.
12. У 1982 році, після періоду забуття, інтерес до нейромереж знову зріс. Дж. Хопфілд показав, що нейронна мережа із зворотніми зв'язками може представляти собою систему, що мінімізує енергію (мережу Хопфілда). Кохонен представив моделі мережі, що навчаються без вчителя (нейронна мережа Кохонена), що вирішує завдання кластеризації, візуалізації даних (карта Кохонена, що самоорганізується) та інші задачі попереднього аналізу даних.
13. У 1986 році Девідом І. Румельхартом, Дж. Є. Хінтоном і Рональдом Дж. Вільямсом і одночасно з С. І. Барцевим та В. А. Охоніним (Красноярська група) був перевідкритий та суттєво розвинений метод зворотного розповсюдження помилки. Це призвело до вибуху інтересу до навчаючихся нейронних мереж.
14. У 2007 році Джеффрі Хінтоном в університеті Торонто були створені алгоритми глибокого навчання багатошарових нейронних мереж за

допомогою обмеженої машини Больцмана (RBM – Restricted Boltzmann Machine).

Хоча спочатку у штучних нейронних мережах було багато досліджень та розробок, їх популярність знизилася через технічні обмеження. Інтенсивність обчислень для штучних нейронних мереж була надто важкою для комп'ютерів того часу. Тоді комп'ютерам не вистачало потужності, і підготовка нейронних мереж займала б надто багато часу. Це призвело до того, що інші методи машинного навчання стали більш популярними, і штучні нейронні мережі були в основному проігноровані.

Проте один важливий алгоритм, пов'язаний зі штучними нейронними мережами, був розроблений у той час – зворотне розповсюдження, відкрите Полом Вербосом. Цей алгоритм дозволив вченим набагато швидше тренувати штучні мережі.

В кінці 2000-х років із зростанням використання методів машинного навчання на великих наборах даних, зібраних від повсякденних користувачів, штучні нейронні мережі знову стали популярними. В наш час ці алгоритми головним чином використовуються для глибокого навчання, яке спрямоване на моделювання більш складних відносин, включаючи нелінійні зв'язки.

Історично, першою ключовою працею, що заложила теоретичний фундамент для розробки штучних моделей нейронів та нейронних мереж, вважається стаття Уоррена С. Мак-Каллока та Вальтера Піттса, опублікована в 1943 році, під назвою "Логічні обчислення, пов'язані з нервовою активністю". Основна ідея теорії Маккалока та Піттса полягає в тому, що будь-які явища, пов'язані з вищою нервовою діяльністю, можуть бути проаналізовані та зрозумілі як активність у мережі, що складається з логічних елементів, які приймають лише два можливих стани ("все або нічого"). При цьому, для будь-якого логічного виразу, який задовольняє умовам, визначеним авторами, можливо побудувати мережу логічних елементів, яка відображає поведінку, описану цим виразом. Ці ідеї були подальше розвинені американським нейрофізіологом

Френком Розенблаттом. Він представив схему пристрою, що моделює процеси сприйняття людиною та назвав його "перцептроном" [18]. Перцептрон передавав сигнали від фотоелементів, що утворюють сенсорне поле, блокам електромеханічних осередків пам'яті. Ці осередки сполучалися між собою випадковим чином відповідно до принципів коннективізму. Отже, перцептрон, або персептрон, є математичною чи комп'ютерною моделлю сприйняття інформації мозком (кібернетична модель мозку), яку запропонував Френк Розенблатт у 1957 році та вперше реалізував у вигляді електронної машини "Марк-1" у 1960 році. Перцептрон став однією з перших моделей нейромереж, а "Марк-1" – першим нейрокомп'ютером [19].

Отже, у ролі моделі логічного елементу Мак-Каллока і Піттса, який подальше отримав назву "формальний нейрон", була запропонована схема, наведена на (рис.1.6). З сучасної точки зору, формальний нейрон є математичною моделлю простого процесора, який має кілька входів і один вихід. Вектор вхідних сигналів (що надходять через "дендрити") перетворюється нейроном у вихідний сигнал (що передається "аксоном") за допомогою трьох функціональних блоків: локальної пам'яті, блоку сумування та блоку нелінійного перетворення. Вектор локальної пам'яті містить інформацію про вагові коефіцієнти, з якими вхідні сигнали інтерпретуються нейроном. Ці вагові коефіцієнти є аналогами чутливості пластичних синаптичних зв'язків. Вибір цих ваг забезпечує той чи інший інтегральний вигляд нейрона[3-5]. У блоку сумування відбувається накопичення загального вхідного сигналу (результат позначимо символом net), що дорівнює зваженій сумі входів: $net = \sum^{n} Wx. J=1 ii$

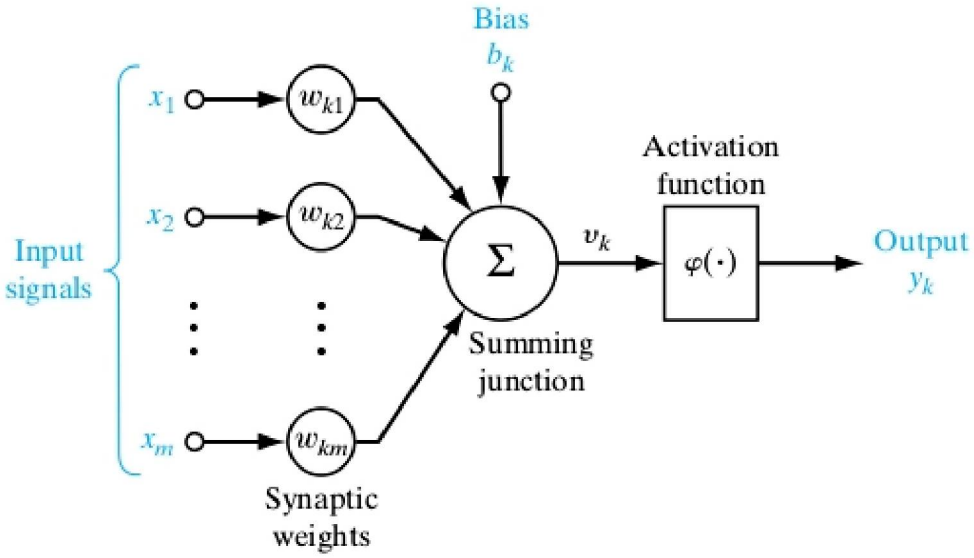

Рисунок 1.6 Функціональна схема формального нейрона Маккалока та Піттса

У моделі Маккалока і Піттса, відсутні тимчасові затримки вхідних сигналів. Отже, значення net визначає повне зовнішнє збудження, сприйняте нейроном. Відгук нейрона подальше описується принципом "все або нічого". Іншими словами, змінна проходить нелінійне порогове перетворення (функція активації), при якому вихід (стан активації нейрона) Y встановлюється рівним одиниці, якщо net > Поріг, і Y = 0 у протилежному випадку. Значення порога (зазвичай рівне нулю) також зберігається в локальній пам'яті.

Формальні нейрони можуть бути об'єднані в мережу шляхом замикання виходів одних нейронів на входи інших. За думкою авторів моделі, така кібернетична система з належно обраними вагами може представляти довільну логічну функцію. Для теоретичного опису нейронних мереж, що виходять таким чином, пропонується математична мова обчислення логічних предикатів.

Важливо зауважити, що сьогодні, через 50 років після роботи Маккалока і Піттса, вичерпної теорії синтезу логічних нейронних мереж із довільною функцією поки немає. Найбільш просунутими виявилися дослідження в галузі багатошарових систем та мереж з симетричними зв'язками. Більшість моделей базуються на різних модифікаціях формального нейрона.

Отже, результат роботи нейрона - це функція активації, отримана як сума вагованих входів нейронів, плюс зміщення (оскільки суматор є лінійною

комбінацією входів X і ваг W, вихід нейрона Y = f(z) = f(X1*W1 + ... XnWn) - фактично є лінійною функцією. Весь спектр лінійних функцій представляє собою y = kx + b, де b - це зміщення, яке додається в суматорі).

Висновок до 1 розділу

Для розв'язання проблеми лексичної неоднозначності в текстових даних існує різноманітні підходи та методи, кожен з яких має свої переваги та обмеження. В даному розділі були детально розглянуті технології, спрямовані на усунення неоднозначності омографів.

Початково вказано, що в сучасному ландшафті обробки природної мови та комп'ютерного зору проблема неоднозначності омографів є надзвичайно актуальною. Розвиток різноманітних технологій для усунення цієї неоднозначності відображається в низці передових підходів та стратегій.

Одним із ключових напрямків є використання машинного навчання та нейронних мереж. Ці методи дозволяють автоматично вивчати залежності між словами та їх контекстом, використовуючи глибокі нейронні мережі, які можуть враховувати семантику слів. Крім того, використання попередньо навчених ембедінгів слів, таких як Word2Vec або GloVe, сприяє кращому представленню слова в векторній формі.

Таким чином було визначено, що для вирішення задачі лексичної неоднозначності омографів найкращим рішенням буде використовувати технологію машинне навчання та нейронні мережі, бо дана технологія, завдяки своїм методам, дає можливість розробити ефективну модель для усунення неоднозначності омографів у текстових даних.

РОЗДІЛ 2.
РОЗРОБКА МОДЕЛІ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ УСУНЕННЯ НЕОДНОЗНАЧНОСТІ ОМОГРАФІВ У ТЕКСТОВИХ ДАНИХ

2.1 Алгоритм нейронної мережі

Нейронна мережа — це серія алгоритмів, які намагаються розпізнати базові зв'язки в наборі даних за допомогою процесу, який імітує роботу людського мозку. У цьому сенсі нейронні мережі відносяться до систем нейронів, органічних або штучних за своєю природою.

Нейронні мережі можуть адаптуватися до змін вхідних даних; таким чином мережа генерує найкращий можливий результат без необхідності переробки критеріїв виводу. Концепція нейронних мереж, яка сягає своїм корінням у штучний інтелект , стрімко набирає популярності в розробці торгових систем. На рис. 2.1 зображена проста схема роботи алгоритму нейронної мережі.

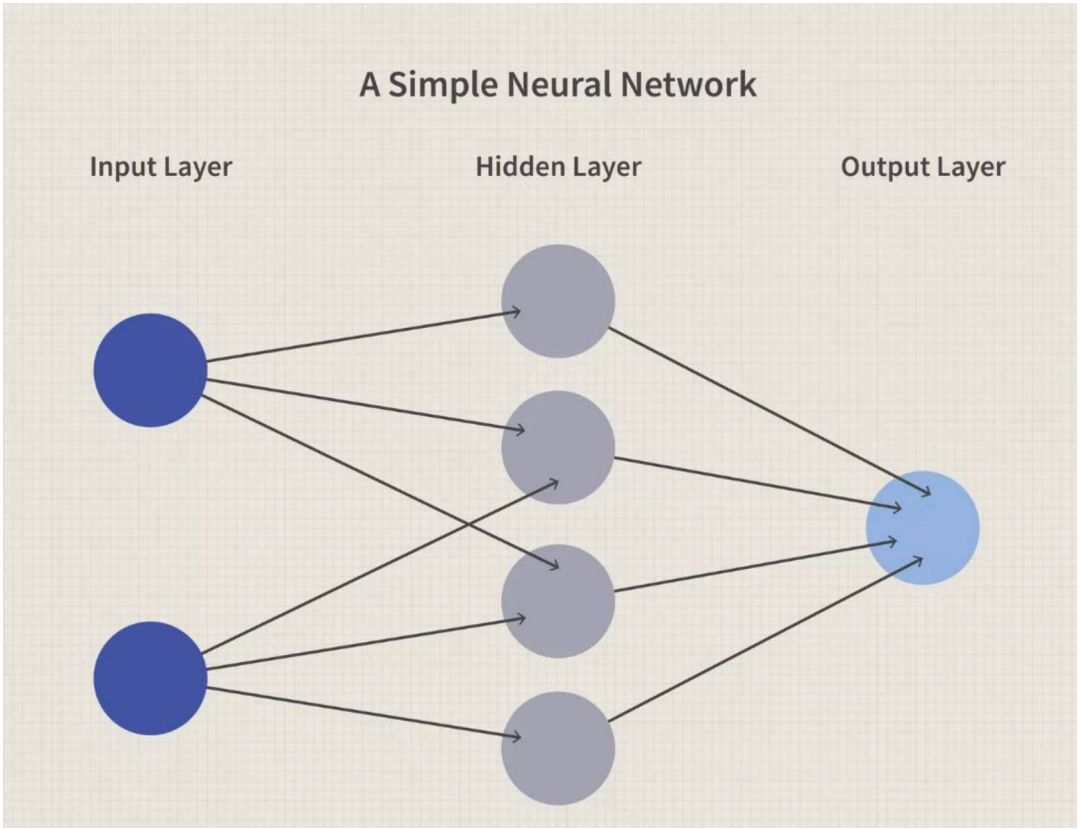

Рисунок 2.1 Алгоритм нейронної мережі

2.2 Види нейронних мереж

У сучасному світі існує безпрецедентний обсяг даних, що створює необхідність вдосконалення методів їх обробки та аналізу. В цьому контексті, нейронні мережі визначаються як ключовий інструмент для вирішення складних завдань з обробки інформації. В межах кваліфікаційної роботи буде розглянуто основні типи нейронних мереж, які можуть підвищити якість і ефективність обробки тексту, зокрема, поліпшення точності розпізнавання слів, які мають більше одного значення.

2.2.1 Нейронні мережі прямого зв’язку

Нейронні мережі прямого зв'язку є ключовим елементом глибокого навчання, і їх роль виявляється надзвичайно важливою в сучасних дослідженнях та застосуваннях штучного інтелекту (рис. 2.2). Основна концепція полягає в тому, що інформація поширюється через мережу в одному напрямку - від входу до виходу, не повертаючись назад. Це робить їх особливо ефективними для розв'язання різних завдань, таких як класифікація, регресія та розпізнавання образів.

