Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та кібербезпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

ВЕКТОРИЗАЦІЯ ТЕКСТІВ НА ОСНОВІ НЕЙРОМЕРЕЖЕВОЇ АРХІТЕКТУРИ

TEXT VECTORIZATION BASED ON NEURAL NETWORK ARCHITECTURE

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІсз-21
Боровик Олександр Васильович

(підпис)

Керівник:
д.пед.н., професор
Чернящук Наталія Леонідівна

(підпис)

Кваліфікаційну роботу
допущено до захисту
« 11 » червня 2024 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2024 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та кібербезпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ проф. Н.Чернящук
« 10 » 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Боровику Олександру Васильовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Векторизація текстів на основі нейромережевої архітектури*

Керівник роботи *д.пед.н., професор Чернящук Наталія Леонідівна*

затверджені наказом закладу вищої освіти від «30» грудня 2023 року № 459/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 11.06.2024р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області та різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Аналітичний огляд в області досліджень
Методологія моделювання української мови
Розробка нейромережевої мовної моделі
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
Нейронна мережа
Згорткові нейронні мережі
Рекурентна нейронна мережа
Алгоритм WORD2VEC

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблеми за темою роботи та постановка завдань дослідження	*Чернящук Н.Л., професор*		
Теоретичне дослідження та практична реалізація	*Чернящук Н.Л., професор*		
Практична реалізація об'єкта проектування	*Чернящук Н.Л., професор*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Лавренчук С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І.,асистент*		

7. Дата видачі завдання 10.01.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Розділ 1. Огляд літератури із досліджуваної проблеми*	до 15.02.2024 р.	Виконано
2.	*Розділ 2. Методологія моделювання української мови*	до 15.03.2024 р.	Виконано
3.	*Розділ 3.Розробка нейромережевої мовної моделі*	до 04.05.2024 р.	Виконано
4.	*Висновки та пропозиції*	до 07.05.2025 р.	Виконано
5.	*Формування списку використаних джерел*	до 10.05.2024 р.	Виконано
6.	*Формування додатків*	до 15.05.2024 р.	Виконано
7.	*Оформлення ілюстративного матеріалу*	до 20.05.2024 р.	Виконано
8.	*Нормоконтроль*	до 01.06.2024 р.	Виконано
9.	*Інструментальна перевірка на академічний плагіат*	до 04.06.2024 р	Виконано
10.	*Представлення кваліфікаційної роботи бакалавра до захисту*	до 11.06.2024 р.	Виконано

Здобувач вищої освіти _____ (підпис) Боровик О.В. (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) Чернящук Н.Л. (прізвище, ініціали)

АНОТАЦІЯ

Боровик О.В. Векторизація текстів на основі нейромережевої архітектури. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2024.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел, трьох додатків.

Проведено детальний аналіз публікацій на тему моделювання природна мова й українська зокрема, інформація про алгоритми та архітектури нейронних мереж, які дозволяють це зробити. Вимоги до процес навчання нейронних мереж – від отримання даних до обробки оптимізація обраної моделі, а також інструменти, що вирішують ці вимоги задовільняти. Для цього пропонується використовувати архітектуру GPT-2 Українське мовне моделювання. Розроблено програмний код для проведення експериментів нейронні мережі в області моделювання природної мови, а також генерування на їх основі текстів.

Модель нейронного мовлення була навчена, що показало кращі результати, ніж аналогічні публікації, які використовували аналогічні обчислення потужності під час навчання, що свідчить про перспективність використання. Архітектура трансформатора для завдань обробки та генерації природної мови.

Ключові слова: мовна модель, нейронна мережа, штучний інтелект, генерація тексту, обробка природної мови.

ANNOTATION

Borovyk O.V. Text vectorization based on neural network architecture. Manuscript.

Bachelor's qualification thesis of the OP «Computer Engineering» specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2024.

The qualification work consists of an introduction, three sections, conclusions, a list of used sources, and three appendices.

A detailed analysis of publications on the topic of modeling natural language and Ukrainian in particular, information on algorithms and architectures of neural networks that allow this to be done. Requirements for the learning process of neural networks – from data acquisition to processing and optimization of the selected model, as well as tools that decide to satisfy these requirements. For this purpose, it is proposed to use the GPT-2 Ukrainian language modeling architecture. A program code was developed for conducting neural network experiments in the field of natural language modeling, as well as generating texts based on them.

A neural speech model was trained that performed better than similar publications that used similar power calculations during training, indicating promising use. A transformer architecture for natural language processing and generation tasks.

Keywords: language model, neural network, artificial intelligence, text generation, natural language processing.

ЗМІСТ

ВСТУП

Актуальність теми роботи. Щоденно генерується величезна кількість текстових даних в Інтернеті (соціальні мережі, блоги, новинні ресурси). Для ефективної обробки та аналізу цих даних необхідні потужні методи векторизації. Сучасні нейронні мережі, такі як трансформери (наприклад, BERT, GPT), демонструють високу ефективність у задачах векторизації текстів, перевершуючи традиційні методи, такі як bag-of-words або TF-IDF. Векторизація текстів на основі нейромереж використовується в багатьох сферах. Пошукові системи для покращення релевантності результатів. Системи рекомендацій, що аналізують текстові відгуки та вподобання користувачів. Системи автоматичного перекладу та обробки мовлення.

Нейромережеві методи забезпечують високу точність у розумінні контексту та семантики тексту, що важливо для завдань, таких як класифікація текстів, видобування інформації, розпізнавання емоцій та настроїв. Постійно розробляються нові архітектури нейронних мереж та алгоритми, що покращують якість векторизації текстів. Це стимулює наукові дослідження та впровадження інновацій у різні прикладні проекти. Розвиток відкритих бібліотек (наприклад, TensorFlow, PyTorch) і платформ (Google Colab) спрощує процес розробки та впровадження нейромережевих моделей для векторизації текстів, що робить цю тему доступною для широкого кола дослідників і розробників.

Таким чином, векторизація текстів на основі нейромережевої архітектури є актуальною темою через її значний внесок у розвиток NLP та її вплив на різні прикладні сфери.

Метою даної роботи є дослідження методів векторизації текстів на основі нейромережевих архітектур та їх застосування в різних задачах обробки природної мови.

Завдання дослідження:

– аналіз існуючих методів векторизації текстів, включаючи традиційні та сучасні нейромережеві підходи;

– дослідження архітектур нейронних мереж, що використовуються для векторизації текстів (LSTM, CNN, трансформери);

– розробка та реалізація моделей векторизації текстів на основі обраних нейромережевих архітектур;

– порівняння ефективності різних методів векторизації на основі експериментальних даних;

– вивчення можливостей адаптації та застосування розроблених моделей для конкретних задач NLP.

Об’єктом дослідження є процес обробки природної мови (NLP), зокрема, методи векторизації текстів, що використовуються для перетворення текстових даних у числові представлення.

Предметом дослідження є конкретні нейромережеві архітектури та алгоритми, що застосовуються для векторизації текстів.

Практичне значення полягає в розробці і впровадженні ефективних методів векторизації текстів на основі нейромережевих архітектур, що забезпечують покращення якості NLP-застосувань, автоматизацію аналітичних задач, оптимізацію роботи з текстовими даними, розвиток нових методів і моделей, а також внесок у наукову спільноту. Ці досягнення можуть мати значний вплив на різні галузі, включаючи інформаційні технології, бізнес-аналітику, маркетинг, освіту та інші.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ВЕКТОРИЗАЦІЇ ТЕКСТІВ НА ОСНОВІ РЕЙРОМЕРЕЖЕВОЇ АРХІТЕКТУРИ

1.1 Аналіз природної мови

Моделювання мови – це область науки про обробку природної мови, що вивчає методи та алгоритми для розуміння, генерації та праці з мовою з точки зору комп’ютера. Основна мета моделювання мови полягає в створенні систем, які можуть розуміти та генерувати природну мову так само, як це робляють люди.

Розуміння мови. Це включає в себе розпізнавання тексту або мовного сигналу, що подані в комп’ютерній формі. Методи розпізнавання можуть використовувати різні підходи, такі як правила, статистику, машинне навчання або глибоке навчання.

Генерація мови. Цей аспект полягає в створенні тексту або мовного висловлювання на основі вхідних даних або шаблонів. Генерація може бути виконана для різних цілей, таких як створення тексту, відповідей на запитання, створення діалогів тощо.

Машинний переклад. Це процес автоматичного перетворення тексту з однієї мови на іншу. Методи машинного перекладу можуть використовувати правила, статистику, нейромережі або їх комбінації.

Аналіз настроїв та сентименту. Це вивчення та визначення емоційного забарвлення тексту. Може використовуватися для аналізу відгуків, соціальних медіа, аналізу настроїв громадськості тощо.

Генерація мовних моделей. Це створення моделей, які можуть передбачати наступне слово або послідовність слів у тексті на основі попереднього контексту. Генерація мовних моделей використовується в багатьох задачах обробки мови.

Розпізнавання та синтез мови. Це визначення мовних одиниць та їх перетворення в або з аудіоформату. Використовується у програмах розпізнавання мови, а також у системах голосового помічника.

Моделі мовної динаміки. Вони вивчають структуру та розвиток мови, включаючи зміни у вживанні слова, граматиці, фонетиці тощо. Моделі мови зазвичай використовують методи машинного навчання, такі як нейромережі, статистичні методи та правила, для досягнення високої точності в завданнях обробки мови.

Класичні методи моделювання мови включають в себе різноманітні підходи, які не використовують глибоке навчання або нейромережі. Ці методи можуть бути ефективними для певних задач, особливо якщо недоступні достатні об'єми даних або обмежені обчислювальні ресурси. Ось кілька класичних методів моделювання мови.

Модель марківських ланцюгів (Markov Chain Model). Цей підхід передбачає ймовірність того, що наступне слово або символ у послідовності залежить від попереднього обмеженого контексту (наприклад, останніх кількох слів). Модель Марківських ланцюгів застосовується для генерації тексту, розпізнавання мови та інших задач.

