МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка веб-сервісу для розпізнавання кіберугроз з використанням нейромережних технологій»

Виконав студент 2 курсу групи МІС-22
спеціальності 122 Комп'ютерні науки
Леонтьєв Михайло Едуардович

Керівник к.техн.наук, доцент
Терещенко Тетяна Михайлівна

Рецензент
Копиченко Іван Юрійович

Одеса 2023

АНОТАЦІЯ

на магістерську кваліфікаційну роботу

«Розробка веб-сервісу для розпізнавання кіберугроз з використанням нейромережних технологій»

Студента Леонтьєва Михайла Едуардовича

В контексті кібербезпеки, важливим є ідентифікація іменованих сутностей, таких як IP-адреси, домени, URL-адреси, імена користувача, сертифікати, хеш-суми файлів та інше. Використання цих іменованих сутностей дозволяє виявляти та аналізувати потенційні загрози та атаки у кіберпросторі. Робота присвячена дослідженню архітектур нейронних мереж та алгоритмів обробки інформації, що використовують для рішення задач детектування кіберзагроз.

Тема магістерської роботи «Розробка веб-сервісу для розпізнавання кіберугроз з використанням нейромережних технологій».

Актуальність магістерської роботи полягає в розробці та застосуванні системи обробки даних та детектування іменованих сутностей для виявлення потенційних кіберзагроз.

Метою роботи є розробка програмної конфігурації для системи детектування об’єктів, що забезпечить аналіз даних і виявлення іменованих сутностей. Функціональні можливості системи повинні охоплювати аналіз завантажених даних і збереження результатів у файл.

Об'єкт дослідження – процеси проектування та розробки архітектури системи, що забезпечить користувачу швидкий аналіз інформації за допомогою використання нейронної мережі.

Предмет дослідження – засоби реалізації процесу детектування та алгоритму роботи рекурентної нейронної мережі.

В роботі було проведено дослідження технологій розробки систем детектування іменованих сутностей та аналіз архітектур нейронних мереж, які

використовують для реалізації їх алгоритмів. Виконано програмну реалізацію системи.

Практична цінність роботи полягає в тому, що створена система забезпечує зручний засіб для користувача по аналізу даних на наявність кіберзагроз.

Ключові слова: CYBER THREAT INTELLIGENCE FEED, ІМЕНОВАНІ СУТНОСТІ, РЕКУРЕНТНІ НЕЙРОННІ МЕРЕЖІ, КІБЕРЗАГРОЗИ, ДАТАСЕТ, ІНФОРМАЦІЯ, АЛГОРИТМ INSIDE-OUTSIDE-BEGIN.

Магістерська робота містить 79 сторінок, 2 таблиці, 27 рисунків, 17 посилань.

SUMMARY

for a master's thesis

«Development of a web service for recognizing cyberthreats using neural network technologies»

by student Leontiev Mikhail

In the context of cyber security, it is important to identify named entities such as IP addresses, domains, URLs, usernames, certificates, file hashes, and more. The use of these named entities allows detection and analysis of potential threats and attacks in cyberspace. The work is devoted to the research of neural network architectures and information processing algorithms used to solve the problems of cyber threat detection.

The topic of the master's thesis is "Development of a web service for recognizing cyberthreats using neural network technologies."

The relevance of the master's thesis lies in the development and application of a data processing system and the detection of named entities for the detection of potential cyber threats.

The purpose of the work is to develop a software configuration for the object detection system, which will provide data analysis and detection of named entities. Functional capabilities of the system should include the analysis of downloaded data and saving the results to a file.

The object of research is the processes of designing and developing the architecture of the system, which will provide the user with a quick analysis of information using a neural network.

The subject of the research is means of implementing the process of detection and the algorithm of the recurrent neural network.

In the work, research was carried out on technologies for developing systems for detecting named entities and analysis of neural network architectures, which are used to implement their algorithms. The software implementation of the system has been completed.

The practical value of the work lies in the fact that the created system provides a convenient tool for the user to analyze data for the presence of cyber threats.

Keywords: CYBER THREAT INTELLIGENCE FEED, NAMED ENTITIES, RECURRENT NEURAL NETWORKS, CYBER THREATS, DATASET, INFORMATION, INSIDE-OUTSIDE-BEGIN ALGORITHM.

The master's thesis contains 79 pages, 2 tables, 27 figures, 17 links

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ 10
ВСТУП 12
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ 14
1.1 Огляд та аналіз кіберзагроз 14
1.2 Платформа розвідки загроз 15
1.3 Розпізнавання іменованих сутностей 19
1.4 Підготовча обробка текстових даних 23
1.4.1 Токенізація 23
1.4.2 Нормалізація 26
1.4.3 Побудова векторної моделі 27
1.5 Огляд та аналіз використання нейроних мереж для виявлення кіберзагроз 30
1.5.1 Рекурентні нейронні мережі та LSTM 30
1.5.2 Transformer та концепція механізму уваги 32
1.5.3 BERT 34
1.5.3 RoBERTa 36
1.6 Постановка завдання 37
Висновки з першого розділу 38
2 ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ ВЕБ-СЕРВІСУ 39
2.1 Опис методу навчання нейронної мережі за допомогою бібліотек 39
2.2 Розробка алгоритму аналізу даних 40
2.3 Вибір засобів розробки 42
2.3.1 Мова програмування 42
2.3.2 Вхідні дані 43
3 ПРОЕКТУВАННЯ 48
3.1 Проектування процесу створення і навчання нейронної мережі засобами IDEF0 48
3.2 Проектування веб-системи за методологією Workflow Diagramming 51
3.3 Моделювання процесу передобробки даних 53
3.4 Діаграма варіантів використання 55

3.5 Підхід до розпізнавання іменованих сутностей 56
4 РЕАЛІЗАЦІЯ 58
4.1 Архітектура веб-сервісу 58
4.2 Програмна реалізація опрацювання даних 59
5 ТЕСТУВАННЯ ВЕБ_СЕРВІСУ 64
5.1 Тестування роботи нейронної мережі 69
5.2 Функціональне тестування 70
5.3 Подальший розвиток-нові набори даних 72
5.3.1 База знать MITRE ATT&CK 73
5.3.2 Набір DNRTI 74
ВИСНОВКИ 76
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 78

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

BERT – Bidirectional Encoder Representations from Transformers

CRF – Conditional Random Fields

CTI – Cyber Threat Intelligence Feed

DOS – Denial of Service

IDS – Intrusion Detection Systems

NER – Namer Entities Recognition

MLM – Masked Language Modeling

LSTM – Long Short-Term Memory

TIP – Threat Intelligence Platform

TF-IDF – TF – term frequency, IDF – inverse document frequency

RNN – Recurrent Neural Network

R2L – Remote to Local Attack

U2R – User to Root Attack

BPE –алгоритм стиснення, який використовується в обробці природної мови

CNN – згорткова нейронна мережа

CBoW – архітектура, яка передбачає поточне слово, виходячи з навколишнього контексту

Input Gate – вхідний вузол в архитертурі LSTM

IDF – Зворотна частота документів

IDEF0 – методологія функціонального моделювання

IOB – Алгоритм Inside-Outside-Begin

Inverse Document Frequency – зворотня частоти документа

Forget Gate – забуваючий вузол в архитертурі LSTM

FSVM – метод нечітких опорних векторів

KDD Cup 1999 Data – набор даних

Machine Learning-Based Approac 2– підхід з урахуванням машинного навчання

Long Short-Term Memory – мережі з довгою короткостроковою пам'яттю

One-Hot Encoding – унітарне кодування

Python –мова програмування

Output Gate – вихідний вузол в архитертурі LSTM

RNN – word-level архитектура

RoBERTa – більш гнучка та оптимізована архитектура версії BERT

Probing Attack – спроба отримати інформацію про мережу комп'ютерів з метою обійти її безпекові заходи

TensorFlow – це відкрита програмна бібліотека

Term Frequency – добуток частоти терміну

TF – частота термінів

Transformer – архітектура глибокої нейронної мережі

ВСТУП

Безперервне зростання складності суб'єктів загроз протягом багатьох років перетворив використання оперативної інформації про загрози на важливу частину захисту від них. Згідно з звітом SANS 2022 Cyber Threat In telligence Survey 72,9% опитаних команд-дослідників кібербезпеки використовують як джерело даних про кіберзагрози CTI постачальників. Структуризація неструктурованих даних – одна з задач, яка повинна вирішуватись під час аналізу таких звітів. Наприклад, є текст (або набір текстів), і дані з нього потрібно ввести в базу даних (таблицю). Класичні іменовані сутності можуть відповідати строкам такої таблиці або же служити змістом яких-то комірок в неї. Відповідно, щоб правильно заповнити таблицю, потрібно перед цим виділити в тексті дані, які не будуть в ньому використовуватись (зазвичай після цього є ще один етап — ідентифікація сутності в тексті).

В контексті кібербезпеки, важливим є ідентифікація іменованих сутностей, таких як IP-адреси, домени, URL-адреси, імена користувача, сертифікати, хеш-суми файлів та інше. Використання цих іменованих сутностей дозволяє виявляти та аналізувати потенційні загрози та атаки у кіберпросторі.

Найчастіше використовуються автоматизовані CTI-постачальники, які є сервісами/додатками, які витягують з текстів звітів CTI-дослідників корисну інформацію про згадану загрозу. Використання такого виду постачальників даних дозволяє зняти навантаження аналітиків інформаційної безпеки та прискорити процес розслідування інциденту загрози безпеці.

Мета роботи: розробка веб-сервісу для розпізнавання іменованих сутностей у текстах звітів дослідників кіберзагроз із застосуванням нейромережевих технологій.

Об'єкт дослідження – процеси проектування та розробки архітектури системи, що забезпечить користувачу швидкий аналіз інформації за допомогою використання нейронної мережі.

Предмет дослідження – засоби реалізації процесу детектування та алгоритму роботи рекурентної нейронної мережі.

Практична цінність роботи полягає в тому, що створена система забезпечує зручний засіб для користувача по аналізу даних на наявність кіберзагроз.

Для досягнення поставленої мети необхідно розв'язати такі завдання:

1) провести огляд аналогів та наукової літератури;

2) підготувати набір даних;

3) реалізувати обрану топологію штучної нейронної мережі та провести її навчання та тестування;

4) розробити веб-сервіс для розпізнавання іменованих сутностей у текстах звітів дослідників кіберзагроз;

5) провести тестування розробленого сервісу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Огляд та аналіз кіберзагроз

Аналіз загроз кібербезпеці – це набір методів, які зазвичай використовуються для боротьби із загрозами. Ці методи в поєднанні з ефективною стратегією допомагають організації оцінити свою інфраструктуру безпеки, протоколи, процеси та процедури, щоб виявити загрози та вразливості та отримати інформацію про потенційну атаку до того, як вона відбудеться.

Групи кібербезпеки можуть краще зрозуміти рівень складності загроз, спрямованих проти організації, і стратегії використання, а також визначити області в інфраструктурі безпеки організації, які можуть бути вразливими до цих загроз, виконуючи аналіз загроз. Це, без сумніву, один із найважливіших методів безпеки, який організація повинна використовувати, щоб отримати більше знань про можливі небезпеки, з якими вона може зіткнутися [1].

Кібератаки належать до ширшого контексту, ніж те, що традиційно називають інформаційними операціями. В інформаційних операціях інтегроване використання основних можливостей електронної війни, психологічних, комп’ютерних мереж, операцій безпеки в координації зі спеціальною підтримкою та відповідними здібностями, а також для проникнення, зупинки, знищення або викрадення людських рішень.

Рисунок 1.1 описує анатомію кібератаки. Згідно зі стратегією USNM для операцій у кіберпросторі, робота комп’ютерної мережі складається з атаки, захисту та активації використання. Останнє відрізняється від мережевих атак і мережевого захисту, оскільки цей тип операцій більше зосереджений на зборі та аналізі інформації, ніж на перериванні мереж, і сам по собі може бути прелюдією до атаки.

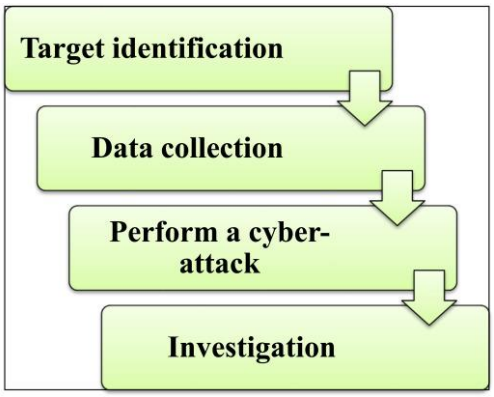

Рисунок 1.1 – Анатомія кібератаки

Ці операції можуть здійснюватися з метою поширення інформації та пропаганди. Операції з використання комп'ютерної мережі також можуть здійснюватися з метою викрадення важливих комп'ютерних даних. У такому контексті Trap Sniffers і Doors є корисними інструментами для кіберрозвідки. Trap Doors дозволяє зовнішньому користувачеві отримати доступ до програмного забезпечення доступності в будь-який час без відома користувача комп'ютера.

1.2 Платформа розвідки загроз

Шалене зростання чисельності просунутих загроз (APT) і таке ж зростання обсягу даних, які потребують обробки для виявлення атак, з кожним днем ускладнює роботу аналітиків безпеки. В останні роки робота інженера інформаційної безпеки часто полягає у ручному просіюванні сотень попереджень у спробах знайти реальні загрози.

Через інформаційні масиви, які генерують організації, команди з ручним пошуком загроз уже неефективні. На боротьбу з цією проблемою використовуються ресурси та програмні засоби, але часто впровадження нових інструментів гальмується під час інтеграції в інфраструктуру організації.

Платформа аналізу інформації про загрози автоматизує, доповнює контекстом базові відомості про загрози – фіди (feeds). Фіди – це потоки даних з індикаторами компрометації, за якими розпізнається потенційна загроза: хеші шкідливих файлів, IP-адреси та домени, пов'язані зі злочинною активністю. Автоматизація процесу звільняє перевантажений персонал, надає точні засоби для аналізу в режимі реального часу, щоб швидко та безпомилково реагувати на загрози [2].

Платформа розвідки загроз, Threat Intelligence Platform (TIP) – це технологічне рішення, яке допомагає організаціям збирати, аналізувати та використовувати інформацію про потенційні кіберзагрози. Він збирає дані з різних джерел, таких як канали безпеки, розвідка з відкритим кодом і системи внутрішньої безпеки, щоб надати розуміння нових загроз, вразливостей і ознак компрометації.

TIPs зазвичай пропонують такі функції, як агрегація даних про загрози, збагачення, кореляція та візуалізація. Вони допомагають командам безпеки визначати пріоритети та ефективніше реагувати на загрози, надаючи сповіщення в реальному часі, контекстну інформацію та рекомендовані дії для усунення.

Ці платформи дозволяють організаціям підвищувати рівень безпеки, активно виявляти та зменшувати ризики, а також залишатися попереду нових кіберзагроз. Вони зазвичай використовуються центрами безпеки (SOC), групами реагування на інциденти та мисливцями за загрозами для посилення захисту від кібератак.

Компанії-постачальники послуги зі збору інформації про загрози – відносні новачки у сфері безпеки, тому типи послуг, що надаються, багато в чому різняться. Деякі з таких сервісів надають лише очищені від хибних спрацьовувань фіди. Найбільш поширені платні сервіси надають агреговані та кореловані фіди (два і більше), оповіщення та попередження, що настроюються, специфічні для клієнтського ландшафту ризику.

Ще один тип служби аналізу загроз займається агрегуванням та кореляцією даних і автоматично включає інформацію до пристроїв безпеки – брандмауери, управління інформацією про безпеку та події, веде оцінку галузевих загроз та надає консультації з питань безпеки.

Кожен тип платформи аналізу загроз надається за підпискою, як правило, на двох або трьох тарифних рівнях і доставляється через хмару, або трохи рідше – локально чи гібридно. У зв'язку з дорожнечею таких платформ, а також через необхідність обладнання для локального розгортання, зараз такі послуги орієнтовані великі організації та підприємства. Однак у міру того, як хмарні послуги виходять на нижчі сегменти ринку, інструменти аналізу загроз теж стануть доступнішими.

Вартість платформ варіюється так само, як і вартість самих служб з розвідки загроз. Одні тільки канали передачі даних можуть коштувати тисячі доларів на місяць, але пов'язані з цим витрати включають витрати на цілодобове обслуговування операційного центру безпеки, укомплектованого технічними фахівцями та аналітиками. Для порівняння, керовані служби безпеки, як правило, коштують десятки тисяч доларів на місяць. Найменш дорогі послуги вимагають більше людського часу та зусиль з боку клієнта [2].

Ключові особливості Threat Intelligence Platform:

1. Збір даних: TIP інтегрує безліч джерел даних, включаючи відкриті та закриті джерела, форуми, блоги, соціальні мережі, підслуховування та інші канали для отримання всебічної інформації про загрози.

2. Обробка та аналіз: отримані дані піддаються автоматичній обробці та аналізу, із застосуванням технологій машинного навчання та штучного інтелекту. Це дозволяє виділити ключові особливості загроз та визначити їхню потенційну небезпеку для організації.