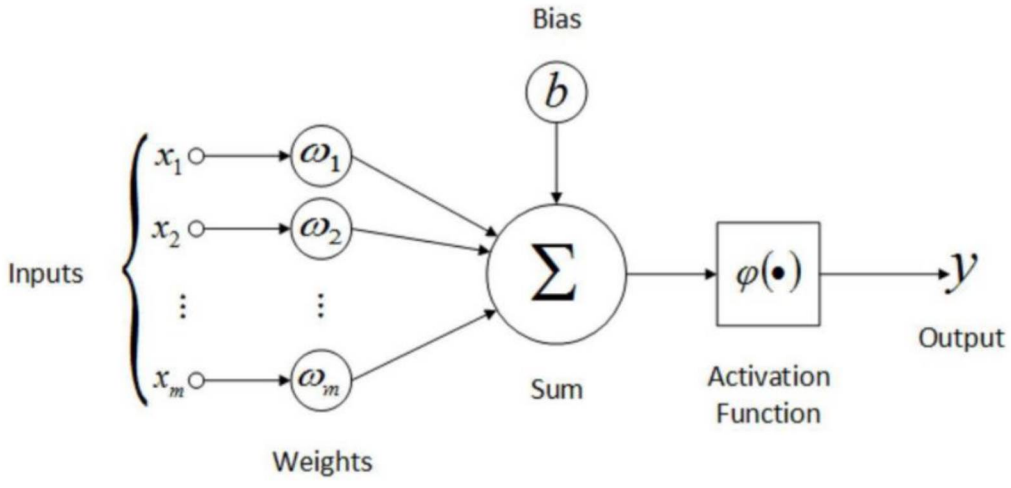

Рисунок 2.2 Нейронні мережі прямого зв'язку

Структура нейронних мереж прямого зв'язку включає в себе три основні типи шарів: вхідний, прихований та вихідний. Вхідний шар приймає вхідні дані,

приховані шари виконують операції трансформації, а вихідний шар генерує кінцеві результати. Процес передачі інформації здійснюється через нейрони, які обчислюють зважену суму вхідних сигналів та застосовують функцію активації.

Функції активації грають важливу роль у визначенні поведінки нейронів. Наприклад, сигмоїдальна функція часто використовується для бінарної класифікації, гіперболічний тангенс може використовуватися для виходів у діапазоні від -1 до 1, а функція ReLU (Rectified Linear Unit) часто застосовується для прихованих шарів для надання нейронам нелінійності.

Однією з ключових переваг нейронних мереж прямого зв'язку є їх здатність адаптуватися до складних залежностей у вхідних даних, навіть якщо ці залежності неочевидні чи важкі для формалізації. Це дає їм перевагу у вирішенні різноманітних завдань, від розпізнавання образів до розуміння природної мови.

Тренування нейронних мереж прямого зв'язку зазвичай включає в себе метод зворотного розповсюдження помилки, де ваги мережі оновлюються з урахуванням відмінностей між передбаченими та фактичними виходами. Цей процес вимагає великої кількості даних для ефективного тренування та правильного налаштування гіперпараметрів.

Нейронні мережі прямого зв'язку є важливим інструментом у сфері глибокого навчання, а їх успіх в різних застосуваннях визначається якістю тренування, архітектурою та обсягом доступних даних. Вони залишаються об'єктом інтенсивних досліджень і розвитку, сприяючи розвитку штучного інтелекту та автоматизованих систем[20].

2.2.2 Рекурентні нейронні мережі

Рекурентні нейронні мережі (RNN) є потужним класом алгоритмів глибокого навчання, спеціально призначених для роботи з послідовнісними даними. Однією з ключових особливостей RNN є їхній здатність враховувати інформацію з попередніх кроків в послідовності, роблячи їх особливо ефективними для

завдань, таких як мовний аналіз, генерація тексту, прогнозування часових рядів та інші сценарії, де важливий контекст.

Основна архітектура RNN включає три основні шари (рис. 2.3): вхідний, прихований та вихідний. Ключовим елементом є зв'язки між нейронами прихованого шару, які дозволяють передавати інформацію від попередніх часових кроків до поточного. Кожен нейрон прихованого шару використовує вхідні дані та виходи своїх власних нейронів з попередніх кроків, що дозволяє зберігати та використовувати контекст.

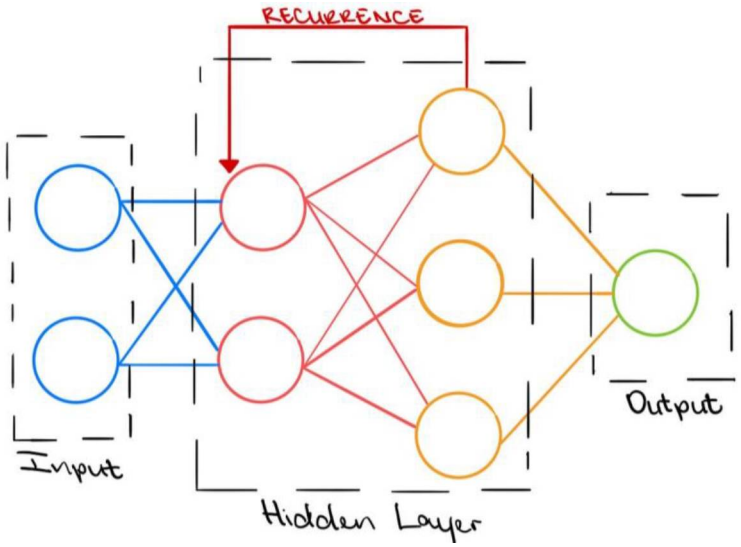

Рисунок 2.3 Архітектура рекурентних нейронних мереж.

У вхідному шарі оброблюються вхідні дані, а кожен часовий крок в послідовності оброблюється окремо. Результат обчислень на кожному кроці використовується для визначення вихідних значень та оновлення стану мережі для наступного кроку.

Однак стандартні RNN мають свої обмеження, такі як проблеми зниклих та вибухлих градієнтів, що може ускладнювати тренування та призводити до втрати інформації на довгих послідовностях. Тому були розроблені покращені версії RNN, такі як довготривалі рекурентні мережі (LSTM) та мережі з відомим градієнтом (GRU), які вирішують ці проблеми та поліпшують здатність моделей до розуміння довгострокових залежностей в даних[21].

Таким чином, RNN залишаються важливим інструментом в глибокому навчанні, що знаходить застосування в різноманітних областях, де необхідно моделювати та розуміти динаміку послідовностей.

2.2.3 Модульні нейронні мережі

Модульні нейронні мережі (MNN) представляють інноваційний підхід до побудови нейромереж. Їхню структуру (рис. 2.4) характеризує розділ функціональності на невеликі автономні модулі, кожен з яких відповідає за власну функцію або завдання. Це робить MNN гнучкими та легко модифікуваними для різних завдань.

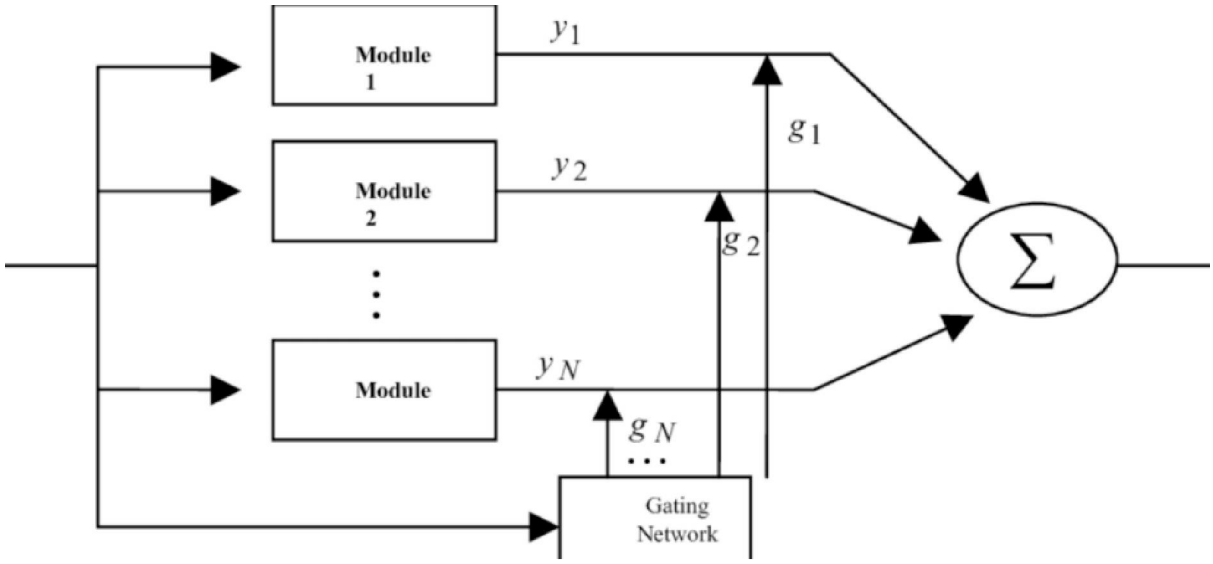

Рисунок 2.4 Архітектура модульних нейронних мереж.

Однією з ключових особливостей модульних нейромереж є їхня універсальність та здатність адаптуватися до різноманітних завдань у різних галузях. Кожен модуль може функціонувати автономно, навчаючись на своїх вхідних даних та оптимізуючи параметри для максимальної ефективності вирішення конкретного завдання.

Ця модульність дозволяє легко модифікувати та розширювати архітектуру MNN, додавати чи заміняти модулі без повного перенавчання мережі. Такий підхід забезпечує швидку адаптацію до нових завдань та змінних умов[22].

Модульні нейромережі виявляють широке застосування в галузі комп'ютерного зору, де вони використовуються для аналізу великих обсягів візуальної інформації та розпізнавання образів. Також, у сфері обробки природної мови, MNN можуть бути ефективно використані для розбиття завдань на модулі, спрощуючи аналіз тексту та мовленнєвих данних. Завдяки своїй гнучкості та універсальності, модульні нейромережі є потужним інструментом для розв'язання різноманітних завдань штучного інтелекту.

2.2.4 LSTM мережі

Мережі LSTM були розроблені спеціально для подолання проблеми тривалої залежності, з якою стикаються рекурентні нейронні мережі RNN (через проблему зникаючого градієнта). LSTM мають зворотні зв’язки , що відрізняє їх від більш традиційних нейронних мереж прямого зв’язку. Ця властивість дозволяє LSTM обробляти цілі послідовності даних (наприклад, часові ряди), не обробляючи кожну точку в послідовності незалежно, а радше зберігаючи корисну інформацію про попередні дані в послідовності, щоб допомогти з обробкою нових точок даних. Як результат, LSTM особливо добре обробляють послідовності даних, такі як текст, мова та загальні часові ряди.

Приклад. Уявіть, що ми намагаємося передбачити місячні продажі морозива. Як і можна було очікувати, вони сильно відрізняються залежно від місяця року, найнижчі в грудні та найвищі в червні.

Мережа LSTM може вивчати цей шаблон, який існує кожні 12 періодів часу. Він не просто використовує попередній прогноз, а радше зберігає довгостроковий контекст, який допомагає йому подолати проблему довгострокової залежності, з якою стикаються інші моделі. Варто зазначити, що це дуже спрощений приклад, але коли шаблон розділений набагато довшими проміжками часу (наприклад, у довгих уривках тексту), LSTM стають дедалі кориснішими[23].

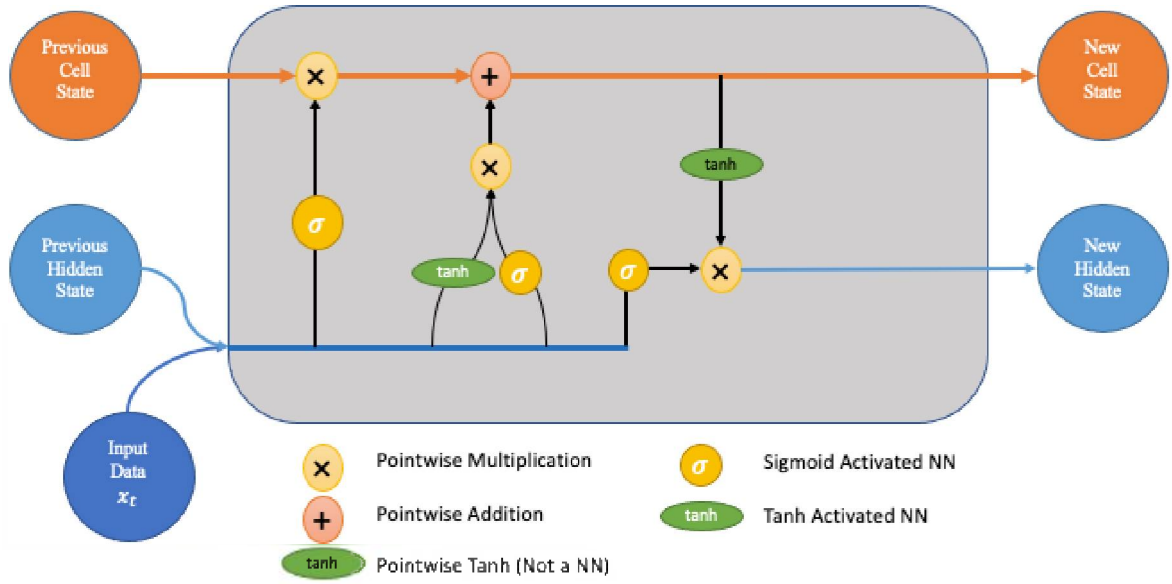

Рисунок 2.5 Діаграма LSTM

На рис. 2.5 зображена діаграма LSTM за допомогою якої буде розроблений алгоритм, для того щоб розробити власну модель нейронної мережі, потрібно зрозуміти яким чином на цій діаграмі все працює.

