МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Система для детектування об'єктів з використанням нейронних мереж»

Виконав студент 2 курсу групи МІС22зф
спеціальності 122 Комп'ютерні науки
Пушкаренко Кирило Григорович

Керівник канд.техн.наук, доцент
Гнатовська Ганна Арнольдівна

Рецензент к. геогр. наук, доцент,
Кузніченко С.Д.

Одеса 2023

АНОТАЦІЯ

на магістерську кваліфікаційну роботу

«Система для детектування об'єктів з використанням нейронних мереж»

Студента Пушкаренко Кирила Григоровича

Збір інформації у мережі інтернет може займати багато часу у зв’язку з великим об’ємом джерел для пошуку та аналізу. Програми-парсери допомагають прискорювати цей процес. Робота присвячена дослідженню технологій парсінгу різних за структурою веб-сайтів, виявлення їх недоліків, концепції рішень та вибору алгоритму парсінгу, враховуючи структуру конкретних веб-ресурсів.

Тема магістерської роботи «Система для детектування об'єктів з використанням нейронних мереж».

Актуальність магістерської роботи полягає в розробці та застосуванні системи парсінгу даних, яка призначена детектувати необхідні об’єкти на веб-сайтах та зберігати їх для подальшого аналізу і використання.

Метою роботи є розробка програмної конфігурації для системи детектування об’єктів, що забезпечить спрощення збору інформації у мережі інтернет та прискорить процес її обробки. Функціональні можливості системи повинні охоплювати вибір класу об’єктів та джерело для збору даних.

Об'єкт дослідження – процеси проектування та розробки моделі системи-парсеру, що забезпечить користувачу швидкий доступ до актуальної інформації за допомогою використання нейронної мережі.

Предмет дослідження – засоби реалізації процесу парсігну та алгоритму роботи згортованної нейронної мережі.

В роботі було проведено дослідження технологій розробки систем парсінгу та аналіз архітектур нейронних мереж, які використовують для детектування об’єктів. Виконано програмну реалізацію системи.

Практична цінність роботи полягає в тому, що створена система забезпечує зручний засіб для збору даних за певними категоріями.

Ключові слова: ПАРСІНГ, ВЕБ-САЙТ, ПОСИЛАННЯ, КОНТЕНТ, ЗГОРТОВАНІ НЕЙРОННІ МЕРЕЖІ, DOCUMENT OBJECT MODEL, БАЗА ДАНИХ, ІНФОРМАЦІЯ, ПРОЕКТУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Магістерська робота містить 66 сторінок, 3 таблиці, 25 рисунків, 15 посилань.

SUMMARY

for a master's thesis

«System for detecting objects using neural networks»

by student Pushkarenko Kyrylo

Gathering information on the Internet can take a lot of time due to the large volume of sources to search and analyze. Parser programs help speed up this process. The work is devoted to the research of parsing technologies of different website structures, identifying their shortcomings, the concept of solutions and the choice of a parsing algorithm, taking into account the structure of specific web resources.

The topic of the master's thesis "System for detecting objects using neural networks".

The relevance of the master's thesis lies in the development and application of a data parsing system, which is designed to detect the necessary objects on websites and store them for further analysis and use.

The purpose of the work is to develop a software configuration for the object detection system, which will facilitate the collection of information on the Internet and speed up the process of its processing. The functionality of the system should include the choice of object class and source for data collection.

The object of research is the process of designing and developing a parser system model that will provide the user with quick access to relevant information using a neural network.

The subject of research is means of implementing the parsign process and the convolutional neural network algorithm.

In the work, research was carried out on technologies for the development of parsing systems and analysis of neural network architectures used for object detection. 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 collecting data by certain categories.

Keywords: PARSING, WEBSITE, LINKS, CONTENT, SNAP NEURAL NETWORKS, DOCUMENT OBJECT MODEL, DATA BASE, INFORMATION, DESIGN, SOFTWARE.

The master's thesis contains 66 pages, 3 tables, 25 figures, 15 links

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

БНМ – Багатошарової нейронної мережі

НМВП – Нейронні мережі високого порядку

НМХ – НМ Хопфілда

СНМК – Самоорганізовуючі нейронні мережі Кохонена

ТПО – Тестування програмного забезпечення

CAPTCHA – Completely Automated Public Turing test to tell Computers and Humans Apart

CSV – Comma-Separated Values

DOM – Document Object Model

DFD – Data Flow Diagrams

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

JSON – JavaScript Object Notation

PHP – Hypertext Preprocessor

URL – Uniform Resource Locator

CNN – згорткові нейронні мережі

MySQL – вільна реляційна система управління базами даних

PostgreSQL – вільна об'єктно-реляційна система управління базами даних

PyTorch – бібліотека Python

Web – система доступу до пов'язаних між собою документів на різних комп'ютерах, підключених до Інтернету

SQLite – СКБД

ВСТУП

Часто інформація, яку потрібно отримати в Інтернеті, є надто великою, щоб її можна було видобути вручну. Ось чому компанії, які використовують інструменти веб-скрапінгу, можуть збирати більше даних за менший проміжок часу за менших витрат. Крім того, компанії, які отримують вигоду від збирання даних, отримують крок попереду в конкуренції між конкурентами в довгостроковій перспективі.

Людський мозок обробляє величезну кількість інформації в ту секунду, коли людина бачe зображення. Кожен нейрон працює у своєму власному рецептивному полі та пов’язаний з іншими нейронами таким чином, що вони покривають усе поле зору. Подібно до того, як кожен нейрон реагує на стимули лише в обмеженій області поля зору, яка називається сприйнятливим полем у системі біологічного зору, кожен нейрон у CNN обробляє дані лише у своєму сприйнятливому полі [1].

Шари розташовані таким чином, щоб спочатку виявляти прості візерунки (лінії, криві тощо), а далі – складніші (обличчя, об’єкти тощо). Використовуючи CNN, можна ввімкнути бачення комп’ютерів. Виявлення об'єктів – це завдання комп'ютерного зору, яка включає ідентифікацію і виявлення об'єктів на зображеннях або відео. Це важлива частина багатьох програм, таких як безпілотні автомобілі, робототехніка та відеоспостереження.

За минулі роки було розроблено безліч методів та алгоритмів для пошуку об'єктів на зображеннях та їх становищі. Найкраща якість виконання цих завдань досягається при використанні згорткових нейронних мереж.

Моделі виявлення об’єктів дозволяють користувачам ідентифікувати об’єкти певних визначених класів. Моделі отримують зображення як вхідні дані та виводять зображення з обмежувальними рамками та мітками на виявлених об’єктах.

Метою роботи є розробка програмної конфігурації для системи детектування об’єктів, що забезпечить спрощення збору інформації у мережі інтернет та прискорить процес її обробки. Функціональні можливості системи повинні охоплювати вибір класу об’єктів та джерело для збору даних.

Об'єкт дослідження – процеси проектування та розробки моделі системи-парсеру, що забезпечить користувачу швидкий доступ до актуальної інформації за допомогою використання нейронної мережі.

Предмет дослідження – засоби реалізації процесу парсігну та алгоритму роботи згортованної нейронної мережі.

В роботі було проведено дослідження технологій розробки систем парсингу та аналіз архітектур нейронних мереж, які використовують для детектування об’єктів. Виконано програмну реалізацію системи.

Практична цінність роботи полягає в тому, що створена система забезпечує зручний засіб для збору даних за певними категоріями.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

Парсинг є досить популярним інструментом у сфері веб-розробки та аналітики даних. Це процес збору та аналізу інформації з веб-сайтів, який дозволяє отримувати цінні дані для різних цілей.

Популярність парсингу визначається його широким спектром застосувань. Веб-парсинг використовується для отримання даних для маркетингових досліджень, конкурентної аналітики, аналізу соціальних мереж, створення власних веб-сервісів та багато іншого.

Завдяки постійному розвитку технологій та зручним інструментам програмування, парсинг стає все більш доступним для широкої аудиторії. Запити до пошукових движків стосовно парсингу та його популярності також збільшуються з часом [2].

Парсинг є корисним знаряддям для багатьох сфер діяльності та його популярність безумовно зростає, привертаючи увагу розробників та дослідників, що стимулює нові інновації та покращення в цій галузі [3].

Парсинг сайтів – це процес автоматичного вилучення даних із веб-сторінок. Веб-скрапінг є підмножиною парсингу сайтів і зазвичай здійснюється за допомогою спеціальних програмних інструментів. При парсингу сайтів бот або програма отримує доступ до HTML-коду веб-сторінок і аналізує його для отримання потрібної інформації, такої як текст, зображення, посилання, ціни, рейтинги товарів та інше [4].

1.1 Опис предметної області

Синтаксичний аналіз означає аналіз і перетворення програми у внутрішній формат, який реально може виконувати середовище виконання, наприклад механізм JavaScript у браузерах.

Браузер аналізує HTML у дерево DOM. Розбір HTML передбачає токенізацію та побудову дерева. HTML-маркери містять початкові та кінцеві теги, а також імена та значення атрибутів. Якщо документ правильно сформований, його розбір є простим і швидшим. Синтаксичний аналізатор аналізує токенізоване введення в документ, створюючи дерево документа.

Виявлення об'єктів широко використовується в комп'ютерному зорі для автономного водіння. Безпілотні автомобілі використовують моделі виявлення об’єктів для виявлення пішоходів, велосипедів, світлофорів і дорожніх знаків, щоб вирішити, який крок зробити.

Крім цього, детектування об’єктів використовують у таких сферах, як:

- відстеження об’єктів у матчах – моделі виявлення об’єктів широко використовуються у видах спорту, де відстежують м’яч або гравця для моніторингу та суддівства під час матчів;
- пошук зображень – смартфони використовують такі моделі для виявлення об’єктів (наприклад, певних місць або об’єктів) і дозволяють користувачеві шукати об’єкти в Інтернеті.
- підрахунок об'єктів – моделі виявлення об’єктів використовуються для підрахунку екземплярів об’єктів на заданому зображенні, це може включати підрахунок об’єктів на складах чи в магазинах або підрахунок кількості відвідувачів у магазині. Вони також використовуються для керування натовпом на заходах, щоб запобігти катастрофам [5].