3. Класифікація загроз: TIP дозволяє класифікувати загрози за різними критеріями, такими як тип шкідливого ПЗ, вразливості, цілі, методи поширення та ін. Це допомагає краще зрозуміти характер загроз та сприяє ефективній розробці контрзаходів.

4. Інформаційні дайджести: TIP надає зручні дайджести та звіти про актуальні загрози, адаптовані до конкретних потреб організації. Це допомагає інформаційним безпекам бути в курсі останніх трендів та нових загроз.

5. Спільна робота та інтеграція: багато TIP-платформ підтримують можливість спільної роботи команди безпечників та обміну інформацією з іншими інструментами інформаційної безпеки (наприклад, SIEM-системами), що забезпечує гармонійну та ефективнішу роботу всієї команди.

Функції Threat Intelligence Platform, які необхідною складовою інформаційної безпеки:

1. Запобігання кібератакам: аналіз інформації про загрози дозволяє виявляти потенційні атаки на ранніх стадіях та вживати заходів щодо їх недопущення.

2. Скорочення реакційного часу: завдяки актуальним дайджестам та звітам команди з інформаційної безпеки можуть оперативно реагувати на нові загрози, скорочуючи час реакції на інциденти.

3. Найкраще розуміння загроз: TIP допомагає організаціям краще зрозуміти характер загроз, виходячи з чого можна розробляти ефективніші стратегії захисту.

4. Зниження ризиків: використання TIP дозволяє знизити ризики кібератак та підвищити загальний рівень безпеки комп'ютерних систем та даних.

Як джерело даних TIP використовують різні канали знань про кіберзагрози (cyber threat intelligence feed або CTI feed) – це технологічне рішення, яке є постачальником даних, пов'язаних з потенційними або поточними загрозами кібербезпеки. Канали CTI надають таку інформацію про об'єкти загрози безпеці як імена шкідливого програмного забезпечення (ПЗ), індикатори компрометації, та інші [3].

CTI канали часто використовують як вихідні дані звіти дослідників кіберзагроз, витягуючи з великих обсягів тексту ємні та конкретні дані про учасників та об'єкти інциденту, згаданого у звіті. Звіти про кіберзагрози щодня

публікується в різних онлайн джерелах, включаючи бази даних виявлених вразливостей, канали груп екстреного реагування на інциденти, соціальні мережі, а також на форумах за тематикою CTI.

Незалежно від рівня відкритості, будь-які CTI канали потребують ретельної перевірки з точки зору точності, щоб, наприклад, уникнути випадкового блокування занадто великої кількості адрес, думці постачальника є шкідливими.

Системи, що використовують методи розпізнавання іменованих сутностей (Namer Entities Recognition, NER) для отримання короткої корисної інформації, показали дуже високу продуктивність у таких областях, як ідентифікація іменованих об'єктів у юридичних документах, публікаціях у соціальних мережах, біографічних текстах [4].

1.3 Розпізнавання іменованих сутностей

Named Entity Recognition (NER) – це завдання в галузі обробки природної мови (NLP), спрямована на виділення та класифікацію іменованих сутностей у тексті, таких як імена людей, назви організацій, дати, розташування, суми грошей та інші типи специфічних об'єктів. NER є важливим компонентом багатьох NLP-додатків, таких як вилучення інформації, аналіз тональності, питання-відповідальні системи та багато інших.

Хоча більшість іменованих сутностей складається з кількох слів, при вирішенні цього завдання зазвичай розглядають окремі слова і вирішують, чи це слово є частиною іменованої сутності чи ні. При цьому розрізняють початок, середину та кінець іменованої сутності.

При розмітці іменованих сутностей прийнято використовувати префікс B (beginning) для позначення першого слова, E – для останнього слова і I

(intermediate) – для всіх слів між. Іноді також використовується префікс S (single) – для позначення іменованої сутності, що складається з одного слова.

Отже, завдання зводиться до пословної класифікації. Сьогодні для побудови класифкатора зазвичай тренують нейромережну модель на велику кількість текстів з розміткою іменованих сутностей. В основі моделей зазвичай стоять модулі, які дозволяють враховувати контекст із двох сторін від аналізованого слова.

Ідентифікація таких іменованих об'єктів у тексті була визнана однією з найважливіших підзавдань вилучення інформації з тексту. Найраніші підходи до NER були засновані на орфографічних патернах, синтаксичних відносинах між словами, вилучення за допомогою регулярних виразів. За цими підходами послідувало використання у розв'язанні задачі алгоритмів машинного навчання, наприклад, метод нечітких опорних векторів (FSVM).

```
Mr.  <ENAMEX TYPE="PERSON">Dooner</ENAMEX> met with <ENAMEX TYPE="PERSON">Martin
Puris</ENAMEX>, president and chief executive officer of <ENAMEX
TYPE="ORGANIZATION">Ammirati & Puris</ENAMEX>, about <ENAMEX
TYPE="ORGANIZATION">McCann</ENAMEX>'s acquiring the agency with billings of <NUMEX
TYPE="MONEY">$400 million</NUMEX>, but nothing has materialized.
```

Рисунок 1.2 – Приклад однієї з перших інструкцій тексту іменованими сутностями

Починаючи з використання машинного навчання в експериментах, нейронні мережі стали більш привабливими для дослідників, оскільки вони не вимагали багато ресурсів, специфічних для конкретної предметної галузі, таких як лексикон або формалізація знань. Це давало їм потенціал бути незалежними від предметної галузі [5].

Сучасні архітектури для NER можуть бути класифіковані в залежності від їх подання слів у тексті: засновані на словах (wordlevel), символах (character-level) або на будь-якій їх комбінації. Часто вибір класу архітектур

залежить від особливостей мови, наприклад, character-level подання більш застосовне до обробки китайської мови, що відрізняється особливостями словотвору при листі. У характерних-рівнях архітектури текст сприймається як послідовність окремих символів.

У характерних рівнях архітектури текст сприймається як послідовність окремих символів. Ця послідовність передається через рекурентну нейронну мережу (RNN), що передбачає мітки для кожного символу. Символьні мітки перетворюються на словесні мітки за допомогою постобробки. Приклад character-level архітектури показано на рисунку 1.3:

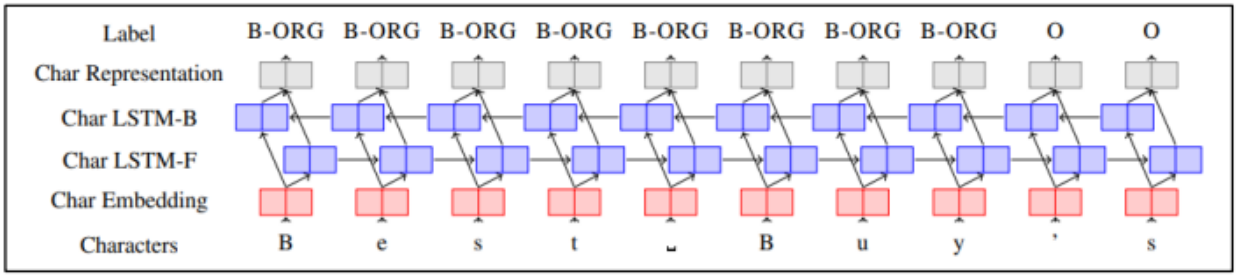

Рисунок 1.3 – Character-level архитектура

В архітектурі word-level текст представлений у вигляді послідовності слів. Слова речення передаються як вхідні дані рекурентним нейронним мережам (RNN), і кожне слово представлене його векторним поданням, як показано на рисунку 1.4.

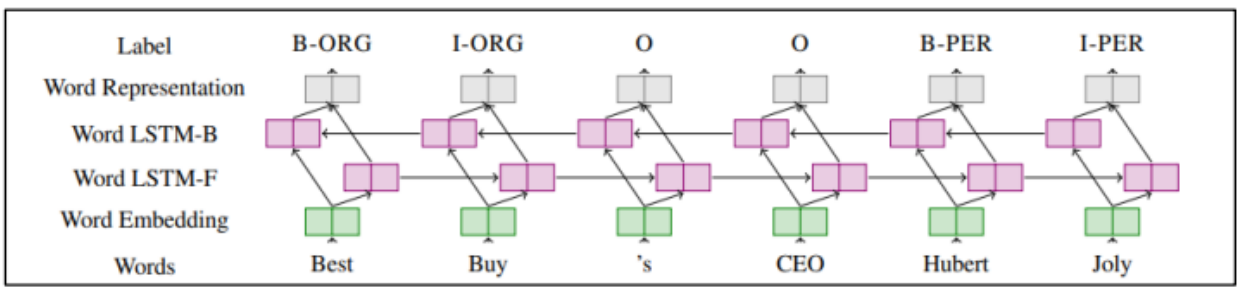

Рисунок 1.4 – Word-level архитектура

Основні підходи NLP:

1. Підхід з урахуванням правил (Rule-Based Approach). У цьому вся підході створюються набори правил, які визначають, які послідовності слів у тексті може бути іменованими сутностями. Ці правила можуть ґрунтуватися на регулярних виразах, шаблонах чи лінгвістичних ознаках. Приклади бібліотек: spaCy (з підтримкою правил, що настроюються), NLTK.
2. Підхід з урахуванням машинного навчання (Machine Learning-Based Approach). У цьому підході використовуються алгоритми машинного навчання, такі як CRF (Conditional Random Fields), LSTM (Long Short-Term Memory) та BERT (Bidirectional Encoder Representations from Transformers), щоб навчити модель розпізнавати іменовані сутності. Модель навчається на розмічених даних, де сутності позначені у тексті. Приклади бібліотек: spaCy (з моделями, що навчаються), Stanford NER, Flair.
3. Поєднаний підхід (Hybrid Approach). Цей підхід поєднує правила та машинне навчання для покращення точності NER. Спочатку можна застосувати правила для виділення сутностей, а потім пропустити текст через модель машинного навчання для уточнення результатів [6].

Логіка роботи NER:

– токенізація тексту, тобто текст розбивається на окремі слова або токени;
– виділення ознак – кожному токену призначаються ознаки, які описують його оточення та контекст, такі як попередні та наступні слова, частини мови та інші лінгвістичні характеристики;
– застосування моделі – модель аналізує ознаки кожного токена і визначає, чи він іменованою сутністю чи ні;

- об'єднання результатів – результати аналізу токенів об'єднуються, щоб сформувати іменовані сутності, і їм призначаються відповідні мітки класів, такі як per (для персон) або org (для організацій);
- постобробка – додаткова обробка для уточнення результатів або виправлення помилок;

Бібліотеки NER надають готові рішення для виділення іменованих сутностей і можуть бути адаптовані під конкретні завдання та типи даних, як наприклад кейси, описані на початку статті. Вибір конкретної бібліотеки залежить від вимог проекту, мови та доступних даних для навчання, звичайно, якщо використовується машинне навчання.

1.4 Підготовча обробка текстових даних

Набори текстових даних (корпусу документів) можуть бути документами різного ступеня довжини, тематики, стилістики і можуть варіюватися від наукових статей до постів у джерелах новин.

Перш ніж використовувати їх під час вирішення завдань обробки природної мови, необхідно провести конструювання ознак текстових даних. Процес конструювання ознак може відрізнятися залежно від конкретної завдання та особливостей використовуваної природної мови. У загальному випадку виділяють такі етапи попередньої обробки текстових даних: токенізація, нормалізація та побудова векторної моделі слів.

1.4.1 Токенізація

Токенізація є першим кроком у будь-якому конвеєрі NLP. Токенізатор розбиває неструктуровані дані та текст природною мовою на фрагменти інформації, які можна розглядати як окремі елементи. Випадки маркера в документі можна використовувати безпосередньо як вектор, що представляє цей документ.

Це негайно перетворює неструктурований рядок (текстовий документ) на числову структуру даних, придатну для машинного навчання. Вони також можуть використовуватися безпосередньо комп'ютером для запуску корисних дій і відповідей. Або вони можуть використовуватися в конвеєрі машинного навчання як функції, які викликають складніші рішення чи поведінку [7].

Токени можуть бути представлені як у вигляді одиничних символів, слів, так і у вигляді n-грам, тобто підпослідовності до n елементів, які були вилучені з текстового рядка. У випадку елементами n грами може бути букви, склади, слова і навіть символи.

Один з варіантів підходів – це токенізація з урахуванням правил, наприклад, поділ на токени по робельним символам і розділових знаків. Але за такого підходу є ризик надмірного збільшення словника за схожими формами слів. Крім того, розділові знаки можуть бути частинами самостійних слів.

Облік описаних вище особливостей дотримується при алгоритмах токенізації на основі підслів. Концепція таких алгоритмів полягає в тому, що слова, що часто зустрічаються, не підлягають розбиттю, а рідкісні слова, навпаки, слід розбивати на значущі підслів. Такі алгоритми особливо корисні при роботі з мовами з багатою морфологією (наприклад, російською) або продуктивним додаванням слів (англійська, німецька та інші) [8].

Кодування пари байтів (BPE) – це алгоритм стиснення, який використовується в обробці природної мови (NLP) для представлення великого словника з невеликим набором одиниць підслів. Він був представлений Sennrich та ін. у 2016 році та широко використовувався в різних завданнях NLP, таких як машинний переклад, класифікація тексту та генерація тексту. Основна ідея BPE полягає в ітераційному об'єднанні найчастішої пари послідовних байтів або символів у текстовому корпусі, доки не буде досягнуто попередньо визначений розмір словника. Отримані підсловні одиниці можна використовувати для представлення оригінального тексту більш компактним і ефективним способом (рис. 1.5).

Поняття, пов’язані з BPE:

– Словник: набір підслів, які можна використовувати для представлення текстового корпусу.
– Байт: одиниця цифрової інформації, яка зазвичай складається з восьми бітів.
– Символ: символ, який представляє письмову або друковану літеру чи цифру.
– Частота: кількість разів, коли байт або символ зустрічаються в текстовому корпусі.
– Об’єднання: процес об’єднання двох послідовних байтів або символів для створення нової одиниці підслова.

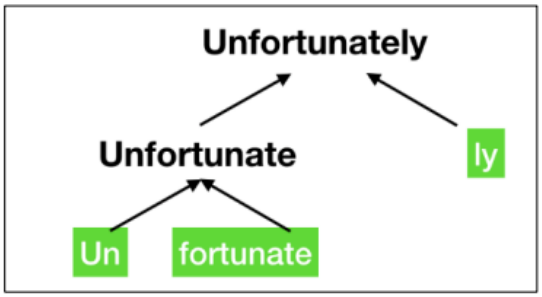

Рисунок 1.5 – Розбій на токени за допомогою BPE

WordPiece – це алгоритм сегментації підслов, який використовується в обробці природної мови. Словник ініціалізується окремими символами мови, потім до словника ітеративно додаються найпоширеніші комбінації символів у словнику. Алгоритм WordPiece був розроблений у рамках вирішення проблем сегментації тексту при створенні системи розпізнавання мови японською та корейською мовами. Він має деякі подібності з BPE: так само задається розмір словника і підслів додаються до словника, доки не буде досягнуто його ліміт. Різниця полягає в умові вибору підслів – замість найчастішої вибирається та, що максимізує ймовірність додавання навчальних

даних до словника. Цей підхід вже встиг проявити себе в реалізації таких моделей нейронних мереж як BERT [9].

1.4.2 Нормалізація

Основна мета нормалізації текстових даних – скоротити загальний обсяг словника мови, що у наслідок допоможе зменшити обсяг обчислень для вирішення обраного завдання. Методи нормалізації залежать часто від особливостей мови, вихідних даних і розв'язуваної задачі, але в загальному випадку це можуть бути наступні операції, наведені нижче.

1. Видалення спеціальних символів, чисел або стоп-слів.
2. Виправлення помилок.
3. Приведення тексту до єдиного регістру.
4. Нормалізація синонімів.
5. Стеммінг.
6. Лематизація.

Зазвичай тексти містять різні граматичні форми однієї й тієї ж слова, і навіть можуть зустрічатися однокореневі слова. Лематизація і стемінг мають на меті привести всі зустрічаються словоформи до однієї, нормальної словникової форми. Лематизація і стемінг – це окремі випадки нормалізації і вони відрізняються.

Стеммінг – грубий евристичний процес, який відрізає "зайве" від кореня слів, часто це призводить до втрати словотворних суфіксів.

Лематизація – це тонший процес, який використовує словник і морфологічний аналіз, щоб у результаті призвести слово до його канонічної формі – лемі.

Відмінність у тому, що стеммер (конкретна реалізація алгоритму стеммінгу – прим. перекладача) діє без знання контексту і, відповідно, не розуміє різницю між словами, які мають різний зміст залежно від частини мови. Однак у стеммерів є свої переваги: їх простіше впровадити і вони

працюють швидше. Плюс більш низька «акуратність» може не мати значення в деяких випадках [10].

1.4.3 Побудова векторної моделі

Для того, щоб використовувати розбиті на токени документи у обчислювальних задачах, необхідно перетворити їх на набір числових векторів. Тому наступний крок у підготовці текстових даних – перетворити лексичні одиниці тексту на послідовність векторів.