Я розбив роботу на декілька кроків, щоб пояснити принцип роботи LSTM.

Крок 1

Першим кроком у цьому процесі є пропуск . Тут ми вирішимо, які біти стану комірки (довгострокової пам’яті мережі) є корисними, враховуючи як попередній прихований стан, так і нові вхідні дані. Для цього попередній прихований стан і нові вхідні дані подаються в нейронну мережу. Ця мережа генерує вектор, де кожен елемент знаходиться в інтервалі [0,1] (забезпечується використанням сигмоїдної активації).Ця мережа (всередині шлюзу забуття) навчена таким чином, що вона виводить значення, близьке до 0, коли компонент вхідних даних вважається нерелевантним, і ближче до 1, якщо є релевантним. Корисно розглядати кожен елемент цього вектора як свого роду фільтр/сито, яке пропускає більше інформації, коли значення наближається до 1.

Ці виведені значення потім надсилаються вгору та поточково множаться на попередній стан клітинки. Це поточкове множення означає, що компоненти стану комірки, які були визнані нерелевантними мережею забутих воріт, будуть

помножені на число, близьке до 0, і, таким чином, матимуть менший вплив на наступні кроки.

Підсумовуючи, шлюз забуття вирішує, які фрагменти довготривалої пам’яті тепер слід забути (мають меншу вагу), враховуючи попередній прихований стан і нову точку даних у послідовності.

Крок 2

Наступний крок включає нову мережу пам'яті та вхідний шлюз . Мета цього кроку — визначити, яку нову інформацію слід додати до довготривалої пам’яті мережі (стан комірки), враховуючи попередній прихований стан і нові вхідні дані.

І нова мережа пам’яті, і вхідний шлюз самі по собі є нейронними мережами, і обидва приймають однакові вхідні дані, попередній прихований стан і нові вхідні дані. Варто зазначити, що вхідні дані тут фактично такі ж, як і вхідні дані для воріт забуття !

1. Нова мережа пам’яті — це активована нейронна мережа tanh, яка навчилася поєднувати попередній прихований стан і нові вхідні дані для створення «нового вектора оновлення пам’яті». Цей вектор по суті містить інформацію з нових вхідних даних з урахуванням контексту з попереднього прихованого стану. Цей вектор повідомляє нам, скільки оновлювати кожен компонент довгострокової пам’яті (стан комірки) мережі з урахуванням нових даних. Зауважте, що тут ми використовуємо tanh, оскільки його значення лежать в [-1,1] і тому можуть бути від’ємними. Можливість негативних значень тут необхідна, якщо ми хочемо зменшити вплив компонента на стан клітини.
2. Однак у першій частині вище, де ми генеруємо новий вектор пам’яті, є велика проблема: насправді не перевіряється, чи варто запам’ятовувати нові вхідні дані. Ось тут і з’являється вхідний шлюз . Вхідний шлюз — це сигмоїдна активована мережа, яка діє як

фільтр, ідентифікуючи, які компоненти «нового вектора пам'яті» варто зберегти. Ця мережа буде виводити вектор значень у [0,1] (через сигмоїдну активацію), дозволяючи їй діяти як фільтр через поточкове множення. Подібно до того, що ми бачили у воротах забуття, вихідний сигнал, близький до нуля, говорить нам, що ми не хочемо оновлювати цей елемент стану комірки.

3. Результати частин 1 і 2 поточково множаться. Це призводить до того, що величина нової інформації, яку ми вибрали в частині 2, регулюється та встановлюється на 0, якщо це необхідно. Отриманий комбінований вектор потім додається до стану комірки, що призводить до оновлення довгострокової пам'яті мережі.

Крок 3

Тепер, коли оновлення довгострокової пам'яті мережі завершені, ми можемо перейти до останнього кроку, вихідного шлюзу , вирішуючи новий прихований стан. Щоб вирішити це, ми використаємо три речі; щойно оновлений стан клітинки, попередній прихований стан і нові вхідні дані.

Можна подумати, що ми можемо просто вивести оновлений стан комірки; однак це можна порівняти з тим, як хтось вивантажує все, що вони коли-небудь дізналися про фондовий ринок, коли його запитують, чи він думає, що він піде вгору чи впаде завтра!

Щоб запобігти цьому, ми створюємо фільтр, вихідний шлюз , точно так само, як ми робили в мережі забутого шлюзу. Вхідні дані однакові (попередній прихований стан і нові дані), а активація також сигмоїдна (оскільки ми хочемо, щоб властивість фільтра була отримана з виходів у [0,1]).

Як згадувалося, ми хочемо застосувати цей фільтр до щойно оновленого стану комірки. Це гарантує виведення лише необхідної інформації (збереження в новому прихованому стані). Однак перед застосуванням фільтра ми пропускаємо стан комірки через tanh, щоб примусово ввести значення в інтервал [-1,1] [24].

Поетапний процес для цього останнього кроку такий:

- Застовуємо функцію tanh до поточного стану комірки поточечно, щоб отримати стан стиснутої комірки, який тепер знаходиться в [-1,1].
- Передається попередній прихований стан і поточні вхідні дані через активовану сигмовидною нейронною мережею, щоб отримати вектор фільтра.
- Застосовується цей вектор фільтра до стану здавленої комірки шляхом поточкового множення.
- Виводиться новий прихований стан.

2.3 Модель з ембедінгами

Ембедінги, як перший шар у моделі, відіграють ключову роль у представленні слів або токенів тексту у векторній формі фіксованої довжини. Їхнє використання дозволяє створити простір, в якому слова з схожими семантичними значеннями розташовані близько одне до одного, надаючи моделі можливість узагальнювати залежності між словами на основі їхнього смислового контексту.

Процес перетворення слів в вектори фіксованої довжини може бути реалізований двома основними методами. Перший - це навчання ембедінгів безпосередньо на великому корпусі тексту під час тренування моделі. Другий - використання попередньо навчених ембедінгів, таких як Word2Vec або GloVe, які вже мають векторні представлення для слів. Останній метод може бути особливо ефективним, коли відсутність великого обсягу данних для навчання.

Модель ембедінг може бути відображена на рисунку 2, де видно, як слова перетворюються в вектори фіксованої довжини. Цей процес відбувається через навчання внутрішніх параметрів моделі, які оптимізуються для забезпечення максимальної відтворюваності семантичних відносин між словами [25].

Ембедінги стали важливим інструментом для роботи з текстовими даними, дозволяючи моделям здійснювати ефективний аналіз та розуміння семантичного контексту слів. Використання ембедінгів в моделях глибокого навчання стало стандартною практикою, що покращує якість та ефективність обробки текстової

інформації.

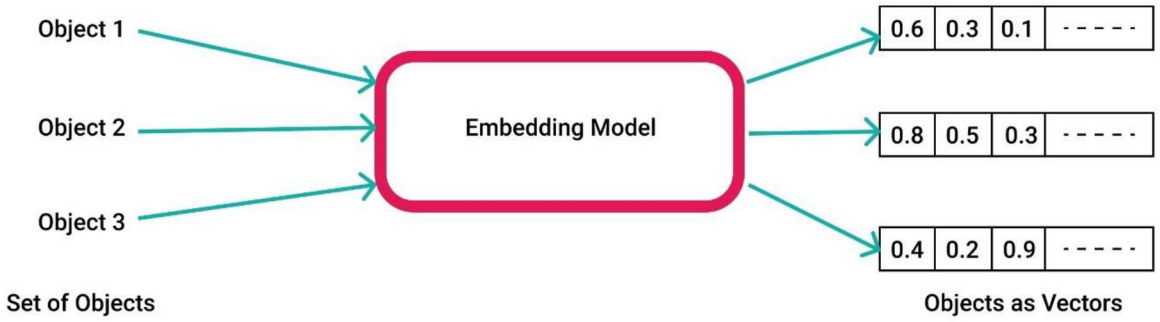

Рисунок 2.6 Модель ембедінг

2.4 Повнозв'язані шари

Нейронні мережі — це набір залежних нелінійних функцій. Кожна окрема функція складається з нейрона (або перцептрона). У повністю зв’язаних шарах нейрон застосовує лінійне перетворення до вхідного вектора через матрицю ваг. Потім до продукту застосовується нелінійне перетворення за допомогою нелінійної функції активації f (2.1) .

$$y_{jk}(x) = f(\sum_{i=1}^{n_H} w_{jk}x_i + w_{j0} \qquad (2.1)$$

Тут ми беремо скалярний добуток між матрицею ваг W і вхідним вектором x. Член зсуву (W0) можна додати всередину нелінійної функції. Я ігноруватиму це протягом решти статті, оскільки це не впливає на розміри вихідних даних або прийняття рішень і є лише додатковою вагою.

Якщо ми візьмемо шар у повністю зв’язаній нейронній мережі з розміром входу дев’ять і розміром виходу чотири, операцію можна візуалізувати таким чином на рис. 2.8 ілюстрація, яка представляє вхідні дані повністю підключеного шару:

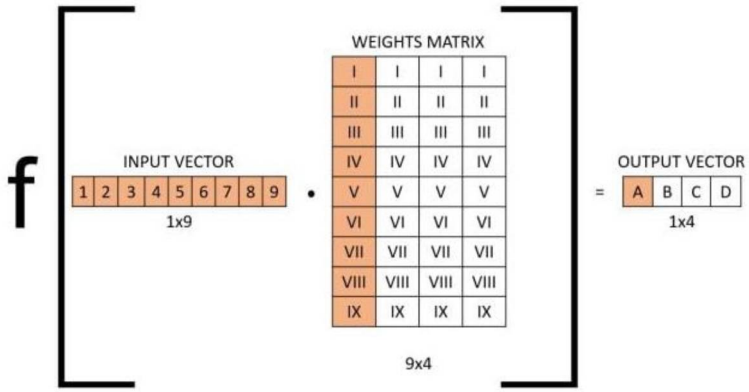

Рисунок 2.7 Ілюстрація, що представляє вхідні дані повністю підключеного шару, помножені на матрицю вагових коефіцієнтів для отримання вихідного вектора.

Функція активації “f” обертає скалярний добуток між вхідними даними шару та матрицею ваг цього шару. Зауважте, що всі стовпці в матриці ваг матимуть різні номери та оптимізовані під час навчання моделі.

Вхідні дані — вектор 1x9, матриця ваг — матриця 9x4. Взявши скалярний добуток і застосувавши нелінійне перетворення з функцією активації, ми отримуємо вихідний вектор (1x4) [26].

Візуалізація цього шару на рис. 2.9:

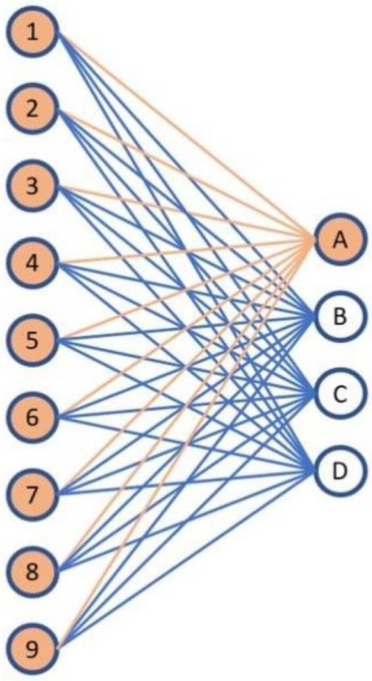

Рисунок 2.8 Ілюстрація повністю підключеного шару.

2.5 Математична модель для усунення неоднозначності омографів

Після проведених досліджень слід побудувати математичну модель для усунення неоднозначності омографів у текстових даних.

Ембедінги. Так як було визначено, що модель має введення тексту і кожне слово індексується, тому Ембедінг-шар перетворює ці індекси у вектори фіксованої довжини. Нехай input_size - розмір словника, а embedding_dim - розмірність вектора ембедінга. Побудова математичної моделі у формулі (2.2)

$$Embedding\ Layer: E(x) \in R^{(sequence\_length, embedding\_dim)} \quad (2.2)$$

де sequence_length - довжина введеного тексту, а E - матриця ембедінгів.

LSTM-шар. Цей шар призначений для роботи з послідовністю даних, зберігаючи деяку інформацію в пам'яті. LSTM-шар взаємодіє з векторами ембедінгів на кожному кроці часу, при цьому враховує попередній прихований стан h_{t-1} та клітинний стан c_{t-1}. На кожному кроці часу t, LSTM вирішує, які частини інформації важливі для збереження у прихованому стані h_t та клітинному стані c_t.

Нехай hidden_size - розмірність прихованого стану LSTM. Побудова математичної моделі у формулі (2.3)

$$h_t, c_t = LSTM(E(x)) \quad (2.3)$$

Де $h_t \in R^{hidden\_size}$ – вектор прихованого стану,

$c_t \in R^{hidden\_size}$ – вектор клітинного стану.

Повнозв'язані шари. Далі додається кілька повнозв'язаних шарів для обробки інформації, отриманої з LSTM-шару. Таким чином вектор прихованого стану h_t передається через повнозв'язані шари для отримання прогнозу значення слова. Для того щоб відображати різні можливі значення слів необхідно використовувати функції активації та шари. Отже нехай output_size - розмір виходу мережі. Математично це виражено у формулі (2.4)

(2.4)

Повнозв’язаний шар 1: $z_1 = W_1 * h_t + b_t$

Функція активації (ReLU): $a_1 = ReLU(z_1)$

Повнозв’язаний шар 2: $z_2 = W_2 * a_1 + b_2$

У цій формулі W1, b1, W2, b2 – ваги і зсуви повнозв’язаних шарів

Вихідний шар. Після обробки всієї послідовності використовується функція активації softmax для отримання ймовірностей різних можливих значень слова. Математично це виражено у формулі (2.5)

$$\hat{y} = Softmax(z_2) \quad (2.5)$$

Де Softmax – є функція активації для отримання ймовірностей по різних класах.