Метод байєсівських мереж (Bayesian Networks). Це графічна модель, яка використовує ймовірності та правила байєсівської статистики для моделювання зв'язків між мовними одиницями. Вони можуть використовуватися для аналізу тексту, класифікації тексту, визначення тематики тощо.

Статистичні моделі. Ці моделі використовують статистичні методи для моделювання мови, такі як n-грами (послідовності з n слів) або функції відстані між словами. Статистичні моделі часто використовуються в машинному перекладі, автозавершенні тексту, визначенні семантичної схожості тощо.

Методи кластеризації та категоризації. Ці методи використовуються для групування схожих текстових документів за їх змістом або темою. Вони можуть використовувати алгоритми, такі як k-means, ієрархічна кластеризація тощо.

Правила та експертні системи. Деякі задачі моделювання мови можуть бути вирішені за допомогою правил або експертних систем, які використовують знання про мову та граматику. Це може бути корисно для

задач, таких як генерація тексту за шаблонами або визначення граматично правильних конструкцій.

Ці класичні методи досі використовуються в багатьох системах обробки мови, особливо там, де обмежений обсяг даних або коли важлива ефективність в обчисленнях. Однак з розвитком глибокого навчання і нейромережі стають домінуючими у багатьох задачах моделювання мови завдяки їх здатності до автоматичного вивчення складних залежностей у даних.

Нейронні мережі (Neural Networks) – це клас моделей машинного навчання, які моделюють роботу людського мозку і використовуються для розв'язання різноманітних завдань, включаючи моделювання мови. Основними перевагами нейронних мереж є їх здатність до автоматичного вивчення складних залежностей у вхідних даних та адаптації до нових сценаріїв.

Деякі типи нейронних мереж, які часто використовуються для завдань моделювання мови, включають. Рекурентні нейронні мережі (RNN). Ці мережі призначені для роботи з послідовними даними, такими як тексти або мовні висловлювання. Вони мають здатність запам'ятовувати попередні стани, що дозволяє їм ефективно моделювати залежності в часі в текстах.

Лонг-трм пам'ять (LSTM) і мережі відновлення від блоку (GRU). Ці варіанти рекурентних нейронних мереж мають додаткові механізми для керування забуванням та зберіганням інформації в пам'яті на довготривалий період. Вони особливо ефективні для роботи з великими корпусами текстів та моделюванням складних лінгвістичних залежностей.

Згорткові нейронні мережі (CNN). Ці мережі зазвичай використовуються для обробки зображень, але їх можна також успішно застосовувати для аналізу текстів, зокрема для завдань класифікації або розпізнавання послідовностей.

Трансформери (Transformer). Це недавно розроблена архітектура нейронних мереж, яка показала вражаючі результати в багатьох завданнях обробки мови. Вона базується на механізмах уваги і покращує швидкість та якість обробки текстів.

Крім цього, для досягнення кращої ефективності в моделюванні мови, нейронні мережі часто комбінуються з іншими методами, такими як векторна

репрезентація слів (word embeddings), аугментація даних та методи оптимізації, щоб покращити якість та швидкість роботи моделі.

Рекурентні нейронні мережі (RNN) та seq2seq (sequence-to-sequence) є ключовими компонентами багатьох моделей обробки природної мови. Давайте розглянемо кожну з них окремо та їхню взаємодію.

RNN – це тип нейронних мереж, призначений для роботи з послідовними даними, такими як тексти, часові ряди або сигнали.

Головна особливість RNN полягає в тому, що вона має зв’язки з попередніми станами, тобто може «запам’ятовувати» попередні вхідні дані та використовувати їх для обробки поточного вводу.

Однак у звичайних RNN може виникати проблема зі зниклим градієнтом, коли інформація про віддалені від моменту часу вхідні дані зникає, що призводить до погіршення якості прогнозів.

Seq2Seq – це архітектура нейронних мереж, яка складається з двох рекурентних нейронних мереж. енкодера та декодера.

Енкодер приймає послідовність вхідних даних і перетворює її в складний вектор фіксованої довжини, який кодує усю інформацію про вхідну послідовність.

Декодер отримує вектор-кодування вхідної послідовності та починає генерувати вихідну послідовність один елемент за одним.

Seq2Seq широко використовується в машинному перекладі, генерації тексту, чат–ботах та інших завданнях генерації послідовностей.

Взаємодія між RNN та Seq2Seq полягає в тому, що RNN часто використовується як базова модель для як енкодера, так і декодера в архітектурі Seq2Seq. Енкодер перетворює вхідну послідовність за допомогою RNN у фіксований вектор, який потім передається до декодера для генерації вихідної послідовності. Така комбінація дозволяє ефективно вирішувати завдання моделювання мови та генерації тексту зображено на рисунку 1.1.

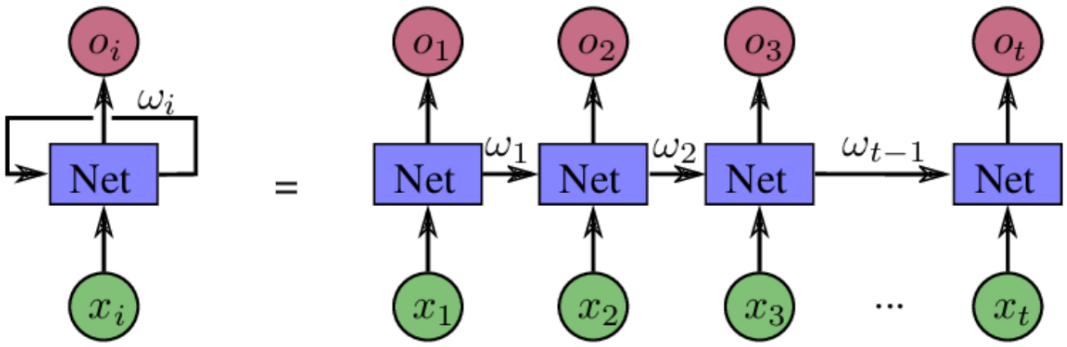

Рисунок 1.1 – Нейронна мережа [1]

Рекурентні нейронні мережі (RNN) є типом глибоких нейронних мереж, які призначені для роботи з послідовностями даних, такими як текст, звук або часові ряди. Ці мережі можуть ефективно моделювати залежності в часі, оскільки вони мають зв’язки, що повертаються, які дозволяють використовувати інформацію з попередніх кроків в обробці поточного вхідного сигналу. Один з основних варіантів застосування рекурентних нейронних мереж – це моделювання послідовних перетворень, таких як переклад тексту або генерація тексту. Для таких завдань часто використовується архітектура seq2seq (sequence-to-sequence), яка базується на RNN зображено на рисунку 1.2.

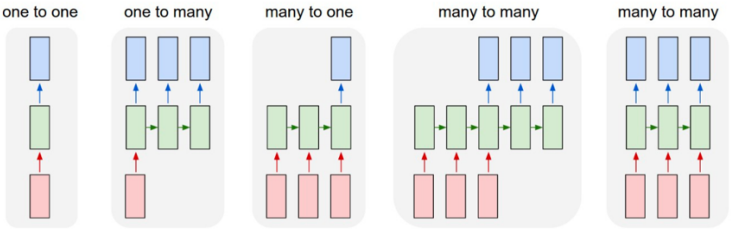

Рисунок 1.2 – Фреймворк [1]

Архітектура seq2seq складається з двох основних частин. енкодера і декодера. Енкодер приймає на вхід послідовність даних (наприклад, речення тексту) і перетворює його в контекстний вектор, який представляє собою закодовану інформацію про вхідну послідовність. Цей вектор потім передається декодеру. Декодер, у свою чергу, використовує контекстний вектор, отриманий

від енкодера, для генерації вихідної послідовності (наприклад, перекладу або продовження тексту) зображено на рисунку 1.3.

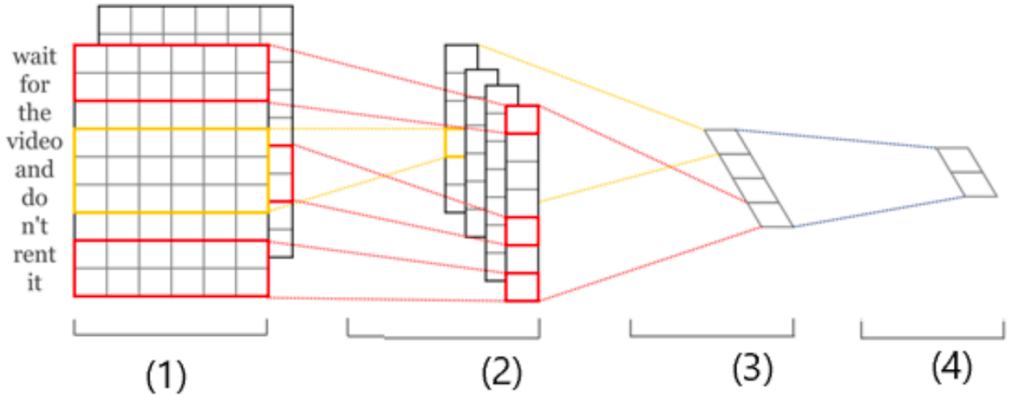

Рисунок 1.3 – Згорткові нейронні мережі [1]

Основна перевага seq2seq полягає в здатності працювати з послідовними даними різної довжини вхідного та вихідного рядів. Такий підхід широко використовується в машинному перекладі, генерації тексту, синтезі мови та інших задачах обробки послідовностей. Механізм уваги є ключовою складовою архітектури трансформерних моделей, таких як GPT (Generative Pre–trained Transformer) і BERT (Bidirectional Encoder Representations from Transformers). Ці моделі стали домінуючими в обробці природної мови завдяки їхній здатності ефективно моделювати контекст та залежності між словами в тексті. Механізм уваги дозволяє моделі зосереджувати увагу на різних частинах вхідного сигналу, приділяючи більше уваги важливим елементам і менше – менш важливим. Це досягається за допомогою механізму операції, який називається «Self-Attention» (самоувага), де кожен вхідний елемент взаємодіє з усіма іншими елементами, а ваги цих взаємодій визначаються в процесі навчання. В результаті модель може «фокусуватися» на різних частинах вхідного сигналу залежно від завдання. Трансформерні моделі, в яких застосовується механізм уваги, зазвичай складаються з кількох шарів самоуваги, які дозволяють моделі аналізувати текст на різних рівнях абстракції. Це робить їх ефективними для широкого спектру завдань, таких як машинний переклад, розпізнавання мови та

аналіз тексту. Узагальнюючи, механізм уваги в трансформерних моделях є ключовим елементом, що дозволяє їм досягати високих результатів у завданнях обробки природної мови за рахунок ефективного моделювання контексту та залежностей між словами в тексті зображено на рисунку 1.4.

