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: «**Модель нейронної мережі для цензурування текстових даних**»

Захищено на засіданні
Атестаційної комісії № 42
протокол № __ від __.12.2023 р.
Оцінка _____ / _____
Голова Атестаційної комісії
_____**СКОБ Ю. О.**

Виконав:

студент 2 курсу, групи КУ– 61
Галузь знань: 151 – Автоматизація
та комп'ютерно-інтегровані технології.
Галузь знань 15 – Автоматизація та
приладобудування
ТИТАРЕНКО Тимур Олегович_____

Керівник:

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

Рецензент:к.ф.-м.н., доцент, в.о. зав.
кафедри електроніки і управляючих
систем
ХРУСЛОВ Максим

Михайлович_____

Харків – 2023

АНОТАЦІЯ

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, переліку використаних джерел і трьох додатків. Загальний обсяг роботи становить 84 сторінки, з яких 64 сторінок основної частини з 37 рисунками, 1 таблиця, 2 сторінок списку використаних джерел із 20-х найменувань, 4 додатків на 8 сторінках.

Це дослідження приділяє увагу можливості використання нейронних мереж для цензурування текстових даних. Мета кваліфікаційної роботи забезпечити більш якісний та безпечний контент для користувачів, які залежать від надійної та безпечної інформації в Інтернеті, за допомогою розробки та впровадження нейронної мережі, яка буде здатна аналізувати текстові дані в реальному часі. Під час виконання дослідження розглядалися різні типи нейронних мереж та їх потенціал у вирішенні завдань цензури.

Результатом даного дослідження є розроблена та оптимізована модель нейронної мережі, яка здатна ефективно фільтрувати та цензурувати текстовий контент. Застосування цієї моделі може бути актуальним у сферах, де необхідно автоматизовано контролювати та фільтрувати неприпустимий текстовий матеріал, а саме соціальні мережі, форуми, блоги та інші джерела інформації.

Ключові слова: нейронні мережі, цензура тексту, обробка природної мови, LSTM, PYTHON, MACHINE LEARNING

ABSTRACT

The qualification work consists of an introduction, three sections, conclusions, a list of used sources and three appendices. The total volume of the work is 84 pages, of which 64 pages are the main part with 37 figures, 1 table, 2 pages of the list of used sources from 20 names, 4 appendices on 8 pages.

This study focuses on the possibility of using neural networks to censor textual data. The goal of the qualification work is to provide greater security and quality of content for users who depend on reliable and secure information on the Internet, through the development and implementation of a neural network that will be able to analyze text data in real time. During the research, different types of neural networks and their potential in solving censorship tasks were considered.

The result of this study is a developed and optimized model of a neural network, which is able to effectively filter and censor text content. The application of this model can be relevant in areas where it is necessary to automatically control and filter unacceptable textual material, namely social networks, forums, blogs and other sources of information.

Keywords: neural networks, text censorship, natural language processing, LSTM, PYTHON, MACHINE LEARNING

ЗМІСТ

ВСТУП

У сучасному інформаційному суспільстві, де потоки текстової інформації невпинно зростають, важливість ефективної фільтрації та цензурування текстових даних визнається як ключова область для забезпечення безпеки, конфіденційності та етичності в онлайн середовищі. Інновації в галузі штучного інтелекту та глибокого навчання створюють унікальні можливості для розв'язання цього виклику.

В умовах стрімкого розвитку інтернет-комунікацій та зростання обсягів текстового контенту, актуальність роботи обумовлена необхідністю забезпечення ефективного цензурування текстових даних. Особливо в онлайн середовищі, де важливо забезпечити безпеку та етичність спілкування. Непередбачуваність та динаміка змін у сучасному мовному просторі створюють виклик для розробки інноваційних моделей, які здатні точно та автоматично визначати неприпустимий контент, забезпечуючи безпеку та етичність онлайн-спілкування.

Мета кваліфікаційної роботи забезпечити більш якісний та безпечний контент для користувачів, які залежать від надійної та безпечної інформації в Інтернеті, за допомогою розробки та впровадження нейронної мережі, яка буде здатна визначати недопустимий контент у текстових дані в реальному часі.

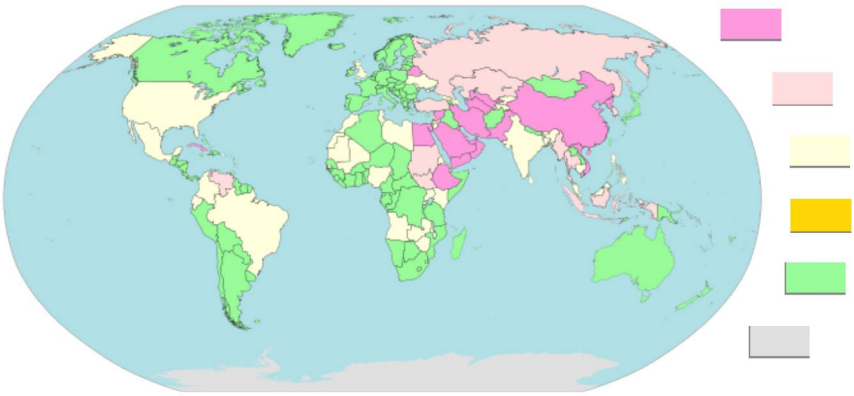
Об'єктом розробки є процес цензурування небажаного та шкідливого контенту з текстів та збереження корисного контенту.

Предметом розробки є модель нейронної мережі для цензурування текстових даних, яку можливо ефективно використовувати для точного виявлення та видалення небажаного та шкідливого контенту.

Методи дослідження, для досягнення поставленої мети використовуються методи глибокого навчання, теорія нейронних мереж, теорія штучного інтелекту математичний аналіз, методи аналізу інформативності, методи оцінки якості класифікації, дослідження практичного застосування.

Завдання дослідження

1. Аналіз підходів цензурування текстових даних.
2. Аналіз можливості застосування нейронних мереж у завданні цензуруванні
3. Вибір типу моделі нейронної мережі.
4. Аналіз параметрів моделі
5. Розробка архітектури нейронної мережі.
6. Вибір даних для навчання.
7. Розробка програмної реалізації.
8. Проведення експериментів для визначення оптимальних параметрів моделі.
9. Оцінка продуктивності та ефективності розробленої моделі.
10. Визначення можливостей застосування моделі в різних сферах та обґрунтування перспектив її використання.
11. Практичне застосування навченої моделі



Блокування ключових слів є одним з основних методів цензурування текстового вмісту. Цей підхід передбачає створення списку слів або фраз, які вважаються небажаними, образливими, чи порушують правила певної платформи чи сервісу. Ці слова можуть бути взяті з мовного фільтру, стандартів спілкування чи регламенту спільноти.

Коли користувач намагається розмістити текст або коментар, система перевіряє його на наявність цих ключових слів. Якщо слова чи фрази співпадають з елементами списку, система може приймати рішення про блокування або фільтрацію контенту. Цей метод особливо ефективний для автоматичного виявлення та блокування особливо неприйнятного вмісту. Однак, важливо враховувати, що іноді ця стратегія може бути надто жорсткою, особливо коли враховувати контекст чи виразову вільність користувачів.

Контент-модерація включає в себе використання людського фактора для перегляду та оцінки контенту. Основна ідея полягає в тому, що модератори, які є реальними людьми, розглядають текстовий, візуальний чи аудіо контент, який користувачі публікують на платформі.

Модератори діють відповідно до заздалегідь встановлених правил та стандартів, які можуть включати в себе обмеження щодо непристойного вмісту, насильства, розпалювання ненависті, порушення авторських прав і т. д. Вони мають завдання визначити, чи відповідає конкретний контент цим стандартам і чи може бути опублікованим чи видаленим.

Контент-модерація може бути важливим елементом управління безпекою та дотриманням правил на різних інтернет-платформах, але вона також може викликати питання стосовно об'єктивності та свободи висловлення, оскільки рішення модераторів може впливати на доступ до інформації та вираження різних поглядів [2].

Деякі платформи дозволяють користувачам налаштовувати рівень цензури, додавати власні слова до списку або визначати, наскільки строгим має бути фільтр. Це робить систему гнучкою та більш придатною для різноманітних спільнот і аудиторій.

Моделі машинного навчання — це підхід, який використовує алгоритми машинного навчання для автоматичного виявлення та фільтрації неприйнятного контенту. Замість того, щоб розробляти жорсткі правила або списки ключових слів, система вивчає зразки вмісту з великої кількості даних. Моделі навчаються розпізнавати патерни та характеристики, які вказують на небажаний вміст. Це може включати аналіз вживання мови, виявлення тональності, розпізнавання образливих висловлювань, та інші аспекти. Важливо, що моделі можуть адаптуватися до нових трендів та змінюючогося мовного вживання.

Однак, моделі машинного навчання не є бездоганними. Вони можуть допускати помилки, особливо в ситуаціях, коли контент має складний або контекстуальний характер. Також, важливо забезпечити, щоб дані, на яких вони навчаються, були репрезентативними та враховували різноманіття користувачів.

Цей підхід може бути ефективним для автоматичного виявлення різноманітних форм неприйнятного вмісту, проте йому потрібно бути постійно вдосконалюваним, оновлюваним та контрольованим, щоб враховувати змінюючіся тенденції та уникати помилкових фільтрів.

Блокування за допомогою IP-адрес - це метод цензури, який використовує ідентифікацію IP-адресу для блокування доступу до певного контенту. Кожен пристрій, підключений до Інтернету, має свій унікальний IP-адрес, який ідентифікує його в мережі. Коли користувач намагається отримати доступ до контенту, система перевіряє його IP-адресу. Якщо цей адрес збігається з адресою, яка була попередньо асоційована із забороненим контентом, то доступ може бути заблокований [3].

Цей метод може бути ефективним, оскільки IP-адреса можуть бути пов'язані із конкретними користувачами чи групами користувачів. Однак важливо враховувати, що деякі користувачі можуть використовувати обходження IP-блокування за допомогою використання віртуальних приватних мереж (VPN) або проксі-серверів.

Цей метод зазвичай використовується в ситуаціях, де необхідно заблокувати доступ до конкретного вмісту для всіх користувачів з певних мереж або географічних областей. Однак він може бути менш ефективним у випадках,

коли користувачі використовують техніки обходження або маскування свого IP-адреси.

Системи батьківського контролю використовується для обмеження доступу до контенту для певних користувачів чи груп користувачів, зазвичай з метою захисту дітей від непридатного чи небажаного вмісту. Системи батьківського контролю дозволяють батькам або адміністраторам встановлювати обмеження та фільтри відповідно до віку чи інших параметрів користувачів.

Ці системи можуть включати в себе кілька функцій.

- Фільтрація за віком: дозволяє обмежувати доступ до контенту в залежності від вікової категорії користувача.
- Блокування конкретних сайтів: дозволяє встановлювати блокування для певних веб-сайтів чи категорій контенту.
- Часові обмеження: дозволяє обмежувати час, протягом якого користувач може використовувати Інтернет чи певні додатки.
- Моніторинг активності: надає можливість батькам або адміністраторам переглядати звіти про використання та активність користувачів.

Цей метод є ефективним для забезпечення контролю над тим, який контент може бути доступний для конкретних користувачів, особливо у випадках, коли важливо враховувати вікові обмеження. Використання систем батьківського контролю може допомогти створити безпечне та відповідальне середовище для використання Інтернету.

1.1.2 Етичні, соціальні та технічні проблеми

Цензурування текстового контенту породжує різні етичні, соціальні та технічні проблеми, такі як:

- свобода слова;
- суб’єктивність;
- неспроможність визначити контекст;
- обхід цензури;
- виклики в області техніки.

Цензура може порушити принцип свободи слова, який вважається ключовим для демократичних суспільств. Обмеження доступу до певної інформації може призвести до обмеження свободи вираження та обміну ідеями.

Суб'єктивність виникає з того, що визначення того, що є неприйнятним або образливим, може значно варіюватися залежно від культур, групи людей чи індивідуальних переконань. Одна й та ж сама фраза чи вислів може бути сприйнятими різними людьми по-різному. Спроби визначити "неприйнятний" вміст можуть враховувати такі аспекти:

- культурні різниці: поняття образливості чи неприйнятності може різнитися в різних культурах. Те, що є прийнятним в одній культурі, може бути образливим в іншій;
- індивідуальні переконання: індивіди мають різні моральні та етичні переконання. Те, що одна людина може вважати неприйнятним, інша може сприйняти як вираження свободи слова чи художню експресію;
- зміна соціокультурного контексту: мінливі соціальні та культурні норми можуть призводити до зміни у сприйнятті того, що вважається прийнятним чи образливим. Інтернет та соціальні мережі активно впливають на формування нових виразових засобів та стандартів;
- визначення контексту: поняття образливості часто залежить від контексту, в якому вислів використовується. Той самий термін може мати різне значення в різних ситуаціях [3].

Суб'єктивність цього аспекту ставить під сумнів універсальність стандартів цензури та вимагає уваги до індивідуальних розумінь та визначень в різних соціокультурних контекстах. Таким чином, важливо знаходити баланс між регулюванням контенту та збереженням різноманітності виражень і свободи слова.

Неспроможність визначити контекст: автоматизовані системи цензури, зокрема засновані на штучних нейронних мережах, можуть маюти складнощі в розумінні контексту, що може призвести до помилкового блокування чи фільтрації текстів.

Алгоритми цензури можуть стикатися із складнощами в розрізненні іронії, гумору, або того, як певний вислів може змінювати своє значення в залежності від контексту.

Деякі аспекти цієї проблеми включають.

- Іронія та гумор: висловлювання може мати іронічний або гумористичний контекст, що зрозуміле для людини, але може бути важким для алгоритмів, які працюють на основі синтаксичних та семантичних правил.
- Зміна значення слова в залежності від контексту: слова можуть мати різне значення в різних контекстах, і важко або навіть неможливо визначити, яке саме значення малося на увазі в конкретному випадку.
- Неоднозначність висловів: деякі вислови можуть бути неоднозначними та мати кілька інтерпретацій, що ускладнює завдання визначення, чи є вони прийнятними чи ні.
- Еволюція контексту: зміна суспільного та культурного контексту може призводити до зміни сприйняття виразів та слів, але алгоритми можуть не завжди враховувати ці зміни.

Обхід цензури: цей аспект вказує на те, що користувачі можуть використовувати різні техніки для ухилення від систем цензури та отримання доступу до забороненого або обмеженого контенту. Обхід цензурних обмежень може стати проблемою для тих, хто намагається контролювати доступ до певного вмісту. Використання альтернативних слів, користувачі можуть змінювати слова чи використовувати схожі терміни, щоб уникнути фільтрів. Маскування контенту, використання специфічних символів, реєстрації чи інших маскувальних технік для ухилення від фільтрів.

Виклики в області техніки: розробка ефективних та справедливих технологій цензури текстових даних виправдовує важливість вирішення кількох ключових викликів в області техніки. Перш за все, обробка природної мови виявляється складною завданням, оскільки вона вимагає розуміння не лише синтаксичних та семантичних правил, але і врахування великого різноманіття мовних виразів та ідіом.

Одним із викликів є уникання великої кількості помилок в процесі фільтрації. Алгоритми цензури повинні бути точними, щоб уникати "ложних позитивів" (фільтрація невинних висловлювань) та "ложних негативів" (невиявлення справжньої образливості чи неприйнятного контенту). Точність грає важливу роль у забезпеченні справедливості та ефективності системи. Врахування різноманітності мови та контекстів є ще однією ключовою аспектом. Терміни та вислови можуть мати різні значення в різних культурах чи соціокультурних групах, тому системи цензури повинні бути адаптовані до різних контекстів використання мови. Це включає врахування іронії, гумору, та інших виразових форм, які можуть мати місце в різних ситуаціях [3].

Використання штучного інтелекту, зокрема штучних нейронних мереж, виявляється найбільш ефективним у вирішенні викликів, пов'язаних із цензурою текстових даних. Штучні нейронні мережі володіють здатністю ефективно адаптуватися до складних структур текстів та контекстів, що робить їх особливо корисними у виявленні та фільтрації небажаного контенту. Вирішення цих викликів потребує впровадження передових методів обробки природної мови, застосування технологій машинного навчання, а також постійного навчання систем для адаптації до змін у мовному вживанні та соціальних стандартах. Безперечно, розробка та вдосконалення таких технологій повинні враховувати етичні аспекти, такі як конфіденційність даних та забезпечення свободи виразу.