Для більш детального розуміння, побудуємо графічне відображення моделі (рис. 2.9).

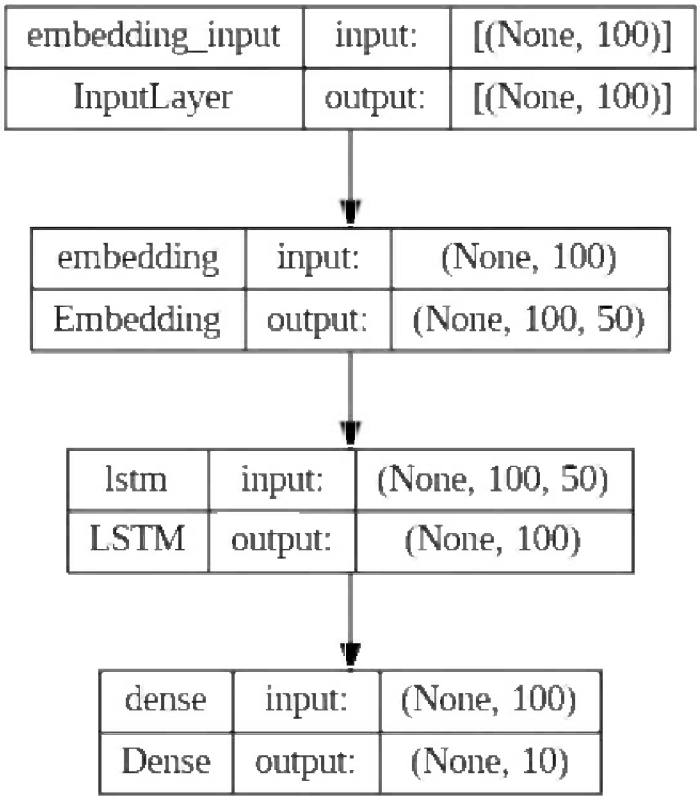

Рисунок 2.9 Модель нейронної мережі для усунення неоднозначності омографів у текстових даних

Input Layer (Вхідний шар): цей шар відображає вхідні дані в модель. Даним вхідним шаром є Embedding з параметрами, які визначають розмір словника (vocab_size), розмірність вектора ембедінгу (embedding_dim) і довжину вхідної послідовності (input_length).

Embedding Layer (Шар ембедінгу): цей шар використовується для перетворення індексів слів у вектори ембедінгу. Вектор ембедінгу - це числовий вектор, який представляє слово у просторі високої розмірності. Величина input_length вказує на максимальну довжину вхідних послідовностей.

LSTM Layer (Шар LSTM): шар довготривалої короткострокової пам'яті (LSTM) використовується для моделювання довготривалих залежностей в послідовностях. Він приймає вхід від шару ембедінгу і генерує внутрішній стан, який передається наступним шарам.

Dense Layer (Повнозв'язаний шар): цей шар використовується для класифікації або регресії. Для вирішення поставленої задачі використовується функція активації softmax для визначення ймовірностей кожного класу.

Output Layer (Вихідний шар): шар визначає вихідні ймовірності для різних класів у задачі класифікації.

Зв'язки між шарами: стрілки між шарами представляють потік даних від одного шару до іншого. Вони також вказують напрямок передачі інформації.

Висновок до 2 розділу

Для вирішення поставленої задачі були проаналізовані види нейронних мереж, що спричинило вибору нейронної мережі з довгою короткостроковою пам’яттю (LSTM), яка призначена для вирішення проблеми зникнення градієнта, що часто виникає в рекурентних нейронних мережах під час тренування на довгих послідовностях і саме тому було виявлено, що для розробки даної моделі, LSTM є найкращим вибором. В ході дослідження було також виявлено, що для досягнення кращого результату використання LSTM у поєднанні з моделлю ембедінгів і повнозв'язаних шарів може підвищити якість моделі.

Таким чином після проведення досліджень було розроблену модель нейронної мережі, зокрема використання ембедінгових моделей, LSTM-шару та повнозв’язаних шарів, що є новизною використання цих архітектурних компонентів для вирішення поставленої задачі. Застосування цих компонентів сформувало архітектуру моделі, спеціально адаптовану для усунення неоднозначності омографів у текстових даних.

У такій архітектурі нейронної мережі виокремлюється здатність моделі розуміти контекст та семантику текстових даних, що дозволяє їй ефективно вирішувати завдання неоднозначності омографів та проводити класифікацію на основі текстової інформації. LSTM-шар забезпечує можливість моделі враховувати довгострокові залежності у тексті, що є важливим аспектом у роботі з послідовними даними, такими, як тексти. Така комплексна архітектура відкриває широкі можливості для використання моделі в завданнях, пов'язаних із текстовою обробкою та аналізом.

РОЗДІЛ 3.
РОЗРОБКА ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ДЛЯ МОДЕЛІ УСУНЕННЯ НЕОДНОЗНАЧНОСТІ ОМОГРАФІВ У ТЕКСТОВИХ ДАНИХ

3.1 Вибір джерел данних для навчання

Сучасні задачі машинного навчання, зокрема розробка моделі для усунення неоднозначності омографів у текстових даних, вимагають відповідних та репрезентативних даних для ефективного тренування та валідації моделі.

Перед початком розробки моделі важливо з'ясувати, чому необхідні репрезентативні дані для навчання. Репрезентативні дані забезпечують відтворення реального світу та враховують різноманітність мовленнєвих ситуацій, що допомагає моделі бути більш універсальною та точною у своїх передбаченнях.

Існує різноманіття лексичних ресурсів, доступних для використання в машинному навчанні, таких як WordNet, SemCor, YARN і багато інших.

WordNet - це лексичний ресурс, який включає синсети, словники, омографи та інші лінгвістичні елементи (рис. 3.1). Його структуровані дані дозволяють отримати глибокий контекст для слів, допомагаючи вирішувати завдання семантичної дисамбігуації. Даний ресурс містить синсети, що об'єднують слова за семантичним значенням, а також відносини між цими синсетами, які допомагають розуміти семантику слів[27].

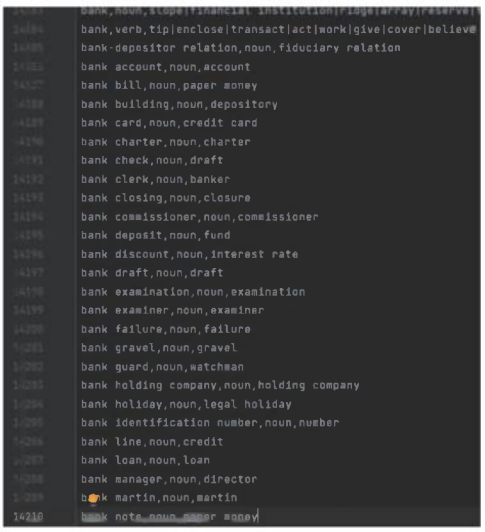

Рисунок 3.1 Вигляд у якому зберігаються дані

SemCor - це корпус тексту, анотований з використанням WordNet, який надає текстові дані з призначеними сенсами для деяких слів. Даний ресурс допомагає взаємодіяти з текстовим матеріалом та вирішувати проблему семантичної дисамбігуації, де декілька сенсів призначаються одному слову[28].

YARN - це ресурс, який надає пари синонімів для слів англійської мови, а також семантичні відносини. Він допомагає у вирішенні проблеми вибору синонімів для слів у тексті та розумінні семантичних зв'язків[29].

Порівнюючи ці ресурси, WordNet вирізняється своєю великою кількістю семантичних відносин та структурованими синсетами, що робить його підходящим для вирішення задач семантичної дисамбігуації. SemCor надає анотований корпус для тренування моделей семантичної дисамбігуації, в якому слова в тексті пов'язані з конкретними сенсами з WordNet. YARN, з іншого боку, допомагає у побудові семантичних зв'язків через синоніми, що може бути корисним при вирішенні завдань, пов'язаних з вибором слів або побудовою різноманітних варіантів тексту.

В даному випадку, WordNet обраний як ключовий джерело даних. Він визначається своєю здатністю надавати широкий лінгвістичний контекст, включаючи семантичні відносини, омографи, гіпероніми, синоніми та інші.

Аналізуючи якість даних у WordNet, ми проводимо етап аудиту даних, щоб виявити можливі аномалії чи шуми. Після цього застосовуємо стратегії

очищення та підготовки, такі як видалення застарілих визначень чи невірних зв'язків, для покращення якості даних.

Створюємо структуроване сховище даних, щоб забезпечити організований доступ та зручне використання WordNet під час етапів навчання та експериментів.

Вибір WordNet для навчання моделі семантичної дисамбігуації обґрунтований його важливими перевагами в якості лексичного ресурсу. Його структуровані дані, багатий лінгвістичний контекст і різноманіття семантичних відносин надають моделі достатній контекст для вирішення завдання. Безперечно, вибір джерела даних є ключовим етапом у розробці моделі, і WordNet є високоякісним інструментом для досягнення цієї мети[27].

3.2 Огляд існуючих програмних засобів для розробки моделі нейронної мережі

В рамках вирішення поставленої задачі розглянемо використання програмного засобу для розв'язання, і конкретно, оберемо мову програмування Python. Python визначається своєю простотою та виразністю синтаксису, що робить його ідеальним вибором для широкого спектру завдань, включаючи обробку текстових даних, розробку моделей машинного навчання та вирішення завдань обробки природної мови.

Використання Python надає доступ до багатофункціональних бібліотек та інструментів, спеціально призначених для обробки текстової інформації, а також для розробки та навчання моделей машинного навчання. Зокрема, бібліотеки, такі як NLTK (Natural Language Toolkit), SpaCy, та scikit-learn, забезпечують ряд інструментів для токенізації, лематизації, векторизації тексту, а також для побудови та навчання моделей.

Однією з переваг використання Python є активна спільнота та велика кількість відкритих джерел, що дозволяє швидко знаходити рішення для різноманітних завдань та вдосконалювати якість програмного забезпечення.

Такий підхід робить Python ефективним інструментом для вирішення завдань, пов'язаних з обробкою текстової інформації та мовним аналізом.

PyCharm — це інтегроване середовище розробки (IDE). IDE – це інструменти кодування, які дозволяють тестувати та налагоджувати свій код, а також писати його (рис. 3.2).

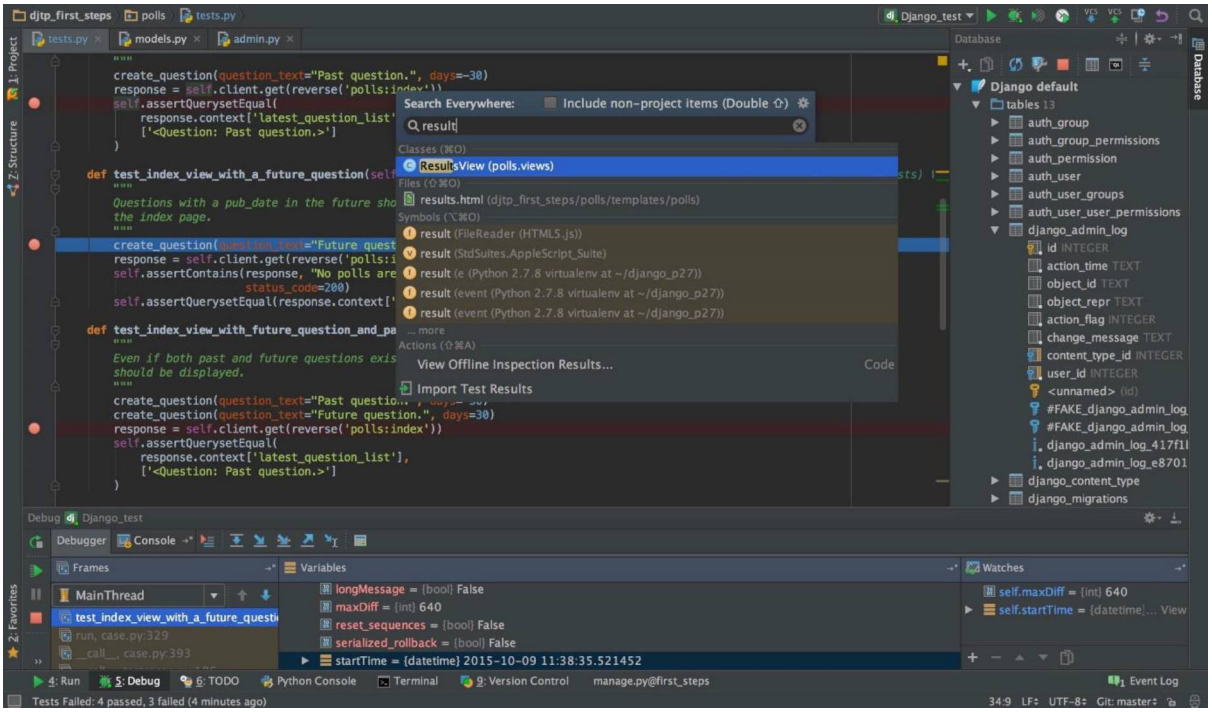

Рисунок 3.2 Вікно роботи у PyCharm.

PyCharm забезпечує першокласну підтримку Python, включаючи тестування коду, виявлення помилок та виправлення коду в режимі онлайн. Інтелектуальний пошук дозволяє перейти до будь-якої категорії, файлу, символу, навіть до дії IDE або окну інструменту. Одним кліком миші можна переключати оголошення, супер-методи, тести, використання, реалізації та багато іншого.