Унітарне кодування (one-hot encoding) – метод логічного векторного кодування, де значення 1 присвоюється елементу вектора, що означає існуючий у документі токен. Таке кодування є основою моделі «мішка слів» (bag-of-words, BOW), яка відображає наявність або відсутність токену із заданого словника в документі (рис. 1.6). У такому підході векторизований корпус документів є масивом документів, де кожен документ є вектором довжини словникового запасу корпусу. Очевидним мінусом такого підходу є те, що він описує зміст документа автономно, без урахування контексту.

Рисунок 1.6 – Приклад використання bag-of-words

TF-IDF (TF – term frequency, IDF – inverse document frequency) – статистична міра, що оцінює важливість слова в контексті документа щодо корпусу документів [40]. Міра є добутком частоти терміну (term frequency) та зворотної частоти документа (inverse document frequency).

Частота термінів (TF) – частота кожного слова у документі, яка розраховується як відношення числа входжень слова до загальної кількості слів у документі. Зворотна частота документів (IDF) – інверсія частоти слова, щодо всього корпусу документів.

Позначимо t як слово, d – документ, у якому зустрічається, D – загальний корпус документів. Таким чином, загальний розрахунок TF-IDF міри представлений у формулі (1):

$$TF\text{-}IDF(t, d,D) = tf(t, d) \times idf(t,D) = nt\ |d| \times |D|\ |\{di \in D \mid t \in di\}| \qquad (1),$$

де nt – число входжень слова t у документ, $|d|$ – загальна кількість слів у документі, $|D|$ – загальна кількість документів у корпусі, $|\{di \in D \mid t \in di\}|$ – число документів в корпусі, в яких зустрічається t.

TF-IDF міра дозволяє використовувати вектор документів для порівняння по якій-небудь метриці відстань, наприклад для кластерного аналізу корпусу. Також її використання підходом bag-of-words допомагає вирішити проблему обліку контексту.

Вкладення слів (word embedding) – загальна назва для різних підходів до моделювання природної мови, спрямованих на зіставлення слів із деякого словника векторів невеликої розмірності. Маючи колекцію вкладень слів для певної мови, можна використовувати її для подання окремих слів у реченнях і документах, написаних цією мовою [11].

На відміну від унітарного кодування, такий підхід дозволяє позбутися векторів великої розмірності, тим самим скоротити кількість обчислень і ресурсів, що використовуються для цього (рисунок 1.7).

Word2vec – алгоритм, який використовує штучну нейронну мережу для отримання векторних уявлень слів природною мовою з великого корпусу документів.

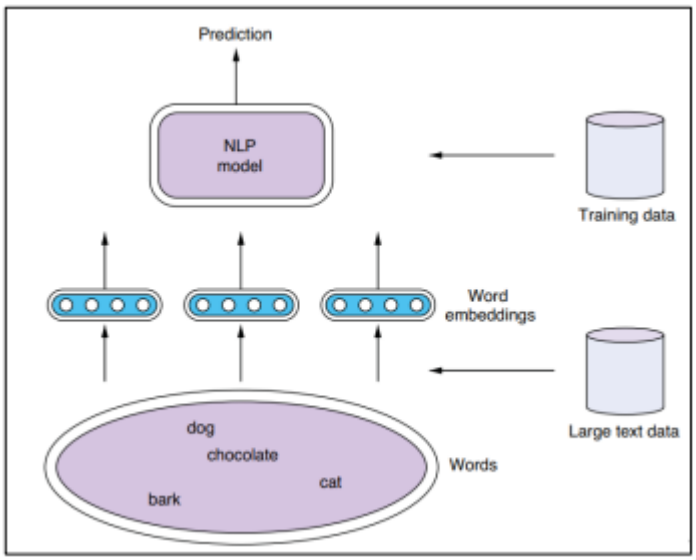

Рисунок 1.7 – Використання вкладень слів у задачі NLP

У word2vec використовуються дві архітектури нейронних мереж: CBoW (Continuous Bag of Words) та Skip-gram. CBoW – архітектура, яка передбачає поточне слово, виходячи з навколишнього контексту. Архітектура типу Skip-gram діє навпаки: вона використовує поточне слово, щоб передбачати навколишні слова (рисунок 1.8).

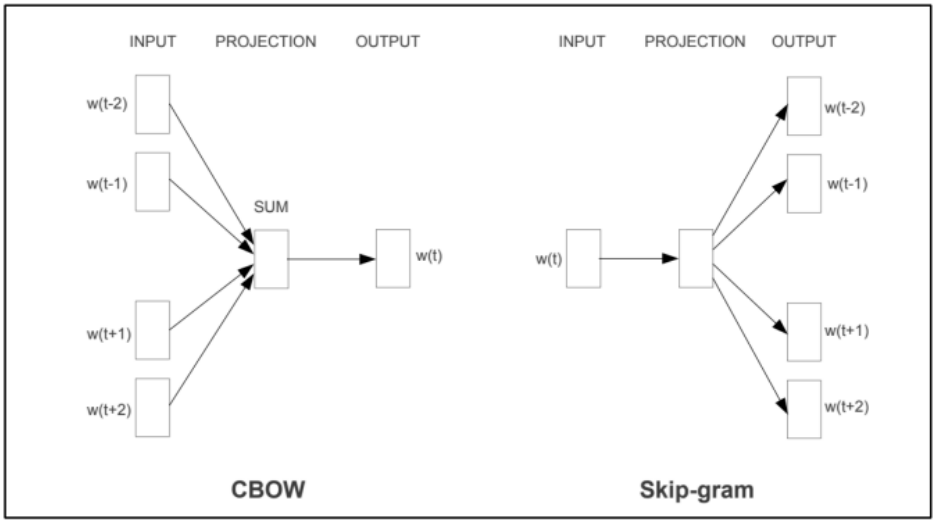

Рисунок 1.8 – CBOW та Skip-gram архітектури

Ефективність алгоритму обумовлена його здатністю групувати вектори схожих слів. Тобто слова, що зустрічаються в тексті поруч із однаковим набором слів, матимуть близькі за косинусною відстанню вектори.

1.5 Огляд та аналіз використання нейроних мереж для виявлення кіберзагроз

1.5.1 Рекурентні нейронні мережі та LSTM

Рекурентна нейронна мережа (Recurrent Neural Network, RNN) – вид штучної нейронної мережі, яка містить цикли у зв'язку з чим добре підходить для обробки послідовностей. У рекурентних нейронні мережі зв'язку між нейронами можуть йти не тільки від нижнього шару до верхнього, але і від нейрона до попереднього значення самого цього ж нейрона або інших нейронів того ж шару [12] (рисунок 1.9).

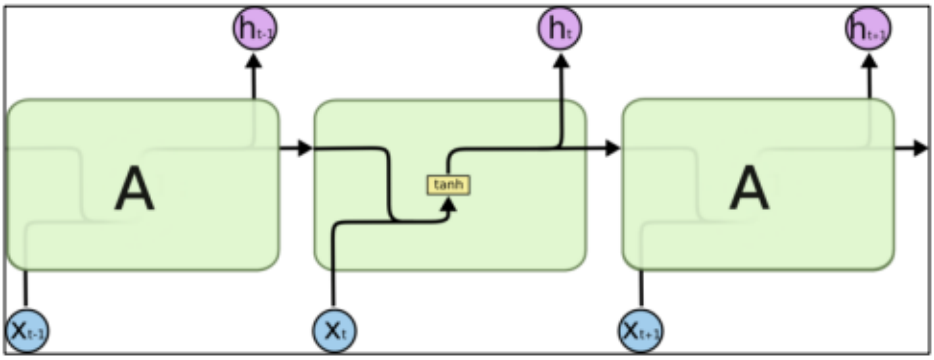

Рисунок 1.9 – Рекурентна нейронна мережа

Такий підхід має свої недоліки, як наприклад «вибух» (exploding gradients) та згасання (vanishing gradients) градієнтів. Якщо ваги помітно збільшують норму вектора градієнта при проході через один шар зворотного розповсюдження, то при проході через N шарів ця норма може зростати або зменшуватися експоненційно від N.

Інша проблема полягає в тому, що в міру збільшення довжини вхідної послідовності, ознаки, отримані на початку послідовності поступово

«забуваються», тому що стан всіх вузлів більшою мірою залежить від ознак, отриманих в останню чергу. Тому класичні RNN не підходять для завдань розпізнавання довгострокових залежностей даних.

Мережі з довгою короткостроковою пам'яттю (Long Short-Term Memory, LSTM) – особливий вид RNN, здатний до навчання довгострокових залежностей. LSTM відрізняється від звичайних RNN своєю здатністю ефективніше працювати з довгостроковими залежностями в послідовності. Це досягається за рахунок використання спеціальних блоків пам'яті, які називаються «комірками пам'яті» (memory cells), які можуть запам'ятовувати інформацію протягом багатьох кроків часу [13].

Класична архітектура LSTM містить рекурентний блок із прихованим станом (комірка пам'яті) і три види вузлів (вентилів): вхідний (input gate), забуваючий (forget gate) та вихідний (output gate) (рисунок 1.10).

Забуваючий вентиль вирішує, яку інформацію із попереднього стану слід забути, вхідний вентиль визначає, яку нову інформацію слід додати, а вихідний – яку інформацію потрібно використовуватиме обчислення виходу.

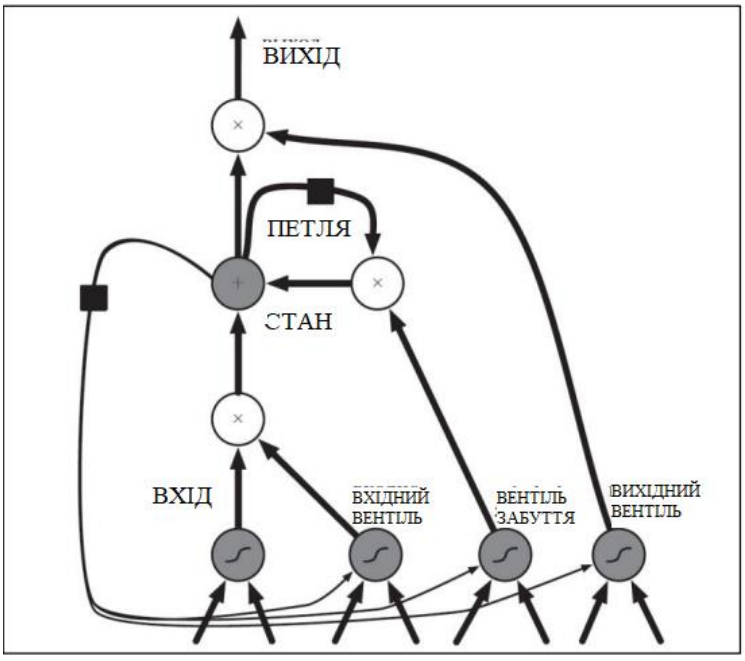

Рисунок 1.10 – Мережа з довгою короткостроковою пам'яттю

1.5.2 Transformer та концепція механізму уваги

Архітектура глибокої нейронної мережі, заснована на механізмах уваги – Transformer, стала заміною рекурентності, яка у свою черга вимагала великих кількостей операцій зі зростанням відстані між токені. Він дозволяє виявити, як кожне слово пов'язане з іншими словами в послідовності, включаючи вихідне аналізоване слово. У такій архітектурі як Transformer використовується не один, а вісім механізмів уваги паралельно, щоб прискорити обчислення. Цей процес називається «багатовузлова увага» (multi-head attention) і крім прискорення обчислень дозволяє проводити більш глибокий та широкий аналіз послідовностей.

Основна складова архітектури Transformer – блок, що складається з кодувальника (encoder) та декодувальника (decoder). Transformer складається із 6 таких блоків. На рисунку 1.11 зліва зображено кодувальник, праворуч – декодувальник Transformer.

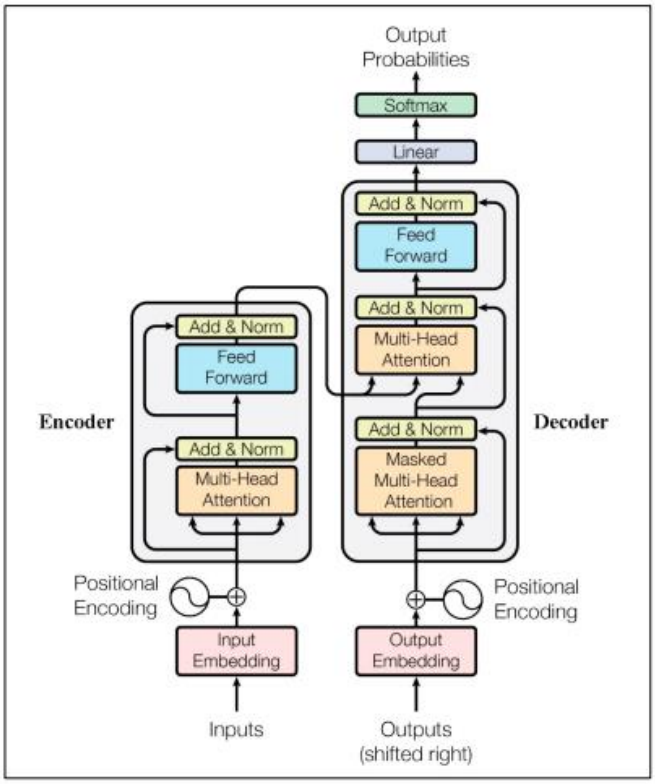

Рисунок 1.11 – Архітектура блоку Transformer

Multi-Head Attention – механізм, який дозволяє блоку "фокусуватися" на різних частинах послідовності. Вхідний вектор послідовності має розмірність 512. В рамках даного механізму вона ділиться на 8 блоків по 64 токена, які паралельно обробляються функцією уваги. Кожен блок має три векторні уявлення: вектор запиту (Q), ключовий вектор (K), вектор значень (V). Функція уваги ("Scaled Dot-Product Attention") зіставляє запит і набір пар ключ-значення з вихідним вектором. Вихідні дані обчислюються як зважена сума значень, де вага, присвоєна кожному значенню, обчислюється функцією сумісності запиту з відповідним ключем. На рисунку 1.12 зображено використання Scaled Dot-Product Attention в багатовузловому механізмі уваги.

Feed-Forward Neural Network – повнозв'язний шар, який застосовує лінійне перетворення до кожного вектора ознак, а потім використовує нелінійну функцію активації ReLU. Ці два компоненти застосовуються послідовно в кожному блоці кодувальника, а потім вихід останнього блоку використовується як приховане уявлення (hidden representation) вхідної послідовності, яке передається декодувальнику.

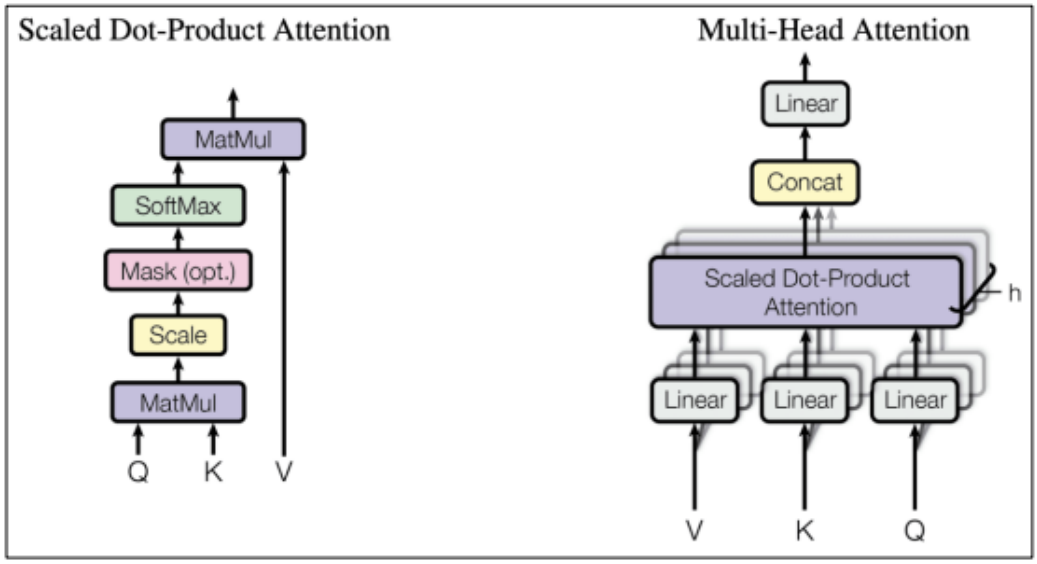

Рисунок 1.12 – Multi-Head Attention

1.5.3 BERT

На відміну від Transformer, BERT (Bidirectional Encoder Representations from Transformers) у своїй архітектурі використовує механізм двоспрямованої уваги. Вона призначена для попередньої підготовки глибоких двонаправлених уявлень немаркованого тексту шляхом спільного узагальнення лівого та правого контексту у всіх шарах.