Отже, використання штучних нейронних мереж є перспективним, оскільки це дозволяє не лише вдосконалювати процес цензури текстів, а й забезпечує адаптивність до зростаючої складності та різноманітності мовленнєвих виразів.

1.2 Штучні нейронні мережі

Розвиток штучних нейронних мереж тісно пов'язаний із біологічними принципами, використовуючи терміни, що характеризують організацію мозкової діяльності, при дослідженні мережних конфігурацій та алгоритмів. Однак ця аналогія має свої межі через обмеженість наших знань про мозок, що

ускладнює використання його як взірця. Розробникам мереж доводиться виходити за межі сучасних біологічних знань, шукаючи структури, що можуть виконувати корисні функції.

Нейрони, як базові елементи мозку, відрізняються від інших клітин тіла своєю здатністю запам'ятовувати та застосовувати попередній досвід. Кора головного мозку людини, об'єм якої приблизно відповідає площі стандартної клавіатури, містить мільярди нейронів та трильйони взаємозв'язків. Мозок людини здатен генетично програмуватися та навчатися, визначаючи потужність розуму.

Окремий нейрон, що має свої складові, підсистеми та механізми керування, передає інформацію через велику кількість електрохімічних зв'язків. Загалом, різноманітність нейронів та з'єднань між ними формують недвійковий, нестійкий та несинхронний процес, відмінний від обчислень традиційних комп'ютерів. Штучні нейромережі моделюють лише основні аспекти мозку, проте стимулюють науковців та розробників до пошуку нових підходів до вирішення завдань [4].

Ключові події у історії дослідження та використання штучних нейронних мереж.

- 1943 р. – У. Маккалок та У. Піттс вперше формалізують концепцію нейронної мережі у статті про логічні обчислення та нервову активність.
- 1948 р. – Н. Вінер та його команда публікують роботу з кібернетики, в якій запропоновано математичні моделі для складних біологічних процесів.
- 1949 р. – Д. Хебб вводить перший алгоритм навчання для нейронних мереж.
- 1958 р. – Ф. Розенблатт створює одношаровий перцептрон, що демонструє успішність у завданнях класифікації та стає популярним для розпізнавання образів та прогнозування.
- 1960 р. – Уідроу та Хофф розробляють Адалін, використовуючи дельта-правило для передбачення та адаптивного управління.
- 1963 р. – В Інституті проблем передачі інформації АН СРСР проводиться докладне дослідження завдань "важких" перцептронів.

- 1969 р. – М. Мінський публікує формальний доказ обмеженості перцептрону та вказує на його нездатність вирішувати певні завдання.
- 1972 р. – Т. Кохонен та Дж. Андерсон представляють новий тип нейронних мереж, здатних працювати як пам'ять.
- 1973 р. – Б. В. Хакімов пропонує нелінійну модель із синапсами на основі сплайнів для вирішення завдань у медицині, геології та екології.
- 1974 р. – Пол Дж. Вербос та А. І. Галушкін винаходять алгоритм зворотного розповсюдження помилок для навчання багатошарових перцептронів.
- 1975 р. – Фукусіма представляє когнітрон – самоорганізуючуся мережу для інваріантного розпізнавання образів, хоча це досягається шляхом запам'ятовування практично всіх станів образу.
- 1982 р. – Після періоду забуття відновлюється інтерес до нейромереж. Дж. Хопфілд показує, що нейронна мережа із зворотніми зв'язками може діяти як система, що мінімізує енергію (мережа Хопфілда). Кохонен представляє моделі мережі, що навчаються без вчителя, розв'язуючи завдання кластеризації, візуалізації даних (карта Кохонена, яка самоорганізовується) та інші задачі попереднього аналізу даних.
- 1986 р. – Девід І. Румельхарт, Дж. Є. Хінтон і Рональд Дж. Вільямс, разом із С. І. Барцевим та В. А. Охоніним (Красноярська група), значно вдосконалюють метод зворотного розповсюдження помилок. Це спричиняє вибух інтересу до навчання нейронних мереж.
- 2007 р. – Джеффрі Хінтон, працюючи в університеті Торонто, розробляє алгоритми глибокого навчання для багатошарових нейронних мереж. Успіх обумовлений використанням обмеженої машини Больцмана (RBM – Restricted Boltzmann Machine) під час навчання нижніх шарів мережі [5].

Хоча на початку існувало багато досліджень та розробок в галузі штучних нейронних мереж, їхня популярність швидко впала через технічні обмеження. Обчислювальна складність штучних нейронних мереж була надто високою для комп'ютерів того часу. Комп'ютери не володіли достатньою обчислювальною потужністю, і підготовка нейронних мереж займала надто багато часу. У зв'язку

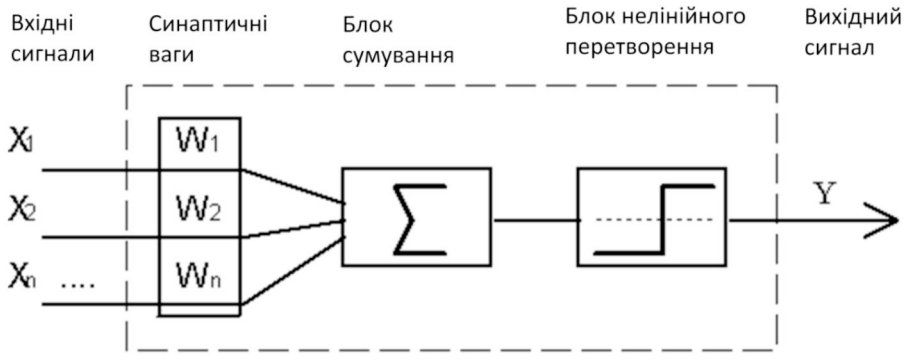
з цим інші методи машинного навчання стали більш популярними, а штучні нейронні мережі втратили свою актуальність.

Важливим досягненням цього періоду є розробка алгоритму зворотного розповсюдження, який винайшов Пол Вербос. Цей алгоритм став ключовим методом навчання штучних нейронних мереж, спрямованим на мінімізацію помилок. Зворотне розповсюдження значно полегшило тренування штучних мереж, роблячи його ефективнішим та швидшим.

У кінці 2000-х років штучні нейронні мережі відновили свою популярність, особливо після того, як компанії, такі як Google і Facebook, продемонстрували переваги використання методів машинного навчання на великих обсягах даних, зібраних від звичайних користувачів. В сучасний час ці алгоритми переважно використовуються для глибокого навчання, що представляє собою галузь машинного навчання, спрямовану на моделювання більш складних відносин, таких як нелінійні взаємодії [6].

1.2.1 Математична модель нейрона

Історично першою роботою, що заклала теоретичний фундамент для створення штучних моделей нейронів та нейронних мереж, прийнято вважати опубліковану у 1943 р. статтю Уоррена С. Мак-Каллока і Вальтера Піттса "Логічні обчислення ідей, що відносяться до нервової активності". Головний принцип теорії Маккалока і Піттса полягає в тому, що довільні явища, що відносяться до вищої нервової діяльності, можуть бути проаналізовані та зрозумілі як деяка активність у мережі, що складається з логічних елементів, що приймають лише два стани ("все або нічого"). При цьому для будь-якого логічного вираження, що задовольняє зазначеним авторами умовам, може бути знайдена мережа логічних елементів, що має поведінку, що описується цим виразом. Ці ідеї через кілька років розвинув американський нейрофізіолог Френк Розенблатт. Він запропонував схему пристрою, що моделює процес людського сприйняття і назвав його «перцептроном»[7]. Перцептрон передавав сигнали від фотоелементів , що являють собою сенсорне поле, блоки електромеханічних осередків пам'яті. Ці осередки з'єднувалися між собою


Вхідні сигнали
Синаптичні ваги
Блок сумування
Блок нелінійного перетворення
Вихідний сигнал
X1
X2
Xn
....
W1
W2
Wn
Y

У моделі Маккалока і Піттса відсутні тимчасові затримки вхідних сигналів, тому значення net визначає повне зовнішнє збудження, сприйняте нейроном. Відгук нейрона далі описується за принципом "все або нічого", тобто 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 (часто рівне нулю) також зберігається в локальній пам'яті.

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

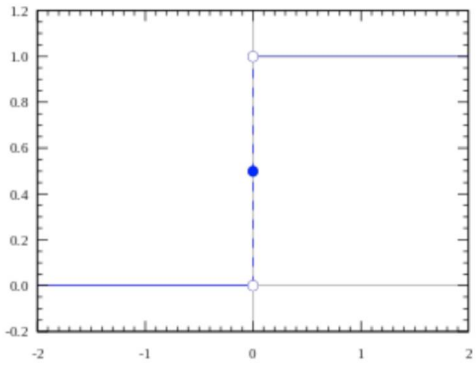
Слід зазначити, що сьогодні, через 50 років після роботи Маккалока і Піттса, вичерпної теорії синтезу логічних нейронних мереж із довільною функцією немає. Найбільш просунутими виявилися дослідження в галузі багатошарових систем та мереж із симетричними зв'язками. Більшість моделей спираються у своїй основі на різні модифікації формального нейрона.

Таким чином результат роботи нейрона – це функція активації взята від суми входів нейронів, помножена на ваги із додаванням зміщення (з-за того, що суматор є лінійною комбінацією входів X і ваг W, то вихід нейрона $Y = f(z) = f(X_1*W_1 + ... X_nW_n)$ – це насправді лінійна функція, а весь спектр лінійних функцій – це y = kx + b, де b – це зміщення, яке ми додаємо в суматорі).

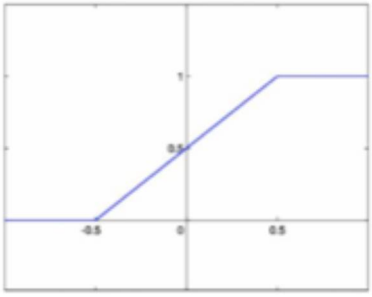
1.2.2 Функції активації. Види функції активації.

Функція активації визначає вихідне значення нейрона в залежності від результату зваженої суми входів та порогового значення.

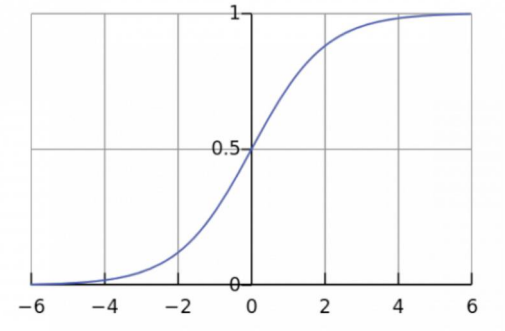
Розглянемо нейрон:


1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-2
-1
0
1
2

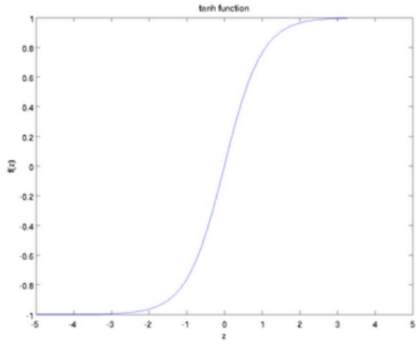
$$A = cx$$

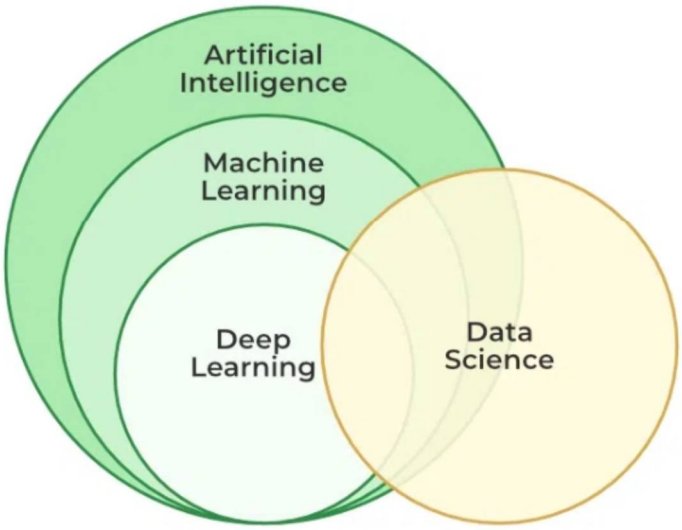


$$A = \frac{1}{1 + e^{-x}}$$



$$f(x) = tanh(x) = \frac{2}{1 + e^{-2x}} - 1 = 2sigmoid(2x) - 1$$




Artificial
Intelligence
Machine
Learning
Deep
Learning
Data
Science

зв'язки в наборах даних. Алгоритми глибокого навчання, такі як автокодери та генеративні моделі, використовуються для неконтрольованих завдань, таких як кластеризація, зменшення розмірності та виявлення аномалій.

Машинне навчання з підкріпленням — це техніка машинного навчання, за допомогою якої агент вчиться приймати рішення в середовищі, щоб максимізувати сигнал винагороди. Агент взаємодіє з навколишнім середовищем, виконуючи дії та спостерігаючи за отриманими винагородами. Глибоке навчання можна використовувати для вивчення політик або набору дій, які максимізують сукупну винагороду з часом. Алгоритми глибокого навчання з посиленням, як-от мережі Deep Q і глибокий детермінований градієнт політики (DDPG), використовуються для посилення таких завдань, як робототехніка та ігри тощо [12].

1.3 Базові алгоритми навчання нейронних мереж

1.3.1 Алгоритм зворотного розповсюдження помилки

Алгоритм зворотного поширення помилок призначений для швидкого обчислення градієнта. Для розрахунку градієнта необхідно обчислити окремі похідні: $\frac{\partial C}{\partial w_j^i}$, $\frac{\partial C}{\partial b_j^i}$. Розраховувати їх кожен раз вручну - складне завдання, оскільки нейронна мережа може бути глибокою і мати багато параметрів. При цьому робити це необхідно швидко. Для швидкого обчислення градієнта функції витрат був винайдений алгоритм зворотного поширення помилок. Метод був вперше винайдений у 1970-х роках, потім значного розвитку отримав у 1986 році. Розглянемо далі.

Символ ⊙ позначає операцію Адамара.

Визначення 1. Операція Адамара — це операція над двома матрицями однакової розмірності, результатом якої є матриця однакової розмірності. Значення елемента з індексом ij виходить як добуток елементів з однаковим індексом вихідних матриць.

Введемо помилку нейрона j шару l:

$$\delta_j^l = \frac{\delta C}{\delta z_j^l} \tag{1.2}$$

Тоді загальний алгоритм зворотного розповсюдження помилки буде таким:

1. Розрахуємо активацію для кожного з шарів нейронної мережі:

$$a^l = \sigma(w^l a^{l-1} + b^l). \tag{1.3}$$

2. Обчислюємо вектор помилки останнього шару:

$$\delta^L = \nabla C \odot \sigma'(w^L a^{L-1} + b^L). \tag{1.4}$$

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

$$\delta l = ((wl+1)T\delta l+1) \odot \sigma'(wl-1al-1 + bl) \tag{1.5}$$

4. Обчислюємо градієнт функції вартості за формулами:

$$\frac{\delta C}{\delta b_j^l} = \delta_j^l, \ \frac{\delta C}{\delta w_{ij}^l} = a_j^l \delta_j^l. \tag{1.6}$$

Алгоритм називається зворотним поширенням помилки, оскільки помилка поширюється назад до перших рівнів мережі. Похибка обчислюється від останнього шару і закінчуючи першим. І розраховують всі необхідні параметри нейронної мережі за два проходи.

1.3.2 Градієнтний спуск

Завданням навчання нейронної мережі є зменшення значення функції вартості, а саме мінімізація функції з багатьма змінними. Оскільки ця функція є

вкрай складною і, зазвичай, включає велику кількість параметрів, традиційний підхід до пошуку екстремуму функції стає неефективним. Спершу потрібно розрахувати похідні першого порядку, вирішити систему рівнянь та визначити потенційні точки екстремуму. Після цього, для перевірки, яка з цих точок є локальним мінімумом, необхідно обчислити похідні другого порядку. Однак цей метод є обчислювально витратним і складним. Саме тому в алгоритмах мінімізації функцій, використовуваних у сфері машинного навчання, переважно використовується градієнтний спуск.