PyCharm включає в себе безліч інструментів, інтегрований інтерпретатор і засіб запуску тестів, інструменти відновлення роботи, вбудований термінал, інтегровану основну систему контролю версій
(включаючи Git, SVN, Mercurial), видалену розробку, інтегрований термінал ssh. PyCharm має підтримку багатьох бібліотек, за допомогою яких можна проводити аналіз даних [30].

Numpy — це бібліотека Python, яка надає математичні функції для обробки масиву великих розмірів. Вона надає різні методи/функції для масивів, метрик та лінійної алгебри. NumPy розшифровується як числовий Python. Він надає багато корисних функцій для операцій з n-масивами та матрицями в Python. Бібліотека забезпечує векторизацію математичних операцій над типом масиву NumPy, що підвищує продуктивність і прискорює виконання. За допомогою NumPy дуже легко працювати з великими багатовимірними масивами та матрицями [31].

Pandas є однією з найпопулярніших бібліотек Python для маніпуляції та аналізу даних. Pandas надає корисні функції для маніпулювання великою кількістю структурованих даних. Pandas надає найпростіший метод для виконання аналізу. Бібліотека забезпечує великі структури даних і маніпулює числовими таблицями та даними часових рядів. Pandas — ідеальний інструмент для контролю даних. Pandas розроблено для швидкого та легкого маніпулювання даними, їх агрегації та візуалізації. У Pandas є дві структури даних. Series – обробляє та зберігає одновимірні даних. DataFrame – обробляє та зберігає двовимірні дані [32].

Matplotlib — ще одна корисна бібліотека Python для візуалізації даних. Описовий аналіз і візуалізація даних дуже важливі для будь-якої організації. Matplotlib надає різноманітні методи для більш ефективної візуалізації даних. Matplotlib дозволяє швидко створювати лінійні графіки, кругові діаграми, гістограми та інші фігури професійного рівня. Використовуючи Matplotlib, можна налаштувати кожен аспект фігури. Matplotlib має інтерактивні функції, такі як масштабування, планування та збереження графіка в графічному форматі [33].

SciPy — ще одна популярна бібліотека Python аналізу даних. Scipy надає чудові функціональні можливості для наукової математики та обчислювального програмування. SciPy містить підмодулі для оптимізації, лінійної алгебри, інтеграції, інтерполяції, спеціальних функцій, обробки сигналів і зображень [34].

Sklearn – це бібліотека Python для машинного навчання. Sklearn надає різні алгоритми та функції, які використовуються в машинному навчанні. Sklearn

побудований на NumPy, SciPy і matplotlib. Sklearn надає легкі та прості інструменти для датамайнінгу та аналізу даних. Бібліотека надає користувачам набір загальних алгоритмів машинного навчання через узгоджений інтерфейс. Scikit-Learn допомагає швидко впроваджувати популярні алгоритми в набори даних і вирішувати реальні проблеми [35].

3.3 Розробка програмної реалізації

3.3.1 Підготовка даних

Попередня обробка даних є важливим етапом у підготовці даних для використання їх у моделі машинного навчання. Основні мети попередньої обробки даних включають:

Очищення даних: Видалення непотрібної або неінформативної інформації, такої як зайві символи, пробіли, знаки пунктуації. Це допомагає у покращенні якості даних та ефективності моделі.

Токенізація: Розбиття тексту на окремі "токени" або слова. Це необхідно для подальшого представлення тексту у вигляді числових векторів, зрозумілих для моделі.

Лематизація та стемінг: Зменшення слова до його базової форми (леми) або кореневої форми (стему). Це допомагає моделі у виборі ключових слов для розпізнавання семантики.

Видалення стоп-слів: Слова, які не несуть значущої інформації (так звані "стоп-слова"), можуть бути вилучені для поліпшення ефективності моделі та зменшення розміру набору даних.

Побудова словника: Створення словника, який містить відображення слів на їх числові індекси. Це необхідно для подальшого використання слів у векторному представленні.

Обробка пропущених значень: Обробка та заповнення відсутніх даних, якщо такі є, для уникнення проблем при навчанні моделі.

Всі ці етапи спрямовані на те, щоб підготувати дані до використання у моделі, забезпечити їхню якість та зрозумілість для алгоритмів машинного навчання. Також попередня обробка даних допомагає уникнути перенавчання та забезпечити більш стабільні та точні результати.

В рамках роботи було розроблено функцію, задача якої очищення даних перед тим, як передавати їх до моделі. Функція ініціалізує порожній список semcor_data, який буде використовуватися для зберігання інформації про слова, частини мови та ключі сенсів. Потім відбувається ітерація по файлових ідентифікаторах корпусу SemCor.

Для кожного файлу отримуються розмічені речення, і для кожного речення виконується ітерація по словах. У випадку, якщо слово є складним (представленим деревом), відбувається додаткова ітерація по його листках.

Для кожного слова (листка) отримується інформація, така як слово, частина мови та, за наявності, ключ сенсу. Ця інформація додається до списку.

Якщо слово є простим, витягується інформація (слово, частина мови, ключ сенсу за наявності) та додається до списку.

На завершення, функція повертає зібрану інформацію у вигляді списку (рис. 3.3). Такий підхід дозволяє отримати структуровані дані, які можна використовувати для навчання моделі семантичної дисамбігуації.

```
('u', 'n', 'x'), ('t', 'h', 'm'), ('h', 'a', 'x'), ('s', 't', 'o'), ('a', 'l', 'l'), ('o', 'v',
('t', 'h', 'x'), ('a', 'r', 'f'), ('t', 'h', 'l'), ('e', 'n', 'k'), ('s', 'o', 'f'), ('y', 'o',
```

Рисунок 3.3 Вигляд списку даних після очищення.

3.3.2 Векторизація тексту

Векторизація тексту є одним з основних етапів при розробці моделі, бо більшість моделей машинного навчання та глибокого навчання працюють із числовими даними. Векторизація тексту перетворює слова чи токени в числові представлення, що можуть бути використані для подальшого навчання моделі.

Так як текст містить велику кількість слів, але не всі з них можуть бути важливі для конкретної задачі. Векторизація дозволяє стиснути інформацію, представляючи текст у вигляді векторів з меншою кількістю вимірів. Цей підхід дозволяє створити числові представлення слів чи фраз, що можуть узагальнити семантичні відносини між ними. Це важливо для завдань, таких як класифікація текстів, де потрібно враховувати сенс слів. Саме тому коли текстові дані векторизовані, їх можна легко поєднувати з іншими числовими ознаками для створення повноцінного вхідного сигналу для моделі.

Визначившись в необхідності векторизації тексту розробимо функцію для побудови нашої моделя, яка приймає речення та словник (де кожному слову відповідає індекс) і повертає список індексів, де кожен індекс відповідає слову в реченні. Якщо слово не знаходиться в словнику, використовується індекс для невідомого слова.

Використовуючи функцію word_tokenize, яка використовується для токенізації кожного речення, і результат зберігається у списку (рис. 3.4). Після цього створюється словник, де кожному унікальному слову в тексті призначається унікальний індекс. Також створюється зворотний словник, що дозволяє здійснювати зворотнє відображення з індексів у слова.

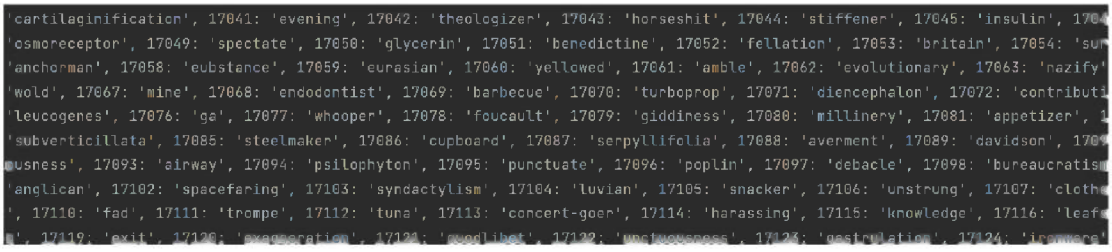

Рисунок 3.4 Список токенізованих речень і слів.

Після проробленних дій використовується готова бібліотека MultiLabelBinarizer, яка потрібна для перетворення зазначених міток у бінарну матрицю. Кожна мітка перетворюється у вектор, де кожен елемент вказує на наявність (1) чи відсутність (0) мітки у відповідному реченні.

3.3.3 Архітектурний огляд та технічна реалізація моделі

Розроблена нейронна мережа розпочинається з вхідного шару, де кожен вхід представляє собою індекс слова в словнику, створеному на основі тренувального корпусу. Ці індекси подаються на вхід ембедінг-шару, що перетворює їх у векторні представлення фіксованої розмірності.

Ембедінг-шар відповідає за навчання контекстуальних представлень слів. Векторні представлення слів, отримані в результаті цього шару, включаються у вхід трансформерного енкодера для обробки послідовностей.

Основною архітектурною складовою є трансформерний енкодер, який відповідає за моделювання внутрішньої структури послідовності. Він складається з кількох трансформерних блоків, де кожен блок містить механізм вимірювання уваги та позапослідовні підсумовування.

Вихід з трансформерного енкодера піддається глобальному середньому пулінгу, щоб отримати фіксовано-розмірний вектор. Цей вектор передається через повністю з'єднаний шар, який відповідає за остаточний класифікаційний вихід.

Така архітектура дозволяє моделі ефективно враховувати контекст інтерпретації омографів та навчати корисні репрезентації для вирішення завдань визначення омографії словникових слів.

Для технічної реалізації потрібно встановити певні параметри. Першим чином встановлюється розмір словника, який визначає кількість унікальних слів, що використовуються у завданні. Наприклад, для текстового завдання це може бути кількість унікальних слів у корпусі текстів, який використовується для тренування моделі.

Далі, параметр розміру ембедінгів визначає кількість вимірів у векторному просторі, в якому представлені слова. Вибір цього параметра залежить від обсягу даних та складності завдання, а типово використовується значення в діапазоні від 50 до 300.

Кількість вихідних класів і розмір вихідного шару залежить від задачі поставленої задачі, а саме, тому в програмі виділяють кількість унікальних класів, на які розділені текстові дані.

Для того щоб вибрати оптимальні параметри для ефективної компіляції і роботи моделі потрібно провести тестування на різних значеннях (табл. 3.1).

Таблиця 3.1

Результати експериментів з різними параметрами моделі

	LSTM = 64 Demes = 128 Batch_size = 64 Epochs = 10 Adam = 0.001	LSTM = 64 Demes = 128 Batch_size = 64 Epochs = 10 Adam = 0.001	LSTM = 32 Demes = 64 Batch_size = 64 Epochs = 10 Adam = 0.001	LSTM = 128 Demes = 256 Batch_size = 64 Epochs = 5 Adam = 0.01	LSTM = 64 Demes = 256 Batch_size = 64 Epochs = 10 Adam = 0.01	LSTM = 64 Demes = 128 Batch_size = 64 Epochs = 10 Adam = 0.0001
Accuracy	0.8532	0.8802	0.8647	0.8569	0.8690	0.8693
AUC-ROC	0.8751	0.8916	0.8407	0.8749	0.8801	0.8634

Accuracy та AUC-ROC є метриками, які використовуються для вимірювання ефективності класифікаційних моделей.

Accuracy визначає відсоток правильно класифікованих прикладів відносно загальної кількості прикладів у наборі даних. Вона є простим індикатором того, наскільки точно модель класифікує дані. Висока точність свідчить про ефективність моделі у визначенні класів.

AUC-ROC (Area Under the Receiver Operating Characteristic curve) - це метрика, яка враховує якість бінарної класифікації для різних значень порогу. Крива ROC показує відношення між чутливістю (True Positive Rate) та специфічністю (1 - False Positive Rate) при зміні порогу класифікації. AUC-ROC визначає площу під цією кривою, де висока величина вказує на високу ефективність моделі.

Проведений експеримент з різними конфігураціями моделі показав, що варіант (друга колонка) з параметрами LSTM = 64, Dense = 128, batch_size = 64, Epochs = 10, Adam = 0.001 демонструє найвищі значення обох метрик - високу точність (Accuracy = 0.8802) та значення AUC-ROC (0.8916). Це свідчить про те, що ця конфігурація моделі ефективно вирішує задачу класифікації омографів, надаючи найкращі результати у порівнянні з іншими варіантами.

Після цього огляду були обрані оптимальні параметри для поставленої задачі, а саме наведено у табл. 3.2.

Таблиця 3.2

Параметри для Ініціалізації моделі

Параметр	Значення
Розмір словнику	25000
Розмір ембедінгів	100
Кількість вихідних класів	5
Розмірність прихованого шару	64
Кількість шарів трансформера	2
Швидкість навчання	0.001

Обираючи значення параметрів для нейронної мережі, важливо було керуватися природою завдання, обсягом доступних даних та рекомендаціями з практики.

Розмірність ембедінгів була обрана саме 64, тому що менше значення призводить до втрати важливої інформації (рис. 3.5), а більше значення збільшує кількість параметрів і обчислювальні витрати.

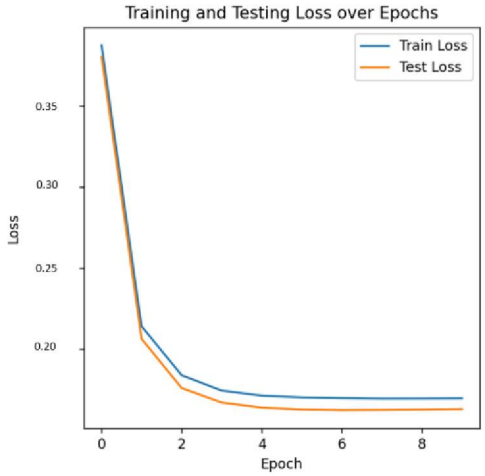

Рисунок 3.5 Графічне відображення втрати даних.