Інструменти веб-скрапігну шукають нові дані вручну або автоматично. Вони отримують оновлені або нові дані, а потім зберігають їх для легкого доступу. Ці інструменти корисні для тих, хто намагається збирати дані з Інтернету.

Наприклад, інструменти веб-збирання можна використовувати для збору даних про нерухомість, готелів із найпопулярніших туристичних порталів, цін на продукти, даних відгуків для веб-сайтів електронної комерції тощо.

В процесі парсинга застосовуються скриптові мови програмування: PHP, Perl, Ruby, Python, JavaScript і багато інших. Він здійснюється шляхом написання так званого парсеру. Парсер – це програма або скрипт, який обробляє структуровані дані, зазвичай у форматі тексту, та витягує з них необхідну інформацію.

Парсери використовуються для автоматичного оброблення великих обсягів даних, наприклад, веб-сторінок, файлів логів або баз даних. Вони можуть аналізувати та розділяти текст на складові елементи, такі як заголовки, параграфи, таблиці, посилання тощо, щоб зробити його більш зрозумілим та зручним для подальшого використання..

Для розв'язуваної задачі на сайтах інтернет-магазинів можна виділити наступні типи сторінок: картка товару та сторінка, яка не містить інформації про товар. Картка товару – це окрема сторінка, на якій міститься вся інформація про товар: назва, фото, ціна тощо. Сторінки, які не містять інформації про продукт, це такі сторінки, як домашня сторінка, контакти, загальна інформація, різні документи тощо.

Оскільки система намагатиметься автоматизувати алгоритм пошуку потрібних сторінок і визначення необхідної інформації за допомогою обробки зображень, слід використовувати алгоритми глибокого навчання.

Система, що розробляється, вимагає вирішення наступних підзадач машинного навчання: завдання класифікації сторінок і завдання виявлення об’єктів на зображенні.

У рамках завдання класифікації система визначить, чи підходить веб-сторінка умові картки продукту чи ні. У рамках завдання виявлення об’єктів на зображенні система визначає вибрану інформацію на ідентифікованих зображеннях картки товару. Наприклад, назву, фотографію, ціну.

Парсинг сайтів є ефективним рішенням для автоматизації збору і зміни інформації. У порівнянні з людиною, комп'ютерна програма-парсер:

- швидко обійде тисячі веб-сторінок;
- акуратно відокремить технічну інформацію від «людської»;

- безпомилково відбере потрібне і відкине зайве;
- ефективно упакує кінцеві дані в необхідному вигляді.

Результат (будь то база даних чи електронна таблиця) потребує подальшої обробки. Втім, подальші маніпуляції із зібраною інформацією вже до теми парсинга не належать.

Таким чином, основними етапами роботи є класифікація сторінок, виявлення об'єктів на потрібних сторінках і запис отриманої інформації в базу даних.

1.2 Етапи парсінгу

Етапи парсінгу можуть варіюватися в залежності від конкретного проекту, але основні етапи наступні:

1. Збір URL: перший крок – це збір URL-адрес веб-сторінок, які ви хочете проаналізувати. Це може включати пошук пошукових систем, сканування сайтів або отримання URL з бази даних.

2. Завантаження сторінок: після збору URL сторінки потрібно завантажити. Це може бути зроблено за допомогою HTTP-запиту до кожної веб-сторінки та отримання вихідного коду HTML або іншого вмісту сторінки.

3. Вилучення даних: після завантаження сторінки потрібно вилучити потрібні дані. Це може бути зроблено шляхом розбору HTML і витягування необхідних елементів, таких як заголовки, текст, зображення тощо.

4. Обробка даних: отримані дані можуть потребувати подальшої обробки, наприклад, очищення від HTML-тегів, форматування або перетворення в інший формат.

5. Збереження даних: остаточні дані можуть бути збережені у базу даних або іншу форму для подальшого використання або аналізу.

6. Повторення процесу: якщо є багато сторінок, які потрібно проаналізувати, процес може бути повторений для кожної сторінки [2].

Ці етапи можуть бути змінені або розширені в залежності від особливостей конкретного проекту та потреб користувача. Обхід всіх потрібних сторінок сайту забезпечується різними способами.

- обробляючи чергову сторінку, парсер можна навчити не тільки отримувати необхідні дані, але і заносити в свою базу даних всі внутрішні посилання, що зустрічаються на шляху. Звертаючись до свого сховища лінків, програма послідовно відвідує сторінки сайту, до тих пір поки не обійде їх всіх;
- при первинному аналізі сайту найчастіше можна простежити логіку формування url для сторінок. І потім, генерувати адреси відповідно до виявлених закономірностей;
- деякі парсери розраховані, як не дивно, на «ручний» обхід веб-ресурсу. Користувач, клікаючи по посиланнях, сам вирішує які сторінки відвідувати, які ні. А програма у фоновому режимі запам'ятовує необхідні дані.

В цілому парсинг: визначення структури сайту і шаблону даних і, на цьому етапі корисним буває вивчити файл robots.txt, xml карту сайту, пошук по сайту, видачу пошуковиків для сайту; підготовка виразів (xpath або css селектори) для отримання необхідних даних зі сторінок [1].

1.2.1 Імпорт контенту та синтаксичний аналіз

Парсинг, як і будь-яка технологія, стикається із низкою проблем. Найактуальніші:

- блокування доступу до даних: використання CAPTCHA, блокування IP-адрес та інше;
- швидкість виконання: великий обсяг даних потребує багато ресурсів та часу;
- складність обробки помилок: помилки з'єднання, помилки синтаксису та інші;

- робота з динамічним контентом: необхідно розробляти спеціальні інструменти для аналізу сайтів, які використовують технології Ajax та Javascript [6].

Зазвичай рішення зводиться до завантаження інтернет-сторінки стандартними методами, проте не завжди все виявляється так просто. Деякі сайти з легкістю розпізнають ботів для парсінга і блокують отримання сторінок. Переважно непросто дістатися до вмісту фреймів і ділянок, підвантажуваних через Ajax. Дуже часто для отримання потрібних сторінок необхідна авторизація, виникає проблема у вигляді сесій і coockie.

Окремо потрібно відмітити випадки, коли веб-сторінка повністю генерується на стороні клієнта, динамічно формуючись за допомогою JavaScript. Іноді отримання інформації є набагато більш складним завданням ніж наступний синтаксичний розбір.

1.2.2 Експорт даних

Матеріал, отриманий з розпарсенного сайту, необхідно упакувати у вигляді, придатному для використання. Конкретний формат залежить від того як в подальшому буде оброблятися зібрана інформація. Найчастіше це бази даних MySQL або PostgreSQL. Тут кожна сторінка заносяться в базу даних. При чому, це можна робити не тільки за допомогою запитів SQL, але і за допомогою JSON через Ajax. Однак часто наповненням бази даних справа не закінчується. Інформацію з SQL доводиться конвертувати в інші формати: JSON, текстовий CSV (який, в свою чергу, є зручним буфером для подальшої обробки), таблично-процесорний XLS (в PHP є спеціальні додаткові бібліотеки для перетворення SQL-даних в електронні таблиці Excel), XML.

У багатьох випадках із спарсенного контенту за допомогою XML формується RSS-потік, що зручно при використанні даних, без процедури рерайтинга. Іноді результат парсинга поміщають в CSV-файл, оскільки цей текстовий формат дуже простий у подальшій обробці, легко конвертується в

SQL-запити і без проблем відкривається в Excel (рис. 1). Один рядок – один запис в таблиці.

	A	B	C
1	First Name	Last Name	Email Address
2	Jane	Doe	jane@gmail.com
3	John	Doe	john@gmail.com
4	Chris	Perkins	cperk@gmail.com
5	Eva	Davis	eva@gmail.com
6	Mitchell	Tarver	mitch@gmail.com
7	Nathan	Woodward	nathanwoodward@gmail.com

Рисунок 1 – Приклад структури CSV-файлу

Формат CSV можна легко:

- правити в будь-якому текстовому редакторі;
- відкривати в Microsoft Excel і OpenOffice.org Calc;
- змінювати функціями PHP для роботи з файлами;
- конвертувати в SQL і назад.

У міру обходу сторінок сайту-донора часто зібрана інформація тут же за допомогою запитів SQL заноситься в базу даних. Альтернативою є JSON – текстовий формат обміну даними оснований на JavaScript. У даного методу чимало безперечних переваг перед тим же SQL. JSON відрізняють мовонезалежність, крайня простота і лаконічність синтаксису, наявність готових рішень для обробки даних. Головна перевага – це можливість прямої взаємодії браузера і сервера, що дозволяє реалізовувати алгоритми пов'язані з фоновим занесенням інформації в базу даних, роботою з файловим кешем, серіалізацією, десеріалізацією динамічних структур та ін [2].

DOM (Document Object Model) – багатоплатформовий багатомовний програмний інтерфейс, що надає додаткам і скриптам доступ (в тому числі і на зміну) до структури та змісту HTML, XHTML, XML-документів. DOM дозволяє ефективно вибудувати ієрархічне дерево веб-документа (рис. 2) для подальшої роботи з ним.

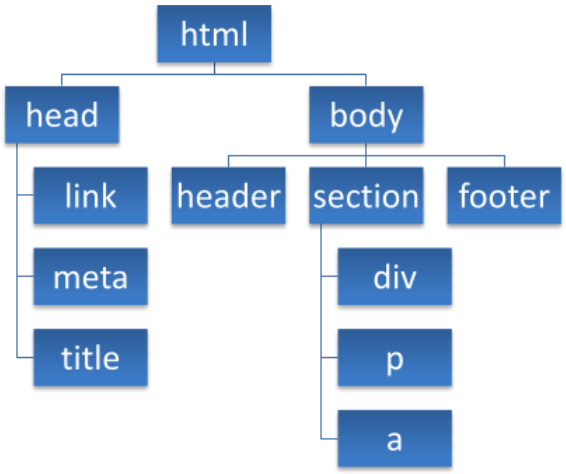

Рисунок 2 – Приклад ієрархічного дерева веб-документа

За допомогою DOM мова JavaScript отримує доступ до структури та змісту HTML-документів.