Основним будівельним блоком архітектури є блок кодувальник із архітектури Transformer. На відміну від Transformer, де ці блоки було 6, базова модель BERT (BERT base) має 12 блоків-кодувальників, а велика (BERT large) – 24 блоки (рисунок 1.13):

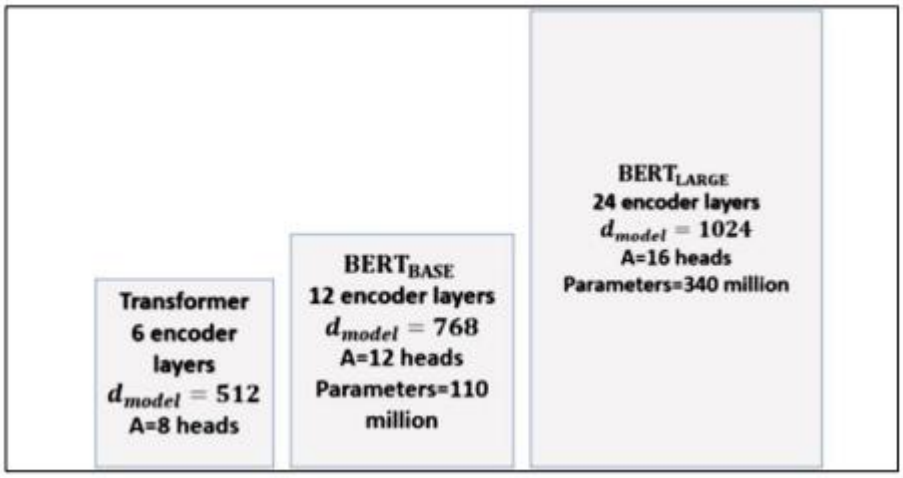

Рисунок 1.13 – Порівняння архітектур BERT з Transformer

Основні кроки у Masked Language Modeling (MLM) виглядають так. Перш ніж використовувати вибірку даних для навчання BERT, 15% випадкових токенів вибірки маскуються токеном "[MASK]". Потім модель намагається передбачити значення замаскованого слова на основі контексту немаскованих токенів послідовності. Для цього додано шар класифікації поверх вихідного вектора кодувальника, значення якого перемножуються на матрицю вкладень слів (embedding matrix).

Для обчислення передбачення маскованих токенів використовується softmax, узагальнення логістичної функції, переважно застосовне у завданнях класифікації, у яких класів більше двох.

Функція втрат обчислюється лише за прогнозуванні замаскованих токенів. Це спричиняє складнощі під час фактичного використання, оскільки в цьому випадку у словнику ще немає токенів «[MASK]». Для Вирішення цієї проблеми, 10% замаскованих токенів замінюються оригінальним токеном, а ще 10% замаскованих токенів замінюються випадковим токеном. Це навчає модель давати уявлення про фактичний токена незалежно від того, чи є вхідний токен у цій позиції [MASK] чи ні.

Завдання пророкування наступної пропозиції (Next Sentence Prediction) була поставлена з метою подання у мовних моделях відносин між реченнями. В рамках цього завдання модель навчається бінарному передбаченню щодо наступної пропозиції: є чи воно разом із поточною частиною однієї підпослідовності [14].

Перед навчанням моделі цього завдання, навчальний набір даних готують так: 50% даних є пари пропозицій, у яких перша та друга пропозиція є загальною підпослідовністю. Решта даних є пари пропозицій, де друга пропозиція вибирається випадково із загального набору даних.

Для підготовки вхідних даних додаються токени «[CLS]» та "[SEP]", що позначають кінець і початок пропозицій відповідно. Також для вхідних даних використовуються вкладення пропозицій (sentence embedding) для позначення поточної та наступної пропозиції у парі. Крім того, з Transformers були запозичені позиційні вкладення (positional embedding), які вказують положення кожного токена у послідовності.

Загальне представлення вхідних даних для BERT показано на рисунку 1.14. При навчанні моделі обидві завдання MLM та NSP вирішуються одночасно, щоб мінімізувати комбіновану функцію втрат обох стратегії.

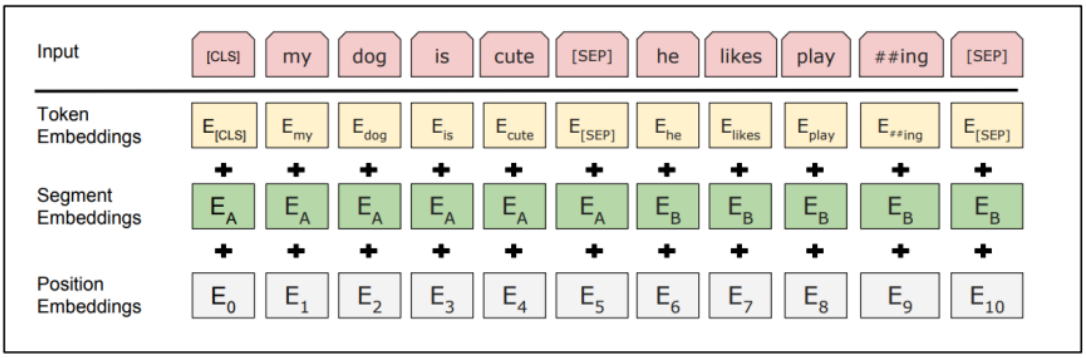

Рисунок 1.14 – Подання вхідних даних для BERT

1.5.3 RoBERTa

Архітектура RoBERTa є більш гнучкою та оптимізованою версією BERT. За будовою обидві архітектури ідентичні, головна відмінність полягає у покращених підходах навчання, які використовує BERT. Першим значним зміною є використання динамічної маскування слів (Dynamic MLM). У BERT маскування послідовності проводиться один раз під час попередньої обробки даних. Щоб уникнути навчання на однаково замаскованій послідовності кожну епоху, RoBERTa вхідна послідовність маскується щоразу перед використанням моделлю.

Ще однією особливістю є те, що RoBERTa не використовує NSP стратегію під час навчання. Автори архітектури провели експерименти, порівнюючи побудовану модель з використанням NSP і без, у зв'язку з чим з'ясували що NSP знижує загальну продуктивність моделі та негативно впливає навчання довгостроковим залежностям [14].

Для отримання попередньої моделі RoBERTa використовували набір даних, що перевищує в загальному обсязі набір для BERT в 10 разів і містила більш різноманітні за стилістикою тексти, наприклад, були додані статті статті. Навчання також тривало довше: 500 000 ітерацій з розміром вибірки 8 000 послідовностей. Загалом було відзначено приріст метрик якості для RoBERTa в порівнянні з BERT large на 2-20% при експериментах на наборах

даних для таких завдань як побудова питання-відповідей, класифікації пропозицій тощо.

1.6 Постановка завдання

Створення веб-сервісу для розпізнавання іменованих сутностей в текстах звітів дослідників кіберзагроз – це завдання, яке можна реалізувати за допомогою певних технологій і інструментів. Нижче наведено загальні кроки та компоненти, які можна використовувати для реалізації цього веб-сервісу:

1. Збір даних: отримання текстових даних звітів від дослідників кіберзагроз.
2. Передобробка тексту: очистка тексту від зайвих символів, перетворення тексту у вигляд, зрозумілий для моделі.
3. Модель для розпізнавання іменованих сутностей (NER): Використання існуючих моделей машинного навчання.
4. Розробка API: створення веб-сервісу, який надає API для взаємодії з моделлю розпізнавання іменованих сутностей.
5. Створення інтерфейсу користувача: розробка веб-інтерфейсу або додатку, який дозволяє користувачам завантажувати тексти та отримувати результати розпізнавання іменованих сутностей.
6. Розгортання і тестування: розгортання веб-сервісу на веб-сервері та тестування його працездатності та продуктивності.
7. Підтримка та покращення: підтримка сервісу, виявлення проблем і виправлення помилок, а також постійне покращення моделі NER для більш точного розпізнавання.

Під час реалізації веб-сервісу для розпізнавання іменованих сутностей в текстах кіберзагроз важливо враховувати аспекти безпеки та конфіденційності даних, особливо якщо звіти містять конфіденційну інформацію. Також важливо забезпечити високу точність розпізнавання для ефективної роботи сервісу.

Висновки з першого розділу

Огляд аналогів показав, що на даний момент існує безліч відкритих програмних рішень для розпізнавання іменованих сутностей у текстах природної мови.

Серед існуючих рішень для реалізації проекту найпопулярнішим був відзначений клас архітектур глибоких нейронних мереж Transformers, зокрема BERT та RoBERTa. Дані архітектури популярні за гнучкість, доступність і двоспрямовані механізми обліку контексту, значущі завдання розпізнавання іменованих сутностей.

2 ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ ВЕБ-СЕРВІСУ

Теоретичні концепції та ключові параметри, необхідні для створення нейронних мереж, є вкрай суттєвими, оскільки вони дають змогу моделювати та навчати складні нелінійні зв'язки у наборах даних. Ваги та зсуви визначають силу зв'язків між окремими нейронами та сприяють виконанню обчислень у мережі. Функції активації вводять нелінійність у виходи нейронів, що робить мережу більш гнучкою та здатною відтворювати складні взаємозв'язки. Цей теоретичний апарат дозволяє нейронним мережам ефективно вирішувати завдання, такі як класифікація, регресія, розпізнавання образів та інші.

Використання теоретичного апарату та належне налаштування параметрів є ключовими для досягнення високої точності та продуктивності мережі при аналізі реальних даних. Це робить нейронні мережі потужним інструментом у різних галузях, включаючи машинне навчання, обробку природної мови та комп'ютерне зору.

2.1 Опис методу навчання нейронної мережі за допомогою бібліотек

TensorFlow – це відкрита програмна бібліотека, розроблена Google для реалізації алгоритмів машинного навчання та глибокого навчання. Вона надає інструменти для створення та навчання різних типів нейронних мереж, включаючи згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN) та інші.

Процес навчання нейронної мережі в TensorFlow включає наступні основні кроки:

1. Підготовка даних: для навчання нейронної мережі необхідно підготувати тренувальні дані, які використовуватимуться для навчання моделі. Це включає етапи завантаження даних, їх попередню обробку, поділ

на тренувальні, перевірочні та тестові набори даних та інші перетворення в залежності від завдання.

2. Створення моделі: за допомогою TensorFlow визначаються архітектура та структура нейронної мережі. Це включає визначення шарів (повнозв'язні шари, згорткові шари, рекурентні шари і т. д.), функцій активації, функцій втрат (loss functions), оптимізаторів та інших параметрів моделі.

3. Компіляція моделі: після створення моделі необхідно скомпілювати її, вказавши функцію втрат, оптимізатор та метрики, які використовуватимуться для оцінки продуктивності моделі в процесі навчання.

4. Навчання моделі: навчання моделі відбувається шляхом передачі тренувальних даних через модель. У процесі навчання параметри моделі змінюються за допомогою методів оптимізації для мінімізації функції втрат, що покращує здатність моделі робити прогнози.

5. Оцінка та тестування моделі: після закінчення навчання модель оцінюється на перевірочних даних з метою оцінки її продуктивності. Після цього модель може бути протестована на тестовому наборі даних з метою оцінки її узагальнюючої здатності.

6. Використання навченої моделі: після успішного навчання модель може бути використана для віщування передбачень на нових даних [15].

TensorFlow надає гнучкий набір інструментів та високорівневі абстракції для навчання нейронних мереж, роблячи процес навчання більш зручним та ефективним для різних завдань машинного навчання та досліджень у галузі штучного інтелекту.

2.2 Розробка алгоритму аналізу даних

Процес розробки алгоритму для аналізу та прогнозування включає кілька ключових етапів. Розглянемо більш детально кожен крок:

1. На початку проводиться збір та попередня обробка даних, необхідних для подальшого аналізу та прогнозування.

Це включає збір даних із різних джерел, очищення від відсутніх або неузгоджених значень, а також перетворення для забезпечення сумісності з вибраною моделлю.

2. Далі важливо вибрати або витягти відповідні функції з набору даних.

Цей процес включає в себе визначення найінформативніших змінних, які сприятимуть аналізу та прогнозуванню. Можуть бути використані методи вибору ознак, такі як кореляційний аналіз, взаємна інформація або експертне знання з предметної області.

Крім того, для створення нових функцій можна використовувати методи виділення ознак, такі як аналіз основних компонентів (PCA) або алгоритми зменшення розмірності.

3. Після підготовки характеристик важливо обрати відповідну модель для вирішення завдання аналізу та прогнозування.

Вибір моделі залежить від природи проблеми, доступних даних і бажаних результатів. Серед популярних варіантів – лінійна регресія, дерева рішень, випадкові ліси, опорні векторні машини та нейронні мережі. Варто розглянути переваги та обмеження кожної моделі для вибору тієї, яка найкраще відповідає конкретній задачі.

4. Обрана модель потребує навчання на підготованому наборі даних.

Це передбачає розділення даних на набори для навчання та перевірки.

5. Наступною стадією є навчання моделі за допомогою навчального набору, після чого її ефективність оцінюється на перевірочному наборі.

Під час процесу навчання можна застосовувати різні методи, такі як перехресна перевірка, регуляризація чи ансамблеві методи, для поліпшення загальної здатності моделі до узагальнення та зменшення перенавчання.

6. Після завершення процесу навчання модель слід оцінити щодо її ефективності.

Це робиться шляхом тестування на окремому тестовому наборі даних, який не використовувався під час навчання. Для оцінки роботи моделі часто використовуються метрики, такі як точність, відсоток правильних відповідей, чутливість, специфічність, F1-оцінка або площа під ROC-кривою. Крім того, для кращого розуміння роботи моделі можуть бути використані візуалізації, наприклад, матриці плутанини або криві ROC-AUC.

Після успішної оцінки моделі вона може бути використана для прогнозування та аналізу. Нові, невідомі дані можна вводити в цю навчену модель, де на основі навчених шаблонів та зв'язків генеруються прогнози. Отримані результати можна подальше проаналізувати для отримання додаткової інформації, яка допоможе у прийнятті обґрунтованих рішень або вживанні відповідних заходів на основі цих прогнозів.

2.3 Вибір засобів розробки

2.3.1 Мова програмування

Python – це одна з найбільш популярних мов програмування у сфері розробки нейронних мереж. Існує кілька причин, чому Python часто обирають для розробки нейронних мереж:

1. Простота використання та читабельність коду.

Python має простий та зрозумілий синтаксис, що робить його легким для вивчення та розуміння, навіть для початківців. Читабельність коду сприяє розвитку та підтримці складних проектів, включаючи роботу з нейронними мережами.

2. Велика кількість бібліотек.

У Python існує велика кількість бібліотек для машинного навчання та роботи з нейронними мережами, таких як TensorFlow, Keras, PyTorch, scikit-learn та інші. Ці бібліотеки забезпечують величезний набір інструментів для розробки, навчання та експериментів з нейронними мережами.

3. Спільнота та підтримка.

Python має велику активну спільноту розробників, яка постійно вносить удосконалення та розвиває інструменти для машинного навчання та нейронних мереж. Це забезпечує широкий доступ до інформації, підтримку та ресурси для вирішення проблем та вдосконалення розроблених моделей.

4. Велика кількість інтеграцій.

Python має велику кількість інтеграцій з іншими мовами та інструментами. Це дозволяє легко поєднувати функціональність Python з іншими технологіями та інструментами, що розширює можливості розробки нейронних мереж.

Ці фактори зробили Python однією з основних мов програмування для роботи з нейронними мережами та машинним навчанням загалом.

2.3.2 Вхідні дані (Datasets)

Проектування систем машинного навчання вимагає коректної формалізації завдання, а саме уявлення об'єктів реального світу векторами ознак у просторі Rn, де кожен вектор уособлює один навчальний приклад, а кожен елемент вектора відповідає певній властивості (ознаці) класу об'єктів, що вивчається (запитів, requests). Формалізація проблеми, вибір технології та алгоритму навчання є нетривіальними завданнями. Універсальних алгоритмів на на даний момент не існує, а тому системи машинного навчання становлять інтерес для наукового дослідження.

Тому з метою навчання та подальшої перевірки ефективності роботи систем машинного навчання для того чи іншого протоколу використовуються відповідні набори даних (Datasets). Як правило, ці набори даних мають таку структуру:

- безліч ознак об'єктів – кінцевий набір ознак запитів, обумовлений роботою в рамках певного протоколу, наприклад, для протоколу TCP це будуть атрибути : елементи заголовка та вмісту, часові параметри;

- навчальна вибірка – набір даних, в яких є як аномальні, так і неаномальні запити, за допомогою яких здійснюється тренування системи;
- тестова вибірка – як правило, безліч, більше навчального, що містить в собі аномальні та неаномальні запити, за допомогою якого здійснюється перевірка роботи навченої системи.

Багато наборів є за фактом стандартом перевірки якості адаптивних алгоритмів. Набір даних для навчання нейронної мережі з розпізнавання кіберзагроз може включати різноманітні дані про вразливості, вторгнення, типи атак, аномальні активності тощо. Такі дані зазвичай вимагаються для тренування моделі на відповідних прикладах.