Наше завдання — мінімізувати функцію багатьох змінних, ваг і зсувів.

Визначення 2. Глобальний мінімум функції $f{:}X{\to}\mathbb{R}, x0 \in \min f <=>$

$\forall\ x \in X$ задовольняє $f(x) \geq f(x0)$

Означення 3. Локальний мінімум функції $f{:}X{\to}\mathbb{R}, x0 \in loc\ \min f <=>$

$\exists O\epsilon(x0)$, що $\forall\ x \in O\epsilon(x0)$ буде виконано $f(x) \geq f(x0)$

Означення 4. Градієнт функції $f{:}\mathbb{R}n \to \mathbb{R}$ називається вектором приватних похідних.

Означення 5. Похідна функції $f{:}\mathbb{R}n \to \mathbb{R}$ у напрямку e називається проекцією градієнта в цьому напрямку.

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

Щоб мінімізувати функцію f, необхідно знайти напрямок, у якому f спадає найшвидше. Тобто необхідно знайти $\min e \nabla f = \min \|e\| \|\nabla f\| \cos\theta$, вважаючи довжину вектора сталою, вираз зводиться до $\min \cos\theta$, $\theta = \pi$ тобто значення похідної за напрямком мінімальна, якщо її напрямок протилежний напрямку вектора градієнта.

Таким чином, функція може бути приведена в напрямку зворотного градієнта відповідно до наступного правила:

$$x_j' = x_j - \eta \frac{\delta f}{\delta x_j}, \tag{1.7}$$

де η — швидкість навчання, додатне число, що вказує розмір кроку, на який дана функція зменшується. Зазвичай для η вибирають невелике позитивне

число. Таким чином, можливо зменшити значення функції, застосувавши це правило кілька разів поспіль. Переносячи (1.3) до функції витрат, отримуємо:

$$w_i' = w_i - \eta \frac{\partial C}{\partial} \qquad (1.8)$$

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

Оскільки функція витрат (1.7) є усередненою по всіх вхідних даних, щоб обчислити нові значення, необхідно буде обчислити градієнт ∇C для кожного вхідного елемента, а потім усереднити його. Іноді кількість таких входів може бути більшою. І такий розрахунок займе багато часу. Тому зручніше виконувати обчислення градієнта на випадкових невеликих наборах даних (нам не потрібно знати точне спадання функції, нам головне рухатися в правильному напрямку). Такі входи називають міні-серією. При цьому вважається, що значення градієнта на таких сторонах приблизно таке ж, як і на всіх даних. Такий алгоритм навчання називається стохастичним градієнтним спуском.

Наприклад, весь набір вхідних даних X випадковим чином розділено на кілька підмножин: $X1$, $X2$,... Xk, нехай розмір кожної підмножини буде m. Далі алгоритм навчається на кожній із цих підмножин, тобто.

$$w_i' = w_i - \eta \frac{\partial C_{X_j}}{\partial} \qquad (1.9)$$

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

де C_{Xj} – усереднена за множиною Xj функція. Значення ваг і переміщень змінюватимуться після кожного тренування в сетах Xj. Навчальний набір із усіх пакетів даних Xj називається епохою. Коли одна епоха закінчується, дані знову розбиваються на підмножини і для кожної з підмножин проводиться навчання, тобто починається наступна епоха.

Давайте розглянемо можливі труднощі, пов'язані з градієнтним спуском. На (рис. 1.8) зображена функція однієї змінної з кількома локальними мінімумами.

Якщо під час роботи алгоритму градієнтного спуску початкова позиція ваги та переміщення вибрані поблизу одного з локальних мінімумів, існує ймовірність уникнути цього локального мінімуму. Якщо значення цього локального мінімуму значно відрізняється від глобального, результат роботи нейронної мережі може віддалитися від бажаного.

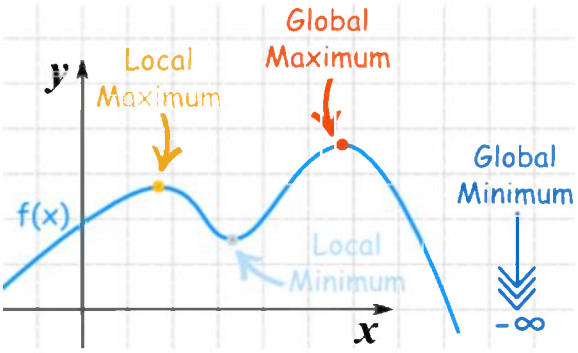


Рисунок 1.8 – Локальний мінімум.

У глибокому навчанні, де функція багатьох змінних має численні локальні мінімуми та сідлові точки, оточені плоскою областю, оптимізація ускладнюється. Тому зазвичай вони згодні шукати значення функції, яке дуже низьке, але не обов'язково є глобальним мінімумом.

Висновки за розділом 1

У світі, де обмін інформацією на різних платформах досягає свого піку, проблема цензури текстового контенту стає крайньою необхідністю. Негативний вплив небажаних висловлювань та образливості викликає загрозу безпеки та порушує нормальний обмін інформацією. Отже, розробка ефективних інструментів для автоматичного виявлення та цензури такого контенту стає актуальною задачею.

Спроби вирішення цієї проблеми через використання нейронних мереж визначаються як перспективні. Ці мережі дозволяють автоматизувати процес

виявлення неприпустимого тексту та забезпечують гнучкість у роботі з різноманітним контентом. Обраний тип нейронної мережі, а саме рекурентні нейронні мережі з підтипом LSTM, обрано з урахуванням їхньої здатності ефективно враховувати контекст та довгострокові залежності в текстових даних. Проте, на фоні всіх переваг такого підходу, вибір нейронних мереж у цензурі текстового контенту викликає і ряд проблем, включаючи етичні, соціальні та технічні:

- Етичні проблеми: зазначений метод може породжувати питання щодо обмеження свободи слова та вибору, а також створювати потенційні етичні дилеми визначення та класифікації контенту.
- Соціальні виклики: важливим аспектом є вплив цензури на соціокультурний ландшафт, включаючи сприйняття та спілкування в суспільстві.
- Технічні складнощі: розробка та підтримка таких моделей потребує постійного вдосконалення, оновлень та адаптації до нових викликів та штучних обхідних методів.

Отже, вирішення цих проблем вимагатиме комплексного підходу, який враховуватиме не лише технічні можливості нейронних мереж, але й їхні соціальні та етичні наслідки.

РОЗДІЛ 2. РОЗРОБКА МОДЕЛІ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ ЦЕНЗУРУВАННЯ ТЕКСТОВИХ ДАНИХ

2.1 Типи нейронних мереж

2.1.1 Персептрон (Perceptron)

Персептрон - це найпростіший вид нейронної мережі, який був винайдений Френком Розенблаттом у 1957 році. Він є основою для багатьох інших типів нейронних мереж.

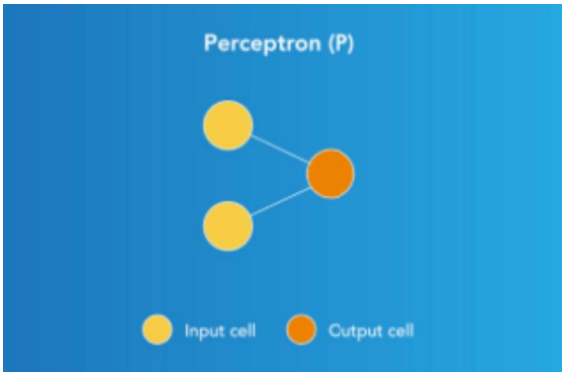


Рисунок 2.1 Персиптрон.

Основна ідея персептрона полягає в тому, що він може взяти вхідні дані, ваги для цих даних, виконати їх лінійну комбінацію та передати результат через функцію активації для вирішення задачі класифікації. Основною функцією активації може бути, наприклад, функція Хевісайда або сигмоїда.

Проте простий персептрон має свої обмеження, такі як неспроможність вирішувати складні проблеми, які включають не лінійні залежності. Це привело до розвитку багатошарових перцептронів (MLP), які складаються з кількох шарів нейронів та використовують функції активації для нелінійного моделювання.

Персептрони та їх розширені форми використовуються в багатьох сферах машинного навчання, особливо в задачах класифікації та регресії.

Переваги

- Простота та ефективність: Персептрони - це прості нейронні мережі, що робить їх ефективними для деяких простих завдань класифікації.

- Легка інтерпретація: оскільки персептрони мають один шар, їх легше інтерпретувати та розуміти.
- Швидкість навчання: в персептронах відсутні складні алгоритми зворотнього поширення помилок, тому вони можуть навчатися швидше, особливо на простих завданнях.

Недоліки

- Обмежена здатність до вирішення складних завдань: Персептрони обмежені у здатності вирішувати завдання, які вимагають розпізнавання складних патернів чи залежностей.
- Неможливість вирішення проблем XOR: найбільш відомий одношаровий персептрон не може вирішити проблему XOR (включаючи інші нелінійні проблеми).
- Схильність до перенавчання: якщо надто складні структури даних, може виникнути проблема перенавчання.
- Відсутність можливості моделювання нелінійних залежностей: Персептрони обмежені у здатності моделювати складні нелінійні залежності в даних.

2.1.2 Багатошаровий персептрон (Multilayer Perceptron)

Точка входу до складних нейронних мереж, де вхідні дані проходять через різні шари штучних нейронів. Кожен окремий вузол з'єднаний з усіма нейронами на наступному рівні, що робить його повністю пов'язаною нейронною мережею. Присутні вхідні та вихідні шари з кількома прихованими шарами, тобто загалом щонайменше три або більше шарів. Він має двонаправлене поширення, тобто пряме поширення та зворотне поширення.

Вхідні дані множаться на вагові коефіцієнти та передаються до функції активації, а при зворотному поширенні вони змінюються, щоб зменшити втрати. Простими словами, ваги - це значення, отримані машиною з нейронних

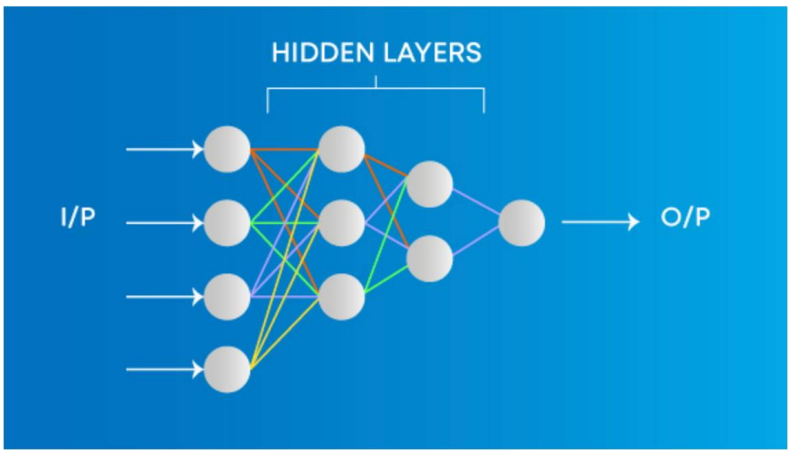


Рисунок 2.2 Багатошаровий персептрон.

мереж. Вони самоналаштовуються залежно від різниці між прогнозованими результатами та навчальними входами. Використовуються нелінійні функції активації, а потім softmax як функція активації вихідного рівня.

Переваги

- Здатність до вивчення складних функцій: MLP може навчатися складнішим завданням, оскільки вони мають більше одного прихованого шару, що дозволяє їм виражати більше вищезазначених функцій.
- Універсальність: за теорією апроксимації універсальність мережі, які мають хоча б один прихований шар, дозволяє MLP наближати будь-яку неперервну функцію.
- Адаптивність: здатність адаптуватися до різних видів даних, таких як текст, зображення, аудіо і т.д.
- Можливість вирішувати проблеми класифікації та регресії: використовується для задач класифікації (наприклад, розпізнавання об'єктів на зображеннях) і регресії (наприклад, прогнозування числових значень).

Недоліки

- Схильність до перенавчання: може виникнути проблема перенавчання, особливо при наявності великої кількості параметрів і обмеженої кількості даних для навчання.

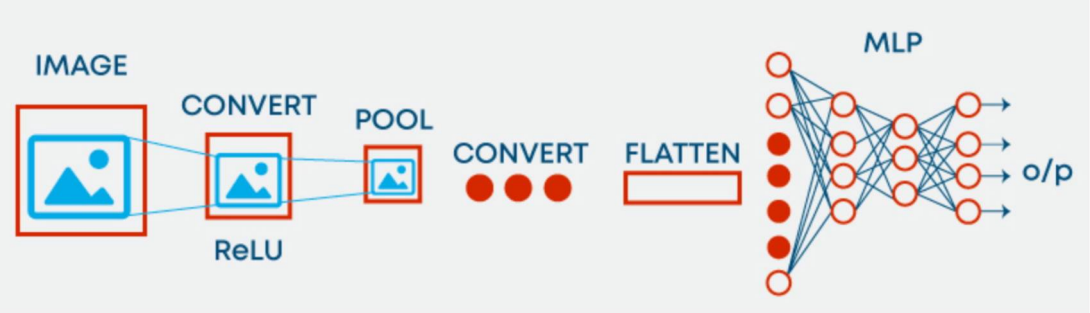


IMAGE
CONVERT
ReLU
POOL
CONVERT
FLATTEN
MLP
o/p

класифікації зображень, як показано на схемі вище. Фільтри використовуються для виділення певних частин зображення. У MLP вхідні дані множаться на ваги та передаються до функції активації. Convolution використовує RELU, а MLP використовує нелінійну функцію активації, за якою слідує softmax. Згорткові нейронні мережі показують дуже ефективні результати в розпізнаванні зображень і відео, семантичному розборі та виявленні перефразів [12].

Дана мережа май наступні переваги

- Розпізнавання образів: висока ефективність у завданнях розпізнавання образів, таких як класифікація зображень.
- Спільне використання параметрів: застосування фільтрів дозволяє спільно використовувати параметри та виявляти особливості в різних частинах зображення.
- Інваріантність до зсувів і масштабувань: згорткові шари можуть реагувати на особливості в зображенні, незалежно від їх точного положення.
- Зменшення просторових розмірів: використовуючи шари пулінгу, CNN зменшують просторові розміри, зберігаючи важливі особливості.
- Робота зі зображеннями в реальному часі: ефективність для обробки зображень у режимі реального часу, наприклад, у системах відеоспостереження чи в автономних автомобілях.

У цієї мережі є наступні недоліки

- Потреба в об'ємних даних: CNN може вимагати великої кількості даних для ефективного навчання, особливо для глибоких мереж.
- Витрати обчислень: Глибокі CNN можуть бути витратними з точки зору обчислень, що може впливати на час навчання та роботу моделі.
- Неявність глобального контексту: у деяких випадках CNN може не враховувати глобальний контекст, оскільки вони фокусуються на локальних особливостях.
- Підгонка до конкретних завдань: глибокі моделі можуть виявитися перенавченими для конкретних завдань та не ефективно вирішувати нові завдання.

2.1.4 Рекурентні нейронні мережі (Recurrent Neural Networks)

Recurrent neural networks, RNN – це мережі типу FFNN, але з особливістю: нейрони отримують інформацію не тільки від попереднього шару, а й від себе від попереднього проходу (рис. 1.5). Це означає, що порядок, в якому подаються дані та навчаєте мережу, стає важливим. Великою складністю мереж RNN є проблема зникаючого (або вибухового) градієнта, яка полягає у швидкій втраті інформації з часом. Звичайно, це впливає лише на ваги, а не стан нейронів, але саме в них накопичується інформація. Зазвичай, мережі такого типу використовуються для автоматичного доповнення інформації.