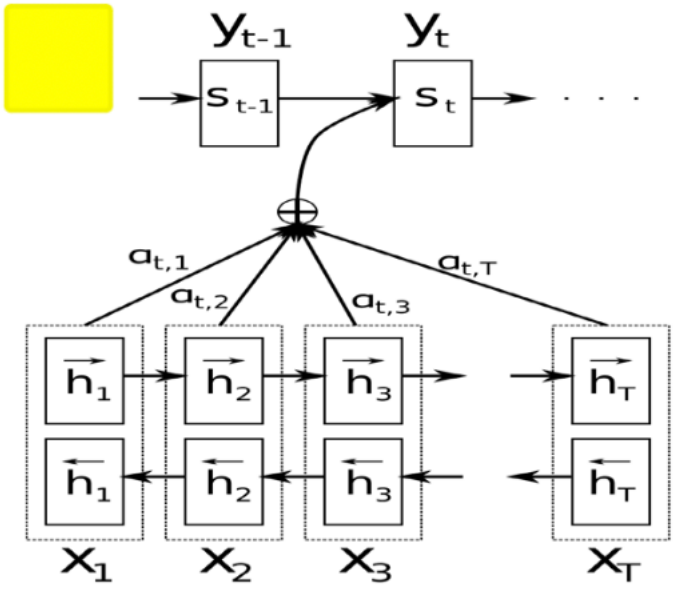

Рисунок 1.4 – Рекурентна нейронна мережа [2]

Механізм уваги в нейромережах, зокрема у моделі «Трансформер», відіграє ключову роль у забезпеченні ефективного опрацювання великих обсягів даних, зокрема в природній мові, зображеннях або інших типах вхідної інформації. Модель «Трансформер», яка вперше була представлена в роботі «Attention is All You Need», відмінною особливістю має саме механізм уваги. Механізм уваги в «Трансформерах» дозволяє моделі віддавати приоритет окремим частинам вхідного сигналу, при цьому забезпечуючи зв’язок між різними елементами послідовності. Замість того, щоб кодувати вхідну послідовність в фіксований вектор фіксованої довжини, як це робиться, наприклад, у звичайних рекурентних або згорткових нейронних мережах, увага дозволяє «Трансформерам» приділяти різні рівні уваги різним частинам вхідної послідовності. Це робить модель більш гнучкою та здатною до опрацювання послідовностей різної довжини. У моделі «Трансформер» механізм уваги використовується на двох рівнях. механізм уваги всередині кожного блоку

уваги (self–attention) та механізм уваги між різними блоками уваги, які зазвичай об’єднуються у стек. Це дозволяє моделі ефективно опрацьовувати інформацію з різних частин вхідної послідовності та взаємодіяти з нею, що робить «Трансформери» дуже потужними для завдань обробки мовлення, машинного перекладу, генерації тексту та багатьох інших завдань, пов’язаних з послідовнісною моделлю даних.

1.2 Теоретичний аналіз моделей державної мови

На сьогоднішній день розвиток нейромережевих моделей для української мови зростає. Важливо зазначити, що багато з розроблених моделей базуються на існуючих архітектурах, таких як BERT, GPT, або Transformer, і адаптовані для української мови. Ось кілька прикладів застосування нейромереж для української мови.

Машинний переклад. Нейромережеві моделі, такі як Seq2Seq або Transformer, використовуються для машинного перекладу української мови. Ці моделі навчаються перекладати текст з однієї мови на іншу, що дозволяє забезпечити автоматичний переклад між українською та іншими мовами.

Моделі для обробки тексту. Нейромережеві моделі, такі як BERT, використовуються для різних завдань обробки тексту на українській мові, таких як класифікація текстів, виявлення спаму, сентимент-аналіз та інші.

Генерація тексту. Моделі, які базуються на GPT (Generative Pre-trained Transformer), використовуються для генерації тексу українською мовою. Ці моделі можуть бути навчені на великих корпусах тексту для генерації нових речень, статей або навіть поезії.

Розпізнавання мовлення. Нейромережеві моделі можуть використовуватися для розпізнавання мовлення на українській мові. Вони можуть бути навчені розпізнавати індивідуальні слова або фрази у великих аудіо– або відеофайлах.

Інтенсивний розвиток нейромережевих моделей для української мови свідчить про зростаючий інтерес до застосування штучного інтелекту для

обробки української мови в різних галузях, від перекладу до аналізу тексту та мовлення.

Аналіз нейромережевих моделей для української мови є важливим напрямком досліджень у галузі обробки природної мови (Natural Language Processing, NLP). Тут представлено кілька ключових аспектів аналізу таких моделей.

Машинний переклад. Дослідження у сфері машинного перекладу української мови за допомогою нейромережевих моделей є активним напрямком. Моделі «Трансформер» та їхні варіанти виявилися ефективними у завданнях машинного перекладу, де українська мова є або джерелом, або мовою призначення.

Аналіз тексту. Нейромережеві моделі використовуються для аналізу тексту на українській мові, такі як класифікація тексту, виявлення сутностей, аналіз настроїв тощо. Деякі з цих моделей можуть бути адаптовані з мов, які мають більш розвинені корпуси даних. Генерація тексту. Нейромережеві моделі також використовуються для генерації тексту на українській мові. Це може бути використано для створення автоматичних систем генерації контенту, а також для створення літературних творів, новинних статей тощо.

Розпізнавання мови. Нейромережеві моделі використовуються для розпізнавання мови на українській мові. Це може включати розпізнавання мовлення, а також розпізнавання мовленнєвих команд для систем штучного інтелекту, таких як голосові помічники. Розвідувальні завдання. Українська мова також використовується у завданнях розвідки та аналізу текстів для виявлення паттернів, зв’язків та трендів у текстах, що може бути важливим для різних сфер, включаючи політику, економіку, соціальні науки тощо. Усі ці аспекти досліджуються та розвиваються в контексті української мови з використанням нейромережевих технологій, що відкриває широкі можливості для автоматизації обробки природної мови в українськомовному середовищі.

1.3 Постановка завдань векторизації текстів на основі нейромережевої архітектури

Науково-технічна задача полягає у розробці та аналізі нейромережевих моделей для обробки природної мови (Natural Language Processing, NLP) на українській мові. Це може включати такі завдання, як машинний переклад, розпізнавання та синтез мовлення, класифікація текстів, генерація текстів, витягування інформації, аналіз тональності тощо. Формулювання такої задачі може виглядати наступним чином. «Розробити та оцінити ефективні нейромережеві моделі для обробки природної мови на українській мові. Завдання включає розробку моделей для машинного перекладу, класифікації текстів, генерації текстів та інших завдань NLP, які специфічні для української мови. Дослідження має включати порівняння різних архітектур нейромереж, методів попередньої обробки даних та підходів до навчання моделей з метою вибору оптимальних стратегій для досягнення найкращих результатів у вищезазначених завданнях» Формулювання науково-технічної задачі для аналізу нейромережевих моделей української мови може бути таким. «Розробити та оцінити нейромережеві моделі для обробки природної мови на українській мові з метою досягнення кращих результатів у завданнях, таких як машинний переклад, розпізнавання мовлення, аналіз тональності та інші. Задача передбачає використання сучасних методів глибокого навчання, зокрема заснованих на архітектурі ’Трансформер’, та оптимізацію параметрів моделей для досягнення максимальної точності та швидкодії на відповідних наборах даних. Оцінка моделей передбачає порівняння їх з існуючими підходами, які можуть бути базовими лінгвістичними моделями або іншими методами, що використовуються для обробки української мови. Оцінка моделей також включає аналіз їхньої ефективності в різних сценаріях використання, а також оцінку їхньої здатності до адаптації до нових завдань та умов». Ця задача може служити основою для дослідження та розробки нових нейромережевих моделей для української мови, а також для оцінки їхнього потенціалу та порівняння з існуючими підходами.

РОЗДІЛ 2
МЕТОДИ ВЕКТОРИЗАЦІЇ ТЕКСТІВ НА ОСНОВІ НЕЙРОМЕРЕЖЕВОЇ АРХІТЕКТУРИ

2.1 Оцінка нейромережевих моделей для обробки природної мови

Для розробки та оцінки нейромережевих моделей для обробки природної мови на українській мові потрібен відповідний датасет. Датасет є ключовим елементом у навчанні та оцінці моделей машинного навчання. Ось деякі характеристики, які можуть мати значення для створення такого датасету. Мовна різноманітність. Датасет повинен містити різноманітні тексти на українській мові, щоб моделі могли навчатися різним аспектам мови. Різноманітність жанрів. Тексти можуть включати статті з новин, літературні тексти, соціальні медіа, наукові публікації тощо, щоб моделі мали змогу навчатися розпізнавати та генерувати тексти різних жанрів. Масштаб. Датасет повинен бути достатньо великим, щоб моделі могли навчитися репрезентативним особливостям української мови. Мітки (якщо потрібно). Якщо задача вимагає міток (наприклад, у випадку класифікації текстів за темами), датасет повинен містити відповідні мітки для навчання та оцінки моделей. Збалансованість (якщо потрібно). Якщо задача вимагає рівномірного представлення різних класів (наприклад, у випадку класифікації), датасет повинен бути збалансованим за цим параметром. Якість даних. Тексти повинні бути чистими, без помилок та інших відхилень, щоб уникнути негативного впливу на навчання моделей. Датасет можна скласти з відкритих джерел, таких як українські веб–сайти, літературні тексти, соціальні медіа, корпуси мовлення тощо. При цьому важливо дотримуватися авторських прав та забезпечувати конфіденційність особистої інформації, якщо така міститься у датасеті. Також можна розглянути можливість використання вже існуючих корпусів або датасетів для української мови, які вже існують у відкритому доступі. Розбиття даних на частини для навчання, валідації та тестування є важливим етапом в процесі розробки нейромережевих моделей. Зазвичай дані розбиваються на тренувальний, валідаційний та тестовий набори з метою ефективного навчання,