1.3 Захист від парсенгу та методи його обходу

Захист від парсингу (витягування даних з веб-сторінок) є важливим аспектом безпеки веб-додатків. Ось кілька методів захисту від парсингу та способи їх обходу:

1. Обмеження швидкості: встановлення розумних обмежень на швидкість запитів до сервера може допомогти убезпечити веб-додаток від автоматичних парсерів. Проте, деякі парсери можуть по-справжньому повільно виконувати запити, щоб обійти це обмеження.

2. Капча: використання капчі може ускладнити парсинг, оскільки ботам важко розпізнати та вирішити капчу. Але часто зловмисники використовують розподілені системи з розумними алгоритмами, які можуть обходити більшість капч.

3. Змішування даних: додавання випадкових значень до ідентифікаторів URL або сторінок може ускладнити або навіть унеможливити парсинг, оскільки потрібно зробити багато запитів для отримання всієї інформації.

4. Розпізнавання шаблонів: веб-додаток може використовувати аналіз поведінки запитів, щоб виявити автоматичних парсерів, дивлячись на шаблони запитів, час запитів та інші характеристики.

5. Анти-парсинг служби: існують спеціалізовані служби, які надають захист від парсингу, шляхом блокування або ускладнення доступу для автоматичних парсерів.

Найпростішим і поширеним засобом визначення спроб парсингу сайту є аналіз частоти і періодичності запитів до сервера: у разі виявлення активності йде бан IP-адреси. Тут слід знайти межу між природною частотою і кількістю запитів і спробами скрейпінга щоб не заблокувати ні в чому не винних користувачів. Зазвичай це визначається за допомогою аналізу поведінки нормальних користувачів сайту. Прикладом використання цього методу може служити Google, який контролюється кількість запитів з певної адреси і видає відповідне попередження з блокуванням IP адреси і пропозицією ввести каптчу [2].

Використання облікових записів допомагає у захисті доступу до даних, яке здійснюється тільки авторизованим користувачам. Тут легше контролювати поведінку користувачів і блокувати підозрілі акаунти незалежно від того, з якої IP адреси працює клієнт. Прикладом може служити Facebook, активно контролює дії користувачів і блокує підозрілих. Цей захист обходиться шляхом створення (в тому числі автоматичного) безлічі облікових записів (є навіть сервіси, які торгують готовими обліковими записами для відомих соціальних мереж).

Недоліком використання CAPTCHA є незручність користувача в необхідності її вводу. Тому цей метод найкраще застосовувати в системах, де доступ до даних здійснюється окремими запитами і не дуже часто. Обходиться каптча за допомогою програм і сервісів по її розпізнаванню. Вони діляться на дві основні категорії: автоматичне розпізнавання без участі людини (OCR, наприклад програма GSA Captcha Breaker) і розпізнавання за допомогою людини (коли десь в Індії сидять люди і в режимі онлайн обробляють запити

на розпізнавання картинок, наприклад може служити сервіс Bypass CAPTCHA).

У разі використання JS-логіки запиті до сервера браузер відсилає спеціальний код (або декілька), які сформовані складною логікою написаною на JavaScript. При цьому, часто код цієї логіки обфусцированний і розміщений в одному або декількох підвантажуваних JavaScript-файлах. Типовим прикладом використання даного методу захисту від парсинга є Facebook. Обходиться це за допомогою використання для парсинга реальних браузерів (наприклад, за допомогою бібліотек Selenium або Mechanize). Але це дає цим методом додаткових переваг: виконуючи JavaScript, парсер буде проявляти себе в аналітиці відвідуваності сайту (наприклад Google Analytics), що дозволить вебмайстру відразу помітити недобре [1].

Один з ефективних способів захисту від автоматичного парсинга – це часта зміна структури сторінки. Це може стосуватися не тільки зміна назв ідентифікаторів і класів, але навіть і ієрархії елементів. Це сильно ускладнює написання парсеру, але з іншого боку ускладнює і код самої системи.

Щоб обійти такий захист потрібне створення більш гнучкого і «розумного» парсеру або ж (якщо зміни робляться не часто) просто ручне виправлення парсеру, коли ці зміни відбулися.

1.4 Огляд програмних засобів

Для розробки веб-додатку, спрямованого для детектування об'єктів на сторінках сайтів інтернет-магазинів, слід розглянути існуючі аналоги для того, щоб виявити основні недоліки та переваги при розробці даних систем, які допоможуть покращити сприйняття користувачем системи.

Перший аналог «Web Scraper» надає хмарну платформу для доступу до витягнутих даних (рис. 3). Він має простий у використанні інтерфейс, тому ним можуть користуватися й початківці. Крім того, це дозволяє отримувати дані або вміст навіть із динамічних веб-сайтів.

Особливості Web Scraper:

- він дозволяє витягувати дані з веб-сайтів із категоріями та підкатегоріями;
- змінює вилучення даних у міру зміни структури сайту.

Плюси Web Scraper:

- це хмарний веб-ресурс;
- витягнуті дані доступні через api.

Мінуси Web Scraper:

- має надати додаткові кредити в пробному плані;
- висока ціна для маленьких користувачів;
- кілька внутрішніх помилок сервера;
- відповідь веб-сайту іноді дуже повільна;
- він повинен містити більше відеодокументації.

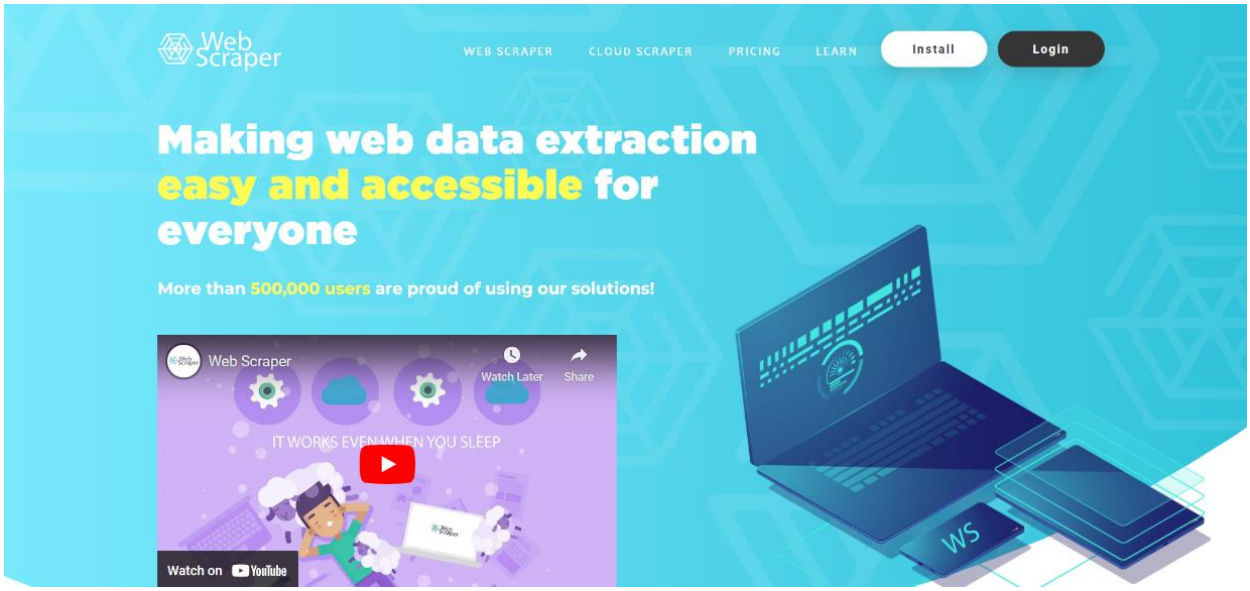

Рисунок 3 – Web Scraper

Наступний аналог – ресурс «Scraper API» (рис. 4) призначений для збору загальнодоступних веб-даних у реальному часі майже з будь-якої сторінки. Він служить надійним рішенням для швидкого та надійного вилучення даних.

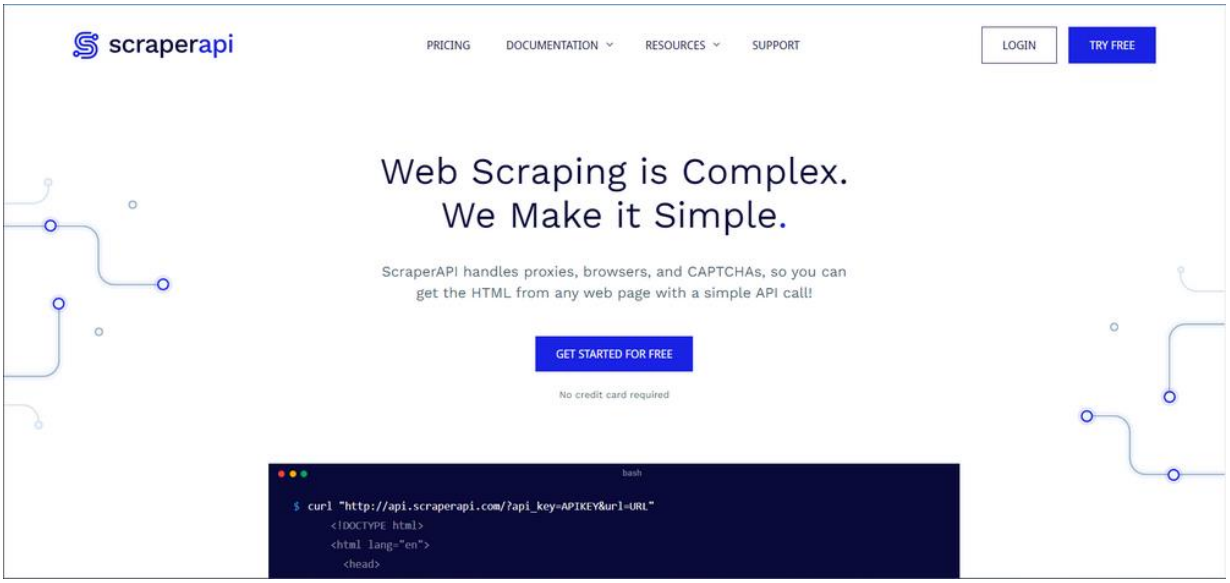

Рисунок 4 – Scraper API

Як наслідок, Scraper API найкраще підходить, але не обмежується, такими випадками використання, як захист від шахрайства, дослідження ринку та моніторинг вартості проїзду. Він надає безкоштовну пробну версію протягом одного тижня.