Наприклад, набір даних може включати:

- Журнали вторгнень: це дані про спроби несанкціонованого доступу до системи, включаючи дані про IP-адреси, часи, спроби входу, використання підозрілих команд тощо.
- Дані про мережевий трафік: інформація про типи пакетів, їх розмір, джерела та призначення, протоколи, використані порти тощо.
- Дані про вразливості та програмне забезпечення: це може бути інформація про відомі вразливості програмного забезпечення, патчі, актуалізації та інші деталі, які допомагають ідентифікувати потенційні кіберзагрози.
- Дані про аномальність в роботі системи: інформація про незвичні або непередбачувані зміни в системі, що можуть свідчити про потенційні атаки або вторгнення.
- Приклади типових кіберзагроз: набір даних може також містити відомі приклади кіберзагроз або атак, щоб навчити модель розпізнавати їх у майбутньому.

Ці дані допомагають моделі навчитися відзначати та класифікувати потенційні загрози або небезпеку, що може допомогти в обліку та захисті систем від кібератак.

Для магістерської роботи було прийнято рішення обрати у якості вхідних даних KDD Cup 1999 Data та на його основі провести навчання нейронної мережі. Він є одним з найвідоміших та широко використовуваних наборів даних для вивчення та тестування алгоритмів виявлення вторгнень (Intrusion Detection Systems, IDS) в комп'ютерних мережах.

Основні характеристики KDD Cup 1999 Data:

1. Тип даних: цей набір даних містить інформацію про мережевий трафік, який був створений за допомогою симуляції різних типів атак та нормальних дій в мережі.
2. Класифікація: дані розділені на кілька класів атак, таких як DoS (Denial of Service), R2L (Unauthorized access from a remote machine), U2R (Unauthorized access to local superuser), а також нормальний мережевий трафік.
3. Кількість атрибутів: набір даних має більше 40 атрибутів, які включають інформацію про TCP/IP пакети, з'єднання, порти, протоколи, середні значення, стандартні відхилення тощо.
4. Об'єм даних: в оригінальному наборі даних було понад 4 мільйони записів.

Цей набір даних став класичним інструментом для тестування алгоритмів виявлення вторгнень та розробки систем захисту від кібератак. Однак, важливо враховувати, що він може бути застарілим і не враховувати сучасні типи кіберзагроз, оскільки дані відносяться до періоду понад 20 років тому.

Цей набір даних включає 43 колонки, які містять як цифрову, так і категоріальну інформацію. Його підготовка для навчання моделей вимагає уважного перетворення категоріальних даних у формат, придатний для моделювання. Також слід провести аналіз з використанням Python для вилучення взаємокорельованих колонок перед навчанням. Це може змінити кількість стовпців у процесі підготовки даних.

Остання колонка у цьому наборі даних вказує на стан системи: значення "normal" означає нормальну роботу системи, а інші вказують на можливі збої або атаки на неї.

Атаки у цьому наборі поділені на чотири категорії:

- DOS (Denial of Service) – атака, під час якої зловмисник заважає нормальній роботі системи, заповнюючи ресурси чи переповнюючи пам'ять, тим самим утруднюючи законним користувачам отримання доступу до машини [16].
- User to Root Attack (U2R) – зловмисник отримує доступ до нормального облікового запису користувача та використовує вразливості для отримання привілеїв на системі.
- Remote to Local Attack (R2L) – зловмисник відправляє пакети до машини через мережу, не маючи облікового запису на цій машині, але використовує вразливості для отримання локального доступу.
- Probing Attack – спроба отримати інформацію про мережу комп'ютерів з метою обійти її безпекові заходи.

Інші колонки датасету:

- duration – час з`єднання у секундах;
- protocol – тип протоколу (tcp, udp, icmp) ;
- service – мережевий сервіс для підключення (http, ftp) ;
- flag —– нормальний чи помилковий статус з`єднання;
- src_bytes – кількість відправлених байт;
- dst_bytes – кількість прийнятих байт;
- wrong_fragment – загальна кількість невірних фрагментів (0, 1, 2…);
- urgent – кількість urgent packets;
- hot – кількість hot індикаторів;
- num_failed_logins – кількість невдалих спроб авторизуватись;
- logged_in – якщо вхід в систему пройшлв успішно – 1, інакше – 0;
- num_compromised – число скомпрометованих станів;

- root_shell – показник, що вказує, чи був отриманий віддалений доступ з привілеями root (суперкористувача);
- num_shells – кількість оболонок (shell), відкритих у системі;
- num_access_files – кількість файлів, до яких був здійснений доступ;
- num_outbound_cmds – кількість вихідних команд (не використовується у даному наборі даних);
- is_host_login – показник, що вказує, чи був вхід в систему здійснений за допомогою віддаленого хоста;
- is_guest_login – показник, що вказує, чи був вхід в систему здійснений як гість
- count – кількість з'єднань до однієї й тієї ж цілі протягом останнього вікна часу;
- srv_count кількість з'єднань до однієї й тієї ж цілі на відповідній службі;
- serror_rate – відсоток з'єднань, які призвели до помилки;
- srv_serror_rate – відсоток з'єднань до цілі на відповідній службі, які призвели до помилки;
- rerror_rate – відсоток з'єднань, які отримали помилкову відповідь;
- srv_rerror_rate – відсоток з'єднань до цілі на відповідній службі, які отримали помилкову відповідь;
- same_srv_rate – відсоток з'єднань до тієї ж самої служби;
- diff_srv_rate – відсоток з'єднань до різних служб;
- srv_diff_host_rate – відсоток з'єднань до різних цілей серед з'єднань до тієї ж самої служби;
- dst_host_count – кількість з'єднань до однієї й тієї ж цілі на віддаленому хості;
- dst_host_srv_count – кількість з'єднань до однієї й тієї ж цілі на відповідній службі на віддаленому хості;
- dst_host_same_srv_rate – відсоток з'єднань до тієї ж самої служби на віддаленому хості.

3 ПРОЕКТУВАННЯ

3.1 Проектування процесу створення і навчання нейронної мережі засобами IDEF0

Для виконання наступного етапу розробки системи – проектування веб-системи була обрана методологія функціонального моделювання SADT (Structured Analysis and Design Technique), що є стандартом IDEF0. IDEF0 – використовується для створення функціональної моделі, що відображає структуру і функції системи, а також потоки інформації і матеріальних об'єктів, що зв'язують ці функції.

Для нових систем застосування IDEF0 направлено на визначення вимог і зазначення функцій для подальшої розробки системи, що відповідає поставленим вимогами і реалізує виділені функції. Результатом застосування IDEF0 до деякої системи є модель цієї системи, що складається з ієрархічно упорядкованого набору діаграм, тексту документації та словників, пов'язаних один з одним за допомогою перехресних посилань [17]. Для відображення основних робот по створенню і навчанню нейронної мережі використуємо діаграму IDEF0, головна роботи якої зображено на рисунку 3.1:

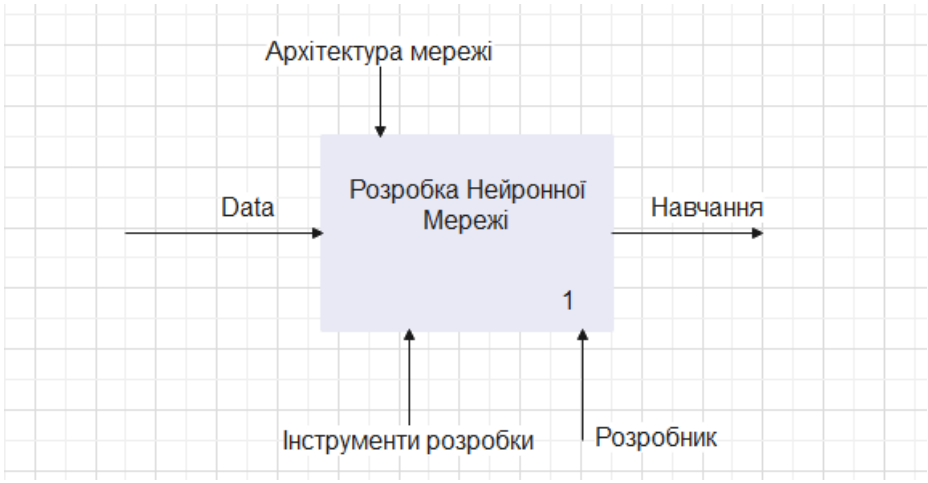

Рисунок 3.1 – Головна робота діаграми IDEF0

У подальшому функціональному моделюванні необхідно розкласти контекстну діаграму веб-системи для віртуального аукціону на менші компоненти, щоб досягти необхідної деталізації системи. Це дозволить розробляти складові системи за модульним принципом.

Декомпозиція передбачає обов'язкову експертизу процесів, відображених на діаграмах, для відповідності реальним процесам. Таку експертизу проводять фахівці з вибраної предметної області. Якщо виявляються неузгодженості в процесах, після обговорення виправляють помилки та усувають неточності. Більш детальний рівень декомпозиції виконується лише після коригування та уточнення на попередньому рівні. Механізм декомпозиції моделі дозволяє досягти необхідного рівня деталізації системи.

Наступний крок – це декомпозиція першого рівня, в процесі якої головну роботу буде поділено на послідовність декількох її складових (рис. 3.2):

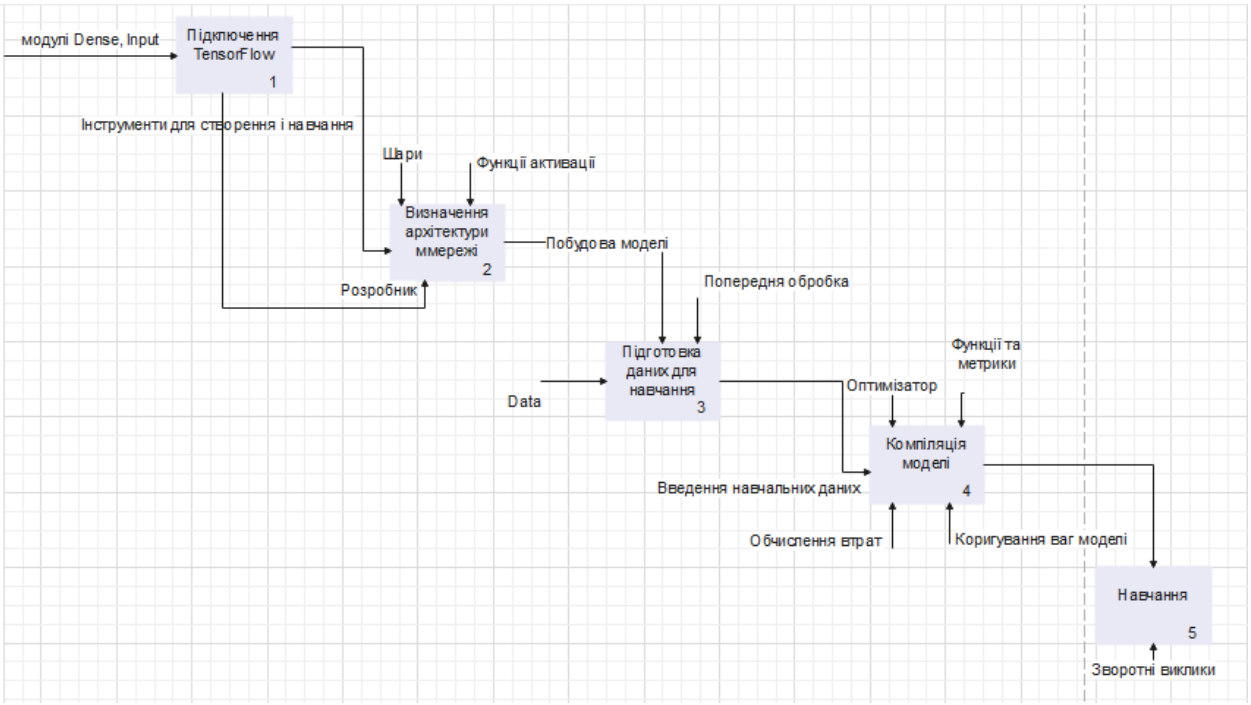

Рисунок 3.2 – Декомпозиція першого рівня діаграми IDEF0

Початковий код включає імпорт необхідних бібліотек, таких як TensorFlow, який зазвичай імпортується під псевдонімом tf, та окремі модулі, наприклад, Dense, Input і зворотні виклики, які імпортуються з tensorflow.keras.layers відповідно. Ця бібліотека надає необхідні інструменти для створення та навчання нейронних мереж.

Після цього визначається архітектура нейронної мережі, що включає в себе визначення кількості та типів шарів, їх функцій активації та розмірів входів. Наприклад, Dense шар представляє собою повністю зв'язаний шар, а вхідний шар визначає форму входу мережі. Ці шари використовуються для побудови моделі нейронної мережі.

Після визначення архітектури мережі необхідно підготувати дані перед їхнім використанням у навчанні. Зазвичай це включає завантаження та попередню обробку навчальних даних, таких як очищення, нормалізація та вилучення функцій.

Після підготовки даних модель нейронної мережі компілюється та навчається. Компіляція включає в себе визначення оптимізатора, функції втрат та метрик для оцінки моделі. Навчання моделі включає введення навчальних даних, обчислення втрат та коригування ваг моделі за допомогою зворотного поширення та градієнтного спуску.

Для моніторингу ходу навчання можна використовувати зворотні виклики, які забезпечують різноманітні можливості, такі як збереження контрольних точок моделі та виконання передчасної зупинки.

Після завершення процесу навчання модель можна оцінити за допомогою тестових даних, щоб оцінити її ефективність на нових невідомих даних. Такі показники, як точність, запам'ятовування та оцінка F1, можуть допомогти зрозуміти, наскільки добре модель узагальнює нові дані.

Отже, реалізація навчання нейронної мережі за допомогою TensorFlow та його модулів передбачає імпорт бібліотек, визначення архітектури мережі, підготовку даних, компіляцію, навчання, використання зворотних викликів для моніторингу та оцінку продуктивності моделі. Ці кроки є важливими для

навчання нейронної мережі та можуть бути використані для адаптивного захисту комп'ютерної системи.

3.2 Проектування веб-системи за методологією Workflow Diagramming

Методологія Workflow Diagramming використовується для проектування веб-систем, щоб детально відобразити послідовність робочих процесів та їх взаємозв'язки в системі. Вона базується на створенні графічних діаграм, які уявно показують послідовність етапів в робочому процесі. Кожна діаграма відображає різні аспекти функціонування системи, що дозволяє зрозуміти, як працюють різні компоненти, як вони взаємодіють між собою та яким чином виконуються робочі завдання. Основна мета – візуалізувати та проаналізувати робочі процеси системи для подальшого вдосконалення та оптимізації роботи всього комплексу (рис. 3.3).

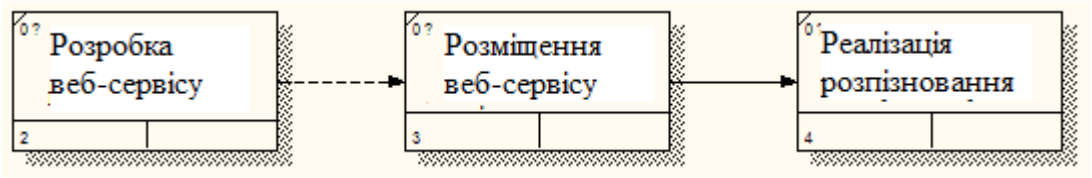

Рисунок 3.3 – Діаграма декомпозиції веб-сервісу

Для реалізації основної роботи системи проведено декомпозицію та визначені наступні послідовності виконання робіт: першою роботою системи є «Розробка веб-сервісу», наступна робота – «Розміщення веб-сервісу». Зв'язок між цими роботами означає, що робота-приймач може завершитись ще до закінчення роботи-джерела.

Остання робота «Реалізація розпізнавання» пов'язана з блоком «Розміщення веб-сервісу» старшим зв'язком, що означає завершення всіх

попередніх робот для того, щоб можна було без перешкод проводити перевірку даних на наявність кіберзагроз.

Виконаємо наступну декомпозицію для блоку «Розробка веб-сервісу» – отримуємо три блоки, які представлено роботами з двома перехрестями (рис. 3.4). Обидва елементи-перехрестя на діаграмі – це «асинхронне І». Перше з них вказує на те, що наступні етапи, а саме «Розробка програмного коду» та «Створення інтерфейсу», можуть розпочинатися не одночасно, але обов'язково мають бути запущені. Коли вони завершуються і виходять на перехрестя, передбачено, що ці етапи завершуються, проте це може відбутися у різний час.