відбору гіперпараметрів та оцінки результатів моделі. Ось кілька загальних рекомендацій щодо розбиття даних. Тренувальний набір (Train set). Використовується для навчання моделі. Зазвичай складає від 60% до 80% від усіх даних. Валідаційний набір (Validation set). Використовується для відбору гіперпараметрів моделі та оцінки її продуктивності під час навчання. Зазвичай складає від 10% до 20% від усіх даних. Тестовий набір (Test set). Використовується для остаточної оцінки продуктивності моделі після навчання та налаштування. Не використовується під час навчання моделі. Зазвичай складає від 10% до 20% від усіх даних. Стратифікація (Stratification). При розбитті даних можна забезпечити, щоб різні класи або категорії були представлені у кожному з наборів. Це допомагає уникнути випадкових перекосів у розподілі класів між наборами. Випадкове розбиття (Random splitting). Дані можна розбивати на набори випадковим чином, щоб забезпечити репрезентативність кожного набору. Хресна перевірка (Cross-validation). Замість фіксованих тренувальних та валідаційних наборів можна використовувати метод хресної перевірки, щоб отримати більш об’єктивну оцінку продуктивності моделі. Наприклад, якщо маєте датасет для розробки моделі обробки природної мови на українській мові, ви можете розбити його на тренувальний, валідаційний та тестовий набори у співвідношенні близько 70% – 15% – 15%. При цьому важливо врахувати різноманіття даних у кожному наборі, щоб вони були репрезентативними для ваших цілей. Препроцесинг або попередня обробка даних – це важливий етап у підготовці даних перед їхнім використанням для навчання моделі машинного навчання. У випадку обробки текстових даних на українській мові, препроцесинг може включати наступні кроки. Токенізація. Розбиття текстів на окремі слова або токени. В українській мові це може включати розрізнення слів за пробілами або іншими роздільниками, а також врахування особливостей мови, таких як складність деяких словосполучень чи відсутність пробілів у деяких текстах.

Перетворення в нижній регістр. Перетворення всіх символів у тексті на нижній регістр, щоб уникнути дублювання токенів через різні варіанти написання. Вилучення стоп-слів. Вилучення часто зустрічаються слів, які

мають мало семантичного навантаження, таких як сполучники, прийменники, займенники тощо. Лематизація або стемінг. Зменшення слова до його базової форми (леми або стеми). Це допомагає уніфікувати слова з однаковими коренями. Вилучення символів пунктуації та спеціальних символів. Це може включати вилучення ком, крапок, знаків питання, символів нового рядка та інших непотрібних символів. Токенізація чисел та адрес електронної пошти. У випадку текстів, що містять числа або електронні адреси, їх можна замінити спеціальними токенами. Векторизація тексту. Перетворення тексту на числову форму, яку можна використовувати для подальшої обробки моделлю машинного навчання. Це може включати використання методів векторизації, таких як Bag of Words, TF-DF або Word Embeddings. Ці кроки можуть варіюватися в залежності від конкретних потреб вашого проекту та властивостей даних. Після проведення препроцесингу дані будуть готові до використання для навчання моделей машинного навчання. На рисунку 2.1 показано схему роботи цих підходів.

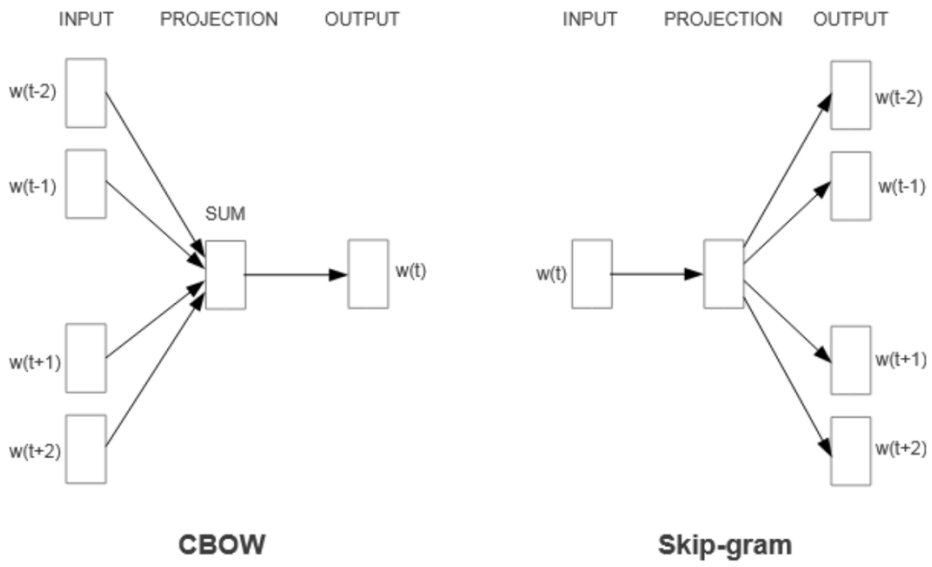

Рисунок 2.1 – Алгоритм WORD2VEC [6]

Препроцесинг даних є важливим етапом у розробці нейромережевих моделей для обробки природної мови. Він включає в себе ряд операцій, спрямованих на підготовку та очищення даних перед їхнім подальшим

використанням у моделі. Ось деякі типові операції препроцесингу даних для української мови. Токенізація. Розбиття тексту на окремі слова або токени. Для української мови можна використовувати вбудовані токенізатори або створити власний на основі правил та регулярних виразів. Перетворення регістру. Стандартизація регістру тексту (наприклад, переведення всіх літер у нижній або верхній регістр), щоб уникнути дублювання слів через відмінювання або неврахування слів через великі літери. Вилучення знаків пунктуації та спеціальних символів. Видалення зайвих символів, які не несуть значення для аналізу тексту. Вилучення стопслів. Вилучення часто зустрічаються слів, які не несуть значення (наприклад, сполучники, частки, прийменники). Лематизація та стемінг. Зменшення слова до її базової форми. Українська мова має складну морфологію, тому лематизація може бути особливо корисною. Токенізація на рівні символів. Розбиття тексту на окремі символи, що може бути корисним у деяких випадках, таких як генерація тексту. Обробка невідомих слів. Обробка слів, які відсутні у словнику, наприклад, шляхом заміни їх на спеціальні токени (наприклад, <UNK>). Падіння даних. Заповнення або видалення порожніх значень, якщо вони присутні в даних. Нормалізація числових значень. Нормалізація числових значень, якщо вони присутні у даних, для забезпечення однакового масштабу. Ці операції можуть варіюватися в залежності від конкретної задачі та особливостей даних. Важливо експериментувати з різними методами препроцесингу та обирати ті, які найбільше підходять для вашого конкретного випадку за допомогою емпіричних спостережень та оцінки результатів моделі.

2.2 Моделі обробки природної мови

Побудова моделі для обробки природної мови на українській мові включає кілька кроків, які можуть варіюватися в залежності від конкретної задачі та обраної архітектури моделі. Ось загальний огляд кроків, які можна виконати при побудові моделі. Вибір архітектури моделі. Виберіть підходящу архітектуру моделі для вашої задачі. Наприклад, можна використовувати

модель «Трансформер», рекурентні нейронні мережі (RNN), згорткові нейронні мережі (CNN) або комбінації цих архітектур. Побудова моделі. Створіть модель, використовуючи бібліотеку для глибокого навчання, таку як TensorFlow або PyTorch. Визначте архітектурні елементи моделі, такі як шари, функції активації, оптимізатори та інші. Компіляція моделі. Визначте функцію втрат та метрики для оцінки продуктивності моделі під час тренування. Оберіть оптимізатор для навчання моделі та визначте параметри оптимізатора. Навчання моделі. Завантажте дані та тренуйте модель на тренувальному наборі. Виконуйте ітерації навчання, підганяючи модель до тренувальних даних за допомогою оптимізатора та міні-пакетів даних. Оцінка моделі. Оцініть продуктивність моделі на валідаційному наборі, використовуючи задані метрики. Використовуйте ці результати для відбору гіперпараметрів та налаштування моделі.

Тестування моделі. Оцініть остаточну продуктивність моделі на тестовому наборі, щоб оцінити її здатність до узагальнення на нових даних.

Настройка моделі. Виконайте налаштування моделі, включаючи зміну архітектури, гіперпараметрів та інших параметрів, для покращення її продуктивності.

Використання моделі. Після вдосконалення моделі та перевірки її продуктивності, ви можете використовувати її для різних завдань, таких як машинний переклад, розпізнавання мовлення, генерація тексту тощо. Це загальний підхід до побудови моделі для обробки природної мови на українській мові. Конкретні кроки та методи можуть відрізнятися в залежності від конкретної задачі та вимог дослідження.

Методологія моделювання української мови та побудова моделі ґрунтуються на різноманітних підходах і техніках, які використовуються в області обробки природної мови (Natural Language Processing, NLP) та машинного навчання (Machine Learning). Нижче наведено загальну архітектуру та кроки побудови такої моделі.

Збір корпусу текстів українською мовою з різних джерел (веб-сторінки, книги, статті тощо). Очищення текстів від зайвої інформації (HTML-теги,

спеціальні символи, стоп-слова тощо). Токенізація текстів на окремі слова та фрази. Побудова векторних представлень. Використання методів векторної репрезентації слів (наприклад, Word2Vec, GloVe, FastText) для перетворення слів у вектори фіксованої довжини. Можливе використання попередньо навчених моделей на великих корпусах текстів. Побудова моделі. Використання різних архітектур нейронних мереж, таких як рекурентні нейронні мережі (RNN), згорткові нейронні мережі (CNN), трансформери тощо. Адаптація архітектур до специфіки української мови, яка може включати різні рівні аналізу, такі як морфологічний, синтаксичний, семантичний. Навчання моделі на зібраних та підготовлених даних. Оцінка та налаштування моделі. Використання метрик, що відображають якість моделі (наприклад, точність, відтворюваність, F1-мера). Налаштування параметрів моделі для покращення її результатів.