Scraper API дозволяє легко інтегрувати; просто потрібно отримати запит і URL-адресу. Крім того, користувачі можуть отримати в документації більш розширені випадки використання. Він також надає геолокаційні обертові проксі-сервери, які допомагають направляти запит через проксі-сервери.

Особливості Scraper API: дозволяє легко інтегрувати та дозволяє користувачам також очищати сторінки, відтворені за допомогою javascript.

Плюси Scraper API:

- простий у використанні;
- повністю настроюється;
- це швидко і надійно.

Мінуси Scraper API:

- на деяких веб-сайтах цей інструмент не працює;

– фівнансові витрати мна використання ресурсу;
– деякі функції, такі як копіювання javascript, дуже дорогі;
– має покращити здатність масштабувати виклики плану;
– під час виклику api заголовки відповіді відсутні [5].

«ParseHub» це відомий інструмент для збирання веб-сторінок із простим у використанні інтерфейсом (рис. 5). Він забезпечує простий спосіб отримання даних із веб-сайтів. Крім того, він може витягувати дані з кількох сторінок і взаємодіяти з AJAX, розкривним меню тощо.

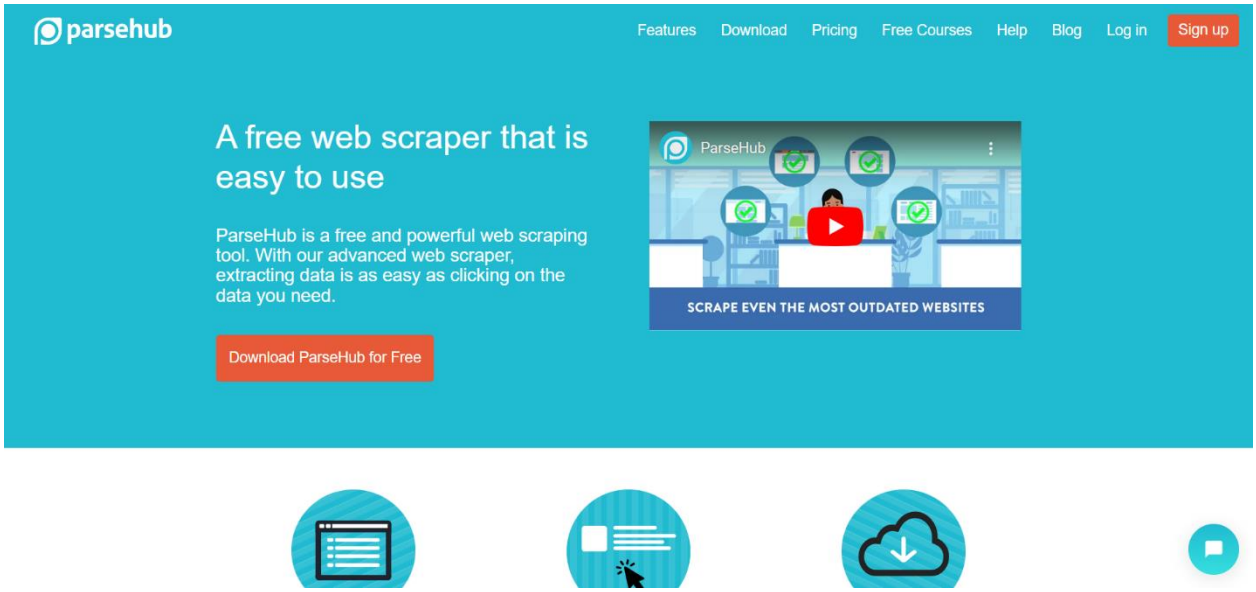

Рисунок 5 – ParseHub

Особливості ParseHub: дозволяє агрегувати дані з кількох веб-сайтів, REST API для створення мобільних і веб-додатків. ParseHub орієнтований майже на всіх, хто хоче пограти з даними. Це може бути будь-хто: від аналітиків і спеціалістів із обробки даних до журналістів.

Плюси ParseHub:

– він має простий у використанні інтерфейс;
– початківці також можуть використовувати його.

Мінуси ParseHub:

- це настільна програма;
- користувачі стикаються з проблемами з помилками;
- дорогий інструмент для сканування веб-сторінок;
- ліміт сторінок для розпакування в безкоштовній версії дуже низький [7].

«Diggernaut» – це хмарний сервіс для веб-збирання, вилучення даних та інших завдань ETL (вилучення, перетворення, завантаження) (рис. 6).

Все, що потрібно зробити, це створити копач, крихітного робота, який може робити веб-збирання від вашого імені та отримувати дані з веб-сайтів замість вас, нормалізувати їх і зберігати дані в хмарі.

Після цього ви можете завантажити його у форматі CSV, XLS, JSON або навіть отримати за допомогою нашого Rest API [6].

Рисунок 6 – Стартова сторінка «Diggernaut»

Користувач може використовувати готові скомпільовані парсери для отримання даних, але при цьому отримати неповні дані доступні для перегляду. Також користувач може зареєструватися та отримати доступ до

створення власного парсера, що має назву «дігер», але при цьому в рамках безкоштовного тарифу отримати обмеження на кількість запитів до сайтів та інформації, що зберігається.

Також є Google Chrome розширення Excavator, за допомогою якого можна дослідити сторінки, на яких ти перебуваєш в даний момент.

Можна виділити такі переваги даного сервісу:

- різноманітність тематик сайтів;
- каталог вже опрацьованих сайтів із попереднім переглядом;
- можливість створення власного парсера свого сайту.

Але у даного сервісу також є й недоліки:

- для створення власного парсеру даних користувачеві потрібні навички програмування;
- обмеженість каталогу деяких оброблених сайтів;
- мала кількість доступної пам'яті, якої вистачає на один сайт.

На рисунку 7 представлений каталог безкоштовних скомпільованих парсерів.

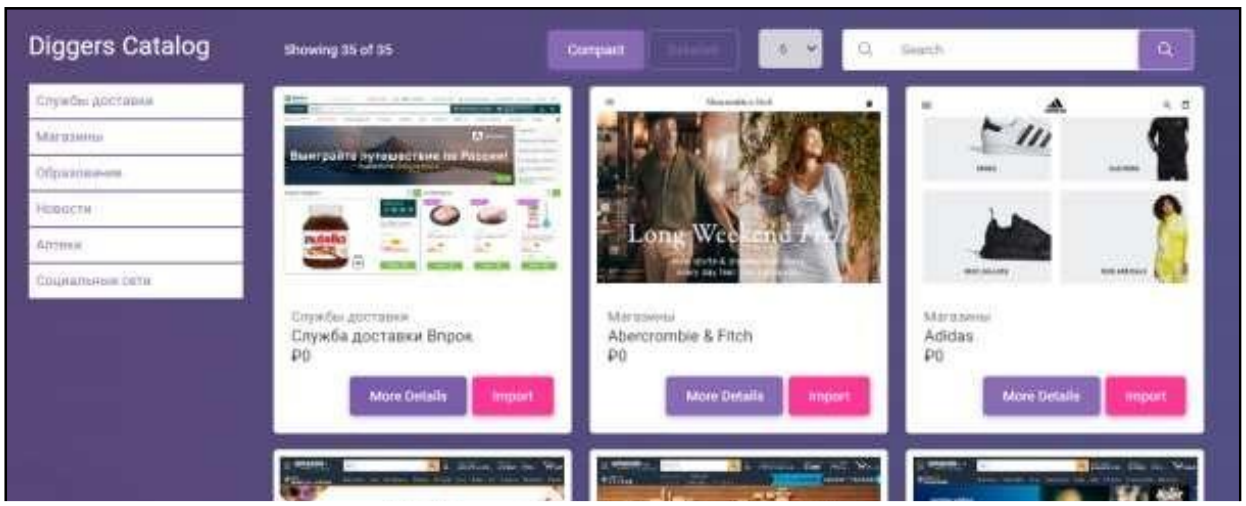

Рисунок 7 – Каталог безкоштовних скомпілованих парсерів

1.5 Огляд нейронних мереж, що використовуються для детектування об’єктів

Абсолютна більшість сучасних нейромереж застосовується для аналізу зображень. Для розпізнавання окремих зображень використовують наступні види нейронних мереж:

1. Багатошарові нейронні мережі.

Архітектура багатошарової нейронної мережі (БНМ) складається з послідовно з'єднаних шарів, де нейрон кожного шару своїми входами пов'язаний з усіма нейронами попереднього шару, а виходами - наступного. НМ з двома вирішуючими шарами може з будь-якою точністю апроксимувати будь-яку агатовимірну функцію. НМ з одним вирішуючим шаром здатна формувати лінійні розділяючі поверхні, що сильно звужує коло завдань, що вирішуються, зокрема така мережа не зможе вирішити завдання типу "виключаюче або".