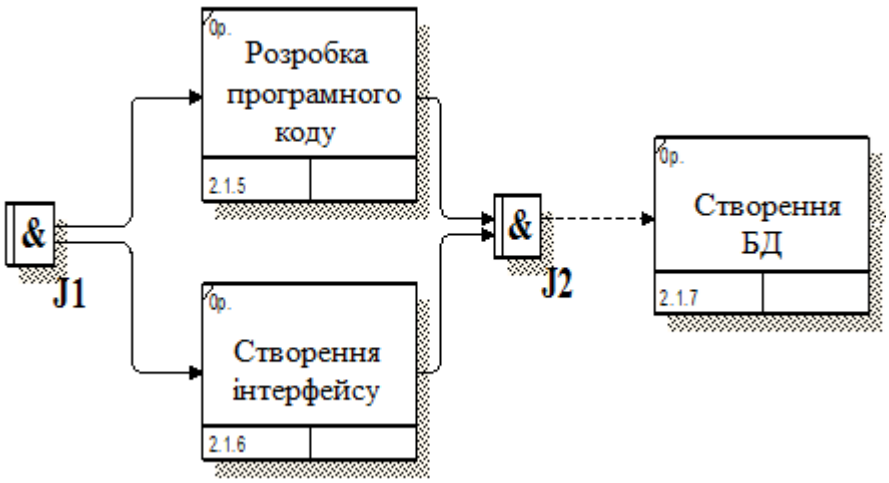

Рисунок 3.4 – Діаграма декомпозиції роботи «Розробка веб-сервісу»

Блок «Реалізація розпізнавання» є ключовим, так як саме тут виконується основне завдання проекту – детектування кіберзагроз. Виконаємо декомпозицію цього блоку. Маємо два перехрестя «асинхронне І». В першому випадку мова йде про три етапи: «Завантаження даних», «Формування датасетів», і головне, «Навчання нейронної мережі» повинні бути виконані. І тільки після цього буде можливість реалізувати сервіс детектування кіберзагроз (рис. 3.5).

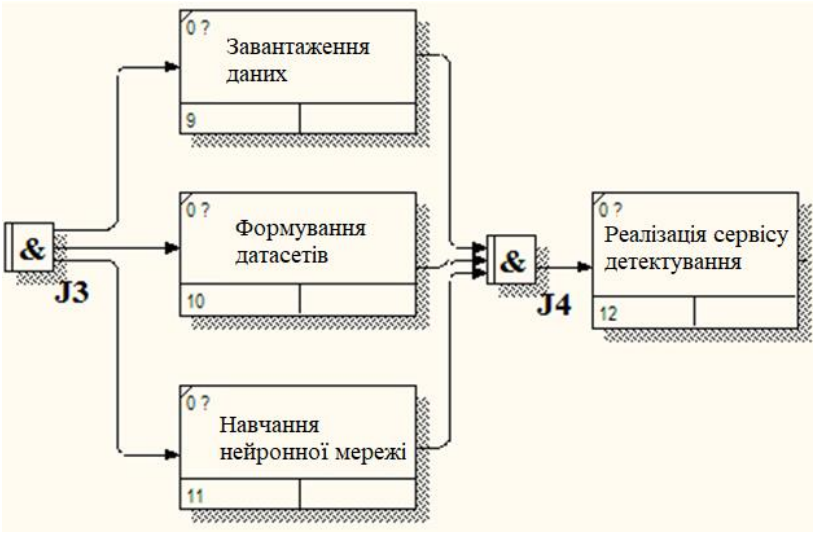

Рисунок 3.5 – Діаграма декомпозиції роботи «Реалізація розпізнування»

3.3 Моделювання процесу передобробки даних

Діаграма станів мови моделювання UML показує, як об'єкт переходить із одного стану до іншого. Діаграми станів служать моделювання динамічних аспектів системи. Ця діаграма корисна при моделюванні життєвого циклу об'єкта. Діаграми станів можуть бути вкладені одна в одну, утворюючи вкладені діаграми для детальнішого уявлення станів окремих елементів моделі. Для розуміння семантики конкретної діаграми станів необхідно представляти особливості поведінки моделюється сутність, а також мати загальні відомості з теорії кінцевих автоматів.

Від інших діаграм діаграма станів відрізняється тим, що описує процес зміни станів лише одного екземпляра певного класу – одного об'єкта, причому об'єкта реактивного, тобто об'єкта, поведінка якого характеризується його реакцією на зовнішні події (рис. 3.6).

Ці етапи передобробки даних є загальними і можуть варіюватися в залежності від специфіки даних та конкретних вимог задачі розпізнавання кіберзагроз.

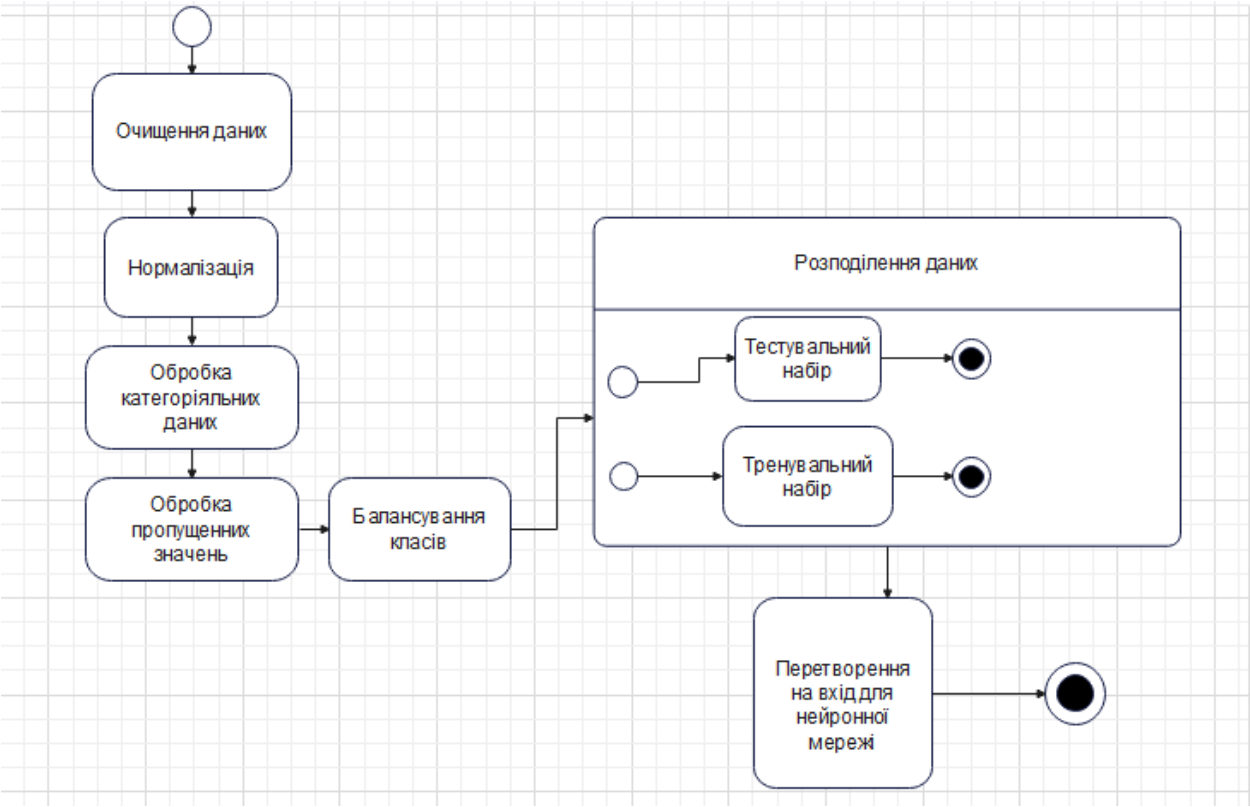

Рисунок 3.6 – Діаграма станів передобробки даних

Щоб підготувати дані для навчання нейронної мережі для розпізнавання кіберзагроз, потрібно виконати кілька етапів передобробки тексту:

1. Очищення даних: видалення зайвих або непотрібних стовпців (колонок) або рядків (записів), які можуть бути неінформативними або несприятливо впливати на навчання моделі.
2. Нормалізація даних: масштабування числових значень до одного діапазону для покращення швидкості та стабільності навчання моделі.
3. Обробка категоріальних даних: перетворення категоріальних даних у числовий формат, наприклад, за допомогою кодування категорій (наприклад, методом кодування One-Hot).
4. Обробка пропущених значень: розгляд можливостей заповнення відсутніх даних або вилучення записів, де бракує важливих атрибутів.

5. Балансування класів (опціонально): якщо класи в наборі даних незбалансовані (наприклад, аномалії зустрічаються значно рідше, ніж нормальні дії), слід розглянути методи балансування класів для покращення навчання моделі.
6. Розділення на тренувальний та тестовий набори: розділіть дані на дві частини: одну для навчання моделі (тренувальний набір) та іншу для перевірки ефективності моделі (тестовий набір).
7. Перетворення на вхід для нейронної мережі: слід підготувати дані у формат, який може бути використаний в якості вхідних значень для нейронної мережі.

3.4 Діаграма варіантів використання

Діаграма варіантів використання – (діаграма прецедентів, сценарій використання, use case) – дозволяє уявити типи ролей та їх взаємодію із системою. Проте не показує порядок виконання кроків. Діаграма варіантів використання описує, який функціонал програмної системи, що розробляється, доступний кожній групі користувачів.

Діаграма варіантів використання UML є основною формою вимог до системи/програмного забезпечення для нової недостатньо розробленої програми. Варіанти використання вказують на очікувану поведінку (що), а не на точний спосіб її здійснення (як). Випадки використання після визначення можуть позначатися як текстовим, так і візуальним представленням (тобто діаграмою варіантів використання). Ключова концепція моделювання варіантів використання полягає в тому, що воно допомагає нам проектувати систему з точки зору кінцевого користувача. Це ефективна техніка для передачі інформації про поведінку системи в термінах користувача шляхом визначення всієї видимої ззовні поведінки системи.

Зображує функціональні вимоги (те, що система може зробити) з точки зору користувача. Може описуватись текстом або у вигляді діаграми (рис. 3.7).

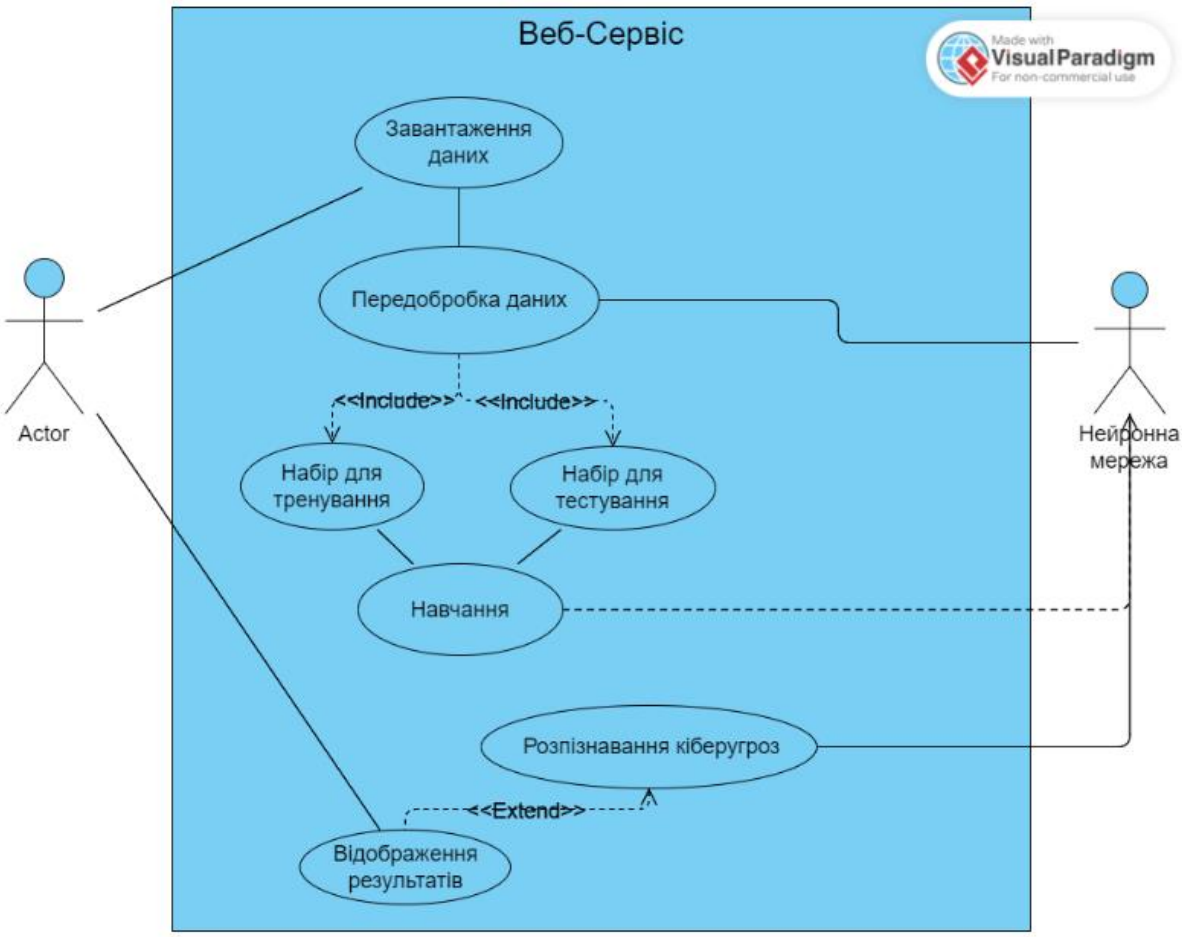

Рисунок 3.7 – Діаграма прецендентів

3.5 Підхід до розпізнавання іменованих сутностей

Алгоритм Inside-Outside-Begin (IOB) є одним із методів розмітки даних, що використовуються у розв'язанні задачі Named Entity Recognition (NER). Цей метод допомагає розмічати слова у тексті відповідно до сутностей, таких як імена, організації, місця тощо.

Покроковий алгоритм IOB зазвичай застосовується так: вихідний текст має бути розбитий на токени (слова чи фрази). Кожен токен має бути

позначений або як частина іменованої сутності, або як нічого (що не належить до суті).

Кожен токен позначається міткою відповідно до його ролі у сутності. Мітки мають три можливі значення: "I" (всередині сутності), "O" (поза сутністю) і "B" (початок сутності).

- "B" означає початок сутності.
- "I" позначає токени, що прямують безпосередньо за токеном "B" і також відносяться до цієї ж сутності.
- "O" означає токени, які не є частиною сутності.

Покроковий алгоритм IOB:

1. Пройти по кожному току тексту.
2. Якщо токен є початком сутності, то позначте його міткою "B", а всі наступні токени сутності позначайте міткою "I", доки не закінчиться сутність.
3. Токени, що не належать до сутності, позначаються міткою "O".

Приклад розмітки тексту за допомогою IOB: вихідний текст: "Google розмістив свій офіс у Нью-Йорку"

Розмітка IOB:

```
"Google" - B-ORG (початок сутності організації)
"розмістив" - O (не відноситься до сутності)
"свій" - O (не відноситься до сутності)
"офіс" - O (не відноситься до сутності)
"в" - O (не відноситься до сутності)
"Нью-Йорку" - B-LOC (початок сутності розташування)
```

Це дозволяє алгоритмам машинного навчання краще розуміти межі сутностей під час навчання на розмічених даних і робить процес вилучення іменованих сутностей точнішим.

4 РЕАЛІЗАЦІЯ

4.1 Архітектура веб-сервісу

Діаграма компонентів для веб-сервісу розпізнавання кіберзагроз може включати різні ключові елементи, необхідні для його функціонування. Це загальний опис ключових компонентів, які можуть бути включені в діаграму компонентів для веб-сервісу для виявлення кіберзагроз. Однак конкретні компоненти та їх взаємодія можуть відрізнятися залежно від специфіки та вимог веб-сервісу.

Ось основні компоненти, які можуть бути присутніми на такій діаграмі:

1. Інтерфейс користувача (UI):
   - Фронтенд програми: компонент, з яким користувачі взаємодіють із веб-сервісом. Може бути у вигляді веб-сторінки, мобільного додатка та інше. В даному випадку це веб-сторінка.
2. Веб-сервер: приймає запити від користувачів через інтерфейс та передає їх відповідним компонентам.
3. Обробка даних:
   - Модуль попередньої обробки: компонент, який відповідає за попередню обробку даних, що надходять від користувачів перед передачею на моделі або алгоритми виявлення кіберзагроз.
   - Модель машинного навчання або алгоритм виявлення загроз. Цей компонент містить моделі, алгоритми або інструменти для виявлення кіберзагроз на основі оброблених даних.
   - Сховище даних: компонент, який може включати базу даних або інші методи зберігання даних, необхідних для навчання моделей або зберігання результатів виявлення загроз. Будемо використовувати просто сховище даних, так як на даний момент сервіс тільки тестується і працює з одним набором даних.

Діаграма компонентів представлено на рисунку 4.1. Це клієнт-серверна архітектура, яка в майбутньому буде доповнена компонентами.