Використання перехресної перевірки для оцінки загальної роботи моделі на нових даних. Впровадження та застосування. Інтеграція моделі в різноманітні застосування, такі як системи автоматичного аналізу текстів, машинний переклад, генерація тексту тощо. Постійне вдосконалення та підтримка моделі з урахуванням нових даних та потреб користувачів, що зображено на рисунку 2.2-2.4.

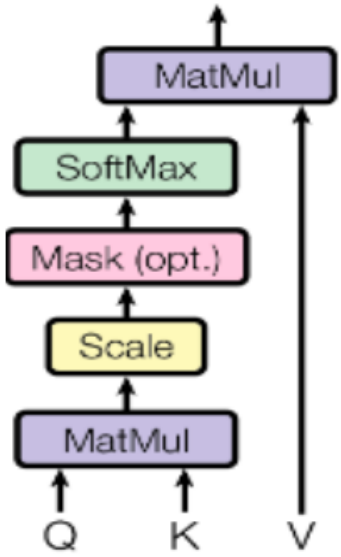

Рисунок 2.2 – Scaled Dot-Product Attention [7]

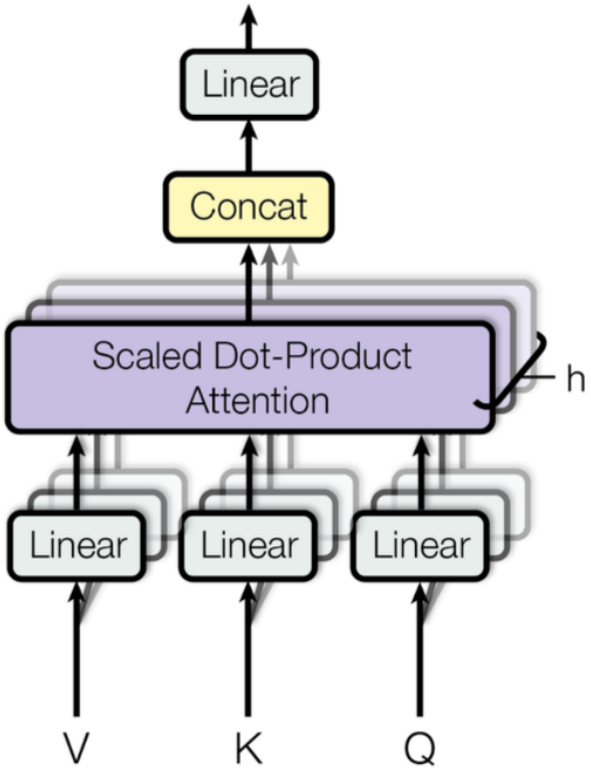

Рисунок 2.3 – Multi-Head Attention [7]

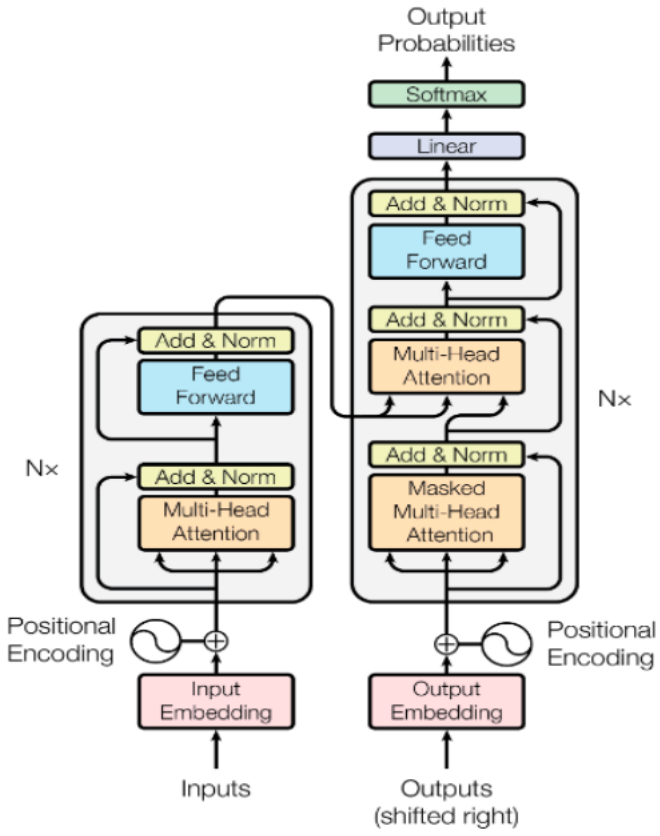

Рисунок 2.4 – Трансформер [7]

Для вивчення нейронної моделі української мови було обрано підхід, описаний у GPT-2 [4], оскільки він був застосований до схожого обсягу тренувальних даних та показав дуже хороші результати для багатьох мов. Функція витрат та оптимізація – це ключові концепції в багатьох галузях, включаючи економіку, фінанси, оптимізацію процесів та аналіз даних. В контексті будь–якої системи, функція витрат – це математична функція, яка оцінює витрати або втрати, пов’язані з різними рішеннями або варіантами дій. Витрати можуть бути розглянуті з різних точок зору, таких як грошові витрати, витрати часу, ресурси тощо. Зазвичай функція витрат визначається як функція від змінних параметрів, які впливають на витрати. Наприклад, у виробництві функція витрат може бути залежною від кількості виробленої продукції, витрат на сировину, праці, обладнання тощо. Оптимізація, з іншого боку, полягає в тому, щоб знайти найкращий варіант серед можливих альтернатив, який мінімізує або максимізує значення функції витрат або іншої цільової функції. Найпоширеніші методи оптимізації включають. Методи градієнтного спуску. Використовуються для знаходження мінімуму або максимуму функції, використовуючи ітеративний процес пошуку в напрямку антиградієнта. Методи лінійного програмування. Використовуються для оптимізації лінійної функції в обмеженому просторі. Методи метаевристики. Включають в себе алгоритми, які шукають рішення, досліджуючи простір можливих рішень без гарантії знаходження оптимального рішення, наприклад, генетичні алгоритми, ройовий імітаційний алгоритм тощо. Методи динамічного програмування. Використовуються для розв’язання задач оптимізації шляхом розбиття їх на менші підзадачі та ефективного розв’язання цих підзадач. Методи оптимізації без обмежень. Використовуються для оптимізації функцій без обмежень, таких як алгоритм Нелдера-Міда. Застосування цих методів оптимізації залежить від конкретного контексту проблеми, доступних ресурсів та обмежень.

У контексті моделювання української мови або будь-яких інших моделей обробки природної мови (NLP), метрики використовуються для оцінки якості моделі та її результатів. Ось кілька типових метрик, які можуть бути використані. Точність (Accuracy). Це відношення кількості правильно

класифікованих прикладів до загальної кількості прикладів. Використовується для задач класифікації. Точність позитивних (Precision). Це відношення кількості правильно класифікованих позитивних елементів до загальної кількості класифікованих позитивних елементів. Використовується для задач класифікації, де важлива точність у визначенні позитивних елементів. Повнота (Recall). Це відношення кількості правильно класифікованих позитивних елементів до загальної кількості позитивних елементів у наборі даних. Використовується для задач класифікації, де важлива повнота у визначенні всіх позитивних елементів.

F1-мера (F1-score). Це гармонічне середнє між точністю і повнотою. Вона дає збалансовану оцінку моделі, враховуючи як точність, так і повноту. Матриця плутанини (Confusion Matrix). Це таблиця, яка показує кількість правильних і неправильних прогнозів, зроблених моделлю, для кожного класу. ROC-крива та площа під ROC-кривою (ROC-AUC). Використовується для оцінки моделей класифікації у випадках, коли є баланс між чутливістю та специфічністю. Втрати моделі (Model Loss). Функціонал, який визначає, наскільки далеко прогнози моделі відрізняються від фактичних значень у навчальному наборі. BLEU (Bilingual Evaluation Understudy). Метрика для оцінки якості машинного перекладу, яка порівнює згенерований текст з референсними текстами. Perplexity. Метрика для оцінки якості моделей мови, що визначає ступінь здивування моделі на нових даних. Ці метрики є лише декількома з багатьох, які можуть бути використані для оцінки моделей NLP. Вибір конкретних метрик залежить від задачі, типу даних та контексту застосування моделі. Для нормалізації результатів, отримане значення ділять на кількість токенів, що дозволяє порівнювати моделі, які оперують послідовностями різної довжини. Існують докази того, що нижче значення перплексії не обов’язково означає природніший текст – особливо в таких мовах, як японська [7], однак наразі немає простого автоматичного способу, який краще корелює і іншими показниками натуральності друкованої мови, таких як швидкість руху очних яблук при читанні, тому ми використовуємо саме цю метрику для оцінки отриманої мовної моделі для української.

РОЗДІЛ 3
ВЕКТОРИЗАЦІЯ ТЕКСТІВ НА ОСНОВІ НЕЙРОМЕРЕЖЕВОЇ АРХІТЕКТУРИ

3.1 Засоби векторизації текстів нейромережевої архітектури

Для розробки нейромережевої мовної моделі ви можете використовувати різні бібліотеки та інструменти, доступні в мові програмування Python. Найпопулярніші з них включають. TensorFlow. TensorFlow – це відкрита бібліотека машинного навчання, розроблена Google, яка надає засоби для створення та навчання різноманітних типів нейронних мереж, включаючи мовні моделі. TensorFlow має широкий спектр інструментів для навчання, оцінки та впровадження моделей. PyTorch. PyTorch – це інша популярна відкрита бібліотека для машинного навчання та глибокого навчання. Вона має динамічний обчислювальний граф, що полегшує розробку моделей, та широкий спектр функцій для обробки тексту та мови. Keras. Keras – це високорівневий інтерфейс для роботи з нейронними мережами, який дозволяє легко створювати, навчати та оцінювати моделі. Він може використовуватися як надбудова над TensorFlow, так і надбудова над Theano.