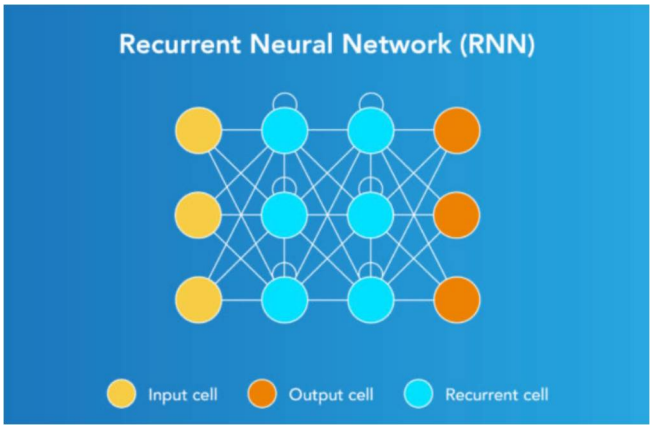


Рисунок 2.4 – Схематичне представлення рекурентних мереж.

Ідея RNN полягає у послідовному використанні інформації. У традиційних нейронних мережах мається на увазі, що всі входи і виходи незалежні. Але для багатьох завдань це не підходить. Якщо, наприклад, треба передбачити наступне слово у реченні, краще враховувати попередні слова. RNN називаються рекурентними, тому що вони виконують одну і ту ж задачу для кожного елемента послідовності, причому вихід залежить від попередніх обчислень. Ще одна інтерпретація RNN – це мережі, які мають «пам'ять», яка враховує попередню інформацію. Теоретично RNN можуть використовувати інформацію у доволі довгих послідовностях, але на практиці вони обмежені лише кількома кроками (докладніше про це пізніше)[13-14].

На рисунку 1.6 показано, що RNN розгортається у повну мережу. Розгорткою ми просто виписуємо мережу для повної послідовності. Наприклад,

якщо послідовність є реченням з 5 слів, розгортка буде складатися з 5 шарів, по шару на кожне слово. Формули, що задають обчислення RNN наступні:

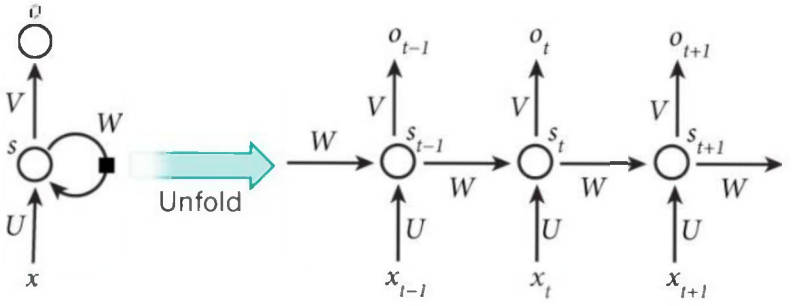


Рисунок 2.5 – Рекурентна нейронна мережа та її розгортка (unfolding).

- x_t – це вхід на часовому кроці t. Наприклад, x_1 може бути вектором з одним гарячим станом (one-hot vector), що відповідає другому слову речення.
- s_t – це прихований стан на кроці t. Це пам'ять мережі. s_t залежить, як функція, від попередніх станів та поточного входу x_t: $s_t=f(Ux_t+Ws_{t-1})$. Функція f зазвичай нелінійна, наприклад, tanh або ReLU. s_{-1}, яке потрібне для обчислення першого прихованого стану, зазвичай ініціалізується нулем (нульовим вектором).
- o_t – це вихід на кроці t. Наприклад, якщо ми хочемо передбачити слово у реченні, вихід може бути вектором ймовірностей у нашому словнику: $o_t = \text{softmax}(Vs_t)$
- Можна інтерпретувати s_t як пам'ять мережі. s_t містить інформацію про те, що сталося на попередніх кроках часу. Вихід o_t обчислюється виключно з урахуванням «пам'яті» s_t. На практиці дещо складніше: s_t не може містити інформацію про занадто велику кількість попередніх кроків;
- На відміну від традиційної глибокої нейронної мережі, яка використовує різні параметри на кожному шарі, RNN має однакові (U, V, W) всіх етапах. Це відбиває той факт, що ми виконуємо те саме завдання на кожному кроці,

використовуючи лише різні входи. Це значно зменшує загальну кількість параметрів, які потрібно підібрати.

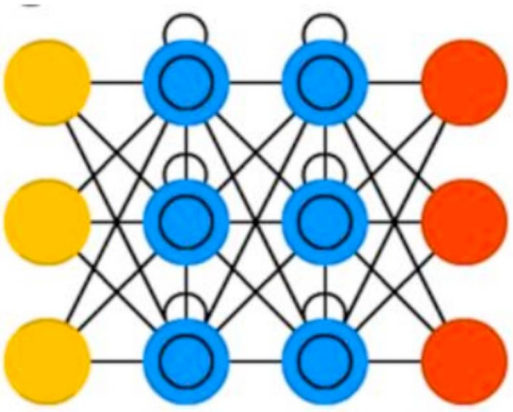
- Діаграма на рисунку 2.5 має виходи на кожному кроці, але, залежно від завдання, вони можуть не знадобитися. Наприклад при визначенні емоційного забарвлення речення, доцільно дбати лише про кінцевий результат, а не забарвлення після кожного слова. Аналогічно, нам може не знадобитися введення даних на кожному кроці. Основною особливістю RNN є прихований стан, який містить деяку інформацію про послідовність.

Переваги цієї мережі

- Обробка послідовностей: здатність ефективно обробляти послідовності даних, такі як текст, часові ряди, аудіо та інші.
- Здатність до врахування контексту: збереження попередніх станів дозволяє RNN враховувати контекст інформації при обробці нових вхідних даних.
- Гнучкість у вхідних та вихідних даних: здатність працювати з різними типами вхідних та вихідних даних, що робить їх універсальними для багатьох завдань.
- Моделювання довготривалих залежностей: можливість зберігати та використовувати інформацію з попередніх моментів часу для моделювання довготривалих залежностей.

Недоліки цієї мережі наступні:

- Проблема зниклих та вибухаючих градієнтів: під час тренування може виникати проблема зниклих або вибухаючих градієнтів, що може призвести до проблем у навчанні.
- Обмежена здатність до моделювання довготривалих залежностей: у деяких випадках RNN може мати проблеми з моделюванням довготривалих залежностей через обмежену пам'ять короткочасної пам'яті.
- Обмежена паралелізація: обчислення у RNN залежать від попередніх кроків, що обмежує можливість паралельної обробки.



- Спроможність утримувати та забувати інформацію: LSTMs мають механізми врегулювання, які дозволяють їм вибирати, яку інформацію тримати та яку забувати, підсилюючи їх здатність зберігати важливі дані.
- Універсальність для різних завдань: LSTMs можна успішно використовувати для різних видів задач, включаючи розпізнавання мови, машинний переклад, генерацію тексту і т. д.
- Можливість паралельного навчання: деякі архітектури LSTMs дозволяють паралельне навчання, що поліпшує ефективність обчислень.

Недоліки

- Витрати обчислень: велика кількість параметрів і обчислень, зокрема в глибоких LSTM, може призвести до збільшення обчислювальних витрат.
- Велика кількість гіперпараметрів: підбір правильних гіперпараметрів для LSTM може бути витратним завданням і вимагати багато експериментів.
- Потреба в великій кількості даних: щоб досягти ефективного навчання, LSTMs можуть вимагати великої кількості даних, особливо в глибоких моделях.

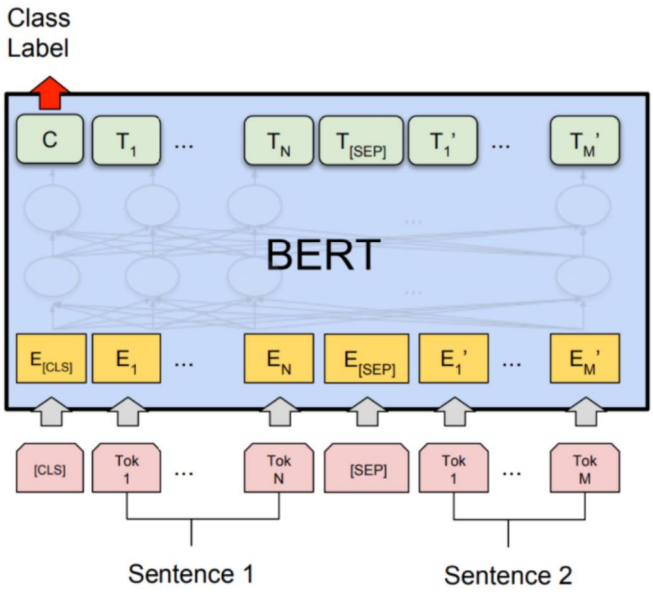
2.2 Моделі нейронних мереж для обробки природної мови

Обробка природної мови (Natural Language Processing - NLP) є важливою галуззю, і для цього використовуються різні моделі нейронних мереж.

Цей розділ призначений для дослідження сучасних моделей та підходів до обробки природної мови, які знаходять своє застосування у вирішенні завдань цензури текстових даних. Заснований на передових досягненнях у галузі NLP та нейронних мереж, розглянемо різноманітні стратегії, що використовуються для ефективного виявлення та обробки природної мови.

2.2.1 BERT (Bidirectional Encoder Representations from Transformers):

BERT — це платформа машинного навчання з відкритим кодом для обробки природної мови (NLP). BERT розроблено, щоб допомогти комп'ютерам зрозуміти значення неоднозначної мови в тексті, використовуючи навколишній


Class
Label
C
T1
...
TN
T[SEP]
T1'
...
TM'
BERT
E[CLS]
E1
...
EN
E[SEP]
E1'
...
EM'
[CLS]
Tok
1
...
Tok
N
[SEP]
Tok
1
...
Tok
M
Sentence 1
Sentence 2

Однак BERT пройшов попередню підготовку, використовуючи лише корпус звичайного тексту без міток (а саме всю англійську Вікіпедію та Корпус Брауна). Він продовжує навчатися без нагляду з тексту без міток і вдосконалюватися, навіть якщо його використовують у практичних програмах (наприклад, пошук Google). Його попередня підготовка служить базовим рівнем «знань», на якому можна будувати. З цього моменту BERT може адаптуватися до постійно зростаючої маси пошукового вмісту та запитів і бути налаштованим відповідно до специфікацій користувача. Цей процес відомий як трансферне навчання.[16]

Як згадувалося вище, BERT стало можливим завдяки дослідженням Google щодо Transformers. Трансформатор є частиною моделі, яка надає BERT підвищену здатність розуміти контекст і неоднозначність мови. Трансформатор робить це, обробляючи будь-яке задане слово по відношенню до всіх інших слів у реченні, а не обробляючи їх по одному. Переглядаючи всі навколишні слова, Transformer дозволяє моделі BERT зрозуміти повний контекст слова, а отже, краще зрозуміти наміри шукача.

Це контрастує з традиційним методом обробки мови, відомим як вбудовування слів, у якому попередні моделі, такі як GloVe та word2vec, відображали кожне окреме слово у вектор, який представляє лише один вимір, фрагмент, значення цього слова.

Ці моделі вбудовування слів вимагають великих наборів даних із мітками. Хоча вони вправно справляються з багатьма загальними завданнями НЛП, вони не справляються з важким контекстом, передбачуваним характером відповідей на запитання, оскільки всі слова в певному сенсі закріплені за вектором або значенням. BERT використовує метод моделювання замаскованої мови, щоб утримати слово у фокусі від того, щоб воно «бачило себе» — тобто мало фіксоване значення незалежно від контексту. Тоді BERT змушений ідентифікувати замасковане слово лише на основі контексту. У BERT слова визначаються їх оточенням, а не попередньо фіксованою ідентичністю. За словами англійського лінгвіста Джона Руперта Ферта: «Ви дізнаєтеся слово по компанії, яку воно веде» [17].

2.2.2 Модель GPT (Generative Pre-trained Transformer):

GPT, або Generative Pre-trained Transformer, — це найсучасніша мовна модель, розроблена OpenAI. Він використовує методи глибокого навчання для створення тексту природною мовою, наприклад статей, історій або навіть розмов, які дуже нагадують текст, написаний людиною.

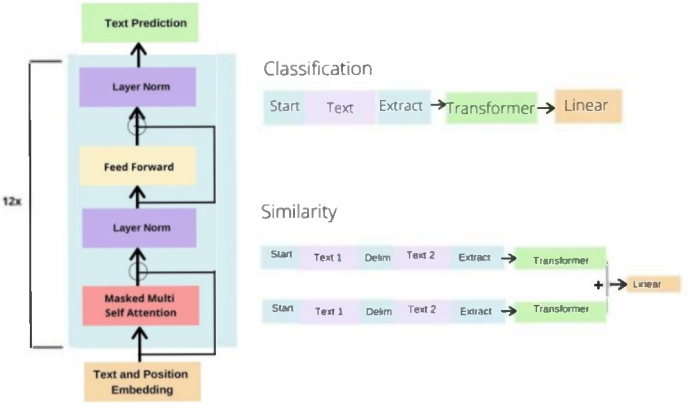


Рисунок 2.8 Схематичне зображення GPT.

GPT був представлений у 2018 році як частина серії мовних моделей на основі трансформаторів, розроблених OpenAI. Його архітектура базується на трансформаторі, моделі нейронної мережі, яка використовує самоувагу для обробки вхідних послідовностей. На відміну від традиційних рекурентних нейронних мереж, трансформатори можуть обробляти вхідні дані паралельно, що робить їх швидшими та ефективнішими.

Основною ідеєю GPT є попереднє навчання. Попереднє навчання — це техніка, яка використовується в глибокому навчанні, яка передбачає навчання моделі на великій кількості даних перед її тонким налаштуванням для виконання конкретного завдання. У випадку GPT модель попередньо навчається на величезній кількості текстових даних, таких як книги, статті та веб-сторінки, щоб вивчати статистичні моделі та структури природної мови. Цей етап попереднього навчання є критично важливим, оскільки він дозволяє моделі розвинути загальне розуміння мови, яке можна застосовувати до різних завдань. Після попереднього навчання модель точно налаштована на певні мовні завдання, такі як мовний переклад, відповіді на запитання чи резюмування,

шляхом додавання шарів вихідних даних для конкретних завдань і точного налаштування ваг попередньо навченої моделі на завдання даних. Етап тонкого налаштування дозволяє адаптувати модель до конкретних нюансів і вимог завдання, водночас використовуючи загальні знання мови, отримані під час попереднього навчання.

Однією з найвидатніших особливостей GPT є його здатність генерувати зв'язний і релевантний контексту текст. Це досягається завдяки використанню механізмів самоконтролю, які дозволяють зважувати важливість різних частин вхідної послідовності під час генерації вихідного тексту. Механізми самоконтролю також дозволяють GPT фіксувати контекст і залежності між різними словами та реченнями, що робить його добре придатним для завдань, які передбачають створення довших текстових послідовностей, таких як статті чи історії.

GPT досяг найсучаснішої продуктивності в різноманітних завданнях обробки природної мови, таких як моделювання мови, відповіді на запитання та класифікація тексту. Його також використовували для створення реалістичних розмов між людьми та чат-ботами та навіть для написання переконливих новинних статей і історій [18].

2.3 Вибір типу нейронної мережі

Вибір типу нейронної мережі є критичним етапом у розробці моделі для цензурування текстових даних. У даному випадку, рекурентні нейронні мережі (RNN) а саме підтип LSTM (Long Short-Term Memory) були обрані з ряду важливих причин, а саме.

Обробка послідовностей: однією з ключових особливостей текстових даних є їхній послідовний характер. Відзначаючись тим, що слова та символи розташовані в конкретному порядку, текст вимагає уваги до взаємозв'язків між елементами послідовності. Рекурентні нейронні мережі (RNN) є ідеальним інструментом для опрацювання таких даних.

RNN мають здатність враховувати контекст попередніх елементів при аналізі поточного елемента послідовності. У контексті цензури текстових даних це дозволяє моделі не лише виявляти конкретні слова чи фрази, але і здатність аналізувати їхнє значення в залежності від контексту.

Наприклад, для правильного визначення неприпустимого висловлювання, модель повинна розуміти, що певне слово чи вираз може бути образливим або неприйнятним лише в певному контексті. RNN, використовуючи свою здатність до обробки послідовностей, можуть ефективно враховувати цей аспект та забезпечувати точну класифікацію текстових даних з урахуванням їхнього послідовного устрою.