В рамках імплементації нейронної мережі було обрано оптимізатор Adam для навчання моделі. Adam (Adaptive Moment Estimation) є популярним алгоритмом оптимізації, який комбінує ідей з алгоритмів Momentum та RMSprop. Основні особливості Adam:

1. Миттєвий момент (Momentum): Використання миттєвого моменту у фреймворці оптимізації дозволяє алгоритму швидше набирати швидкість на областях функції втрат з крутими узбіччями, що сприяє більш ефективному переходу через регіони невизначеності. Цей механізм спрямований на зменшення перепадів швидкості та покращення стабільності оптимізаційного процесу, зменшуючи ймовірність перескоків алгоритму через значні варіації градієнту на великих узбіччях.
2. Коригований миттєвий момент (Bias-Corrected Momentum): Adam враховує початкову нечутливість до обчислень шляхом введення коригованого миттєвого моменту.
3. Квадрат градієнту (RMSprop): Adam використовує коригований квадрат градієнту для адаптивного визначення швидкості навчання для кожного параметра.
4. Адаптивне оновлення швидкості навчання: Adam автоматично адаптує швидкість навчання для кожного параметра на основі історії градієнтів.

Оптимізатор Adam є ефективним в вирішенні поставленної задачі для усунення омографів у текстових даних через свою здатність пристосовуватися до різних умов навчання. Його використання дозволяє швидше та стабільніше навчати модель, адаптуючись до характеристик градієнтів у різних частинах параметричного простору.

3.3.4 Процес навчання

Процес навчання представляє собою важливий етап в розробці та оптимізації нейронної мережі для вирішення завдань класифікації омографів. Цей цикл об'єднує кілька ключових етапів, які мають вирішальний вплив на ефективність моделі (рис. 3.6).

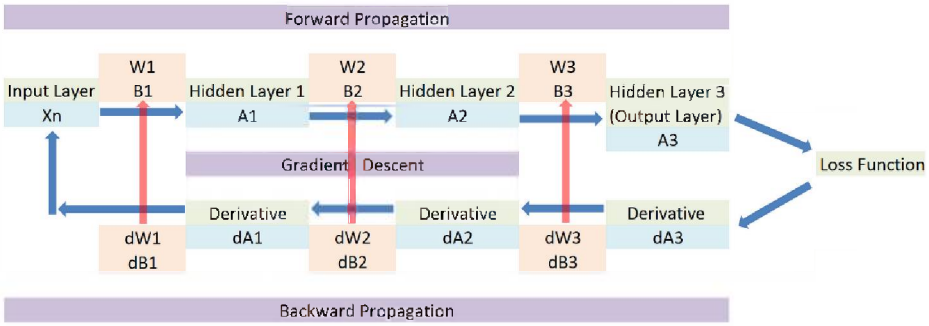

Рисунок 3.6 Цикл навчання нейронної мережі [36].

Перший етап - ініціалізація - визначає початкові параметри моделі та її компонентів. Після цього настає етап оптимізації, де використовується оптимізатор Adam для адаптації ваг моделі під конкретні дані

Після ініціалізації, настав час для етапу форвардного та зворотнього поширення. Під час форвардного поширення вхідні дані подаються в модель, де вони проходять через різні шари та отримують прогнозовані значення. Після цього відбувається обрахунок втрат та початок зворотного поширення.

Зворотнє поширення включає обчислення градієнтів втрат відносно параметрів моделі. Ці градієнти використовуються для актуалізації ваг шарів у

відповідності із заданими правилами оптимізації. Цей процес дозволяє моделі "вчитися" з входов та виправляти помилки, підлаштовуючи параметри.

Кожен ітераційний цикл включає послідовність форвардного та зворотного поширення, а також оновлення параметрів моделі. З кожною ітерацією модель адаптується до вхідних даних, намагаючись мінімізувати втрати та покращити точність передбачень. Цей процес триває до досягнення оптимальної ефективності або завершення визначеної кількості епох навчання.

3.3.5 Оцінка ефективності

Оцінка ефективності моделі для усунення неоднозначності омографів є ключовим етапом в дослідженні. Для цього використовуються різні метрики, які дозволяють кількісно оцінити якість результатів. Однією з основних метрик є точність (accuracy), яка визначає відсоток правильно класифікованих прикладів відносно загальної кількості прикладів у тестовому наборі. Вона вказує на здатність моделі правильно ідентифікувати класи. Дана метрика використовується у моїй роботі і для візуальної оцінки ефективності побудований графік точності на визначених параметрах (рис. 3.7).

Графік точності зображує зміну цієї метрики на протязі кожної епохи (циклу навчання), що дозволяє відслідковувати, як точність змінюється від епохи до епохи. Зростання точності на графіку вказує на поліпшення ефективності моделі.

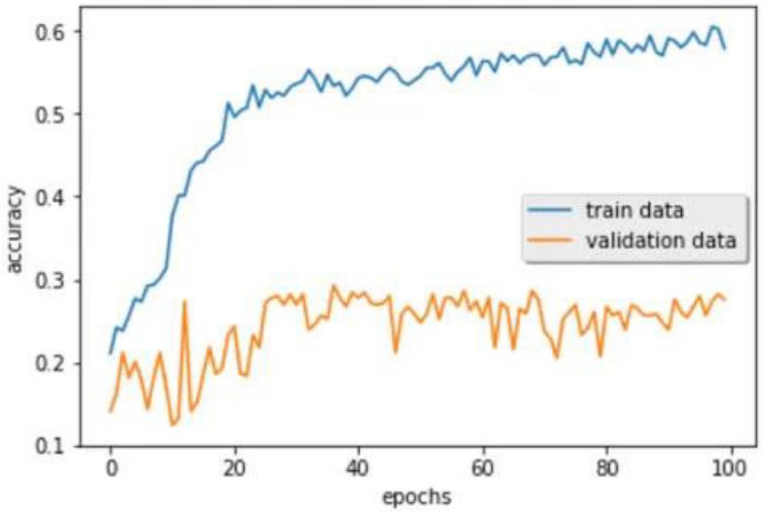

Рисунок 3.7 Графік точності

Додатково до точності були обрані такі метрики, як точність класу (precision) та F1-оцінка. Ці метрики дозволяють отримати більш детальний аналіз результатів, особливо у випадках, коли задача вирішується в рамках багатокласової класифікації.

Графік точності класу надає інформацію про те, наскільки ефективно модель визначає конкретний клас порівняно з іншими класами. Висока точність класу свідчить про те, що модель майже завжди правильно ідентифікує позитивні екземпляри цього класу, тим самим мінімізуючи кількість ложнопозитивних результатів (рис. 3.8).

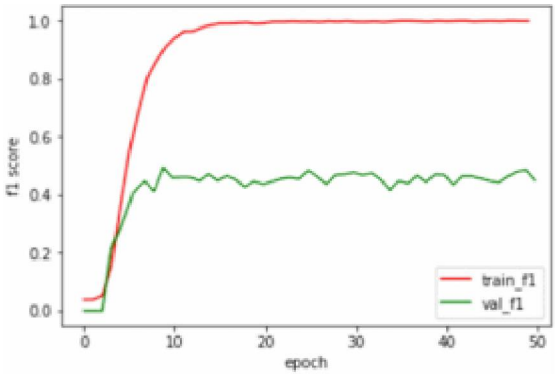

Рисунок 3.8 Графік точності класу (precision)

Графік F1-оцінки вимірює та візуалізує гармонічне середнє між точністю (precision) та чутливістю (recall). F1-оцінка об'єднує ці дві метрики в один

показник, що дозволяє оцінити якість моделі з урахуванням як точності, так і повноти. Це особливо корисно в ситуаціях, коли маємо невідсутність балансу між факторами "точність" і "чутливість" (рис. 3.9).

Цей показник використовується для оцінки якості класифікації в задачі, тому що важливо забезпечити баланс між точністю і повнотою.

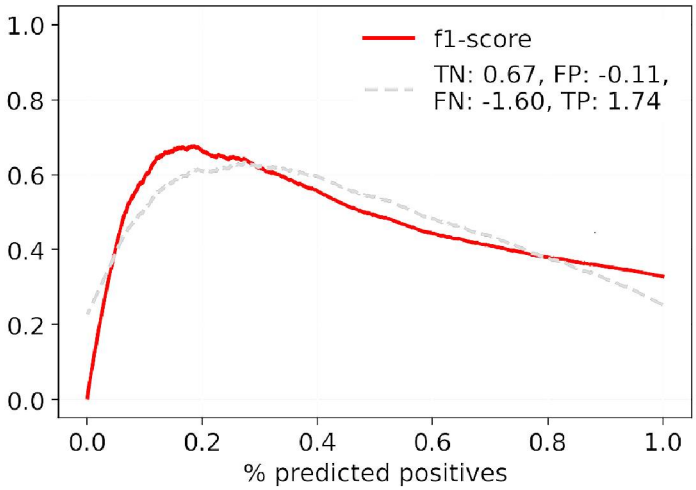

Рисунок 3.9 Графік F1

Для коректної оцінки моделі потрібно також здійснити аналіз помилок, для того щоб виявити момент, коли розроблена модель для усунення неоднозначності омографів неправильно визначила або пропустила омограф у текстових даних за контекстом. Результатом такої перевірки буде матриця плутанини (рис. 3.10), яка покаже, чи потребує модель додаткового навчання чи оптимізації.

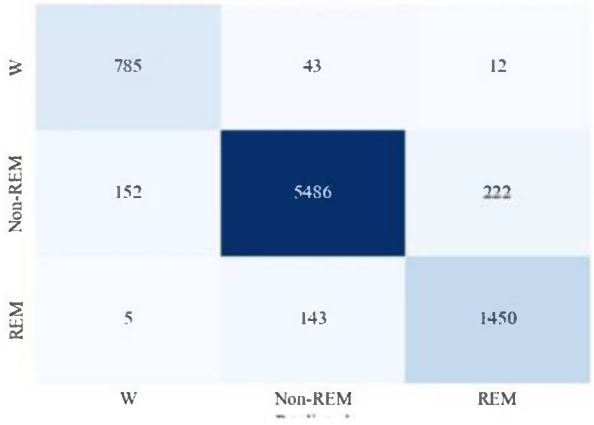

Рисунок 3.10 Матриця плутанини.

3.4 Практичне застосування моделі

Для тестування моделі на реальних даних було прийнято рішення взаємодіяти з зовнішнім сервісом, а саме telegram. Сутність даного тестування полягає в тому, щоб підключити навчену модель нейронної мережі для усунення неоднозначності омографів у текстових даних до власного телеграм боту.

Алгоритм тестування:

1. Відправляємо повідомлення до телеграм боту
2. Телеграм бот повинен передати повідомлення користувача на вхід моделі
3. Модель визначає наявність омографів у текстових даних.
    a. Омограф присутній у текстових даних – модель виправляє контекст речення і надсилає виправлене речення з прикладом перекладу на українську мову
    b. Омограф відсутній у текстових даних – модель надсилає повідомлення з контекстом, що омографів не було знайдено або контекст речення повністю описаний

Для виявлення працездатність моделі в першу чергу було надіслано речення з омографом і зможемо дослідити як поведе себе модель у такій ситуації на (рис. 3.11).

Рисунок 3.11 Тестування моделі з реченням, яке має омограф.

Для перевірки, що модель спроможна виявити речення, яке має гарно описаний контекст або відсутність омографів було відправлене речення до бота згідно задовольняючих потреб (рис. 3.12).

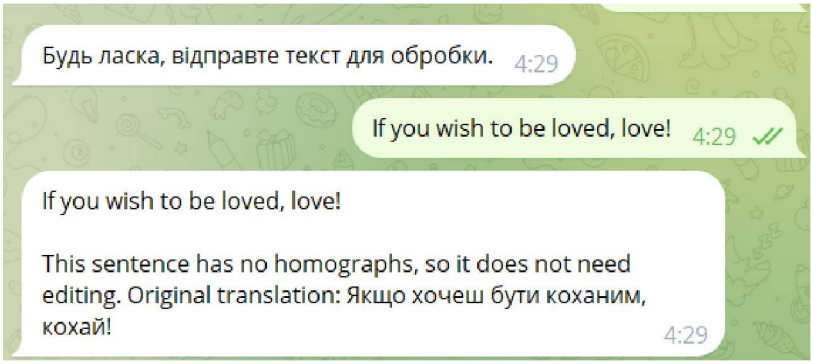

Рисунок 3.12 Тестування моделі з реченням, яке не має омографів

Висновок до 3 розділу

Під час реалізації розділу 3 було виконано реалізацію моделі нейронної мережі для усунення неоднозначності омографів. Розділ включає кілька ключових підрозділів, що представляють етапи розробки та технічні аспекти програмної реалізації.

Визначено та обґрунтовано вибір джерела даних для навчання моделі, а саме WordNet. Це є ключовим етапом, оскільки якість та репрезентативність навчального набору визначають успішність моделі.

Проведено огляд існуючих програмних засобів для розробки моделей нейронних мереж, що дало можливість визначити обрані інструменти та бібліотеки, які використовуються для програмної реалізації.

Під час розробки програмної реалізації було виконано попередню обробку та підготовку даних для навчання моделі. Описано процес векторизації тексту, який перетворює слова у вектори числових значень, зрозумілих для моделі. Надано архітектурний огляд моделі та технічні деталі її реалізації, зокрема використання вбудованих шарів та трансформерів. Описано етап навчання моделі, включаючи використання оптимізатора Adam та інші параметри. Проведено оцінку ефективності моделі за допомогою метрик, таких як точність Accuracy та AUC-ROC.

Додатково до свого завдання було зазначено практичне використання моделі через взаємодію з телеграм-ботом. Визначено алгоритм тестування та передбачено реакцію моделі на різні сценарії взаємодії з текстовими даними.