Scikit-learn. scikit-learn – це бібліотека машинного навчання для Python, яка містить в собі різноманітні алгоритми класифікації, регресії та кластеризації. Вона також містить інструменти для підготовки та обробки даних, що можуть бути корисні при побудові нейромережевих моделей. NLTK (Natural Language Toolkit). NLTK – це популярна бібліотека для обробки природної мови в Python. Вона містить в собі інструменти для токенізації, стемінгу, лематизації та інші функції, які можуть бути корисними при підготовці даних для нейромережевих моделей. Ці інструменти і бібліотеки надають широкий спектр функціональності для розробки нейромережевих мовних моделей в Python. Вибір конкретного інструменту може залежати від ваших потреб, рівня зручності використання та попереднього досвіду роботи з ними.

Конкретно, бібліотека PyTorch є потужним інструментом для розробки нейромережевих моделей в Python. Ось деякі ключові особливості та функціональні можливості PyTorch. Динамічний обчислювальний граф. PyTorch використовує динамічний обчислювальний граф, що робить роботу з ним більш гнучкою порівняно з TensorFlow. Це означає, що ви можете змінювати граф обчислень під час виконання програми. Надійна та швидка реалізація. PyTorch надає простий та інтуїтивно зрозумілий інтерфейс для реалізації нейромережевих моделей. Вона також пропонує оптимізовані алгоритми для швидкого навчання моделей на GPU. Підтримка автоматичного диференціювання. PyTorch автоматично обчислює градієнти функції відносно параметрів моделі за допомогою автоматичного диференціювання. Це дозволяє легко реалізувати алгоритми зворотного розповсюдження помилки та інші оптимізаційні методи. Модульність. PyTorch побудована на принципі модульності, що дозволяє швидко збирати та модифікувати складні нейромережеві архітектури. Зручність використання. PyTorch має простий та інтуїтивно зрозумілий API, що спрощує процес розробки та налагодження моделей. Підтримка перенесення на GPU. PyTorch має гнучку систему для роботи з GPU, що дозволяє прискорити навчання моделей за допомогою обчислень на графічних прискорювачах. Загалом, PyTorch є потужним інструментом для розробки нейромережевих мовних моделей в Python, і він широко використовується в галузі наукових досліджень та промисловості. Hugging Face – це компанія та відкрита спільнота, що спеціалізується на розробці та розповсюдженні інструментів для роботи з обробкою природної мови (NLP) на базі штучного інтелекту (AI). Одним з їх ключових продуктів є бібліотека та платформа Transformers, яка надає доступ до готових до використання моделей для різних завдань у сфері NLP, таких як машинний переклад, розпізнавання іменованих сутностей, аналіз настроїв тощо. Ось деякі ключові аспекти Hugging Face та їх платформи.

Transformers бібліотека. Це бібліотека, що містить найновіші та найкращі моделі глибокого навчання для обробки природної мови, такі як BERT, GPT, RoBERTa та багато інших. Вона надає легкий доступ до цих моделей та

забезпечує зручний інтерфейс для їх використання. Hugging Face Hub. Це онлайн-платформа, де можна знаходити, ділитися та завантажувати моделі, токенайзери та інші ресурси для обробки природної мови. Вона робить доступ до новітніх моделей та ресурсів простим та зручним. Комунітарний аспект. Hugging Face активно підтримує відкритий та спільнотний підхід до розробки та використання моделей NLP. Вони створили активну спільноту розробників та дослідників, яка активно ділиться знаннями, моделями та ресурсами. Зручний API. Платформа Hugging Face надає зручний API, що дозволяє легко використовувати їхні моделі та інші ресурси у власних проектах. Розширені можливості. Вони надають інструменти для настройки та доналаштування моделей, а також підтримують різні завдання в сфері NLP, включаючи машинний переклад, аналіз настроїв, генерацію тексту тощо. Узагальнюючи, Hugging Face є важливим гравцем у світі обробки природної мови, який пропонує широкий спектр інструментів та ресурсів для розробки та використання моделей NLP. Jupyter та Google Colaboratory (Colab) – це обидва популярні інтерактивні середовища для виконання коду Python (і не тільки), особливо в галузі науки про дані, машинного навчання та обробки природної мови. Ось короткий огляд кожного з них.

Jupyter Notebook – це інтерактивне середовище програмування, яке дозволяє створювати та редагувати документи, що містять живий код, текст та графіку. Він дозволяє виконувати код по блоках (комірках), що дозволяє поетапно виконувати та аналізувати результати. Jupyter підтримує багато мов програмування, але основними є Python, R та Julia. Він широко використовується в наукових дослідженнях, аналізі даних, машинному навчанні та інших областях. Google Colab – це безкоштовне інтерактивне середовище для написання та запуску коду Python у хмарі.

Він базується на Jupyter Notebook і має всі його можливості, але з додатковою можливістю використання обчислювальних ресурсів Google, таких як GPU та TPU, що дозволяє прискорити обчислення. Colab дозволяє легко зберігати, завантажувати та ділитися ноутбуками через Google Drive. Він також підтримує встановлення пакетів та залежностей, використовуючи pip або apt-

get, що робить його ідеальним для виконання складних аналізів даних та моделювання машинного навчання. Обидва середовища мають свої переваги і застосування в залежності від конкретних потреб користувача. Jupyter надає широкі можливості для роботи з кодом та документацією, тоді як Google Colab забезпечує додаткові обчислювальні ресурси та легкість спільного використання ноутбуків. DVC, або Data Version Control, є інструментом для керування версіями даних у проектах машинного навчання. Він дозволяє ефективно керувати, відстежувати та організовувати дані, які використовуються в машинному навчанні, а також зберігати їх у віддалених сховищах, таких як Git, Amazon S3, Google Cloud Storage та інші. Версіонування даних. DVC дозволяє створювати версії та відстежувати зміни в даних, що використовуються в машинному навчанні. Це дозволяє зберігати історію змін, переглядати попередні версії та відновлювати дані до попередніх станів.

Керування великими об’ємами даних. DVC дозволяє ефективно керувати великими обсягами даних, які зазвичай використовуються в машинному навчанні, шляхом оптимізації зберігання та передачі даних. Інтеграція з Git. DVC інтегрується з системами керування версіями, такими як Git, що дозволяє зберігати історію змін даних та моделей разом з кодом проекту. Підтримка віддалених сховищ. DVC дозволяє зберігати дані у віддалених сховищах, таких як Amazon S3, Google Cloud Storage, Microsoft Azure, що полегшує спільну роботу та забезпечує надійність зберігання.

Інтеграція з іншими інструментами. DVC може інтегруватися з різними іншими інструментами машинного навчання, такими як TensorFlow, PyTorch, Scikit–learn тощо.

Узагальнюючи, DVC є потужним інструментом для керування версіями даних у проектах машинного навчання, що дозволяє підтримувати порядок та ефективно використовувати дані у розробці та експериментах з моделями.

3.2 Навчання нейромережі

При тренуванні нейронної мережі важливо правильно організувати структуру проекту, щоб забезпечити чистоту, зручність та ефективність розробки. Ось типова структура проекту для тренування нейронних мереж.

Каталог даних (data). raw. Вихідні нередаговані дані, які отримані з джерела або стороннього постачальника. processed. Оброблені дані, які підготовлені для тренування. Це може включати токенізовані тексти, зображення у форматі, придатному для подальшого аналізу, аудіофайли у форматі, придатному для обробки, тощо.

Каталог моделей (models). Файли з описом архітектур моделей, таких як класи або функції, що створюють модель.

weights. Збережені ваги моделей після тренування.

checkpoints. Збережені контрольні точки моделі під час тренування.

Каталог навчання (train).

train.py. Сценарій, який використовується для тренування моделі. Включає код для завантаження даних, визначення моделі, налаштування оптимізатора, визначення критерію втрат та навчання моделі.

utils.py. Допоміжні функції та утиліти для тренування, такі як функції для завантаження даних, підготовки пакетів даних, підрахунку метрик тощо.

Каталог оцінки (evaluation).

evaluate.py. Сценарій для оцінки якості навченої моделі. Включає код для тестування моделі на тестовому наборі даних та оцінки її продуктивності.

Каталог конфігурації (config).

config.yaml. Файл конфігурації, який містить налаштування тренування та оцінки моделі, такі як гіперпараметри, шляхи до даних, шляхи до збереження моделей тощо.

Каталог звітів (reports).

logs. Файли журналу, які містять інформацію про процес тренування та оцінки, таку як втрати, метрики точності, швидкість навчання тощо.

figures. Зображення, що показують графіки метрик, збережені під час тренування.

Інші файли.

README.md. Опис проекту та інструкції щодо використання.

requirements.txt. Файл, що містить перелік залежностей проекту, необхідних для його виконання.

Ця структура проекту забезпечує організоване та зрозуміле розміщення коду, даних та інших компонентів проекту, що робить розробку, тренування та оцінку нейронних мереж більш зручними та ефективними, що зображнгона рисунках 3.1-3.3.

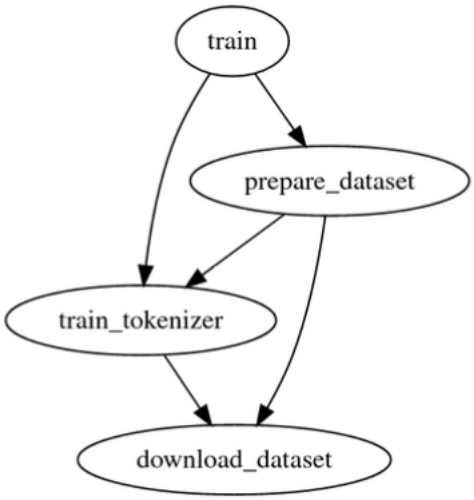

Рисунок 3.1 – Навчання граф

Самі етапи в DVC оперують файлами, запускають команди які їх опрацьовують та продукують файли-результати. Граф залежностей між файлами подано на рисунку 3.2.