НМ з нелінійною функцією активації та двома вирішуючими шарами дозволяє формувати будь-які опуклі області в просторі рішень, а з трьом а вирішуючими шарами – області будьякої складності, в тому числі і неопуклості. При цьому МНС не втрачає своєї узагальнюючої здатності. Навчаються МНС за допомогою алгоритму зворотного поширення помилки, що є методом градієнтного спуску в просторі ваг з метою мінімізації сумарної помилки мережі. При цьому помилки (точніше величини корекції ваг) поширюються в зворотному напрямку від входів до виходів, крізь ваги, що з'єднують нейрони [9].

2. Нейронні мережі високого порядку (НМВП).

Вони відрізняються від БНМ тим, що у них тільки один шар, але на входи нейронів надходять так само терми високого порядку, що є добутком двох або більше компонентів вхідного вектора. Такі мережі так само можуть формувати складні розділяючі поверхні.

Особливість такої мережі полягає в тому, що для навчання деякому класу достатньо продемонструвати його образ без варіацій масштабів та поворотів – після вивчення мережа буде розпізнавати відомі класи з масштабом та поворотом. Така мережа не є повнозв’язною, швидко навчається та працює в порівнянні з БНМ.

3. НМ Хопфілда (НМХ).

Вона буває одношарова і повношарова (зв'язки нейронів на самих себе відсутні), її виходи пов'язані з входами. На відміну від БНМ, НМХ є релаксаційною – тобто будучи встановленою в початковий стан, функціонує до тих пір, поки не досягне стабільного стану, який і буде її вихідним значенням. НМХ застосовуються в якості асоціативної пам'яті і для розв'язання оптимізаційних задач. У першому випадку НМХ навчається без вчителя (наприклад, за правилом Хебба), у другому випадку ваги між нейронами спочатку кодують вирішуване завдання. Таким чином НМХ з початкового стану сходиться до найближчого локального мінімумуенергії мережі, стан нейронів в якому і буде відновленим образом для задач розпізнавання, і рішенням – для оптимізаційних задач [10].

Для пошуку глобального мінімуму стосовно оптимізаційних задач використовують стохастичні модифікації НМХ. Застосування даної мережі відмічається хорошим результатом відновлення трьохвимірної форми (також для зображень облич) і високою швидкодією.

4. Самоорганізовуючі нейронні мережі Кохонена (СНМК).

Такі мережі забезпечують топологічне упорядкування вхідного простору образів. Вони дозволяють топологічно безперервно відображати вхідний n-мірний простір в вихідний m-мірний, m<<n. Вхідний образ проектується на деяку позицію в мережі, кодованих як положення активованого вузла.

На відміну від більшості інших методів класифікації і кластеризації, топологічне упорядкування класів зберігає на виході подобу у вхідних образах, що є особливо корисним при класифікації даних, що мають велику

кількість класів. Мережі такого типу складаються з одного шару (не рахуючи вхідного), який так само може бути організований в n-мірну сітку, в залежності від розмірності вихідного простору. Кожен нейрон зв'язаний з усіма вхідними нейронами. Налаштування ваг мережі здійснюється методом конкурентного навчання, в процесі якого змінюються тільки ваги нейрона-переможця, що має максимальну активність. Для даної мережі характерна висока швидкість навчання.

1.6 Згорткова нейронна мережа

Нейронні мережі це різновид машинного навчання, і вони лежать в основі алгоритмів глибокого навчання. Вони складаються з шарів вузлів, що містять вхідний шар, один або кілька прихованих шарів та вихідний шар. Кожен вузол з'єднується з іншим і має пов'язану з ним вагу та поріг. Якщо вихідні дані будь-якого окремого вузла перевищують зазначене граничне значення, цей вузол активується і надсилає дані на наступний рівень мережі. А якщо ні, то жодні дані не передаються на наступний рівень мережі.

ConvNets (CNN – згорткові нейронні мережі) найчастіше використовуються для завдань класифікації та комп'ютерного зору. До появи CNN для ідентифікації об'єктів на зображеннях використовувалися ручні та трудомісткі методи отримання ознак.

Однак згорткові нейронні мережі тепер забезпечують більш масштабований підхід до завдань класифікації зображень та розпізнавання об'єктів, використовуючи принципи лінійної алгебри, зокрема множення матриць, виявлення закономірностей у зображенні [11].

CNN відрізняються від інших нейронних мереж своєю чудовою продуктивністю з зображенням, мовленням або аудіосигналом. Вони мають три основні типи шарів (рис. 8), а саме:

1. Згортковий шар;
2. Шар об'єднання;

3. Повністю підключений (FC) рівень.

Згортковий шар є першим шаром згорткової мережі. У той час як за згортковими шарами можуть слідувати додаткові згорточні шари або шари об’єднання, повністю зв’язаний рівень є останнім. З кожним шаром CNN зростає у своїй складності, ідентифікуючи більші частини зображення. Попередні шари зосереджені на простих функціях, таких як кольори та краї. Коли дані зображення просуваються між шарами CNN, вони починають розпізнавати більші елементи або форми об’єкта, поки нарешті не ідентифікують запланований об’єкт.

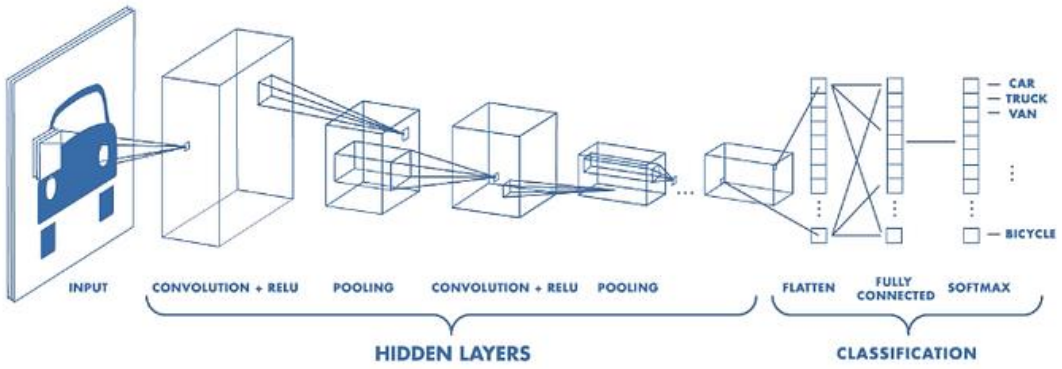

Рисунок 8 – Архитектура CNN

Згортковий шар є основним будівельним блоком CNN, і саме тут відбувається більша частина обчислень. Для цього потрібно кілька компонентів: вхідні дані, фільтр та карта об'єктів. Після кожної операції згортки CNN застосовує перетворення випрямленої лінійної одиниці (ReLU) до карти об'єктів, вносячи нелінійність до моделі (рис. 9).

За вихідним шаром згортки може йти ще один шар згортки. Коли це відбувається, структура CNN може стати ієрархічною, оскільки пізніші рівні можуть бачити пікселі в рецептивних полях попередніх шарів. Наприклад, якщо необхідно визначити, чи містить зображення велосипед, його можна уявляти і як суму частин.

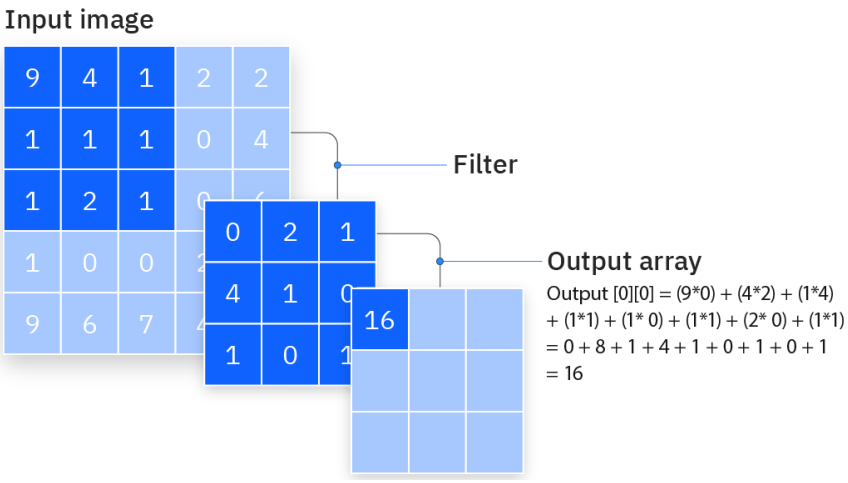

Рисунок 9 – Вихідні дані карти об’єктів

Він складається з рами, керма, коліс, педалей та інше. Кожна окрема частина велосипеда є шаблоном нижчого рівня в нейронній мережі, а комбінація її частин представляє шаблон вищого рівня, створюючи ієрархію функцій усередині CNN [12].

Зрештою згортковий шар перетворює зображення на числові значення, дозволяючи нейронній мережі інтерпретувати та витягувати відповідні шаблони (рис. 10).

Рисунок 10 – Ієрархічна структура CNN для визначення об’єкту «велосипед»

Об'єднання шарів, також відоме як дискретизація, що знижує, знижує розмірність, зменшуючи кількість вхідних параметрів. Подібно до

згортального шару, операція об'єднання охоплює фільтром весь вхідний сигнал, але різниця в тому, що цей фільтр не має ваг. Натомість ядро застосовує функцію агрегування до значень у сприймаючому полі, заповнюючи вихідний масив.

Значення пікселів вхідного зображення не пов'язані безпосередньо з вихідним шаром у частково пов'язаних шарах. Однак на повнозв'язковому рівні кожен вузол вихідного шару безпосередньо з'єднується з вузлом попереднього шару [12].

Цей шар виконує завдання класифікації на основі ознак, витягнутих за допомогою попередніх шарів та їх різних фільтрів. У той час як згорткові шари та шари пулу, як правило, використовують функції ReLu, шари FC зазвичай використовують функцію активації softmax для відповідної класифікації вхідних даних, створюючи можливість від 0 до 1.

1.7 Постановка завдання

Після аналітичного огляду предметної області та існуючих аналогів слід означити завдання до роботи. Система, що розробляється, працюватиме з зображеннями веб-сторінок і для цього знадобиться використання глибокого навчання, зокрема згорткових нейронних мереж, які зарекомендували себе з кращого боку при роботі із зображеннями.