ВИСНОВКИ

В результаті виконання магістерської кваліфікаційної роботи були досягнуті наступні результати:

- Визначено постановку задачі кваліфікаційної роботи, включаючи мету, об'єкт та предмет дослідження.
- Проведено аналіз технологій і методів для усунення неоднозначності омографів у текстових даних.
- Визначено тип нейронної мережі.
- Проведено аналіз додаткових архітектурних компонентів для підвищення якості моделі.
- Розроблена нова математична модель нейронної мережі для усунення неоднозначності омографів у текстових даних.
- Розроблена графічна архітектура нейронної мережі.
- Визначено джерела даних для навчання.
- Розроблена програмна реалізація моделі.
- Проведена оцінка якості моделі.
- Інтегровано модель нейронної мережі у месенджер Telegram-бот для практичного використання моделі.

В результаті виконання магістерської кваліфікаційної роботи була підвищена якість обробки тексту, зокрема, поліпшення точності розпізнавання слів, які мають більше одного значення, шляхом розробки моделі нейронної мережі з довгою короткостроковою пам'яттю з використанням ембедінгових моделей, LSTM-шару та повнозв'язаних шарів, яка має можливість усувати неоднозначності омографів у текстових даних. Використання усіх перелічених архітектурних компонентів для вирішення поставленої задачі є новим у сфері усунення неоднозначності омографів у текстових даних. Нова модель використовує ітеративний процес тренування, що є ефективним для задачі машинного навчання і допомагає моделі підлаштовуватися під тренувальні дані для досягнення кращого результату.

Для отримання бажаного результату було проведено аналіз різноманітних методів і технологій, використовуваних для вирішення проблеми лексичної неоднозначності. Аналіз таких технологій як машинне навчання, статистичні методи, використання лінгвістичних ресурсів, підтримка експертних систем, синтаксичний та семантичний аналіз, дозволив визначити ефективну технологію, а саме машинне навчання, для вирішення поставленої задачі.

Під час дослідження також були розглянуті типи нейронних мереж. Зокрема, розглянуті нейронна мережа з довгою короткостроковою пам'яттю (LSTM), нейронні мережі прямого зв'язку, рекурентні нейронні мережі та модульні нейронні мережі. За результатом аналізу типів нейронних мереж, було визначено, що для того щоб, розробити якісну модель для усунення неоднозначності омографів у текстових даних необхідно вибрати LSTM, так як вона призначена для вирішення проблеми зникнення градієнта, що часто виникає в рекурентних нейронних мережах під час тренування на довгих послідовностях.

Головними напрямками використання розробленої моделі є сфери автоматичного розпізнавання мови, підтримки прийняття рішень на основі текстової інформації та покращення систем пошуку та класифікації даних.

Якість даної моделі усунення неоднозначності омографів у текстових даних була підвищена за рахунок використання додаткових архітектурних компонентів для моделі нейронної мережі довгої короткострокової пам'яті, а саме моделей з ембедінгами та повзнозв'язаних шарів. Ембедінгові моделі дали можливість робити представлення слів у векторній формі і таким чином дозволили нейронній мережі якісніше розуміти смислові взаємозв'язки між словами у тексті. Повнозв'язані шари тим часом дозволили моделі вивчати складні та нелінійні залежності між вхідними та вихідними даними, що дозволило ефективніше вирішувати задачу, коли зв'язки між різними частинами даних є складними та контекстуальними. Використовуючи дану модель і провівши оцінку ефективності, було виявлено, що точність розробленої моделі

становить 88%, а ROC-AUC становить 89%, що є позитивним результатом для поставленої задачі.

Для отримання бажаних результатів була розроблена програмна реалізація моделі для усунення неоднозначності омографів у текстових даних. Для навчання моделі було проведено аналіз джерел даних, а саме WordNet, SemCor, YARN. Після проведеного аналізу було виявлено, що WordNet вирізняється своєю великою кількістю семантичних відносин та структурованими омографами, що робить його підходящим для вирішення задачі неоднозначності омографів у текстових даних. Для реалізації програмного коду була обрана мова програмування python і необхідні бібліотеки для створення моделі, а саме tensorflow і Sklearn.

Розроблена програмна реалізація моделі для усунення неоднозначності омографів у текстових даних була інтегрована до Telegram-боту. Така інтеграція дозволила перевірити модель на реальних даних і дати змогу користувачу взаємодіяти з моделлю за допомогою графічного інтерфейсу.

Під час написання кваліфікаційної роботи був виступ на КМНТ, що надало можливість провести апробацію результатів дослідження.

Таким чином результатом успішно розроблена та імплементована модель, яка здатна якісно вирішувати проблему лексичної неоднозначності омографів. Модель пройшла випробування на реальних даних, зокрема шляхом інтеграції з месенджером Telegram. Отримані результати та високий рівень точності свідчать про ефективність розробленого рішення. Провівши експериментальні дослідження і довівши, що точність нейронної мережі має високі показники, то можна прийти до висновку, що модель може бути успішно використана в різних текстових сценаріях для підвищення якості розуміння контексту та ефективної обробки омографічних слів. Подальше вдосконалення та оптимізація моделі можуть розширити її можливості та застосування в ширшому спектрі завдань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація IBM за темою «What is machine learning?»: https://www.ibm.com/topics/machine-learning (дата звернення 07.02.2022)
2. Стаття «What Is a Neural Network»: https://www.investopedia.com/terms/n/neuralnetwork.asp (дата звернення 01.03.2022)
3. "Deep Learning" Ian Goodfellow, Yoshua Bengio, Aaron Courville, 2016, MIT Press
4. Hongkai W. Zongwei Z. Yingci L. Zhonghua C. Peiou L. Comparison of machine learning methods for classifying mediastinal lymph node metastasis of non-small cell lung cancer from 18F-FDG PET/CT images — 2017
5. Lin, D.; Pantel, P. (2002). Discovering word senses from text. 8th International Conference on Knowledge Discovery and Data Mining (KDD). Edmonton, Canada. pp. 613–619
6. Kilgarriff,A.:Getting to know your corpus. In: Text, Speech and Dialogue, Springer (2012)3–15
7. Baisa,V., Suchomel,V.: Large corpora for turkic languages and unsupervised morphological analysis. In: Proceedings of LREC 2012 Workshop on Language Resources and Technologies for Turkic Languages ,pp.28–32.Istanbul,Turkey,2012.
8. Стаття «7 Types of Statistical Analysis Techniques (And Process Steps)»: https://www.indeed.com/career-advice/career-development/types-of-statistical-analysis (дата звернення 24.09.2022)
9. McCrae, John P.; Labropoulou, Penny; Gracia, Jorge; Villegas, Marta; Rodríguez-Doncel, Víctor; Cimiano, Philipp (2015). "One Ontology to Bind Them All: The META-SHARE OWL Ontology for the Interoperability of Linguistic Datasets on the Web". In Gandon, Fabien; Guéret, Christophe; Villata, Serena; Breslin, John; Faron-Zucker, Catherine; Zimmermann, Antoine (eds.). The Semantic Web: ESWC 2015 Satellite Events. Lecture

Notes in Computer Science. Vol. 9341. Cham: Springer International Publishing. pp. 271–282.

10. Стаття «NLP - Linguistic Resources»: https://www.mygreatlearning.com/natural-language-processing/tutorials/nlp-linguistic-resources (дата звернення 17.12.2022)
11. Jackson, Peter (1998). Introduction To Expert Systems (3 ed.). Addison Wesley. p. 2
12. Natural Language Processing - Syntactic Analysis https://www.tutorialspoint.com/natural_language_processing/natural_language_processing_syntactic_analysis.htm (дата звернення 14.02.2023)
13. NERA 2.0: Improving coverage and performance of rule-based named entity recognition: https://www.researchgate.net/publication/302066635_NERA_20_Improving_coverage_and_performance_of_rule-based_named_entity_recognition_for_Arabic/figures (дата звернення 22.02.2023)
14. Leturiondo, U. Hybrid modelling in condition monitoring [dissertation]. Luleå, Sweden: Luleå University of Technology; 2016: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://ltu.diva-portal.org/smash/get/diva2:1034053/FULLTEXT01.pdf (дата звернення 16.03.2023)
15. Fletcher, R. (2005). "On the Barzilai–Borwein Method". In Qi, L.; Teo, K.; Yang, X. (eds.). Optimization and Control with Applications. Applied Optimization. Vol. 96. Boston: Springer. pp. 235–256.
16. Стрілець В. Є., Шматков С. І., Угрюмов М. Л. та ін. – Харків, Методи машинного навчання монографія, Харківський національний університет імені В. Н. Каразіна, 2020.
17. Error back propagation algorithm https://blog.oureducation.in/error-back-propagation-algorithm/ (дата звернення 08.06.2023)

18. A Concise History of Neural Networks: https://towardsdatascience.com/a-concise-history-of-neural-networks-2070655d3fec (дата звернення 17.07.2023)
19. Novikoff, Albert J. (1963). "On convergence proofs for perceptrons". Office of Naval Research.
20. Zell, Andreas (1994). Simulation Neuronaler Netze [Simulation of Neural Networks] (in German) (1st ed.). Addison-Wesley. p. 73
21. Miljanovic, Milos (Feb–Mar 2012). "Comparative analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction" (PDF). Indian Journal of Computer and Engineering. 3 (1): chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.ijcse.com/docs/INDJCSE12-03-01-028.pdf (дата звернення 01.08.2023)
22. Azam, Farooq (2000). "Biologically Inspired Modular Neural Networks. PhD Dissertation". Virginia Tech: https://vtechworks.lib.vt.edu/items/d9d88e1e-034a-4bc1-8743-b17bec523d32 (дата звернення 20.08.2023)
23. Sak, Hasim; Senior, Andrew; Beaufays, Francoise (2014). "Long Short-Term Memory recurrent neural network architectures for large scale acoustic modeling":https://web.archive.org/web/20180424203806/https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43905.pdf
24. Saiful Islam, Md.; Hossain, Emam (2020-10-26). "Foreign Exchange Currency Rate Prediction using a GRU-LSTM Hybrid Network": https://www.sciencedirect.com/science/article/pii/S2666222120300083?via%3Dihub (дата звернення 23.08.2023)
25. Li, Yitan; Xu, Linli (2015). Word Embedding Revisited: A New Representation Learning and Explicit Matrix Factorization Perspective: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.ijcai.org/Proceedings/15/Papers/513.pdf (дата звернення 23.08.2023)

26. Fully Connected Layers in Convolutional Neural Networks: https://indiantechwarrior.com/fully-connected-layers-in-convolutional-neural-networks/ (дата звернення 20.09.2023)
27. Документація WordNet: https://wordnet.princeton.edu (дата звернення 20.09.2023)
28. Документація Semcor: https://www.sketchengine.eu/semcor-annotated-corpus/ (дата звернення 20.09.2023)
29. Документація YARN: https://classic.yarnpkg.com/en/package/data-set
30. Офіційна документація JetBrains: https://www.jetbrains.com/help/pycharm/getting-started.html (дата звернення 13.10.2023)
31. NumPy documentation: https://numpy.org/doc/stable/ (дата звернення 13.10.2023)
32. pandas documentation: https://pandas.pydata.org/docs/ (дата звернення 13.10.2023)
33. Matplotlib 3.8.2 documentation: https://matplotlib.org/stable/index.html (дата звернення 13.10.2023)
34. SciPy documentation: https://docs.scipy.org/doc/scipy/ (дата звернення 13.10.2023)
35. Sklearn documentation: https://devdocs.io/scikit_learn/ (дата звернення 13.10.2023)
36. Build up a Neural Network with Python: https://towardsdatascience.com/build-up-a-neural-network-with-python-7faea4561b31 (дата звернення 08.11.2023)

ДОДАТКИ

Додаток А

БЛАНК ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **Магістр**
Галузь знань: **15 – Автоматизація та приладобудування**
Спеціальність: **151 – «Автоматизація та комп'ютерно-інтегровані технології»**

ЗАТВЕРДЖУЮ
Завідувач кафедри теоретичної та прикладної системотехніки
_____д.т.н., проф. Шматков С. І.

«08 » грудня 2022 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Єгоров Єгор Сергійович
(прізвище, ім'я, по батькові студента)

1. Тема роботи «Розробка моделі нейронної мережі для усунення неоднозначності омографів у текстових даних»

керівник роботи Шматков Сергій Ігорович, д.т.н., професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «10» листопада 2023 року № 4101-5/3197

2. Строк подання студентом роботи 28 .11.2023

3. Перелік питань, які потрібно розробити
1) Огляд і аналіз існуючих досліджень щодо усунення неоднозначності омографів за допомогою нейромереж.
2) Визначення архітектури нейронної мережі для розв'язання завдання усунення неоднозначності омографів.
3) Вибір оптимальних алгоритмів навчання та оптимізації моделі для даного завдання.
4) Збір та підготовка набору даних для навчання та тестування моделі.
5) Проведення експериментів для оцінки ефективності моделі на тестовому наборі даних.
6) Аналіз результатів експериментів та ідентифікація можливих шляхів покращення моделі.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Проведення літературного огляду з проблематики застосування різних моделей для обробки текстових інформації різного типу.	08.12.2022 — 06.03.2023
2	Постановка задачі розробки моделі нейронної мережі для усунення неоднозначності омографів у текстових даних, підготовка документації проекту.	17.01.2022 — 06.03.2023
3	Аналітичній огляд існуючих програмних засобів для розробки моделі нейронної мережі.	06.03.2023 — 01.04.2023
4	Розробка моделі нейоронної мережі.	01.04.2023 — 27.06.2023
5	Вибір датасету з тестовими даними.	20.06.2023 — 01.07.2023
6	Програмна реалізація моделі з використанням підібраних алгоритмів.	01.07.2023 — 25.08.2023
7	Аналіз результатів тестування та визначення ефективності використання моделі для усунення неоднозначності омографів.	25.08.2023 — 17.09.2023
8	Розробка рекомендацій щодо застосування моделі для усунення неоднозначності омографів у текстових даних.	17.08.2023 — 20.09.2023
9	Оформлення пояснювальної записки.	01.09.2023 — 27.10.2023
8	Представлення кваліфікаційної роботи керівнику та рецензенту.	28.10.2023 - 28.11.2023

5. Дата видачі завдання 08.12.2022 р.