Урахування контексту: тексти, як правило, несуть значення не лише в окремих словах чи фразах, але й у їхньому взаємодії та вкладеності в речення або абзац. У випадку цензурування текстових даних важливо враховувати не лише самі слова, але й контекст, у якому вони вживаються. Рекурентні нейронні мережі (RNN), і зокрема LSTM, проявляють високу ефективність у вирішенні цього завдання.

LSTM відмінно справляються із здатністю зберігати та передавати інформацію на різних рівнях послідовності. Завдяки механізму "воріт" (gate mechanism), вони можуть вирізняти ключові аспекти контексту та враховувати їх в подальших етапах обробки тексту. Такий підхід дозволяє більш точно аналізувати не лише окремі слова, а й їхнє значення в конкретному контексті, що робить LSTM ідеальним вибором для розв'язання завдань цензурування, де контекст має величезне значення.

Здатність до роботи з різними довжинами послідовностей: різноманітність текстових даних полягає в їхній різній довжині та обсягу. Під час розробки моделі для цензурування текстів важливо мати архітектуру, яка може ефективно працювати з послідовностями різної довжини. Рекурентні нейронні мережі (RNN), включаючи LSTM, проявляють вражаючу гнучкість в цьому відношенні.

Основна перевага RNN полягає в їхній здатності адаптуватися до довільної довжини вхідних послідовностей. При роботі з текстовими даними,

де кількість слів чи символів може варіюватися, це важливо для забезпечення оптимального функціонування моделі на текстах різних розмірів. LSTM, зокрема, вміють ефективно управляти різною довжиною послідовностей, допомагаючи зберігати та використовувати інформацію з урахуванням конкретних властивостей кожного тексту. Ця здатність гарантує, що модель залишається гнучкою та адаптованою до широкого спектру текстових вхідних даних.

Зменшення проблеми зниклих градієнтів: проблема зниклих градієнтів стає особливо актуальною при тренуванні глибоких нейронних мереж, коли під час зворотного поширення помилки градієнти можуть зменшуватися експоненційно, або навіть зникають, що ускладнює оновлення ваг моделі. Рекурентні нейронні мережі (RNN) допомагають зменшити цю проблему, забезпечуючи ефективну роботу з послідовністю даних.

У випадку цензурування текстових даних, де потрібно уважно враховувати контекст та завдання виявлення неприпустимого контенту, важливо мати модель, яка може адекватно пристосовуватися під час тренування. Використання RNN дозволяє зменшити вплив проблеми зниклих градієнтів завдяки здатності моделі враховувати попередні елементи послідовності та коректно передавати інформацію через тривалі відстані. В особливості, використання LSTM дозволяє вдало керувати градієнтами через довгострокові залежності, що покращує стабільність і швидкість навчання моделі. Це робить RNN, зокрема LSTM, ефективним вибором для завдань обробки тексту та виявлення неприпустимого вмісту.

Застосування в аналізі тексту: оскільки завдання цензурування текстових даних передбачає високий рівень розуміння семантики та контексту, вибір рекурентних нейронних мереж (RNN), зокрема Long Short-Term Memory (LSTM), обумовлений їхньою ефективністю у вирішенні таких завдань.

RNN здатні ефективно враховувати довгострокові залежності між елементами послідовності, що є ключовим аспектом при аналізі текстових даних. Контекстуальне розуміння слів та фраз у тексті вимагає урахування їхньої взаємодії та взаємозалежності в реченні чи абзаці.

LSTM, завдяки своїй здатності зберігати та враховувати інформацію на тривалий термін, стають ефективним інструментом для виявлення та аналізу семантичних взаємозв'язків у тексті. Враховуючи, що завдання цензурування текстових даних вимагає не тільки виявлення окремих слів чи виразів, але і їхнє розуміння в контексті, використання RNN, особливо LSTM, дозволяє створювати моделі, здатні збагачувати аналіз тексту та приймати розсудливі рішення щодо класифікації вмісту.

2.4 Архітектура нейронної мережі

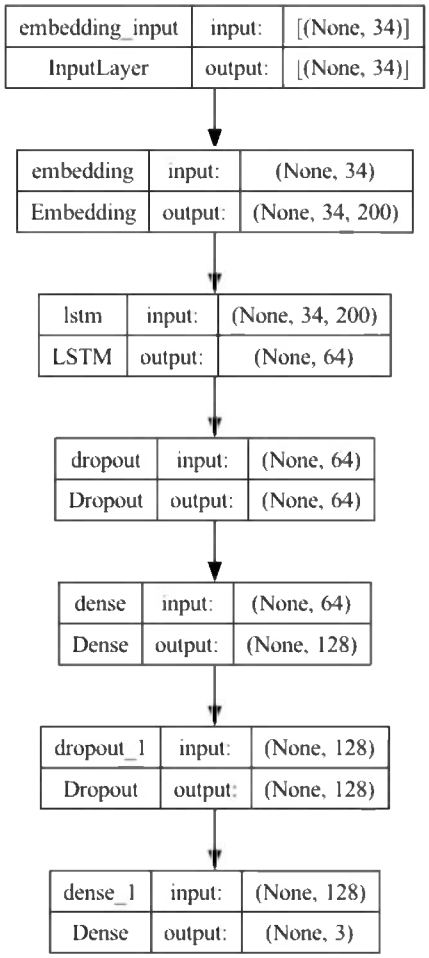


Рисунок 2.9. Архітектура нейронної мережі.

Embedding Layer

Опис: це перший шар у нейронній мережі, призначений для вбудовування слів або токенів у векторний простір фіксованої розмірності. Його основна мета навчити нейронну мережу представленню слів, де семантично схожі слова мають близькі векторні представлення.

Як працює

- Кожне слово або токен у вхідному тексті призначається унікальному числовому ідентифікатору відповідно до словника.
- Кожен цей ідентифікатор перетворюється в вектор фіксованої розмірності за допомогою матриці вбудовування.
- Ця матриця вбудовування навчається під час тренування моделі так, щоб вектори, які вона надає для слів, були такими, щоб схожі слова були близькими одне до одного в цьому векторному просторі.

Переваги

- Embedding дозволяє нейронній мережі розуміти семантичні відносини між словами
- Зменшує розмірність вхідного простору, зберігаючи при цьому важливі семантичні відносини.

LSTM Layer (Long Short-Term Memory)

Опис**:** є типом рекурентного шару, спроектованим для ефективної роботи з проблемами довготривалих залежностей та уникнення проблеми зниклої та затухаючої градієнтів у рекурентних нейронних мережах.

Як працює:

- LSTM має внутрішню структуру, яка дозволяє їй запам'ятовувати та забувати інформацію на тривалий час.
- У кожному кроці часу LSTM приймає вхід, генерує вихід та оновлює свій внутрішній стан.
- Він використовує тривалий та короткотерміновий пам'ятник для ефективного керування інформацією.

Структура LSTM:

- Внутрішній стан LSTM складається з двох компонентів - короткотермінової пам'яті (СТ) та тривалої пам'яті (LT).
- Кожен нейрон у LSTM приймає вхід та генерує вихід на основі поточного введення, попереднього виходу та попереднього внутрішнього стану.
- Ці механізми дозволяють LSTM ефективно працювати з послідовностями та уникати проблем градієнтів у порівнянні з традиційними рекурентними мережами.

Переваги: LSTM є потужним інструментом для роботи з послідовностями та дуже ефективним для завдань обробки природної мови, де важливий контекст та залежності між словами.

Dropout Layer:

Опис: це техніка регуляризації, яка використовується для запобігання перенавчанню в нейронних мережах. Dropout Layer випадковим чином вимикає (анулює) вихідні сигнали нейронів під час тренування з певною ймовірністю, щоб зменшити залежність між конкретними нейронами та поліпшити генералізацію моделі.

Як працює:

- Під час тренування для кожного вхідного батча Dropout Layer випадковим чином вимикає (встановлює в нуль) вихідні сигнали нейронів з ймовірністю rate.
- Це призводить до того, що модель навчається не залежати виключно від певних нейронів, що допомагає уникнути перенавчання.
- Під час валідації або тестування всі нейрони залишаються активними (без випадкового відключення) для коректного передбачення.

Важливість Dropout

- Зменшує перенавчання: запобігає тому, щоб модель навчалася шуму чи випадковим закономірностям тренувальних даних.
- Поліпшує генералізацію: допомагає моделі краще узагальнити на нові, раніше невидані дані.

Dense Layers

Опис: Dense Layer, або повністю з'єднаний шар, є базовим шаром у багатьох нейронних мережах. Всі нейрони в Dense Layer пов'язані з кожним нейроном попереднього та наступного шару, утворюючи "повністю з'єднану" структуру.

Як працює:

- Кожен нейрон у Dense Layer приймає вхідні сигнали від усіх нейронів попереднього шару.
- Кожен вхід має вагу, яка оптимізується під час тренування.
- Виходить сума зважених входів, яка потім піддається функції активації.

Важливість Dense Layers:

- Пов'язаність нейронів: Кожен нейрон взаємодіє з усіма нейронами попереднього та наступного шару, що дозволяє моделі вивчати складні взаємозв'язки в даних.
- Здатність до нелінійності: Використання функцій активації, таких як ReLU, дозволяє моделі вивчати нелінійні залежності у даних.

Висновок за розділом 2

У цьому розділі ми детально розглянули різноманітні типи нейронних мереж, спрямованих на розробку моделі для цензурування текстових даних. Від персептронів до передових трансформерів, ми провели аналіз архітектур, що відкривають нові перспективи у сфері обробки природної мови (NLP).

Персептрони та багатошарові персептрони стали основою для багатьох нейронних мереж, спрямованих на класифікацію. Згорткові нейронні мережі (CNN) виявляють особливості у тексті, рекурентні нейронні мережі (RNN) та мережі з довгою короткостроковою пам'яттю (LSTM) ураховують послідовності та довготривалі залежності.

Також було розглянуто новітні моделі для обробки природної мови такі як BERT та GPT, представники передових трансформерів, відзначаються здатністю

враховувати контекст та семантику слова в обох напрямках, відкриваючи нові можливості для цензури тексту.

Вибір типу нейронної мережі та архітектури є ключовим етапом у розробці моделі для цензури текстових даних. Обрані рекурентні нейронні мережі, зокрема LSTM, виправдали свій вибір завдяки здатності ефективно обробляти послідовності, ураховувати контекст та зменшувати проблему зниклих градієнтів. Це робить їх оптимальним вибором для завдань цензурування та фільтрації текстового контенту.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір даних для навчання

Першим етапом у розробці моделі було уважне відібрання та підготовка наборів даних для тренування. Одним із ключових джерел інформації став набір даних з Twitter, спрямований на виявлення ворожих висловлювань. Текстові дані в цьому наборі класифікуються за кількома категоріями, включаючи мову ненависті, образливу лексику та відсутність образливого змісту. Важливо враховувати, що зміст цього набору даних може включати расистські, сексистські, гомофобні або загалом образливі висловлювання, роблячи його викликливим і суперечливим для обробки.

	count	hate_speech	offensive_language	neither	class	tweet
0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your house. & as a man you should always take the trash out...
1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn bad for cuffin dat hoe in the 1st place!!
2	3	0	3	0	1	!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby4life: You ever fuck a bitch and she start to cry? You be confused as shit
3	3	0	2	1	1	!!!!!!!! RT @C_G_Anderson: @viva_based she look like a tranny
4	6	0	6	0	1	!!!!!!!!!!!! RT @ShenikaRoberts: The shit you hear about me might be true or it might be faker than the bitch who told it to ya 
5	3	1	2	0	1	!!!!!!!!!!!!!!!!!"@T_Madison_x: The shit just blows me..claim you so faithful and down for somebody but still fucking with hoes! 😂😂😂"
6	3	0	3	0	1	!!!!!"@__BrighterDays: I can not just sit up and HATE on another bitch .. I got too much shit going on!'
7	3	0	3	0	1	!!!“@selfiequeenbri: cause I'm tired of you big bitches coming for us skinny girls!!”
8	3	0	3	0	1	" & you might not get ya bitch back & thats that "
9	3	1	2	0	1	" @rhythmixx_ :hobbies include: fighting Mariam" bitch
10	3	0	3	0	1	" Keeks is a bitch she curves everyone " lol I walked into a conversation like this. Smh
11	3	0	3	0	1	" Murda Gang bitch its Gang Land "
12	3	0	2	1	1	" So hoes that smoke are losers ? " yea ... go on IG
13	3	0	3	0	1	" bad bitches is the only thing that i like "
14	3	0	2	0	1	" bitch get up off me "

Рисунок 3.1 Приклад датасету.

Для розробки та навчання моделі цензурування текстових даних був використаний набір даних із використанням декількох класів текстових даних. Цей набір даних використовується для вивчення виявлення ворожих висловлювань у текстових повідомленнях на платформі Twitter.

Характеристики набору даних:

- count: кількість користувачів, які кодували кожен твіт. Мінімальне значення - 3, але іноді було більше користувачів, коли оцінки визначалися ненадійними.
- hate_speech: кількість користувачів, які визначили твіт як "мова ненависті".
- offensive_language: кількість користувачів, які визначили твіт як "образливу мову".

- neither: кількість користувачів, які визначили твіт як "ні те ні інше".
- class: класифікаційна мітка для більшості користувачів. Значення 0 вказує на мову ненависті, 1 - на образливу мову, 2 - на те, що "ні те ні інше".
- tweet: оригінальний текстовий запис твіту.

Цей набір даних містить текст, який може бути вважати расистським, сексистським, гомофобним або загалом образливим. З огляду на характер цього дослідження та специфіку завдання цензурування тексту, важливо було врахувати різноманіття висловлення та можливість виявлення неприпустимого контенту в текстових даних.

3.2 Вибір мови програмування для програмної реалізації

Розробка моделі нейронної мережі для цензорування текстових даних може бути виконана за допомогою різних мов програмування та фреймворків для глибокого навчання.

- Python: це високорівнева мова програмування, яка визначається своєю простотою та читабельністю коду. Завдяки своїй широкій спільноті та великому екосистему бібліотек, Python став однією з найпопулярніших мов для розробки програмного забезпечення та застосувань у сферах науки про дані, штучного інтелекту та машинного навчання.У контексті розробки моделей нейронних мереж для цензорування текстових даних, Python використовується зокрема через такі бібліотеки, як TensorFlow та PyTorch. Ці бібліотеки надають високорівневі API для розробки, навчання та впровадження нейронних мереж [19].

  Наприклад, TensorFlow надає Keras API, який є високорівневим інтерфейсом для конструювання та тренування нейронних мереж. PyTorch також володіє простим та гнучким інтерфейсом для роботи з нейронними мережами. Обидві бібліотеки дозволяють легко використовувати передні плани для обробки текстових даних та реалізації завдань цензури.
- JavaScript: є високорівневою, об'єктно-орієнтованою мовою програмування, яка використовується для розробки веб-додатків та взаємодії з

користувачем на стороні клієнта. За допомогою JavaScript можна створювати динамічний та інтерактивний контент на веб-сайтах. Що стосується використання JavaScript для розробки моделей нейронних мереж, то бібліотека TensorFlow.js надає можливість реалізації нейронних мереж безпосередньо в браузері за допомогою JavaScript. TensorFlow.js дозволяє навчати моделі, виконувати їх на стороні клієнта та інтегрувати їх у веб-додатки. Зокрема, в TensorFlow.js є інструменти для навчання моделей машинного навчання, включаючи нейронні мережі, і вони можуть бути використані для обробки текстових даних та цензурування контенту.[19]

- Java: це об'єктно-орієнтована, високорівнева мова програмування, яка відзначається своєю платформенною незалежністю та широким використанням у великих корпоративних системах. Вона володіє великою спільнотою розробників та має ряд фреймворків для розробки різноманітних застосувань. У контексті розробки моделей нейронних мереж для цензорування текстових даних, Java може бути використана за допомогою бібліотек, таких як Deeplearning4j, яка надає інструменти для розробки та навчання нейронних мереж. Deeplearning4j дозволяє створювати складні моделі, використовуючи різноманітні шари та оптимізатори.
- R: це мова програмування та середовище статистичного аналізу та візуалізації даних. Вона часто використовується у наукових дослідженнях, аналізі даних та машинному навчанні. Для розробки моделей нейронних мереж для цензорування текстових даних можна використовувати різні бібліотеки та фреймворки, доступні для R. Однією з популярних бібліотек для нейронних мереж у R є Keras. Keras - це високорівневий інтерфейс для роботи з нейронними мережами, який працює поверх фреймворків, таких як TensorFlow та Theano. Використовуючи Keras в R, розробник може швидко створювати, навчати та оцінювати моделі нейронних мереж [20-21].