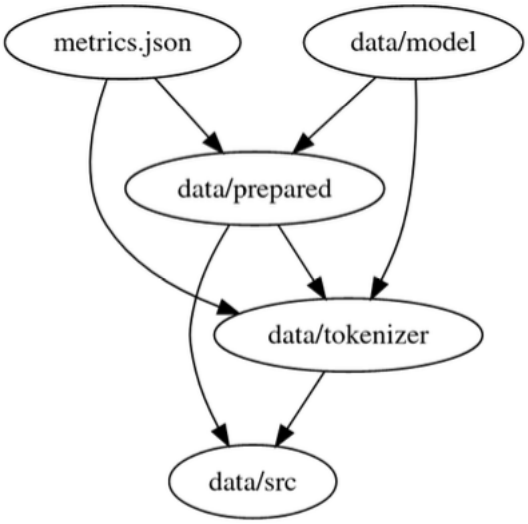

Рисунок 3.2 – Залежності граф

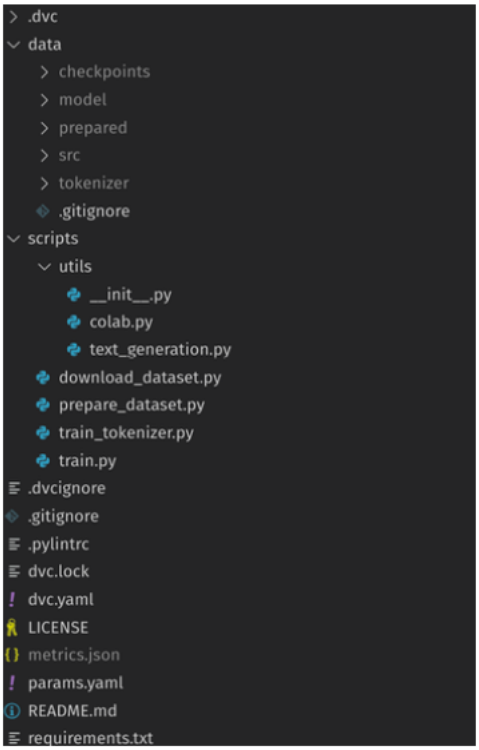

Рисунок 3.3 – Структура файлу

Сценарії, які використовуються в проекті, знаходяться в папці scripts/ і Дані (оригінальні та перетворені) зберігаються в папці data/. У свою чергу, в scripts/ є підпапкою utils/ для додаткових утиліт, таких як створення тексту. IN Корінь проекту містить файли конфігурації та ліцензії. Отримання даних для тренування нейронної мережі може бути виконане різними способами, залежно від конкретної задачі та типу даних, які вам потрібні. Ось кілька загальних

методів отримання даних. Завантаження даних з відкритих джерел. Використовуйте API відкритих даних для отримання доступу до даних у відкритому форматі, які надаються організаціями, такими як уряди, дослідницькі установи та інші. Використовуйте інтернет-ресурси та відкриті набори даних, такі як Kaggle, UCI Machine Learning Repository, Google Dataset Search тощо. Збір даних через веб-скрапінг. Використовуйте бібліотеки Python для веб-скрапінгу, такі як BeautifulSoup або Scrapy, для отримання даних з веб-сайтів. Напишіть скрипт, який автоматично отримує дані з веб-сайтів та зберігає їх у відповідному форматі для подальшого аналізу та обробки. Використовуйте власні даний, якщо вони доступні. Це можуть бути дані з власних досліджень, опитування, лабораторних експериментів тощо. Зберігайте дані у відповідному форматі та структуруйте їх так, щоб вони були готові для використання у тренуванні моделі. Використовуйте публічні або приватні API для отримання даних з онлайн-платформ, таких як Twitter, Facebook, Google Maps тощо. Запитайте API для отримання даних у відповідному форматі, які можна буде використовувати для тренування моделі. Використовуйте ручне введення даних, якщо вони невеликі або необхідно отримати від користувача. Створіть інтерфейс, що дозволяє користувачам вводити дані, і зберігайте їх у відповідному форматі. Тренування токенайзера – це процес створення та налаштування токенайзера для вашого конкретного завдання або мови. Токенайзер використовується для розбиття тексту на токени (наприклад, слова або підрядки символів), які потім можна використовувати для навчання моделей машинного навчання у сфері обробки природної мови. Ось кроки, які зазвичай потрібно виконати для тренування токенайзера. Зібрати великий корпус тексту, який відповідає вашій мові або конкретному завданню. Впевніться, що ваші дані представляють реальні умови використання, оскільки це допоможе покращити якість токенізації.

Очистіть та підготуйте дані перед тренуванням токенайзера. Це може включати видалення непотрібних символів, нормалізацію тексту та інші операції очищення. Виберіть або створіть токенайзер для вашого проекту. Ви можете скористатися готовими токенайзерами з бібліотек, таких як Hugging

Face’s Tokenizers, або створити власний. Виберіть параметри токенізації, такі як розмір вокабуляру, регулярні вирази для розділення слів, правила поділу на токени та інші. Використовуйте ваш корпус тексту для тренування токенайзера. Багато бібліотек, таких як Tokenizers від Hugging Face, надають інтерфейси для тренування токенайзера з використанням навчальних даних.т Після тренування перевірте якість токенізації на випадкових прикладах, щоб переконатися, що ваш токенайзер працює правильно. Оцініть якість токенайзації на тестовому наборі даних. Використовуйте метрики якості, такі як точність та покриття, для оцінки роботи вашого токенайзера. Виконайте корекцію та налаштування параметрів токенайзації, якщо необхідно, для поліпшення результатів. Після успішного тренування використовуйте токенайзер для підготовки тексту перед використанням його у моделі машинного навчання або іншому обробці тексту. Тренування токенайзера – це важливий крок у підготовці даних для використання у моделях машинного навчання, оскільки від якості токенізації може залежати ефективність та точність моделі.

Препроцесінг даних – це важливий етап у підготовці даних для подальшого використання у моделях машинного навчання. Під час препроцесінгу даних зазвичай виконуються різні операції очищення, перетворення та стандартизації даних, які допомагають покращити якість та ефективність моделі. Ось деякі типові кроки препроцесінгу даних. Видалення непотрібних символів, які не мають значення для аналізу. Видалення зайвих пробілів та інших проміжків. Обробка відсутніх значень, наприклад, заповнення пропусків або видалення рядків з неповними даними. Розділення тексту на окремі токени (слова або підрядки символів), що допомагає підготувати текст для подальшої обробки. Використання вбудованих токенізаторів або створення власних на основі правил та правил мови. Перетворення тексту до нижнього регістру для уніфікації. Видалення зайвої пунктуації та спеціальних символів. Лематизація або стеммінг слів для відображення їх базової форми. Перетворення тексту в числовий вектор для подальшого використання у моделі. Використання методів векторизації, таких як мішок слів (bag of words), TF-DF (term frequency-inverse document frequency),

векторизація на основі вбудованих слів тощо. Розділення даних на незалежні набори для тренування та оцінки моделі. Забезпечення того, щоб обидва набори мали подібний розподіл класів або категорій. Зведення різних типів даних до однакового формату. Нормалізація числових даних для забезпечення їхньої подібності та стабільності. Обробка тексту для вилучення важливих функцій або властивостей, таких як ключові слова, назви ентітетів тощо. Використання методів підготовки даних, таких як вирізка або додавання відбитків, для зменшення розмірності або підвищення інформативності даних. Процес препроцесінгу даних є важливим етапом в розробці моделі машинного навчання і може суттєво вплинути на її ефективність і точність. Після того, як дані були підготовлені та токенізовані, а також обрана архітектура моделі, можна переходити до етапу тренування моделі та обчислення метрик. Ось загальний опис цього процесу.

Вибір архітектури моделі, яка найкраще підходить до вашого завдання. Це може бути конволюційна нейронна мережа (CNN), рекурентна нейронна мережа (RNN), трансформер або будь-яка інша архітектура в залежності від типу даних та завдання. Визначення функції втрат (loss function) та оптимізатора для тренування моделі. Вибір оптимізатора та гіперпараметрів оптимізатора (наприклад, швидкість навчання) також є важливим кроком. Підготовка тренувальних та валідаційних даних. Запуск процесу тренування моделі на тренувальних даних. Оцінка моделі на валідаційних даних для визначення її ефективності та перевірки на перенавчання (overfitting).

Обчислення метрик ефективності моделі на тестовому наборі даних. Це може включати точність (accuracy), втрати (losses), прецизію (precision), відгук (recall), F1-оцінку, AUC-ROC тощо, залежно від типу завдання. Візуалізація результатів за допомогою графіків та діаграм для кращого розуміння результатів моделі. Аналіз результатів та виявлення можливостей для покращення. Можливе налаштування гіперпараметрів моделі (наприклад, кількість шарів, розмір пакетів, швидкість навчання) для поліпшення результатів. Повторення тренування та оцінки моделі з оновленими параметрами. Оцінка кінцевої моделі на тестовому наборі даних для оцінки її

загальної продуктивності та готовності до розгортання у виробничому середовищі. Цей процес може варіюватися в залежності від конкретного завдання, типу даних та вибраної архітектури моделі. Важливо провести дослідження та експерименти для забезпечення оптимальних результатів моделі. В ході тренування мовної моделі з конфігурацією GPT-2 з наступними гіпер параметрами зображено на рисунку 3.4.