Студент Є.С. Єгоров _____
ініціали, прізвище підпис

Керівник роботи С.І. Шматков _____
ініціали, прізвище підпис

Додаток Б

Технічне завдання
на розробку програмного вироба
«Модель нейронної мережі для усунення неоднозначності омографів у текстових даних»

Назва розділу	Назва і зміст підрозділу
1. Введення	1.1. Назва програмного виробу: Розробка моделі нейронної мережі для усунення неоднозначності омографів у текстових даних 1.2. Галузь застосування: технології обробки природної мови
2. Підстава для розробки	2.1. Навчальний план за спеціальністю 151 – «Автоматизація та комп'ютерно-інтегровані технології» 2.2. Завдання на кваліфікаційну роботу магістра затверджено наказом ректора № 4101-5/3197 від 10.10.2023.
3. Призначення розробки	3.1. Мета розробки програмного виробу: розробка нейронної мережі, спрямованої на підвищення якості обробки тексту, зокрема, поліпшення точності розпізнавання слів, які мають більше одного значення. Основний акцент кваліфікаційної роботи спрямований на вирішення проблеми неоднозначності омографів у текстових даних. 3.2. Призначення програмного виробу: використання виробу для усунення лексичної неоднозначності омографів в текстових даних 3.3. Вихідні дані для розробки .....
4. Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик: Модель повинна ефективно вирішувати завдання усунення неоднозначності омографів у текстових даних. 4.2. Вимоги до надійності: Модель повинна демонструвати стабільну роботу під час взаємодії з різними типами текстових даних. 4.3. Вимоги до умов експлуатації: Модель повинна працювати ефективно в умовах різної лінгвістичної специфіки та стилів мовлення. 4.4. Вимоги до складу і параметрів технічних засобів: Ефективність моделі повинна забезпечуватися при різних конфігураціях обчислювальних

	пристроїв. 4.5. Вимоги до інформаційної та програмної сумісності: Модель повинна бути сумісною з різними версіями програмних засобів, таких як фреймворки для роботи з нейронними мережами. 4.6. Вимоги до маркування та упаковки: Забезпечення належного опису маркування та параметрів використаних даних для тренування та тестування. 4.7. Вимоги до транспортування і зберігання: Модель та пов'язані дані повинні зберігатися у форматі, який забезпечує легкий доступ та транспортування між різними системами. 4.8. Спеціальні вимоги: Забезпечення можливості подальшого розвитку та покращення моделі з урахуванням нових даних та вимог користувачів.
5. Вимоги до програмної документації.	Програмною документацією до виробу «Модель нейронної мережі для усунення неоднозначності омографів у текстових даних» вважати: 1) Справжнє Технічне завдання на розробку програмного виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи). 2) Програму і методику випробувань розробленого програмного виробу (представити у вигляді Додатку В до пояснювальної записки до кваліфікаційної роботи). 3) Опис програмного виробу (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи).

<table>
<tr><td></td><td colspan="2">4) Текст програми (представити в Додатку Г до пояснювальної записки до кваліфікаційної роботи).</td></tr>
<tr><td>6. Техніко-економічні показники</td><td colspan="2">Орієнтовна оцінка ефективності та якості виконуваної кластеризації даних: точність (accuracy) повинна бути вище 88%, метрика ROC-AUC повинна бути вище 0.89;</td></tr>
<tr><td>7. Стадії і етапи розробки</td><td>Дата</td><td>Назва етапу</td></tr>
<tr><td></td><td>15.04.23-15.05.23</td><td>Дослідження та аналіз існуючих технологій, методів та систем усунення омографів у текстових даних</td></tr>
<tr><td></td><td>16.06.23-16.07.23</td><td>Аналіз засобів створення моделі для вирішення задачі лексичної неоднозначності омографів в текстових даних</td></tr>
<tr><td></td><td>17.07.23-17.09.23</td><td>Розробка моделі для усунення неоднозначності омографів у текстових даних</td></tr>
<tr><td></td><td>17.09.23-30.10.23</td><td>Тестування програмної моделі</td></tr>
<tr><td></td><td>01.11.23-15.11.23</td><td>Розробка рекомендацій щодо застосування програмної моделі</td></tr>
<tr><td></td><td>15.11.23-30.11.23</td><td>Оформлення пояснювальної записки.</td></tr>
</table>

8. Порядок контролю і приймання	В даному розділі повинні бути вказані загальні вимоги до приймання розробленого програмного виробу наприклад: 1) Перевірка ходу розробки програмного виробу. Керівнику робіт виконувати 1 раз в 3 тижні. 2) Випробування програмного виробу відповідно до Програми і методики випробувань провести на базі комп'ютерного класу. 3) Захист розробленого програмного виробу провести на засіданні атестаційної комісії. 4) Пояснювальну записку надати на паперових носіях в одному примірнику, в електронному вигляді - на CD-диску в одному екземплярі.

Виконавець
Студент групи КУ-61

Єгоров Є.С.

_____

Замовник
д.т.н. професор кафедри ТПС

Шматков С.І.

_____

Додаток В

Програма і методика випробувань

програмного виробу

«Модель нейронної мережі для усунення неоднозначності омографів у текстових даних»

1. Об'єкт випробувань

Об'єктом випробовування є програмний застосунок для усунення неоднозначності омографів у текстових даних (розділ 1 ТЗ).

2. Мета випробувань

Перевірка працездатності програмного застосунку для усунення неоднозначності омографів у текстових даних. (Додаток Б).

3. Загальні положення

3.1 Підстави для проведення випробувань

Підставою для проведення випробувань є наказ № _ від _____ про призначення атестаційної комісії та перевірку даного виробу.

3.2 Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу або будь-якого приміщення в період роботи атестаційної комісії.

3.3 Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї програми і методики випробувань.

3.4 Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу

Програмний виріб повинен надавати такі можливості:

1) Натренувати нову модель.
2) Оцінити ефективність роботи моделі за допомогою метрик acurancy та AUC-ROC

Метрика точності AUC на тестовому наборі даних повинна мати значення не менше ніж 0.88.

Для експлуатації програми потрібно, щоб на ПК було встановлено python і необхідні бібліотеки.

Програмний застосунок повинен підтримувати роботу на наступних операційних системах: Windows, Linux, MacOS.

5. Вимоги до програмної документації

Склад програмної документації, що подається на випробування, включає:

1) Технічне завдання на розробку програмного виробу (представлено в Додатку Б до пояснювальної записки до кваліфікаційної роботи).

2) Ця програма і методика випробувань розробленого програмного виробу (представлена в Додатку В до пояснювальної записки до кваліфікаційної роботи).

3) Опис програмного виробу (представлено в розділі 3 пояснювальної записки до кваліфікаційної роботи).

4) Фрагмент програми (представлений в Додатку Г до пояснювальної записки до кваліфікаційної роботи)

6. Засоби і порядок випробувань

6.1. Засоби випробувань

Для виконання програми потрібно, щоб на ПК був встановлений python не нижче версії 3.7.0.

6.2. Порядок проведення випробувань

6.2.1. Перевірка програмної документації

1) Перевірка складу програмної документації. Перевірку здійснювати за критерієм наявності, представленої в ТЗ документації.

2) Перевірка якості програмної документації. Перевірку здійснювати за критерієм відповідності вимогам ЄСПД. «Програма і методика випробувань».

6.2.2. Перевірка працездатності методу

Тест 1

1. Перевірку працездатності здійснюємо шляхом запуску програми. Отримуємо результат запуска у консольному додатку розробленої програмної реалізації.

Рис. В.1 – Консольний запуск програми

Тест 2

2. Перевірку здійснюємо шляхом запуску програми та введенням речення, яке у своєму контексті має неоднозначність омографів. Отримуємо результат роботи програмної реалізації в консольному додатку.

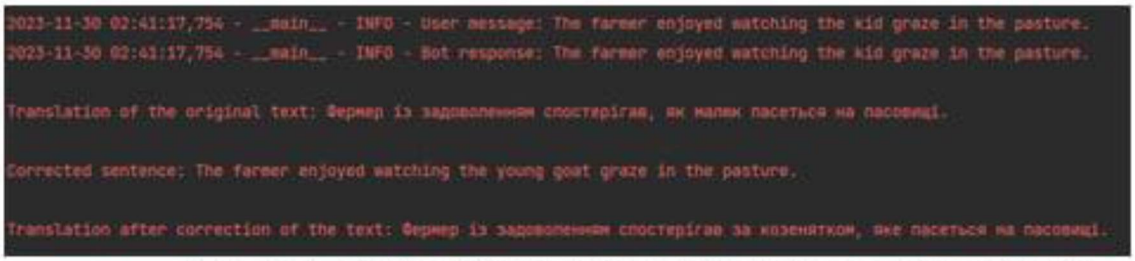

Рис. В.2 – Результат висновку роботи програмної реалізації

Тест 3

3. Перевірку здійснюємо шляхом запуску програми та введенням речення, яке не має у контексті речення неоднозначність омографів. Отримуємо результат роботи програмної реалізації в консольному додатку.

Рис. В.3 – Результат висновку роботи програмної реалізації

Висновок: якщо були пройдені всі тестування у пунктах 1-3, результати випробування вважати успішними.

Виконав
студент групи КУ-61 Єгоров Є.С.

Додаток Г

Фрагменти програмного коду

Рис. Г.1 Фрагмент програмного коду підготовки даних

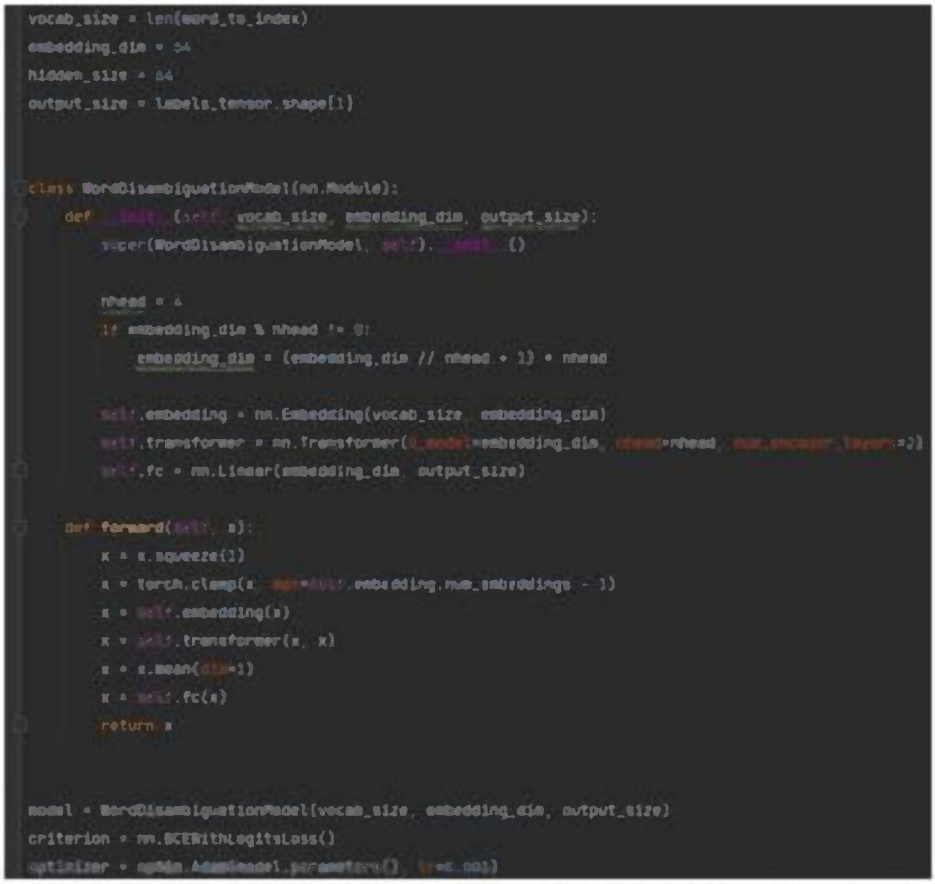

Рис. Г.2 Фрагмент програмного коду ініціалізації моделі

```
num_epochs = 10

for epoch in range(num_epochs):
    model.train()
    for batch_sentences, batch_labels in train_loader:
        optimizer.zero_grad()

        input_data = pad_sequence(
            [torch.tensor(sentence_to_indices(sentence, word_to_index), dtype=torch.long) for sentence in
             batch_sentences],
            batch_first=True)

        output = model(input_data)
        loss = criterion(output, batch_labels)
        loss.backward()
        optimizer.step()
```

Рис. Г.3 Фрагмент програмного коду навчання моделі

```
model.eval()
correct = 0
total = 0
test_loss = 0

with torch.no_grad():
    for batch_sentences, batch_labels in test_loader:
        input_data = pad_sequence(
            [torch.tensor(sentence_to_indices(sentence, word_to_index), dtype=torch.long) for sentence in
             batch_sentences],
            batch_first=True)
        output = model(input_data)

        probabilities = torch.nn.functional.softmax(output, dim=1)
        _, predicted = torch.max(probabilities, 1)

        total += batch_labels.size(0)
        correct += (predicted == torch.argmax(batch_labels, dim=1)).sum().item()

        loss = criterion(output, batch_labels)
        test_loss += loss.item()

    accuracy = correct / total
    average_test_loss = test_loss / len(test_loader)
    print(f'Test Accuracy: {accuracy}, Test Loss: {average_test_loss:.4f}')
```

Рис. Г.4 Фрагмент програмного коду оцінки моделі