Також для зручного використання системи користувачем слід реалізувати веб-додаток, де буде розміщено весь необхідний функціонал.

Існуючі рішення мають низку очевидних проблем:

1) швидкість роботи;

2) вимога до наявності навичок програмування;

3) відсутність вибору корисних елементів;

4) мінімальна автоматизація роботи (доопрацювати);

5) тарифікація послуг.

Таким чином, додаток, що розробляється, повинен реалізувати наявні у аналогів гідності та усунути недоліки. Для реалізації мети необхідно виконати наступне:

1. Провести огляд наукової літератури, виконати аналіз предметної галузі.
2. Вивчити існуючі аналоги та розробити необхідні вимоги до системи.
3. Здійснити збір даних та передобробку даних для навчання нейронної мережі.
4. Навчити нейронну мережу та оцінити результати її роботи.
5. Розробити веб-додаток для визначення об'єктів.
6. Здійснити тестування програми.

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Визначення вимог до системи

Функціональні вимоги – це вимоги, які визначають дії, які має виконувати система, без урахування обмежень, пов'язаних з її реалізацією, тобто визначають поведінку системи у процесі обробки інформації. Нефункціональні вимоги – це умови, за яких продукт повинен працювати, та якості, якими він повинен володіти (наприклад, продуктивність, надійність, масштабованість).

До системи, що розробляється, сформульовані такі функціональні вимоги:

1) потрібна можливість валідації даних;

2) можливість введення сайту, що цікавить;

3) вибір детектованих характеристик сторінки;

4) отримання результату у потрібному форматі файлу.

До системи, що розробляється, є такі нефункціональні вимоги:

1) система має бути продуктивною;

2) система має бути зручною;

3) система має бути надійною;

4) система має бути швидкою.

2.2 Діаграма станів

Діаграма станів дозволяє описувати поведінку системи. В об'єктно-орієнтованому підході розробляється діаграма станів єдиного класу, що демонструє поведінку одного об'єкта протягом його життя (рис. 11).

Діаграма станів – орієнтований граф для кінцевого автомата, в якому:

- вершини позначають стан;
- дуги показують переходи між двома станами.

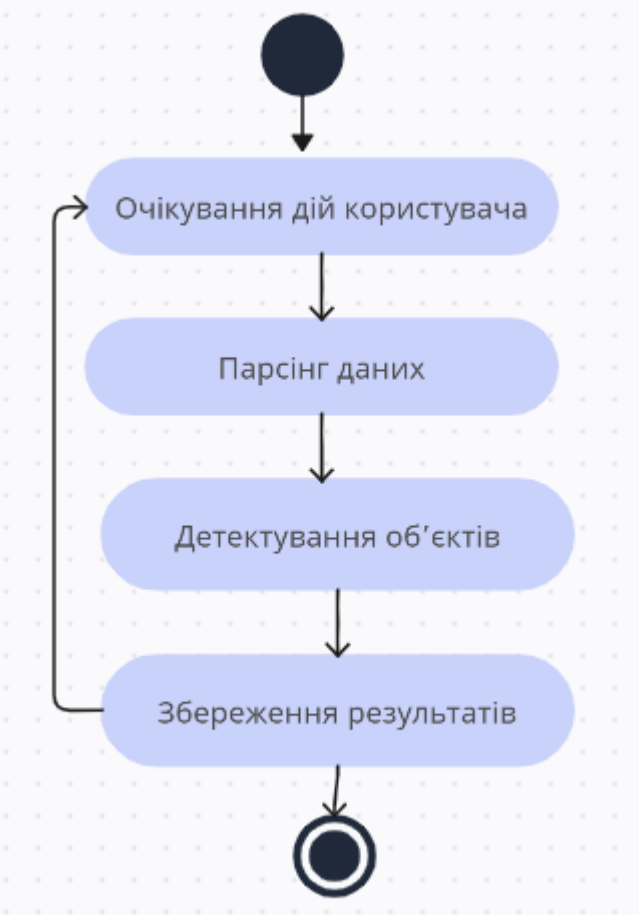

Рисунок 11 – Діаграма станів

2.3 Діаграма IDEF0

IDEF0 – методологія функціонального моделювання. 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 [13].

Найзагальніший опис інформаційної системи та її взаємодії з зовнішнім середовищем можна представити у вигляді контекстної діаграми. На рисунку 11 відображено опис головної роботи розробки системи детектування об’єктів. Так, на вході маємо розробку алгоритму, а також вимоги до проекту. У якості управління для головної роботи виступають технологія завантаження веб-сторінок та технологія створення GUI. Механізми тут – сам виконавець проекту та програмне забезпечення, що було обрано як інструмент розробки системи. На виході головної роботи будемо мати працюючу систему.

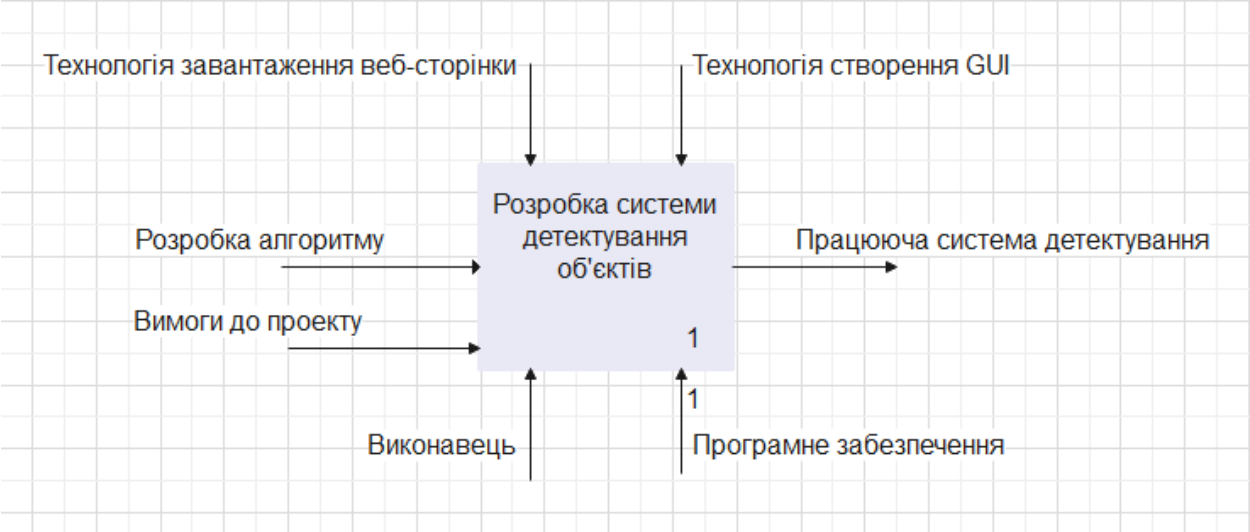

Рисунок 11 – Діаграма IDEF0

Наступний крок – це декомпозиція головної роботи на декілька її складових (рис. 12). Це дозволить більш детально спроектувати вхідні та вихіднідані дані, механізми та управління складовими частинами. На першому етапі декомпозиції розіб’ємо головну роботи на 4 складових: аналіз вимог, вибір нейронної мережі, реалізація функціоналу та тестування отриманої системи.

Для першого етапу «Аналіз вимог» у ролі вхідних даних виступають результати огляду аналогів, механізми – це календарний план, так як слід встигнути реалізувати проект у назначений час. Управляє цим етапом керівник

проекту разом с виконавцем. На виході маємо постановку завдання та функціональні вимоги до проекту.

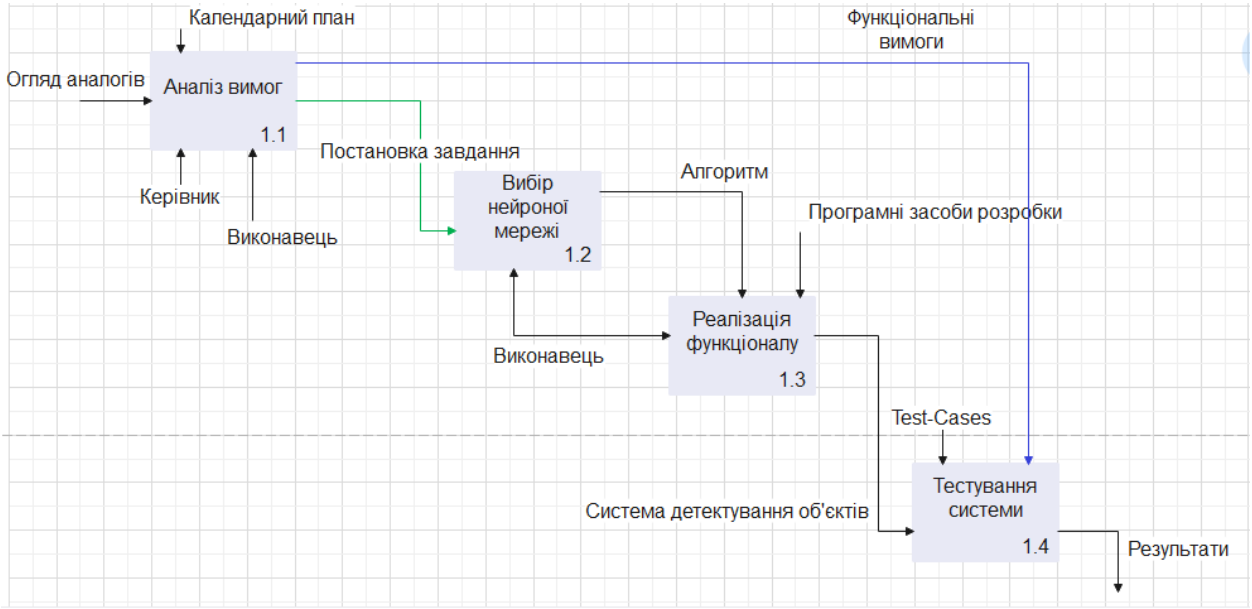

Рисунок 12 – Декомпозиція першого рівня діаграми IDEF0

Наступний крок – це вибір нейронної мережі. З урахуванням наявності великої кількості слід обрати саме ту, що підходить до реалізації поставлених задач. Розроблений алгоритм є результатом цієї роботи та механізмом для етапу «реалізація функціоналу», крім цього, і програмні засоби розробки. Управляння перекладено тільки на виконавця.