розмір батчу (batch size) - 10
кількість тренувальних епох (train epochs) - 1
темп тренування (learning rate) - 2e-5
adam_beta1=0.9,
adam_beta2=0.999,
adam_epsilon=1e-08,
max_grad_norm=1.0,
warmup_ratio=0.0,
warmup_steps=0,
weight_decay=0.01,
vocab_size=50257,
n_positions=1024,
n_embd=768,
n_layer=12,
n_head=12,
n_inner=None,
activation_function="gelu_new",
resid_pdrop=0.1,
embd_pdrop=0.1,
attn_pdrop=0.1,
layer_norm_epsilon=1e-5,
initializer_range=0.02,

Рисунок 3.4 – Мовна модель з конфігурацією GPT-2

Можливе налаштування гіперпараметрів моделі (наприклад, кількість шарів, розмір пакетів, швидкість навчання) для поліпшення результатів. Повторення тренування та оцінки моделі з оновленими параметрами. Оцінка кінцевої моделі на тестовому наборі даних для оцінки її загальної продуктивності та готовності до розгортання у виробничому середовищі. Цей процес може варіюватися в залежності від конкретного завдання, типу даних та вибраної архітектури моделі. Важливо провести дослідження та експерименти

для забезпечення оптимальних результатів моделі. В ході тренування мовної моделі з конфігурацією GPT-2 з наступними гіпер параметрами.

ВИСНОВКИ

Здійснено аналіз існуючих методів векторизації текстів, включаючи традиційні та сучасні нейромережеві підходи. Проведено детальний аналіз традиційних методів векторизації, таких як Bag of Words, TF-IDF, а також сучасних підходів, включаючи Word2Vec, GloVe та FastText.

Досліджено архітектуру нейронних мереж, що використовуються для векторизації текстів (LSTM, CNN, трансформери). Досліджено контекстно-залежні моделі векторизації, такі як ELMo, BERT та GPT, які забезпечують більш точні і глибокі уявлення текстових даних. Розглянуто архітектури нейронних мереж, зокрема RNN, LSTM, GRU, CNN та трансформери, які використовуються для векторизації текстів. Оцінено переваги та недоліки кожної архітектури в контексті різних задач обробки природної мови.

Розроблено та реалізовано моделі векторизації текстів на основі обраних нейромережевих архітектур. Розроблено та реалізовано кілька моделей векторизації текстів на основі нейромережевих архітектур. Проведено експериментальне тестування моделей на різних задачах NLP, включаючи класифікацію текстів, аналіз настроїв та семантичний пошук.

Здійснено порівняння ефективності різних методів векторизації на основі експериментальних даних. Виконано порівняльний аналіз ефективності різних методів векторизації текстів за допомогою метрик точності, швидкості обробки та обчислювальних витрат. Виявлено, що контекстно-залежні моделі, такі як BERT та GPT, демонструють найвищу ефективність в більшості задач NLP. Розроблені моделі та методики можуть бути безпосередньо застосовані в реальних NLP-застосуваннях, таких як машинний переклад, аналіз тексту, пошукові системи та чат-боти. Надані рекомендації щодо вибору та налаштування нейромережевих моделей для різних типів задач NLP.

Вивчення можливостей адаптації та застосування розроблених моделей для конкретних задач NLP. Робота сприяє глибшому розумінню методів векторизації текстів на основі нейромережевих архітектур. Результати

дослідження можуть бути використані для подальших наукових досліджень у галузі обробки природної мови та машинного навчання.

Робота підтверджує значимість нейромережевих архітектур для векторизації текстів та їх переваги в порівнянні з традиційними методами. Векторні уявлення, отримані за допомогою сучасних нейромережевих моделей, демонструють високу якість і контекстну точність, що робить їх незамінними для широкого спектру задач обробки природної мови. Результати дослідження мають як наукову, так і практичну цінність, сприяючи подальшому розвитку NLP-технологій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. How neural networks work. URL: https://geekmatic.in.ua/ua/arduino_osnovyi_programmirovaniya (дата звернення: 11.03.2024).

2. Understanding Activation Functions in Neural Networks. URL: https://arduinka.biz.ua/blok-zhivlennya-9v-1a-p297c75.html (дата звернення: 11.03.2024).

3. Neural networks versus Logistic regression for 30 days all-cause readmission prediction. URL: https://en.wikipedia.org/wiki/ATmega328 (дата звернення: 11.03.2024).

4. Deep Learning for NLP. URL: https://vencon.ua/ua/articles/kak-vybrat-batarejku-ili-akkumulyator-vybiraem-batarejki-akkumulyatornye-i-obychnye (дата звернення: 11.03.2024).

5. Word2vec Made Easy. URL : https://www.hwlibre.com/uk/tft/ (дата звернення: 11.03.2024).

6. How is GloVe different from word2vec? URL: https://electronica.in.ua/ua/p1630184399-ostsilograf-dso-shell.html (дата звернення: 23.03.2024).

7. What are the advantages and disadvantages of Word2vec and GloVe? URL : https://uk.fmuser.net/content/?11010.html (дата звернення: 23.03.2024).

8. How is GloVe different from word2vec?. URL : https://www.guru99.com/uk/analog-vs-digital.html (дата звернення: 23.03.2024).

9. Distributed Representations of Words and Phrases and their Compositionality. Головна | Elib LNTU. URL : https://elib.lntu.edu.ua/sites/default/files/elib_upload/ipv/page10.html (дата звернення: 23.03.2024).

10. Word2Vec Tutorial Part I: The Skip-Gram Model. URL : https://radio-shop.com.ua/uk/osnovni-parametry-ostsylohrafiv (дата звернення: 23.03.2024).

11. Electronics for Beginners: A Practical Introduction to Schematics, Circuits, and Microcontrollers. O’Reilly Online Learning. URL:

https://www.oreilly.com/library/view/electronics-for-beginners/9781484259795/ (date of access: 23.03.2024).

12. ABCs of Electronics: An Easy Guide to Electronics Engineering. O'Reilly Online Learning. URL: https://www.oreilly.com/library/view/abcs-of-electronics/9798868801341/ (date of access: 23.03.2024).

13. Circuit Design and Simulation Quick Start Guide: Create Schematics and Layout Electronic Components. URL: https://www.oreilly.com/library/view/circuit-design-and/9781484295823/ (date of access: 23.03.2024).

14. PCB Design for Absolute Beginners: Layout Printed Circuit Boards in a Web Browser. URL: https://www.oreilly.com/library/view/pcb-design-for/9781484280409/ (date of access: 23.03.2024).

15. Practical Electronic Design for Experimenters. URL: https://www.oreilly.com/library/view/practical-electronic-design/9781260456165/ (date of access: 23.03.2024).

16. DIY Microcontroller Projects for Hobbyists. URL: https://www.oreilly.com/library/view/diy-microcontroller-projects/9781800564138/ (date of access: 23.03.2024).

ДОДАТКИ

Додаток А

Лістинг dvc.yaml

```
stages:
  download_dataset:
     cmd: python scripts/download_dataset.py data/src/
     deps:
     - scripts/download_dataset.py
     params:
     - download_dataset
     outs:
     - data/src/
  train_tokenizer:
     cmd: python scripts/train_tokenizer.py data/src/
data/tokenizer/
     deps:
     - scripts/train_tokenizer.py
     - data/src/
     outs:
     - data/tokenizer/
  prepare_dataset:
     cmd: python scripts/prepare_dataset.py data/src/
data/tokenizer/tokenizer.json data/prepared/ --temp-folder
data/cache/
     deps:
     - scripts/prepare_dataset.py
     - data/tokenizer/
     - data/src/
     params:
     - prepare_dataset
     outs:
     - data/prepared/
  train:
     cmd: python scripts/train.py data/prepared/
data/tokenizer/tokenizer.json data/model/ --checkpoint_root
data/checkpoints/
     deps:
     - scripts/train.py
     - scripts/utils/
```

Додаток Б

Лістинг params.yaml

```
download_dataset:
  dataset:
    path: oscar
    name: "unshuffled_deduplicated_uk"

prepare_dataset:
  batch_size: 64
  block_size: 64

train:
  datasets:
     train_dataset_name: "train"
     eval_dataset_name: "validation"
  model_config: {}
  trainer_config:
     num_train_epochs: 1
     evaluation_strategy: "no"
     learning_rate: 2.e-5
     weight_decay: 0.01
     save_strategy: steps
     per_device_train_batch_size: 10
     save_total_limit: 5
     save_steps: 1000
     do_eval: False
  resume_from_checkpoint: True
```

Додаток В

Лістинг scripts/download

```
from argparse import ArgumentParser
from pathlib import Path
from typing import Dict, Optional

import yaml
from datasets import DatasetDict
from datasets.load import load_dataset
from loguru import logger


def get_params(params_file: Optional[Path] =
Path("params.yaml"), key: Optional[str] = "download_dataset") ->
Dict:
    return
yaml.safe_load(params_file.read_text(encoding="utf-8"))[key]


def describe_datasets(datasets: DatasetDict):
    logger.info(f"Loaded datasets: {datasets}")
    for key in datasets:
    logger.info(f"Dataset[{key}] examples:
{datasets[key][:1]}")

def download_dataset(target_dir: Path):
    params = get_params()
    datasets: DatasetDict =
load_dataset(**params["dataset"])
    describe_datasets(datasets)
    datasets.save_to_disk(target_dir)


if __name__ == "__main__":
    parser = ArgumentParser()
    parser.add_argument("dir", type=Path)
    args = parser.parse_args()

    download_dataset(target_dir=args.dir)
```

Додаток Г

Лістинг scripts/prepare

```
import shutil
from argparse import ArgumentParser
from multiprocessing import cpu_count
from pathlib import Path
from pprint import pformat
from tempfile import gettempdir
from typing import Dict, Optional

import datasets
import yaml
from datasets import DatasetDict
from loguru import logger
from tokenizers import Tokenizer
from transformers import PreTrainedTokenizerFast


def get_params(params_file: Optional[Path] =
Path("params.yaml"),
               stage_name: Optional[str] =
"prepare_dataset") -> Dict:
    return
yaml.safe_load(params_file.read_text(encoding="utf-
8"))[stage_name]


def get_tokenizer(tokenizer_file: Path) -> Tokenizer:
    tokenizer =
PreTrainedTokenizerFast(tokenizer_file=str(tokenizer_file))
    logger.info(f"Loaded {tokenizer}")
    return tokenizer


def get_dataset(input_folder: Path) -> DatasetDict:
    dataset = datasets.load.load_from_disk(input_folder)
    logger.info(f"Loaded {dataset}")
    return dataset
```