Python виявляється найбільш оптимальним вибором для розробки моделей нейронних мереж для цензорування текстових даних. Завдяки своєму багатому екосистемі, Python пропонує широкий набір бібліотек та фреймворків, таких як TensorFlow, Keras і PyTorch, що робить розробку моделей ефективною

та доступною. Простий синтаксис мови полегшує розуміння коду, особливо для розробників з різним рівнем досвіду. Активна спільнота розробників існує для підтримки та обміну досвідом, забезпечуючи велику кількість ресурсів та документації.

3.3 Розробка програмної реалізації

3.3.1 Попередня обробка даних

Важливим аспектом була попередня обробка даних з метою забезпечення якості даних для навчання та точності моделі. Це включало у себе видалення зайвих символів, обробку мовленнєвих елементів, таких як url-адреси чи імена користувачів, та інші прийоми попередньої обробки тексту.

```
Твіт з датасету: @SportsCenter: Eli Manning just threw his NFL-leading 27th interception of the season."

Твіт після обробки: user Eli Manning just threw his NFL-leading 27th interception of the season.
```

Рисунок 3.2 Приклад попередньої обробки.

1. Заміна користувацьких тегів.

   У тексті, що містить твіти, можуть зустрічатися різноманітні користувацькі теги, які вказують на імена користувачів. Заміна цих тегів загальним терміном "user" стандартизує текст та усуває індивідуальні імена, які, ймовірно, не мають суттєвого значення для завдання цензурування. Наприклад, заміна "@JohnDoe" на "user".

   Це дозволяє моделі фокусуватися на суттєвому змісті та зменшує кількість унікальних слів, що може полегшити процес тренування та покращити ефективність моделі при аналізі тексту.

   Такий підхід особливо корисний, оскільки імена користувачів можуть бути дуже різноманітні та нести у собі велику кількість інформації про конкретну особу, що може бути несуттєвим для завдання цензурування тексту.

2. Вилучення URL.

   У текстах твітів можуть міститися гіперпосилання (URL), які вказують на зовнішні ресурси. Вилучення URL допомагає уникнути шуму та фокусуватися саме на текстовому вмісті твіту при аналізі та класифікації.

   Наприклад, перетворення тексту "Check out this amazing article: *https://example.com*" на "Check out this amazing article" спрощує текст і призводить його до більш однорідного формату. Вилучення URL забезпечує стандартизацію текстового контенту та полегшує роботу моделі при визначенні неприпустимого або образливого змісту.

   Цей крок також допомагає уникнути того, щоб модель надто зверталася до зовнішніх ресурсів, що може бути небезпечно з точки зору безпеки та конфіденційності.

3. Видалення HTML-сутностей.

   Текстовий контент з Twitter часто містить HTML-сутності, такі як *&*, *<*, *>*, що представляють символи та теги HTML. Ці сутності можуть виникнути в результаті конвертації спеціальних символів для відображення на веб-сторінці.

   Видалення HTML-сутностей є важливим етапом підготовки даних, оскільки вони не несуть семантичного навантаження для завдання класифікації тексту та можуть створювати додатковий шум у наборі даних. Наприклад, конвертація *&* до звичайного амперсанду (&) або *<* та *>* до відповідно < та > допомагає підвищити читабельність та спростити текст.

   Видалення HTML-сутностей допомагає забезпечити консистентність та чистоту текстових даних, що є важливим для ефективної роботи моделі під час навчання та класифікації.

4. Видалення непотрібних символів.

   У цьому етапі проводилася обробка тексту з метою вилучення непотрібних символів, які не несуть значущої інформації для завдання цензурування текстових даних. До цих символів можуть відноситися різноманітні лапки, апострофи, окликати, подвійні крапки тощо.

Вилучення цих символів спрямоване на створення чистого та легкого для обробки текстового корпусу. Наприклад, конвертація подвійних крапок *(..)* до одинарної крапки *(.)* допомагає уніфікувати представлення тексту та сприяє коректній обробці з боку моделі.

Даний підхід спрямований на вирішення проблеми шуму в даних, що може виникнути внаслідок введення різноманітних символів, що не несуть семантичного навантаження для задачі цензурування. Такий підхід сприяє покращенню ефективності та точності моделі.

3.3.2 Токенізація та трансформація тексту в числові послідовності

Токенізація - це етап обробки тексту, під час якого вхідний текст розділяється на окремі одиниці, так звані токени. Токени можуть бути словами, фразами або іншими одиницями тексту. В контексті цензурування токенізація допомагає розділити текстові фрази на окремі слова, що стає важливим етапом перед подальшим перетворенням тексту для використання в нейронній мережі. Токенізація може бути виконана різними способами. Зазвичай вона включає в себе видалення зайвих символів, поділ тексту на окремі слова та створення токенів. Наприклад, речення «Це речення для цензурування» може бути розділене на токени: ["Це", "речення", "для", "цензурування"].

Токенізація важлива для того, щоб нейронна мережа могла ефективно обробляти текст, враховуючи окремі слова чи фрази як окремі одиниці, які можна аналізувати та класифікувати.

Після токенізації, текстові токени потрібно перетворити в числові значення, які можуть бути подані нейронною мережею. Цей етап називається трансформацією тексту в числові послідовності. Цей етап дозволяє побудувати числовий вектор для кожного текстового фрагменту, що є необхідним для подальшого навчання нейронної мережі.

```
Cleaned dataset: Funny thing isits not just the people doing it. Its the people who seeing these pics and judging the birds. Just as wrong.

Transform to number: [312, 245, 8438, 47, 35, 6, 109, 277, 25, 58, 6, 109, 74, 887, 40, 630, 9, 3790, 6, 195, 35, 73, 319]
```

Рисунок 3.3 Приклад трансформації тексту в числові послідовності.

```
num of tweets: (24783, 7)
num test tweet: 4957
num train tweet: 19826
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 34, 200)           3745200

 lstm (LSTM)                 (None, 64)                67840

 dropout (Dropout)           (None, 64)                0

 dense (Dense)               (None, 128)               8320

 dropout_1 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 3)                 387

=================================================================
Total params: 3821747 (14.58 MB)
Trainable params: 3821747 (14.58 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Після створення архітектури моделі, наступний крок - це її компіляція, тобто конфігурація процесу навчання. Компіляція включає в себе вибір оптимізатора, функції втрат та метрик, які використовуються для оцінки продуктивності моделі під час навчання.

Оптимізатор (Optimizer): це алгоритм, який визначає, яким чином модель адаптує свої ваги в процесі навчання, з метою мінімізації функції втрат. Оптимальний вибір оптимізатора може вплинути на швидкість навчання та здатність моделі знаходити глобальний мінімум.

Adam (Adaptive Moment Estimation): Adam є популярним оптимізатором, особливо ефективним для нейронних мереж. Він комбінує ідеї з двох інших оптимізаторів: "RMSprop" і "Momentum".

Основні переваги Adam включають:

- адаптивність до швидкості навчання: Adam використовує адаптивний розмір кроку для кожного параметра, що дозволяє зберігати ваги актуальними для різних параметрів.
- ефективність в умовах великої кількості даних: Adam добре справляється з великими обсягами даних та підходить для роботи з нейронними мережами в складних завданнях.
- можливість обробки шуму та різниці в швидкості навчання. Алгоритм враховує статистику першого та другого моментів градієнтів для ефективної адаптації до різниці в швидкості навчання та обробки шуму в даних.

Загалом, Adam є добре збалансованим та ефективним оптимізатором для широкого спектру задач машинного навчання, що включає в себе багатошарові нейронні мережі.

Функція втрат (Loss Function): Функція втрат - це метрика, яка визначає, наскільки вдало модель вирішує завдання. У випадку класифікації текстів, як у нашому випадку, часто використовується функція втрати "categorical_crossentropy".

Categorical Crossentropy: Ця функція визначає розріз між розподілом ймовірностей, передбаченим моделлю, та розподілом ймовірностей істинних класів. Вона штрафує модель за невірні прогнози, зокрема враховуючи

ймовірності віднесення кожного прикладу до правильного класу.

Важливі аспекти

- Категорійна класифікація: ця функція призначена для задач, де кожен приклад може належати лише одному класу (наприклад, "позитивний", "негативний", "нейтральний").
- Ймовірність віднесення до класів: функція враховує ймовірність приналежності кожному класу для кожного прикладу, що робить її популярним вибором для багатокласової класифікації.
- Мінімізація розриву між прогнозами та істинними значеннями: модель прагнеться мінімізувати цей розрив, щоб покращити свою точність та відповідність даним.

Використання "categorical_crossentropy" у комбінації з оптимізатором Adam допомагає забезпечити ефективну оптимізацію та вирішення проблеми класифікації текстів у нашій моделі.

Метрики (Metrics): метрики визначають, як ефективно модель вирішує задачу. У нашому випадку ми додаємо метрики, такі як точність (accuracy), чутливість (recall), точність (precision) та F1-показник. Кожна з цих метрик надає різний погляд на продуктивність моделі.

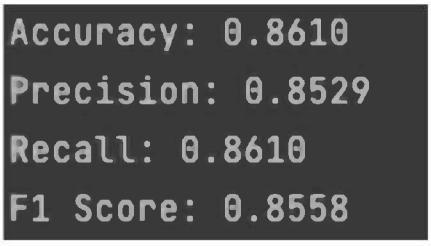


Рисунок 3.5 Результати метрик (accuracy, precision, F1, recall) з точністю до 4 знаків.

- Точність (Accuracy): відношення кількості правильно класифікованих прикладів до загальної кількості прикладів. Визначає загальну ефективність моделі. Важлива в контексті забезпечення загальної точності класифікації.

- Чутливість (Recall): спроможність моделі виявляти всі дійсно позитивні екземпляри. Важлива для виявлення неприпустимого контенту. Ми хочемо уникнути пропуску неприпустимих випадків.
- Точність (Precision): точність визначає, яка частина прикладів, які модель класифікувала як певний клас, дійсно належить до цього класу. Точність є важливою метрикою в ситуаціях, коли неприпустимо помилитися, класифікуючи безпечний контент як неприпустимий. Висока точність означає менше ложно-позитивних результатів, що є ключовим аспектом для систем цензурування текстових даних.
- F1-показник (F1 Score): співвідношення точності і чутливості. Корисний в ситуаціях, коли є дисбаланс між класами, оскільки враховує обидві метрики в одне число.

Функція активації (Activation Function): Функція активації — це математична операція, яка вводить нелінійність в виходи нейронів мережі. Вона дозволяє моделі вирішувати більш складні завдання, які не можна було б вирішити лінійними методами. В рекурентних нейронних мережах (RNN), зокрема LSTM, активаційна функція використовується для контролю потоку інформації, яка проходить через мережу.

У коді, при описі LSTM шару, використовується активаційну функцію "softmax" на останньому шарі. Softmax часто використовується в задачах класифікації, оскільки вона перетворює виходи мережі на ймовірності, що додаються до 1. Це дозволяє вам отримати ймовірнісний розподіл для класів.

Проте, важливо відзначити, що для різних задач і архітектур може бути вигідним використання різних функцій активації. Наприклад, для прихованих шарів у внутрішніх частинах мережі LSTM часто використовується гіперболічний тангенс (tanh) або ReLU (Rectified Linear Unit).

3.3.4 Навчання та валідація

Після відібрання та підготовки наборів даних, модель переходить до етапу навчання. У цьому етапі використовується тренувальний набір даних, і модель намагається оптимізувати свої параметри за допомогою методу градієнтного

спуску та зворотнього поширення помилки. Тренування включає в себе подачу текстових даних в мережу, оцінювання виходів, порівняння їх з очікуваними значеннями та коригування ваг моделі для покращення її точності.

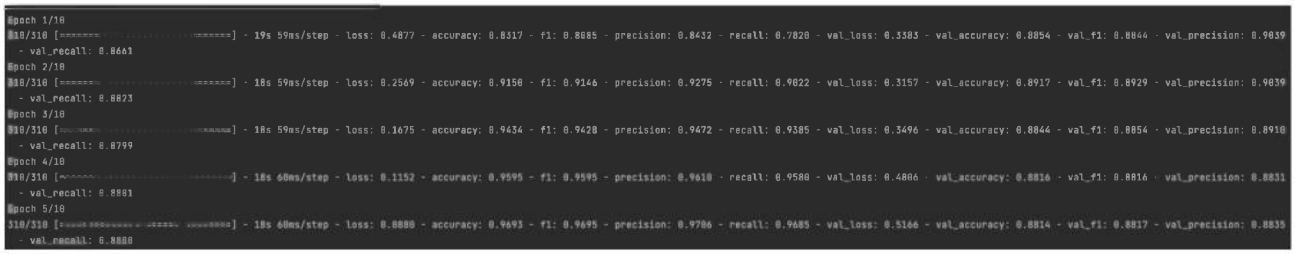


Рисунок 3.6 Результати навчання моделі.

Окремий валідаційний набір даних використовується для перевірки, наскільки добре модель генералізує свої знання на нових, раніше не бачених даних. Валідація дозволяє визначити ступінь, до якої модель уникнула перенавчання (overfitting) або, навпаки, недонавчання (underfitting).

Після кожного циклу тренування, коли модель пройшла через всі дані тренувального набору, проводиться валідація на валідаційному наборі.

Результати валідації служать орієнтиром для покращення параметрів моделі та вдосконалення її продуктивності.

Цей етап є важливим для створення моделі, яка не лише ефективно працює на тренувальних даних, але і може адекватно застосовувати свої знання до нових ситуацій.

3.3.5 Визначення оптимальних гіперпараметрів моделі

	LSTM = 64, Denes = 128, Batch_size = 64, Epochs = 10, Adam = 0.001	**LSTM = 128, Denes = 128, Batch_size = 64, Epochs = 20, Adam = 0.001**	**LSTM = 32, Denes = 64, Batch_size = 64, Epochs = 10, Adam = 0.001**	**LSTM = 128, Denes = 256, Batch_size = 64, Epochs = 5, Adam = 0.01**	**LSTM = 64, Denes = 128, Batch_size = 64, Epochs = 10, Adam = 0.01**	**LSTM = 64, Denes = 128, Batch_size = 64, Epochs = 10, Adam = 0.0001**
Precision	0.8685	0.8572	0.8634	0.8802	0.8715	0.8693
AUC-ROC	0.8800	0.8751	0.8413	0.8895	0.8780	0.8621

Таблиця 3.1 Порівняння зміни гіперпараметрів та метрик.

Аналізуючи результати експериментів з різними наборами гіперпараметрів для моделі, можна зробити декілька важливих висновків. Найефективнішою виявилася конфігурація з LSTM = 128, Dense = 256, Batch_size = 64, Epochs = 5 та швидкістю навчання Adam = 0.01. Цей набір параметрів дозволив досягти високої точності (Precision 0.8802) та високого показника AUC-ROC (0.8895), що свідчить про високу якість моделі в класифікації та роботу з різними класами.

Також важливо зазначити, що збільшення кількості LSTM-шарів не завжди призводить до поліпшення результатів, а зменшення кількості може вплинути на ефективність моделі. Збільшення кількості епох не завжди призводить до покращення, а швидкість навчання може виявитися ключовим фактором: збільшення швидкості може призвести до зростання точності та AUC-ROC.

Додатково, важливо відмітити стабільність моделі при зміні швидкості навчання. Модель показала стійкість результатів при зменшенні швидкості навчання (Adam = 0.0001), що може бути корисним при роботі з великими та складними наборами даних.

У кінці, експерименти з гіперпараметрами вказують на необхідність систематичного та пристосованого підходу до вибору параметрів для кожної конкретної задачі та датасету.

3.3.6 Оцінка продуктивності

На цьому етапі здійснюється оцінка продуктивності навченої моделі. Для цього використовуються різні метрики, щоб отримати об'єктивне уявлення про те, наскільки добре модель справляється з поставленим завданням цензурування тексту.

Метрики, які використовуються

- Точність (Accuracy): це одна з основних метрик, яка визначає, наскільки ефективно модель вирішує поставлене завдання. Ця метрика визначає відсоток правильних передбачень моделі серед усіх передбачень.

  Недоліками точності є те, що вона може бути непоказною в ситуаціях, де

класи нерівномірно розподілені. Наприклад, якщо у вас є 95 екземплярів одного класу і 5 екземплярів іншого, модель може досягти високої точності, класифікуючи всі екземпляри першого класу правильно, але не дуже ефективно визначаючи другий клас через його малий обсяг.

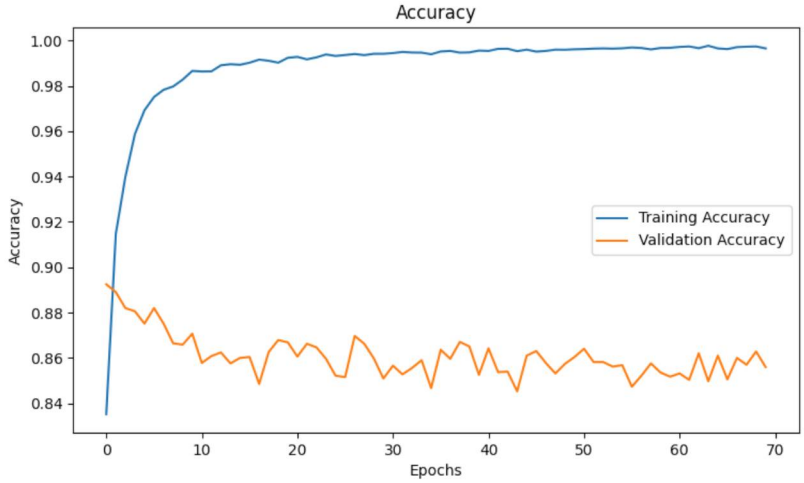


Рисунок 3.7 Графік точності.

- Чутливість (Recall): також відома як True Positive Rate або Sensitivity, є метрикою, яка вимірює, яку частку всіх дійсно позитивних екземплярів модель визначила правильно. Вона важлива у випадках, коли нам потрібно виявити всі можливі позитивні екземпляри, і намалювання уваги на пропусках може мати серйозні наслідки.

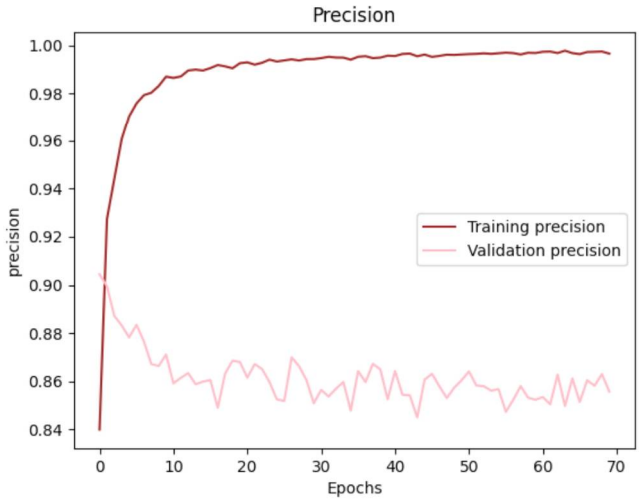
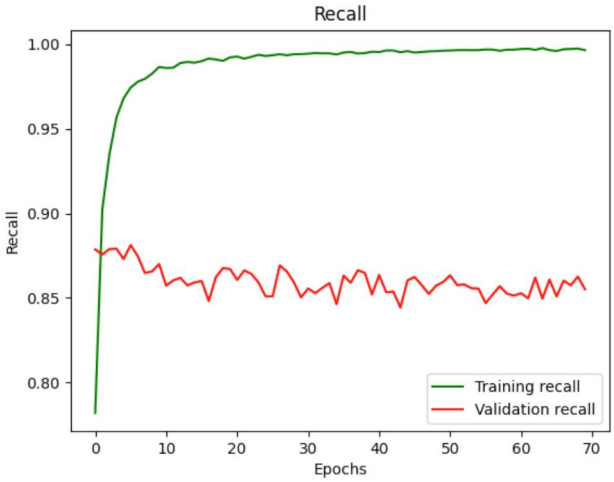
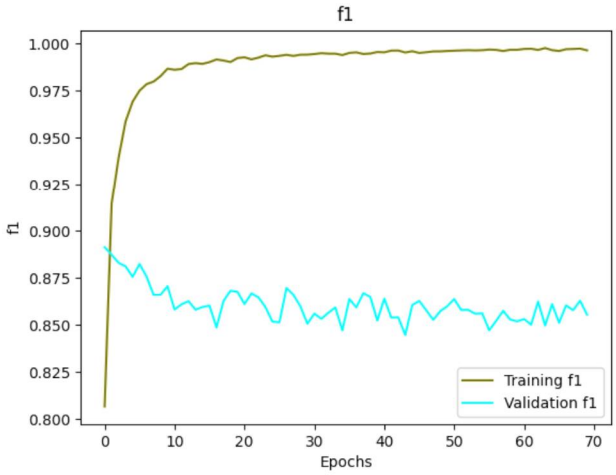


Рисунок 3.8 Графік чутливості.


Recall
1.00
0.95
0.90
0.85
0.80
Recall
0
10
20
30
40
50
60
70
Epochs
Training recall
Validation recall


f1
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
f1
0
10
20
30
40
50
60
70
Epochs
Training f1
Validation f1

Після оцінки моделі може здійснюватися аналіз помилок, щоб виявити ситуації, коли модель неправильно визначала або пропускала неприпустимий

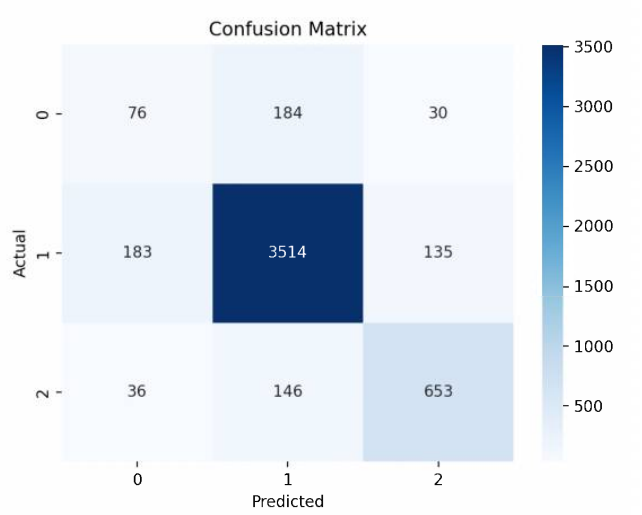


Рисунок 3.11 Матриця плутанини.

контент. Це може призвести до вдосконалення моделі за допомогою додаткового навчання або оптимізації гіперпараметрів.

Ці кроки методики роботи разом із застосуванням LSTM сприяли створенню моделі, яка досягла високої точності 88% та високого показника AUC-ROC 0.8895, що свідчить про високу якість моделі в класифікації та роботу з різними класами.

3.3 Практичне застосування моделі у Telegram

Імплементація моделі цензури тексту у Telegram може значно полегшити управління контентом та забезпечити безпеку серед користувачів. Я реалізував це за допомогою телеграм-бота, який саме буде перехоплювати та при необхідності цензурувати повідомлення.

Етапи розробки:

- створення бота в Telegram;
- отримання токену бота;
- розробка аплікації для обробки повідомлень;
- налаштування взаємодії з Telegram API;

- імпорт моделі;
- запуск бота;
- відповідь бота на повідомлення;
- тестування та вдосконалення.

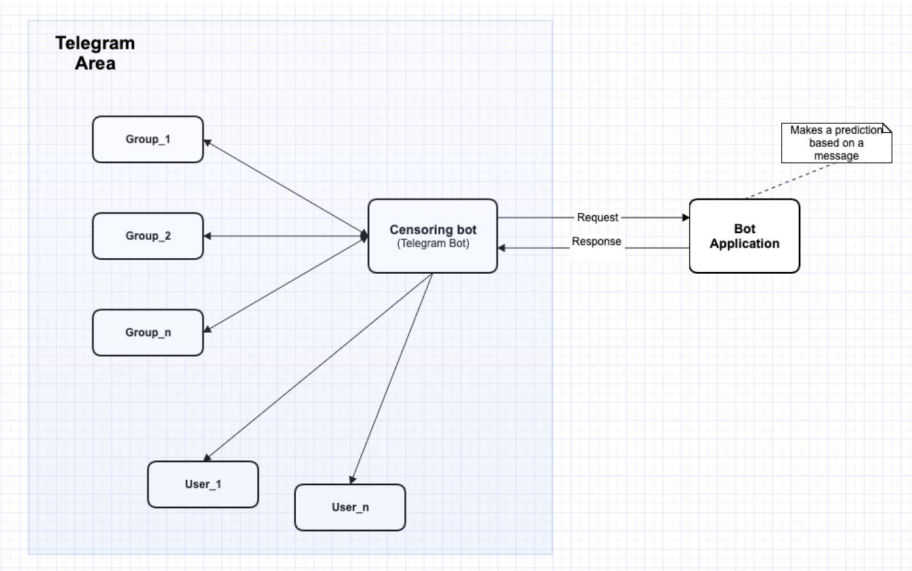


Рисунок 3.12 Архітектура взаємодії сервісів.

З початку треба створити бот в Telegram за допомогою BotFather. Задаємо ім'я та отримуємо унікальний токен для мого бота. Цей токен - ключ для зв'язку аплікації з Telegram API. Використовуватися він буде при кожній взаємодії з API для ідентифікації користувача (бота).

Далі треба налаштувати взаємодію аплікації з Telegram API. Для отримання та обробки повідомлень в режимі реального часу я використовую модуль telebot.