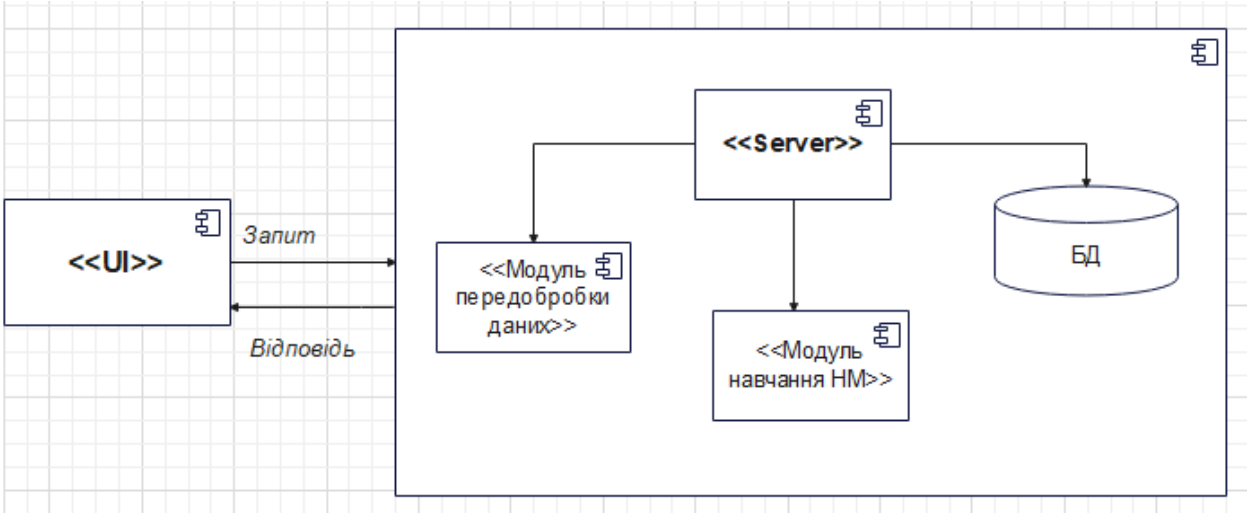

Рисунок 4.1 – Діаграма компонентів

Використання цієї архітектурної концепції для розробки веб-системи передбачає розподіл веб-додатка на кілька компонентів згідно їх функціональності: інтерфейс, бізнес-логіка та доступ до даних.

4.2 Програмна реалізація опрацювання даних

Оскільки дані з KDD Cup 1999 є числові та категоріальні значення, необхідно перетворити їх у текстовий формат, який можна розмітити мітками IOB. Наприклад, можна створити текстовий файл, де кожен рядок є окремою подією або транзакцією, яка описує мережевий трафік, і розмітити його за допомогою міток IOB.

Приклад текстового представлення даних із KDD Cup 1999 для розмітки IOB:

```
    TCP,http,SF,181,5450,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0
.00,0.00,0.00,0.00,1.00,0.00,0.00,9,9,1.00,0.00,0.11,0.00,0.00,0
.00,0.00,0.00,normal
    UDP,other,SF,105,146,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0
.00,0.00,0.00,0.00,1.00,0.00,0.00,19,19,1.00,0.00,0.05,0.00,0.00
,0.00,0.00,0.00,anomaly
    ...
```

Тут кожен рядок представляє окрему подію мережного трафіку. Вона містить різні атрибути (наприклад, протокол, тип сервісу, прапори та інша інформація), що описують цю подію, а останній стовпець представляє мітку, що вказує, чи це подія "normal" (нормальна) або "anomaly" (аномалія).

Для розмітки з мітками IOB краще вибрати певні атрибути (наприклад, тип сервісу, прапори) і використовувати їх для створення текстового подання з мітками IOB, де кожен атрибут буде токен, позначений міткою IOB залежно від його ролі по суті.

Розглянемо реалізацію алгоритму IOB для обробки міток у тексті на Python (функція generate_iob_labels()):

```
def generate_iob_labels(sentence, entities):
    labels = ['O'] * len(sentence.split())
    for entity in entities:
        entity_words = entity.split()
        try:
            start_idx = sentence.index(entity_words[0])
            end_idx = sentence.index(entity_words[-1])
```

Початок іменованої сутності:

```
labels[start_idx] = 'B'  # (begin)
            for i in range(start_idx + 1, end_idx + 1):
                labels[i] = 'I'  # всередені сутністі (inside)
        except ValueError:
            pass
    return labels
```

Приклад використання:

```
iob_labels = generate_iob_labels(text, entities)
print(iob_labels)
```

Функція prepTrainProcAE() відповідає за підготовку та обробку даних для нейронної мережі. Ця функція виконує навчання моделі та порівнює/візуалізує метрики на тестових та тренувальних наборах даних.

Процес підготовки даних складається з кількох кроків: спочатку вихідний датафрейм розділяється на два окремі, один містить рядки з нормальними значеннями в цільовому стовпці (вказуючи на "normal"), а інший – рядки з аномальними значеннями.

```
import tensorflow as tf
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, LSTM, Dense,
Bidirectional

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.models import Sequential
from sklearn.model_selection import train_test_split
```

Створення токенізатору:

```
tokenizer = Tokenizer
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)
```

Далі потрібно додавання нулів для рівної довжини послідовності та виконати поділ даних на навчальну та тестову виборки для нейронної мережі:

```
padded_sequences = pad_sequences(sequences, padding='post')
x_train, x_test, y_train, y_test =
train_test_split(padded_sequences, labels, test_size=0.2)
```

Наступникй крок – це створення моделі RNN та шару ембеддінгу слів:

```
model = Sequential()
model.add(Embedding(input_dim=len(tokenizer.word_index)+1,
output_dim=64))
```

Двонаправлений шар LSTM:

```
model.add(Bidirectional(LSTM(64)))
```

Вихідний шар із сигмоїдною функцією активації для бінарної класифікації та компіляція моделі:

```
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',loss='binary_crossentropy',
metrics=['accuracy'])
```

Наступний крок, це навчання моделі:

model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))

```
data = pd.read_csv('path_to_file.csv')
```

Після перегляду перших кількох рядків даних для ознайомлення йде попередня обробка даних та перетворення категоріальних стовпців на числові за допомогою LabelEncoder:

```
print(data.head())
labelencoder = LabelEncoder()
for col in data.columns:
    if data[col].dtype == 'object':
        data[col] = labelencoder.fit_transform(data[col])
```

Наступний крок це поділ даних на навчальний та тестовий набори:

```
X = data.drop('цільова змінна', axis=1)# Залежить від
параметру даних

y = data[цільова змінна'] # Залежить від параметру даних
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Етап ініціалізації та навчання моделі:

```
model = RandomForestClassifier(n_estimators=100,
random_state=42)
model.fit(X_train, y_train)
```

Отримання передбачень на тестовому наборі даних і оцінка продуктивності моделі:

```
predictions = model.predict(X_test)

print(classification_report(y_test, predictions))
print("Accuracy:", accuracy_score(y_test, predictions))
```

Цей програмний блок, який завантажує дані, перетворює категоріальні змінні, поділяє дані на навчальний та тестовий набори, ініціалізує модель та оцінює її продуктивність.

На даний час у БД веб-сервісу дані заносяться у вигляді окремого файлу. Для подальшої роботи з отриманною інформацією буде розроблена детальна БД для зручного зберігання окремих даних:

1. Звіти про кібератаки: інформація про атаки, типи загроз, час, місцезнаходження, методи атак та інші деталі про події, що відбулися.
2. Сутності для виявлення: набір даних, що містить текстову інформацію зі звітів про кібератаки, що підлягає аналізу щодо іменованих сутностей, таких як імена організацій, адреси, IP-адреси, дати, імена людей та інші ключові сутності.
3. Результати обробки та класифікації: результати роботи системи виявлення та класифікації іменованих сутностей, можливо, включаючи тегування, категоризацію або інші мітки для кожної виявленої сутності.
4. Індекси та метадані: додаткова інформація, наприклад індекси для прискорення пошуку, метадані для забезпечення консистентності даних або додаткові атрибути, необхідні для обробки та аналізу.
5. Логи та аудит інформації: журнали дій користувачів, системних подій та результати аудиту, щоб відстежувати зміни, оновлення та доступ до даних.

5 ТЕСТУВАННЯ ВЕБ-СЕРВІСУ

Тестування відіграє важливу роль у розробці програмного забезпечення та створенні веб-сервісів з кількох причин:

1. Забезпечення якості: тестування допомагає виявити помилки, дефекти та недоліки в програмному продукті, що дозволяє виправити їх до випуску продакшн. Це забезпечує більш високу якість та надійність роботи програми.
2. Підвищення задоволення клієнтів: якісне програмне забезпечення, вільне від помилок та з надійною роботою, сприяє задоволеності користувачів. Це також може покращити репутацію вашої компанії та залучити більше клієнтів.
3. Економія часу та ресурсів: виявлення та усунення помилок на ранніх етапах розробки коштує дешевше та потребує менше часу, ніж виправлення проблем після випуску програми. Тестування дозволяє уникнути додаткових витрат на виправлення помилок у майбутньому.
4. Поліпшення безпеки: тестування допомагає виявити вразливість безпеки програмного забезпечення. Це дуже важливо, особливо для веб-сервісів, що працюють з конфіденційними даними або обробляють критичну інформацію.
5. Підвищення продуктивності: тестування продуктивності дозволяє виявити вузькі місця в роботі програми або веб-сервісу, що дозволяє оптимізувати її для більш ефективної роботи.
6. Дотримання вимог та стандартів: деякі галузеві стандарти, регуляції або вимоги до безпеки вимагають проведення тестування для підтвердження відповідності продукту певним стандартам та нормам.

В цілому, тестування є невід'ємною частиною процесу розробки програмного забезпечення, забезпечуючи його якість, надійність, безпеку та відповідність вимогам користувачів та стандартам галузі.

Головна сторінка веб-сервісу представлена на рисунку 5.1:

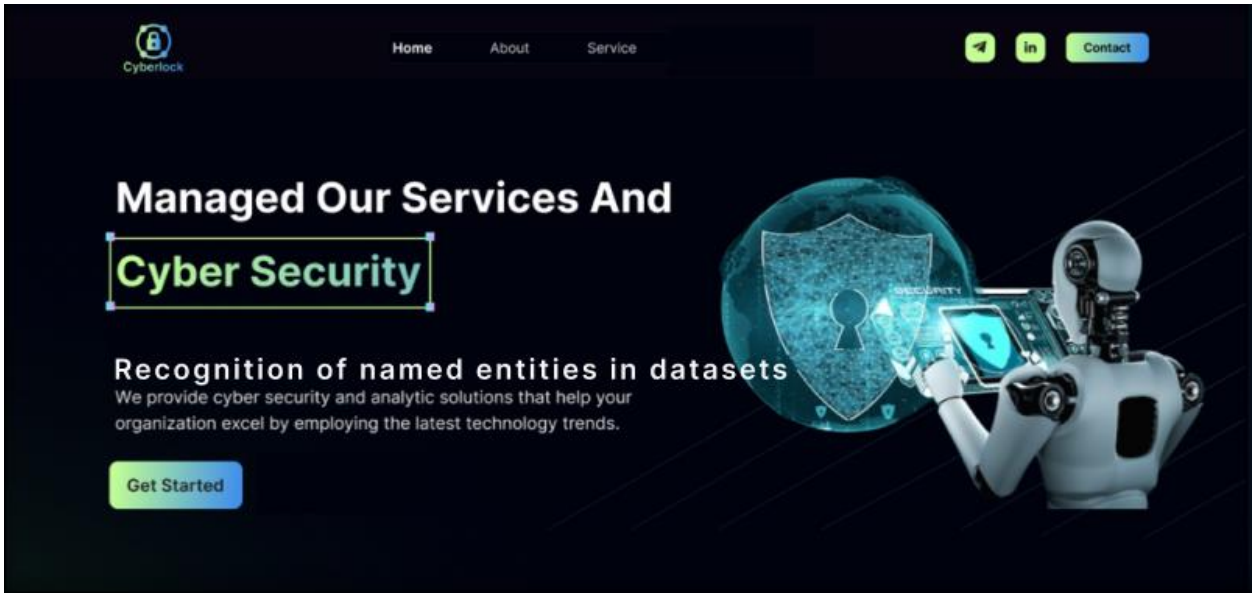

Рисунок 5.1 – Головна сторінка

Як майбутній повноцінний веб-продукт, її структура містить хедер з логотипом та меню на три позиції, а саме, «Home», «About», «Service», а також блоком для регістрації та кнопки відправлення повідомлення. Основний блок сторінки – інформаційний, у футері розміщено кнопку «Get Started».

На першому етапі роботи із сервісом потрібно завантажити датасет. У даний час нейрона мережа працює з одним набором даних. Тож відразу після запуску можна запустити етап навчання і подивитись які результати по епохам ми маємо. Якщо навчання пройде успішно по виборкам даних, результати зберігаються у файл для подальшої обробки.

Коли ви навчаєте нейронну мережу для виявлення вторгнень на даних KDD Cup 1999 або будь-яких інших даних, метрики оцінки моделі відіграють

важливу роль в оцінці її продуктивності. Ось деякі з основних метрик, які часто використовуються для оцінки моделі виявлення вторгнень:

1. Точність (Accuracy): це метрика, яка вимірює загальну правильність передбачень моделі.

Вона визначається як ставлення правильно класифікованих зразків до всіх зразків. Однак для завдання виявлення вторгнень, де класи незбалансовані (наприклад, вторгнення становлять невелику частину загального обсягу даних), точність може дати спотворене уявлення про продуктивність моделі.

2. Повнота (Recall): повнота вимірює здатність моделі виявляти всі фактичні позитивні випадки даних.

Вона окреслюється ставлення правильно класифікованих позитивних зразків до загальної кількості фактичних позитивних зразків. Висока повнота означає, що модель здатна виявляти більшість вторгнень.

3. Точність передбачення (Precision): це метрика, що вимірює частку правильно класифікованих позитивних зразків серед усіх позитивних прогнозів моделі.

Вона визначається як відношення правильно класифікованих позитивних зразків до загальної кількості позитивних прогнозів моделі. Висока точність означає, що модель робить менше хибних позитивних прогнозів.

4. F1-мера (F1-Score): це гармонійне середнє між повнотою та точністю.

F1 міра зазвичай використовується для збалансованої оцінки моделі при незбалансованих класах. Вона дозволяє оцінити модель, враховуючи як точність, і повноту.

5. Матриця помилок (Confusion Matrix): це таблиця, яка показує кількість вірних і невірних передбачень кожного класу.

Матриця помилок допомагає візуалізувати продуктивність моделі, дозволяючи оцінити хибнопозитивні та хибнонегативні передбачення.

При навчанні нейронної мережі даних KDD Cup 1999, ці метрики можуть бути використані для оцінки ефективності моделі. Часто буває корисно застосовувати комбінацію кількох метрик, щоб отримати повніше уявлення у тому, наскільки добре модель працює у різних аспектах завдання виявлення вторгнень.

Приклад процесу навчання:

```
Epoch 1/10
2023-11-16 13:53:32.243442: W tensorflow/core/common_runtime/type_inference.cc:339]
391/391 [==============================] - 43s 88ms/step - loss: 0.6566 - accuracy:
0.5580 - val_loss: 0.5489 - val_accuracy: 0.7505
Epoch 2/10
391/391 [==============================] - 21s 54ms/step - loss: 0.4354 - accuracy:
0.7937 - val_loss: 0.3724 - val_accuracy: 0.8234
Epoch 3/10
391/391 [==============================] - 22s 54ms/step - loss: 0.3451 - accuracy:
0.8468 - val_loss: 0.3403 - val_accuracy: 0.8521
Epoch 4/10
391/391 [==============================] - 21s 52ms/step - loss: 0.3224 - accuracy:
0.8601 - val_loss: 0.3332 - val_accuracy: 0.8573
Epoch 5/10
391/391 [==============================] - 21s 52ms/step - loss: 0.3168 - accuracy:
0.8623 - val_loss: 0.3291 - val_accuracy: 0.8620
Epoch 6/10
391/391 [==============================] - 21s 52ms/step - loss: 0.3088 - accuracy:
0.8658 - val_loss: 0.3370 - val_accuracy: 0.8615
Epoch 7/10
391/391 [==============================] - 22s 52ms/step - loss: 0.3060 - accuracy:
0.8692 - val_loss: 0.3271 - val_accuracy: 0.8448
Epoch 8/10
391/391 [==============================] - 21s 52ms/step - loss: 0.3033 - accuracy:
0.8714 - val_loss: 0.3249 - val_accuracy: 0.8583
Epoch 9/10
391/391 [==============================] - 21s 51ms/step - loss: 0.3017 - accuracy:
0.8695 - val_loss: 0.3293 - val_accuracy: 0.8385
Epoch 10/10
391/391 [==============================] - 21s 52ms/step - loss: 0.2995 - accuracy:
0.8717 - val_loss: 0.3217 - val_accuracy: 0.8630
```

У висновку показано результат виконання кожної епохи. Виводиться номер епохи, кількість пакетів, затрачене на час епохи та помилки. Loss – це розрахована функція втрати, mean_absolute_error – це метрика, указана при компіляції. Якщо вказати додаткові метрики, то вони теж будуть у висновку. Усі помилки з префіксом «val_» – це те саме для валідаційного набору даних.

У цьому вигляді модель можна навчити, але набагато ефективніше це можна зробити, якщо використовувати функціонал зворотних виходів. З

їхньою допомогою можна здійснити ранню зупинку навчання для боротьби з перенавчанням, візуалізувати дані та багато іншого.

Ось приклад деяких із них, реалізовані на мові пайтон:

По-перше, якщо помилка не зменшується протягом зазначеної кількості епох, то процес навчання переривається і модель ініціалізується вагами з найнижчим показником параметра "monitor":