На останньому етапі слід провести тестування отриманої системи. За вимогами тест-кейсів буде протестована система та отримані відповідні результати.

2.4 Діаграма DFD

Діаграма потоку даних (DFD) відображає потік інформації для будь-якого процесу чи системи. Він використовує визначені символи, такі як прямокутники, кола та стрілки, а також короткі текстові мітки, щоб показати

вхідні дані, виходи, точки зберігання та маршрути між кожним пунктом призначення. Блок-схеми даних можуть варіюватися від простих, навіть намальованих від руки оглядів процесів, до поглиблених багаторівневих DFD, які все глибше вивчають, як обробляються дані (рис. 12). Їх можна використовувати для аналізу існуючої системи або моделювання нової.

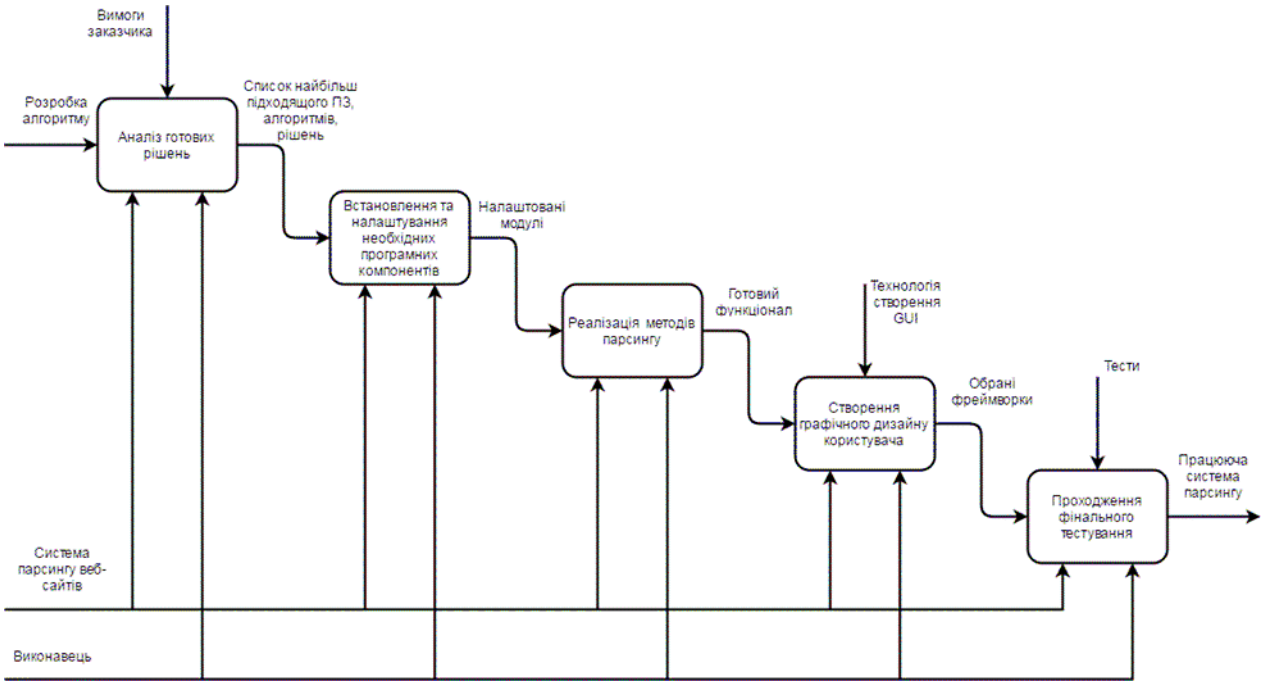

Рисунок 12 – Діаграма потоків даних

2.5 Діаграма Use-Case

Для проектування системи була використана мова графічного опису для об'єктно-орієнтованого програмування UML. Діаграма прецедентів візуально зображає різноманітні сценарії взаємодії між акторами (користувачами) і прецедентами (випадками використання), описує функціональні аспекти системи (бізнес логіку). Діаграми прецедентів відіграють важливу роль не тільки у комунікації між збирачами вимог до проекту і потенційними користувачами [14].

Була побудована модель взаємодії зовнішнього актора із системою веб-додатку у вигляді діаграми варіантів використання. У ході аналізу вимог до системи, що розробляється, були виявлені наступні варіанти використання, показані на рисунку 13:

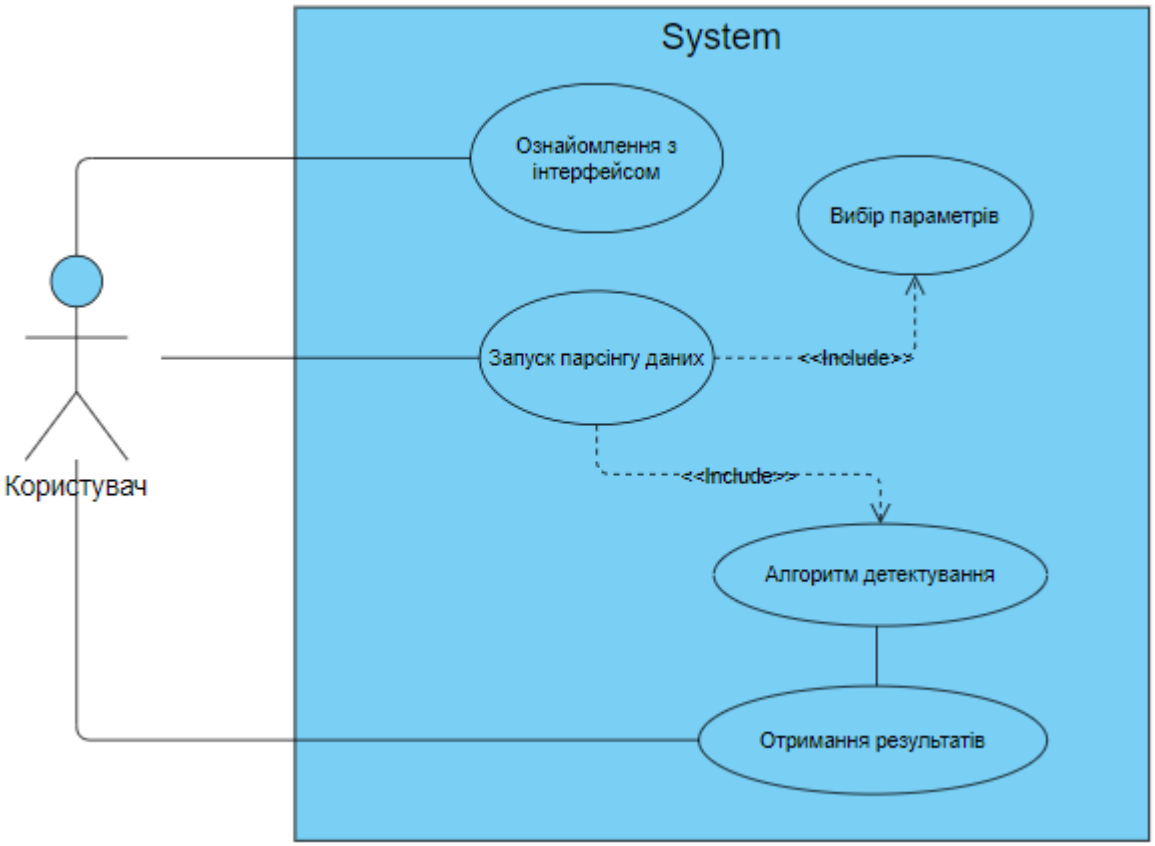

Рисунок 13 – Діаграма прецендентів

2.6 Архітектура нейронної мережі для детектування об'єктів на веб-сторінці

При виборі архітектури потрібно підібрати найкращий варіант. На рисунку 14 представлені результати точності різних архітектур для завдання детектування об'єктів на веб-сторінці. Для вирішення задачі детектування об'єктів на веб-сторінці інтернет-магазину було вибрано архітектуру CovA, на

основі спостережень, наведених у статті «CoVA: Context-aware Visual Attention for Webpage Information Extraction» [15].

Method	Params	Price Acc	Title Acc	Image Acc
Gogar et al. (2016)	1.8m	78.1 ± 17.2	91.5 ± 1.3	93.2 ± 1.9
Random Forest using Heuristic features	-	87.4 ± 10.4	93.5 ± 5.3	97.2 ± 3.8
Fast R-CNN* (Girshick, 2015)	0.5m	86.6 ± 7.3	93.7 ± 2.2	97.0 ± 3.6
Fast R-CNN* + GCN	1.4m	90.0 ± 11.0	95.4 ± 1.5	98.2 ± 2.8
Fast R-CNN* + Bi-LSTM	5.1m	92.9 ± 4.6	94.0 ± 2.1	97.6 ± 3.6
CoVA	1.6m	$\mathbf{95.5 \pm 3.8}$	$\mathbf{95.7 \pm 1.2}$	$\mathbf{98.8 \pm 1.5}$
CoVA++	1.7m	$\mathbf{96.1 \pm 3.0}$	$\mathbf{96.7 \pm 2.2}$	$\mathbf{99.6 \pm 0.3}$

Рисунок 14 – Результати різних архітектур

На вхід даної нейронної мережі подаватиметься зображення веб-сторінки та DOM-дерево документа. На рисунку 15 представлено архітектуру нейронної мережі для детектування.