```
Message: It's a bullshit !; Predicted Class:0; All predict [[9.9849534e-01 1.5046155e-03 5.7598282e-10]]
```

Рисунок 3.13 Приклад Log.

При розробці аплікації важливо враховувати безпеку та конфіденційність даних. Після цього можна імпортувати розроблену модель нейронної мережі

для обробки текстового контенту. Модель буде класифікувати повідомлення на припустимі та неприпустимі.

На рисунку 3.12 ми можемо побачити що модель робить передбачення по трьом класам, на основі передбачення моделі ми вирішуємо що робити далі, виділяти повідомлення чи ні. Тобто при визначенні контента неприпустимим бот буде видаляти повідомлення

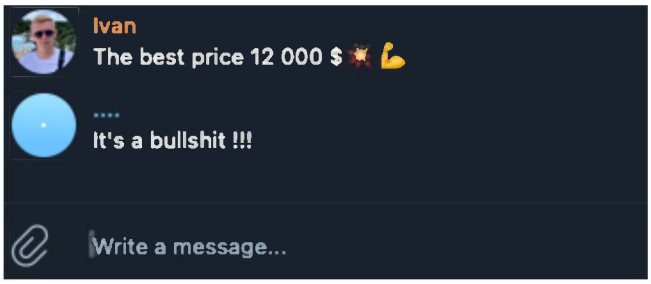


Рисунок 3.14.1 Приклад роботи бота.

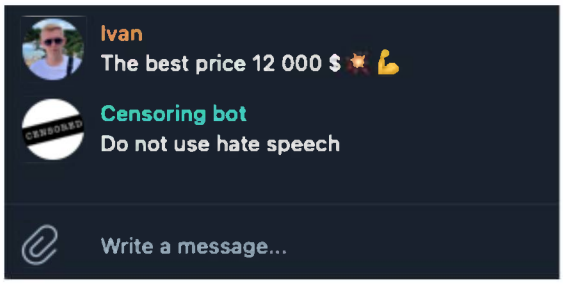


Рисунок 3.14.2 Приклад роботи бота.

Після реалізації логіки обробки повідомлень, аплікацію можна розмістити на власному сервері або в хмарному сервісі.

Останнім етапом є тестування та вдосконалення. Я перевіряю як бот реагує на різноманітні текстові повідомлення(Рис. 3.13.1 - 3.13.2).

Можливі поліпшення :

- Відправляти попередження про порушення правил користування користувачу який надсилав повідомлення.
- Зберігати в базі користувачів які постійно порушення правил користування та застосовувати в їхню сторону санкції.

Області в яких також можливо застосувати модель:

- Соціальні мережі

Використання моделі для автоматичного виявлення та фільтрації образливих чи небажаних коментарів на платформах соціальних мереж може покращити користувачів та створити позитивне середовище для обміну інформацією.

- Медіа та Новини

Впровадження моделі для автоматичного перегляду та фільтрації коментарів на новинних сайтах може допомогти уникнути поширення образливого контенту та покращити якість взаємодії з читачами.

- Організації та Корпорації

Застосування моделі для автоматичного аналізу текстових документів, електронних листів чи інших комунікацій може допомогти забезпечити внутрішню безпеку та уникнути поширення неетичної чи образливої інформації.

- Онлайн-геймінг

Використання моделі для виявлення та фільтрації небажаного мовлення чи образливих висловлювань у чаті гри може створити позитивність в онлайн-спільноті та поліпшити взаємодію гравців.

Висновки за розділом 3

У розділі 3 роботи були розглянуті ключові етапи розробки та впровадження моделі для цензурування текстових даних. Важливими компонентами цього процесу були вибір даних для навчання, розробка програмної реалізації, практичне застосування моделі та оцінка її продуктивності.

Було детально розглянуто питання вибору та підготовки даних для ефективного тренування моделі. Важливим було забезпечити репрезентативність та різноманітність даних, щоб модель могла адекватно виявляти шкідливий контент у різних контекстах.

На етапі розробки програмної реалізації було проведено попередню обробку даних, токенізацію та трансформацію тексту в числові послідовності. Детально описано імплементацію архітектури нейронної мережі та процес компіляції моделі. Далі, навчання та валідація моделі стали ключовими кроками у створенні надійного інструмента для цензурування тексту.

Також у даному розділі роботи було розглянуто конкретне застосування розробленої моделі в середовищі Telegram. Описано кроки створення бота, отримання токену, розробку аплікації для обробки повідомлень та їх цензуру.

ВИСНОВКИ

В даній кваліфікаційній роботі були досягнуті наступні результати:

- визначено постановку задачі даної кваліфікаційної роботи, включаючи мету, об’єкт та предмет дослідження;
- проведено аналітичний огляд існуючих методів цензурування текстових даних, а також метрик для оцінки ефективності моделі нейронної мережі;
- проведено аналітичний огляд існуючих програмних засобів для розробки моделі цензурування текстових даних. В цей огляд включені програмні засоби, реалізовані на різних мовах програмування;
- приведена формалізація вхідних та вихідних даних. Для вхідних даних описані параметри та класи текстових повідомлень;
- виконано опис методів обробки та підготовки вхідних даних та проведено підготовку та обробку вхідних даних для покращення ефективності цензурування текстових даних;
- розроблена модель нейронної мережі для цензурування текстових даних та виконано її опис;
- виконано опис гіперпараметрів які були вибрані для моделі;
- виконано огляд та аналіз отриманих результатів моделі.
- розроблено та представлено практичне застосування моделі у мережі Telegram.

Головним результатом даної кваліфікаційної роботи була розробка моделі нейронні мережі, яка цензурує текстові дані в реальному часі, модель виявляється високомасштабованою і готовою до навчання на даних інших мов. Її архітектура, з використанням LSTM-шарів та ембедінгів, дозволяє ефективно враховувати контекст та взаємодію між словами у текстах. Оскільки використовується токенізація тексту та побудова словника, модель може адаптуватися до різних мовних варіацій. Змінюючи навчальний датасет на дані іншої мови, можна досягти гарної адаптації, оскільки модель вивчає структуру мови та зв'язки між словами. Такий підхід робить модель гнучкою та застосовною до різних завдань та мовних середовищ.

Головним напрямком використання моделі є цензурування текстових даних в онлайн середовищі, де потік текстової інформації росте експоненційно, і ефективна фільтрація та цензура текстових даних стають визначальними аспектами для забезпечення безпеки, конфіденційності та етичності.

Модель демонструє високий рівень ефективності та продуктивності за рахунок розробленої архітектури, різноманіття та якість даних для навчання, та обраних гіперпараметрів. Архітектура складається з LSTM-шарів, ембедінгів та дропаут-шарів які ефективно працюють з текстовими даними, про це свідчить висока точності 88% та високий показника AUC-ROC значення якого складає 0.8895. Дані для навчання, являють собою 25 тисяч текстових повідомлень з мережі Twitter попередньо оброблені, а саме видалені HTML-сутності, URL-адреса та інші непотрібні символи. Проведені експерименти та аналіз результатів дозволили визначити оптимальні гіперпараметри моделі.

Також модульна структура та архітектура програмної реалізації дозволяють легко інтегрувати та адаптувати модель для використання її на різних платформах. Як практичний приклад, модель була успішно імплементована в середовищі Telegram. Це дозволяє в реальному часі аналізувати та цензурувати текстовий контент, що надходить через цю платформу.

На етапі вибору типу моделі нейронної мережі були розглянуті та порівняні різні архітектури, такі як персептрон, багатошаровий персептрон, згорткова нейронна мережа, рекурентні нейронні мережі, зокрема, LSTM, CNN та RNN.

Під час роботи за темою дослідження важливою частиною стала участь у науково-технічній конференції та школі-семінарі, де була представлена доповідь, розкриваючи основні аспекти та досягнення роботи. Ці події надали можливість обміну думками з висококваліфікованими фахівцями галузі та сприяли поглибленню розуміння проблематики.

Однак важливо зауважити, що розвиток таких технологій повинен супроводжуватися відповідальним використанням та урахуванням етичних аспектів. Під час впровадження систем цензури, слід забезпечити прозорість та

можливість контролю для уникнення неправомірного втручання та обмеження свободи вираження.

Дослідження засвідчило важливість інноваційних підходів у вирішенні проблем цензури текстових даних, а використання нейронних мереж та технологій штучного інтелекту стає обіцяючим напрямком для подальших досліджень та впроваджень у цій області.

Наступні кроки можуть включати подальше вдосконалення моделей, розширення мов застосування за рахунок збільшення тренувального датасету, покращення початкової обробки тексту видаленням стоп-слів, видаленням або урахуванням емодзі та пошук оптимального балансу між ефективністю та етичними аспектами у цій динамічній області технологій.

Отже в даній кваліфікаційній роботі були виконані усі сформовані задачі та досягнута мета, а саме забезпечивши більшу якість та безпеку контенту для користувачів, які залежать від надійної та безпечної інформації в Інтернеті, за допомогою розробки та впровадження нейронної мережі, яка здатна аналізувати текстові дані в реальному часі та автоматично видаляти небажаний та шкідливий контент з різних джерел в Інтернеті. Таким чином в практичному застосуванні у мережі Telegram модель видаляє 9 з 10 недопустимих повідомлень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Fort Lewis College, John F. Reed Library. Banned Books, Common Reasons for Banning Books, Censorship & Free Speech. November 15, 2013. Web. March 19, 2014.
2. Ali, Shirin. Books that touched on race were among the most challenged as inappropriate for libraries in 2020, 2021, April 11
3. Smith, Nadine. (n.d.). What are the dangers of censorship?, 2018. URL: https:// editin.cnn.com/2021/04/11/us/challenges-to-books-about-racism-trnd.
4. Harshali M. Deep learning — 2015.
5. Yingci L. Zhonghua C. Hongkai W. Peiou L. Zongwei Z. Comparison of machine learning methods for classifying mediastinal lymph node metastasis of non-small cell lung cancer from 18F-FDG PET/CT — 2017
6. Модельний ряд Raspberry Pi. URL: https://raspberrypi.ru/modeli_raspberry_pi (дата звернення: 10.03.2023).
7. Документація OpenCV. URL: https://opencv.org/ (дата звернення: 20.05.2020).
8. Zhang, X. A Mathematical Model of a Neuron with Synapses based on Physiology. Nat Prec (2008). https://doi.org/10.1038/npre.2008.1703.1
9. Sima J. "Introduction to Neural Networks," Technical Report No. V 755, Institute of Computer Science, Academy of Sciences of the Czech Republic, 1998.
10. Kröse B., and van der Smagt P. An Introduction to Neural Networks. (8th ed.) University of Amsterdam Press, University of Amsterdam, 1996.
11. Y. Bengio. Artificial Neural Networks and their Application to Speech / Sequence Recognition, McGill University Ph.D. thesis. 1991. URL:https:// escholarship.mcgill.ca/concern/file_sets/kw52j887f?local e=en.
12. L. Deng, K. Hassanein, M. Elmasry. "Analysis of correlation structure for a neural predictive model with applications to speech recognition," Neural Networks, vol. 7 (2), 1994, pp. 331–339.
13. Maeda-Gutiérrez, V. Galván-Tejada, C.E.; Zanella-Calzada, L.A. Celaya-Padilla, J.M. Galván-Tejada, J.I. Gamboa-Rosales, H. Luna-García, H. Magallanes-Quintanar, R. Guerrero Méndez, C.A.; Olvera-Olvera, C.A. Comparison of

14. Types of Neural Networks and Definition of Neural Network URL: https:// www.mygreatlearning.com/blog/types-of-neural-networks.
15. В. Є. Стрілець, С. І. Шматков, М. Л. Угрюмов та ін. – Харків, Методи машинного навчання монографія, Харківський національний університет імені В. Н. Каразіна, 2020.
16. Ben Lutkevich, BERT language modelURL: https://www.techtarget.com/ searchenterpriseai/definition/BERT-language-model.
17. Y. Bengio BERT Neural Networks URL:https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b62701.
18. Generative Pre-Trained Transformer (GPT) URL: https://encord.com/glossary/ gpt-definition.
19. Gayle Laakmann McDowell. Cracking the Coding Interview: 189 Programming Questions and Solutions 6th Edition.
20. Aurélien Géron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems 3rd Edition
21. Oliver Theobald, Best Machine Learning Books for Absolute Beginners, 2021 - 50 c.
22. John D. Kelleher, Brian Mac Namee, and Aoife D'Arcy, Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies, 2019 - 133 c.
23. Drew Conway, John Myles White, Machine Learning for Hackers, 2020 - 209 c.
24. Judea Pearl, Madelyn Glymour, Nicholas P. Jewell, Causal Inference in Statistics, 2016 - 45 -50 c.
25. Marc Peter Deisenroth, Mathematics for Machine Learning, 2020
26. Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow 2019.Deep Learning with Python, Second Edition, Francois Chollet, 2021

27. Nikhil Buduma, Nicholas Lacascio, Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms, 2017
28. Davy Cielen, Arno Meysman, Introducing Data Science: Big data, machine learning, and more, using Python tools, 2016
29. Nishant Shukla, Machine Learning with TensorFlow, 2018
30. Satnam Alag , Collective Intelligence in Action, 2008

ДОДАТКИ

Додаток А

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

Титаренко Тимур Олегович
(прізвище, ім'я, по батькові студента)

1. Тема роботи «Модель нейронної мережі для цензурування 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 цензурування текстових даних.
   2. Визначення архітектури нейронної мережі для цензурування текстових даних
   3. Вибір алгоритмів навчання та оптимізації моделі
   4. Пошук та підготовка набору даних для навчання та тестування моделі.
   5. Тестування та дослідження ефективності роботи нейронної мережі для цензурування текстових даних.

4. План роботи

№ з/п	Назви етапів роботи	Термінвикон ання етапівроботи
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 цензурування текстових даних, підготовка документації проекту.	06.03.2023 - 01.04.2023
3	Аналітичній огляд існуючих програмних засобів для розробки моделі нейронної мережі.	01.04.2023 - 27.06.2023
4	Вибір датасету з тестовими даними.	20.06.2023 - 24.08.2023
5	Розробка моделі нейоронної мережі.	15.08.2023 - 17.09.2023
6	Програмна реалізація моделі з використанням підібраних алгоритмів.	17.08.2023 - 20.09.2023
7	Аналіз результатів тестування та визначення ефективності використання моделі для цензурування текстових даних.	01.09.2023 - 17.10.2023
8	Оформлення пояснювальної записки.	18.10.2023- 27.11.2023
9	Представлення кваліфікаційної роботи керівнику та рецензенту.	28.10.202- 28.11.2023

5. Дата видачі завдання 08.12.2022 р.

Студент: Титаренко Т. О.

Керівник роботи: Шматков С. І.

Додаток Б

Технічне завдання

на розробку програмного виробу

«Модель нейронної мережі для цензурування текстових даних»

Назва розділу	Назва і зміст підрозділу
1. Введення	1.1. Назва виробу: модель нейронної мережі для цензурування текстових даних. 1.2. Галузь застосування: онлайн-середовища.
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 No 0210-05/1804 від 10.09.2023.
3. Призначення розробки	3.1. Мета розробки забезпечити більшу безпеку та якість контенту для користувачів, які залежать від надійної та безпечної інформації в Інтернеті. 3.2. Призначення виробу: аналізувати текстові дані в реальному часі та автоматично видаляти небажаний та шкідливий контент з різних джерел в Інтернеті, таких як соціальні мережі, форуми, блоги та інші джерела інформації.
4. Вихідні дані	Набір даних із Twitter використовувався для дослідження виявлення ворожих висловлювань. Текст класифікується як: мова ненависті, образлива лексика та ні те, ні інше. Зважаючи на характер дослідження, важливо зазначити, що цей набір даних містить текст, який можна вважати расистським, сексистським, гомофобним або загалом образливим.

Назва розділу	Назва і зміст підрозділу
5. Технічні вимоги до виробу	5.1. Вимоги до функціональних характеристик: можливість фільтрувати текст на цензурний та нецензурний. 5.2. Вимоги до надійності: забезпечення працездатності. моделі цензурування текстових даних при подачі вхідних даних неправильного формату. 5.3. Вимоги до умов експлуатації: встановлення мови програмування Python та необхідних бібліотек для роботи моделі цензурування текстових даних. 5.4. Вимоги до складу і параметрів технічних засобів: комп'ютер або ноутбук із 4 ГБ оперативної пам'яті та процесором не нижче Intel Core i3 9-го покоління. 5.5. Вимоги до інформаційної та програмної сумісності: підтримка ОС Linux або Windows 8/10, підтримка Anaconda, Python 3. 5.6. Вимоги до маркування та упаковки: жорстка упаковка з пластмаси, маркування на українській та англійській мовах. 5.7. Вимоги до транспортування і зберігання: транспортування в упаковці будь-яким способом, температура транспортування/зберігання +5-+20 Со. 5.8. Спеціальні вимоги: відсутні.
6. Вимоги до документації.	Документацією до виробу «Модель нейронної мережі для цензурування текстових даних» вважати: 1) Справжнє Технічне завдання на розробку виробу (представити як Додаток Б до пояснювальної записки до дипломної роботи). 2) Програму і методику випробувань розробленого виробу (представити як Додаток В до пояснювальної записки до дипломної роботи). 3) Опис виробу (представити у Розділі 3 до пояснювальної записки до дипломної роботи). 4) Фрагменти тексту програми (записати на диск з пояснювальною запискою до дипломної роботи).
7. Техніко-економічні показники	Орієнтовна оцінка ефективності та якості виконуваної цензури данних: точність (accuracy) повинна бути вище 90%. Порівняння ефективності з іншими методами кластеризації представити у розділі 3.

Назва розділу	Назва і зміст підрозділу
8. Стадії і етапи розробки	1. Визначення постановки задачі моделі для цензурування даних. 2. Аналітичний огляд існуючих моделей для роботи з природною мовою. 3. Огляд існуючих програмних засобів для кластеризації даних. 4. Формалізація і уявлення вхідних даних; 5.Підготовка та обробка вхідних даних для ефективності методу. 6. Опис моделі цензурування даних. 7. Опис обчислювального алгоритму вирішення задачі. 8.Опис обраного програмного забезпечення для вирішення задачі. 9. Порівняння ефективності різних моделей. 10. Огляд та аналіз отриманих результатів.
9. Порядок контролю і приймання	Загальні вимоги до приймання розроблюваного виробу: 1) Перевірка ходу розробки програмного виробу керівнику робіт виконувати 1 раз в 3 тижні. 2) Випробування виробу відповідно до Програми і методики випробувань провести на базі комп'ютерного класу або на базі приватного приміщення. 3) Захист розробленого виробу провести на засіданні атестаційної комісії. 4) Пояснювальну записку представити на паперових носіях в одному примірнику та в електронному вигляді.

Додаток В

Програма і методика випробувань

програмного виробу

«Модель нейронної мережі для цензурування текстових даних»

1. Об'єкт випробувань

Об’єктом випробовування є програмний застосунок для цензурування текстових даних (розділ 1 ТЗ).

2. Мета випробувань

Перевірка роботоспособності програмного застосунок для цензурування текстових даних за допомогою згорткових нейронних мереж (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) Оцінити ефективність роботи моделі за допомогою метрик acurancy та AUC

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 було встановлень Python 3.

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 ЕСПД. «Програма і методика випробувань».

6.2.2. Перевірка працездатності методу

6.2.2. Перевірка працездатності методу

Тест 1

Перевірку працездатності здійснюємо шляхом запуску програми. Отримуємо результат запуску консольного додатку.