```
early_stopping = tf.keras.callbacks.EarlyStopping(
     monitor='val_loss',patience=2,mode='min',
     restore_best_weights=True)
```

Тут:

- 'val_loss' вказується параметр, яким здійснюється рання зупинка. Зазвичай це функція потрети на валідаційному наборі (val_loss);
- patience=2 це кількість епох після закінчення яких закінчиться навчання, якщо показники не покращаться;
- mode='min' вказує, в яку сторону має бути покращена помилка;
- restore_best_weights=True – якщо параметр встановлений у true, то після закінчення навчання модель буде ініціалізована вагами з найнижчим показником параметра "monitor".

Далі слід зберігти модель для подальшого завантаження (вказуємо шлях до папки, де буде збережена модель):

```
model_checkpoint = tf.keras.callbacks.ModelCheckpoint(
     filepath='my_model',
     monitor='val_loss',
     save_best_only=True,
```

якщо параметр встановлено в true, то зберігається тільки найкраща модель:

```
   mode='min'
)
```

Зберігає логи виконання навчання, які можна буде переглянути у спеціальному середовищі TensorBoard:

```
tensorboard = tf.keras.callbacks.TensorBoard(
     log_dir='log', #шлях до папки, де будуть збережені логи
)
```

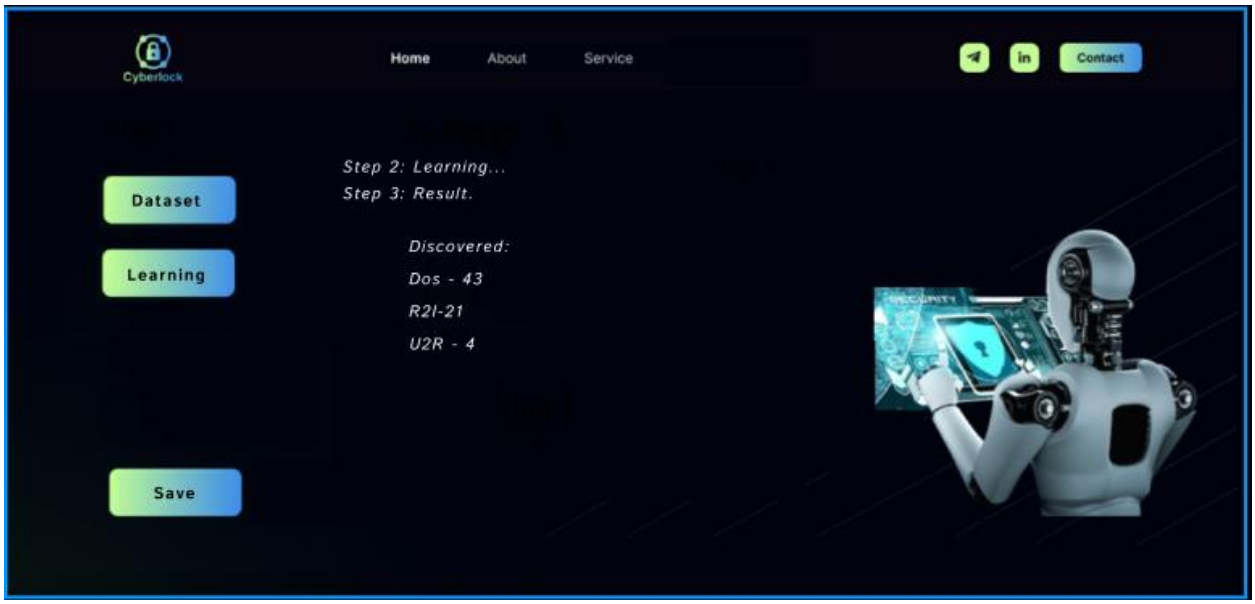

Рисунок 5.2 – Результат

5.1 Тестування роботи нейронної мережі

Тестування моделі нейронної мережі проводилося за таких самих умов, що й під час навчання моделі. Розмір тестової вибірки становить 20% вихідного набору даних. Оскільки модель нейронної мережі, що використовується, має обмеження в вигляді 512 вхідних токенів, необхідно оцінити який обсяг вхідного тексту оптимальний для отримання найкращої якості розпізнавання іменованих сутностей.

Потім були проведено експерименти, у яких модель розмічала підвиборки вихідних документів у вигляді 2, 5, 10, 15 пропозицій. Результати експериментів показані у таблиці5.1.

Таблиця 5.1 – Тестування навченої нейронної мережі на різних розмірах підвиборок

Розмір підвиборки (в реченні)	F1-міра (%)
2 72, 49	2 72, 49
5 80,45	5 80,45
10 78,25	10 78,25
15 78,50	15 78,50

З результатів експериментів можна бачити, що оптимальний розмір підвибірки, що розмічається нейронною мережею, дорівнює 5 реченням. За таких умов якість розпізнавання іменованих сутностей досягає значення F1 міри 80,45%.

5.2 Функціональне тестування

Функціональне тестування веб-системи спрямоване на перевірку функціональності програми, щоб переконатися, що всі його компоненти працюють відповідно до заданих специфікацій та очікувань користувачів. Основною метою функціонального тестування є перевірка того, чи виконуються функції системи відповідно до її вимог.

Після завершення реалізації веб-сервісу було проведено його функціональне тестування відповідно до вимог поставленого на етапі проектування. Результати функціонального тестування відображені у таблиці 5.2.

Таблиця 5.2 – Функціональне тестування

№	Предумова	Дія	Очікуваний результат	Тест пройдено?
1	Веб-сервіс запущений	Клієнт відправляє POST-запит веб-сервіс з використанням методу extract_entities(). Тіло запиту містить текст для аналізу	Сервіс надає інформацію про вилучених з вхідного тексту іменованих сутності	так
2	Веб-сервіс запущений	Клієнт відправляє POST запит на веб-сервіс з використанням методу HTML. Тіло запиту містить	Сервіс надає htmlсторінку з розміченим вхідним текстом.	так

Головною метою було навчити нейронну мережу розпізнавати нормальні дані, тому тренувальний набір складається зі значень першого датасету, що містить нормальні дані.

Після завершення навчання на тренувальних даних, ми використовуємо навчену модель для тестування на "нормальних" даних з першого датасету, а також на даних з другого датасету (які містять аномальні значення). Крім того, модель застосовується до змішаного набору даних, який складається як з нормальних, так і з аномальних даних.

Отримані результати показують, що помилка реконструкції на нормальних даних є малою, у той час як помилка реконструкції на аномальних даних виявляється значно більшою. Це свідчить про те, що модель успішно впізнає та розуміє нормальні дані, водночас виділяючи всі інші дані як аномальні, які потребують уваги в системах безпеки.

5.3 Подальший розвиток-нові набори даних

Існує кілька відкритих наборів даних, які можна використовувати для навчання нейронних мереж або інших методів машинного навчання для детектування різних сутностей у тексті, включаючи кіберзагрози.

Деякі з них включають:

- CoNLL-2003: датасет для отримання іменованих сутностей (Named Entity Recognition, NER) з новинних статей англійською, німецькою та іспанською мовами.
- OntoNotes: цей набір даних містить розмічені тексти новин, телефонних розмов та інших текстів англійською з розміткою іменованих сутностей, парсингом і синтаксичними анотаціями.
- ACE (Automatic Content Extraction): цей набір даних містить розмічені англомовні тексти з анотаціями іменованих сутностей, відносин та подій у статтях новин.
- MITRE ATT&CK: даний набір даних відображає різні види кіберзагроз, атак та технік зловмисників, який може бути використаний для аналізу та навчання в галузі кібербезпеки.
- Kaggle Datasets: на платформі Kaggle є різні датасети для навчання моделей детектування кіберзагроз та інших сутностей. Деякі з них включають дані про шкідливі програмні атаки, мережні вразливості та інші аспекти кібербезпеки.

Ці набори даних надають різні типи анотацій та розмітки, які можуть бути використані для навчання моделей різними мовами та різних аспектів кібербезпеки.

З точки зору подальшого розвитку проекту в першу чергу слід тестувати її функціонал на нових наборах даних. Було проведено аналітичний огляд таких даних, у разі, якщо відсутня можливість отримати звіти з кібербезпеки від реальних підприємств, можна скористатися одним з наступних прпикладів.

5.3.1 База знать MITRE ATT&CK

MITRE ATT&CK – це керівництво з класифікації та опису кібератак і вторжень. Він був створений Mitre Corporation і випущений в 2013 році. Замість того, щоб розглянути результати атаки, він визначає тактику, що вказує на те, що атака триває.

MITRE ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) є базою знань про кібератаків, яка описує тактики, методи та інструменти, використовувані кіберпреступниками та зловмисниками. Тематика «Software» в MITER ATT&CK включає інформацію про програмне забезпечення, його використання та залучення уразливостей, а також про методи, які можуть бути використані зловмисниками для експлуатації цього програмного забезпечення.

Набір включає 5 категорій сутностей: malware (шкідливе програмне забезпечення), indicators of compromise (індикатор компрометації), system (система), organization (організація) та vulnerability (уразливість). Підсумковий розмір набору становить 4530 розмічених сутностей.

В рамках цієї тематики «Software» виділяються такі тактики і методи:

1. Execution (Виконання): Як зловмисники можуть запускати шкідливі програми та код на цільовій системі через різні прийоми, такі як команди командного рядка, виконання програм та інше.
2. Persistence (Стійкість): Як зловмисники можуть забезпечити постійність свого доступу до системи, наприклад, шляхом створення або зміни програмного коду для автозапуску або використання служб і заданих планувальників.
3. Privilege Escalation (Підвищення привілеїв): методи, за допомогою яких зловмисники можуть спробувати отримати додаткові привілеї та повноту права на цільову систему.

4. Defense Evasion (Виявлення виявлення): Тактики для обходу засобів захисту, які дозволяють зловмисникам уникнути виявлення та блокування.
5. Credential Access (Доступ до вхідних даних): як зловмисники можуть отримувати доступ до вчетних даних, таким як паролі та ключі доступу.
6. Discovery (Обнаружение): Методи пошуку інформації про цільове середовище, такі як сканування портів, мережеві запити та інше.
7. Lateral Movement (Бокове переміщення): способи, якими зловмисники можуть поширювати ваш доступ у мережі чи системі для досягнення нових цілей чи системи.
8. Collection (Збір даних): Як злоумышленники могут собирать и копировать данные с целевых систем.
9. Command and Control (Контроль і управління): як зловмисники можуть контролювати пошкоджені системи та встановлювати канали зв’язку з серверами керування.
10. Exfiltration (Винос даних): Методи ексфільтрації даних із зараженою системою.

MITRE ATT&CK надає широкий огляд тактик і методів, які використовуються зловмисниками, що допомагає організаціям краще боротися з загрозами та захистити свої системи та дані.

5.3.2 Набір DNRTI

В якості набору даних використовується набір DNRTI (Набір даних для Named entity Recognition in Threat Intelligence). ДНІЗТ містить більше 300 звітів про кіберугрози, отриманих з відкритих веб-сайтів за тематикою TI. Дані звіти включають в себе 175 220 анотованих слів, що належать 13 класам: хакерська організація, атака, образець файлу, команда ІБ, інструмент атаки, час, ціль, область, відрасль, організація, спосіб, уязвимость, особливість.

Набір даних був розмічений за допомогою схеми IOB. Приклад розміченого пропозиції із зразка набору DNRTI показаний на рисунку 5.3.

```
The O
admin@338 B-HackOrg
has O
largely O
targeted O
organizations O
involved O
in O
financial B-Idus
, O
economic B-Idus
and O
trade B-Idus
policy I-Idus
, O
typically O
using O
publicly B-Tool
available I-Tool
RATs I-Tool
such O
as O
Poison B-Tool
Ivy I-Tool
, O
as O
well O
some O
non-public B-Tool
backdoors I-Tool
. O
```

Рисунок 5.3 – Приклад зразка з набору даних DNRTI

ВИСНОВКИ

Детектування сутностей у звітах з кіберзагроз має критичне значення у сфері кібербезпеки з наступних причин:

1. Ідентифікація загроз: аналіз текстових звітів про кіберзагрози дозволяє виявляти ключові загрози, такі як види атак, ідентифікувати використовувані техніки, інструменти та спосіб дії зловмисників.
2. Прогнозування та запобігання: шляхом детектування сутностей можна передбачати можливі вразливості, передбачувані цілі атак, а також моделі поведінки зловмисників. Це допомагає розробити ефективніші стратегії запобігання кіберзагрозам.
3. Обробка даних: автоматична обробка текстових даних звітів про кіберзагрози прискорює аналіз та полегшує виявлення вразливостей, необхідних для реагування та запобігання атакам.
4. Створення бази знань: результати детектування сутностей можуть бути використані для створення бази знань, що дозволить покращити системи детектування, навчання та контролю безпеки у майбутньому.
5. Автоматизація процесу: використання алгоритмів та методів машинного навчання для детектування сутностей у звітах про кіберзагрози дозволяє автоматизувати процес аналізу, що економить час та ресурси фахівців з кібербезпеки.
6. Швидкий відгук: завдяки швидкому та ефективному детектуванню сутностей у звітах про кіберзагрози, компанії та організації можуть швидше реагувати на нові загрози та атаки, забезпечуючи безпеку своїх систем та даних.

Отже, детектування сутностей у звітах про кіберзагрози відіграє ключову роль у забезпеченні кібербезпеки, надаючи цінну інформацію для розробки більш ефективних стратегій захисту та запобігання кібератакам.

У рамках магістерської кваліфікаційної роботи було розроблено веб-сервіс розпізнавання іменованих сутностей у текстах звітів дослідників кіберзагроз.

У ході роботи було вирішено всі поставлені завдання:

1) проведено огляд існуючих аналогів та наукової літератури;
2) підготовлено набір даних;
3) виконано проектну частину роботи;
4) реалізовано, навчено та протестовано обрану топологію штучної нейронної мережі;
5) розроблено веб-сервіс для розпізнавання іменованих сутностей у текстах звітів дослідників кіберзагроз;
6) проведено тестування розробленого сервісу.

Як подальший розвиток проекту слід розглянути додаткове тестування, яке буде включати реальні звіти з кібербезпеки. Це дозволить використовувати веб-сервіс у комерційних цілях.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments. URL: https://www.sciencedirect.com/science/article/pii/S2352484721007289 (дата звертання 11.09.2023)
2. Шість найдієвіших платформ аналізу інформації про загрози для ваших команд. URL: безпекиhttps://habr.com/ru/companies/roi4cio-articles/528514/ (дата звертання 11.09.2023)
3. What Is a Threat Intelligence Feed? URL: https://www.techtarget.com/whatis/definition/threat-intelligence-feed (дата звертання 11.09.2023)
4. Leitner E., Rehm G., Moreno-Schneider J. Fine-grained named entity recognition in legal documents. // International Conference on Semantic Systems. Springer, 2019. – 272–287 pp.
5. Collins M., Singer Y. Unsupervised Models for Named Entity Classification. // Proc. of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, 1999. – 100–110 pp
6. Що є NER сервіси та як їх застосовують у бізнесі від А до Я (практика). URL: https://habr.com/ru/articles/763542/ (дата звертання 15.09.2023)
7. Tokenization in NLP: Types, Challenges, Examples, Tools. URL: https://neptune.ai/blog/tokenization-in-nlp (дата звертання 15.09.2023)
8. Hapke H., Howard C., Lane H. Natural Language Processing in Action: Understanding, analyzing, and generating text with Python. Simon and Schuster, 2019. – 544 p.

9. Devlin J., Chang M.W., Lee K., Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018. URL: https://arxiv.org/abs/1810.04805 (дата звертання 25.09.2023 г.)
10. Основи Natural Language Processing для тексту. URL: https://habr.com/ru/companies/Voximplant/articles/446738/ (дата звертання 25.09.2023)
11. Mikolov T., Chen K., Corrado G., Dean J. Efficient estimation of word representations in vector space. 2013. URL:https://arxiv.org/abs/1301.3781 (дата звертання 25.09.2023).
12. Introduction to Recurrent Neural Network. URL: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/ (дата звертання 02.10.2023).
13. A Gentle Introduction to Long Short-Term Memory Networks by the Experts. URL: https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/ (дата звертання 02.10.2023).
14. Як покращується обробка мови за допомогою моделі Google BERT з відкритим кодом. URL: https://www.unite.ai/uk/how-language-processing-is-being-enhanced-through-googles-open-source-bert-model/ (дата звертання 12.10.2023).
15. Create production-grade machin. URL: https://www.tensorflow.org/ (дата звертання 19.10.2023).
16. KDD Cup 1999 Data. URL: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (дата звертання 23.10.2023).
17. IDEF0. Знайомство з нотацією та приклад використання. URL: https://trinion.org/blog/idef0-znakomstvo-s-notaciey-i-primer-ispolzovaniya (дата звертання 25.10.2023).