```
2023-11-22 19:55:42,803 - INFO - Starting load model...
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.Adam`
2023-11-22 19:55:43,151 - INFO - Model has been loaded
2023-11-22 19:55:43,158 - INFO - Ready to use...
2023-11-22 19:55:43,158 - INFO - Enter your text:
```

Рисунок В.1 - Консольний вихід.

Тест 2

Перевірку здійснюємо шляхом запуску програми та вводу недопустимого тексту. Отримуємо результат роботи додатку.

```
2023-11-22 20:14:46,759 - INFO - Starting load model...
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.Adam`
2023-11-22 20:14:47,075 - INFO - Model has been loaded
2023-11-22 20:14:47,083 - INFO - Ready to use...
2023-11-22 20:14:47,083 - INFO - Enter your text:That's bullshit
2023-11-22 20:14:47,083 - INFO - Prediction class: 0, You use hate speech
```

Рисунок В.2 - Результат висновку моделі.

Тест 2

Перевірку здійснюємо шляхом запуску програми та вводу допустимого тексту. Отримуємо результат роботи додатку.

```
2023-11-22 20:18:26,535 - INFO - Starting load model...
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.
2023-11-22 20:18:26,904 - INFO - Model has been loaded
2023-11-22 20:18:26,915 - INFO - Ready to use...
2023-11-22 20:18:26,915 - INFO - Enter your text:Hi, my name is Tim
2023-11-22 20:18:26,915 - INFO - Prediction class: 2, Text is valid
```

Рисунок В.3 - Результат висновку моделі.

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

Виконав
студент групи КУ-61 Титарснко Т.О.

Додаток Г

Фрагменти програмного коду

```
# remove html entity:
def remove_entity(raw_text):
    entity_regex = r"&[^\s;]+;"
    text = re.sub(entity_regex, "", raw_text)
    return text

# change the user tags
def change_user(raw_text):
    regex = r"@([^ ]+)"
    text = re.sub(regex, "user", raw_text)

    return text

# remove urls
def remove_url(raw_text):
    url_regex = r"(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.]
                r"[.][a-z]{2,4}/)(?:[^\s()<>]+|\(([^\s()<>]+|(\([^\s()<>]+\)))*\))+(?:\(([^\s()<>]+|(\([^\s()<>]+\)))*\)|[^\s`!()\[\]{};:\'".,<>?«»“”‘’])"
    text = re.sub(url_regex, '', raw_text)

    return text

# remove unnecessary symbols
def remove_noise_symbols(raw_text):
    text = raw_text.replace('"', '')
    text = text.replace("'", '')
    text = text.replace(';', '')
    text = text.replace("`", '')
    text = text.replace("..", '')

    return text
```

Рисунок Г.1 - Приклад програмного коду для попередньої обробки даних.

```
model = Sequential([

    Embedding(vocab_size, output_dim, input_length=max_length),

    LSTM(128, dropout=0.3, recurrent_dropout=0.3),

    Dropout(0.5),

    Dense(256, activation='relu'),

    Dropout(0.5),

    Dense(3, activation='softmax')
])


model.compile(optimizer=Adam(lr=0.01), loss='categorical_crossentropy', metrics=['accuracy',f1_,precision, recall])

history = model.fit(
    X_train,
    y_train,
    batch_size=64,
    epochs=5,
    validation_data=(X_test, y_test)
)
```

Рисунок Г.2 - Приклад програмного коду побудови моделі.