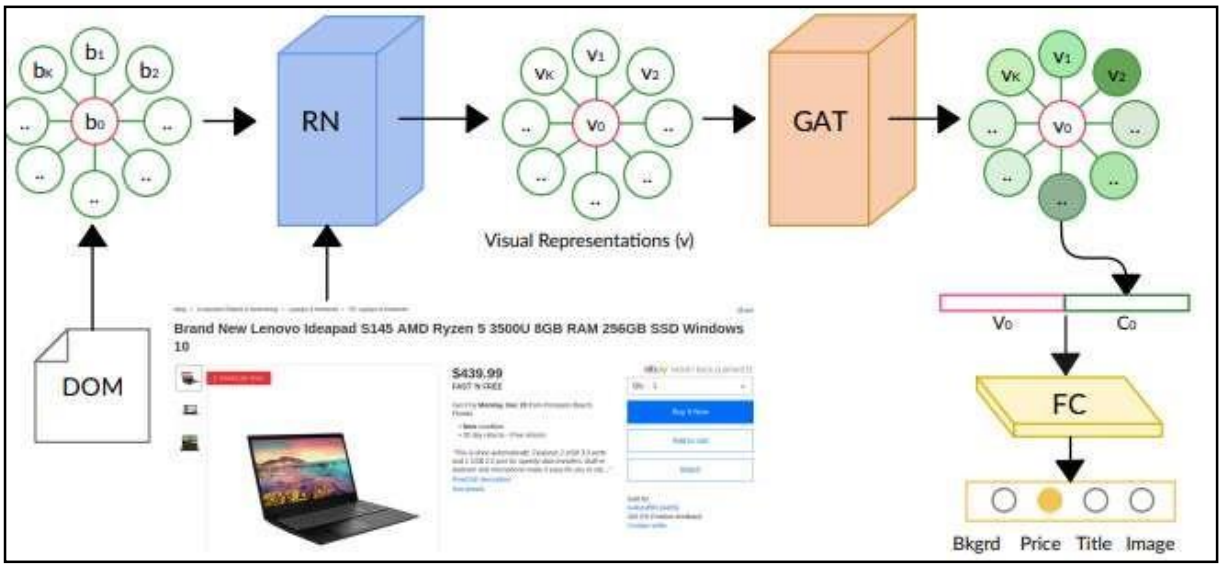

Рисунок 15 – Архітектура НМ для детектування

Архітектура нейронної мережі складається з моделей для обробки візуальної інформації веб-сторінки та контекстної інформації про елементи

цієї сторінки, отриманої з DOM-дерева сторінки, а також шарів для класифікації елемента.

Завдання моделі для обробки візуальної інформації полягає в тому, що модель намагається запам'ятати розмір візуального представлення кожного веб-елемента. Це важливо, тому що різні елементи на сторінці мають різні розміри елементів. Запам'ятовування розмірів проводиться за допомогою отримання високорівневих ознак (лінії, краю, контури фігур) згорткової нейронної мережі. Після отримання розмірів веб- елементів слід область інтересу для отримання фіксованого розміру елемента.

Щоб зафіксувати розмір елемента, модель вчиться підбирати координати елемента на зображенні, такі як висота, ширина, координата по осі X, координата по осі Y. Виходом даної мережі є згладжений список, що складається з даних координат, що описують положення елемента [15]. На рисунку 16 подано елемент моделі, призначений для обробки візуальної інформації.

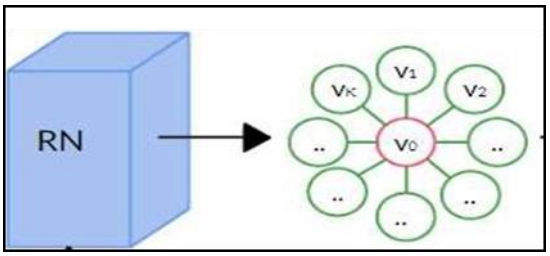

Рисунок 16 – Модель для обробки візуальної інформації

Завдання моделі для обробки контекстної інформації полягає в тому, що модель намагається «зрозуміти» важливість символів, слів, що стоять поруч. Наприклад, символ «$», що знаходиться поруч із набором цифр, буде релевантним для пошуку ціни. Для визначення реляційної важливості символу використовується GraphAttentionLayer.

Алгоритм приймає візуальну інформацію, одержану з моделі для обробки візуальної інформації. Формальний запис таких даних $v = [v1, v2, \ldots, vn]$. Кожна вхідна інформація перетворюється за допомогою проекційних матриць і набуває значення важливості.

На рисунку 17 представлений елемент моделі для обробки контекстної інформації:

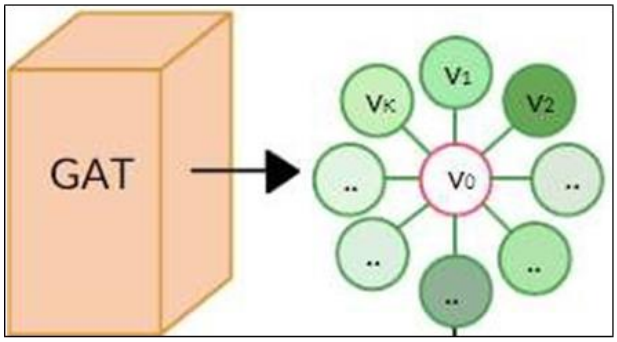

Рисунок 17 – Модель обробки контекстної інформації

Отримані зважені значення важливості складаються для отримання фінальної важливості елемента. На фінальний шар подаються дані про візуальну інформацію веб-сторінки та оброблену контекстуальну, виражену у значенні важливості веб-елементів, для отримання класу веб-елемента. На рисунку 18 представлено елемент фінального шару.

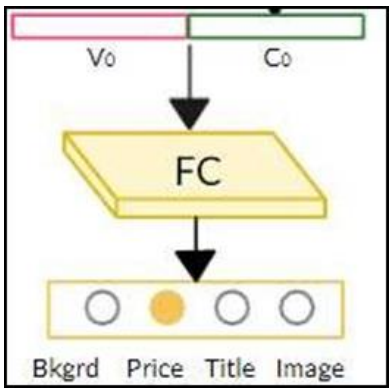

Рисунок 18 – Фінальний шар моделі

2.7 Загальна архітектура веб-програми

Для роботи з сервісом користувач повинен авторизуватись. Після авторизації користувачеві буде доступне введення сайту для детектування об'єктів. Перед тим, як розпочати детектування, користувач вводить адресу сайту і вибирає потрібні характеристики веб-сторінки, які є обов'язковими для вибору.

Для реалізації системи було розроблено діаграму компонентів (рис.19), яка показує розбиття додатка для детектування об'єктів на веб-сторінках на програмні модулі.

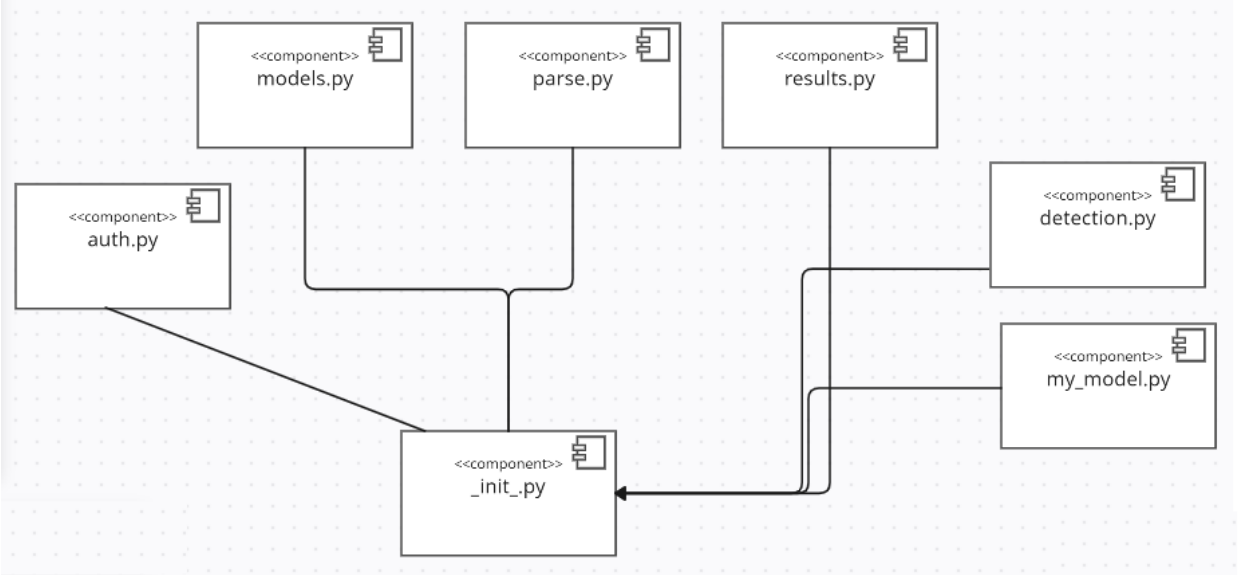

Рисунок 19 – Діаграма компонентів системи

Ця діаграма включає наступні програмні модулі:

1) «auth.py» – відповідає за авторизацію користувача;

2) «models.py» – містить описові класи таблиць у базі даних;

3) «parse.py» – відповідає за обробку даних класифікацію, детектування об'єктів сайтів;

4) «result.py» – відповідає за запис даних до бази даних;

5) «detection.py» – містить у собі архітектуру та алгоритм детектування об'єктів;

6) «my_model.py» – містить у собі архітектуру та алгоритм згорткової мережі;

7) «_init_.py» – відповідає за ініціалізацію проекту.

2.8 Проектування інтерфейсу користувача

Було вирішено створити простий і інтуїтивно зрозумілий інтерфейс для зручності користувача. Макет є зразковим уявленням підсумкового продукту і містить основні необхідні функції (рис. 20). Після авторизації користувача відображається вікно, яке відповідає головному меню. Воно містить такі пункти:

1) профіль;

2) меню;

3) блок, який відповідає за парсинг даних.

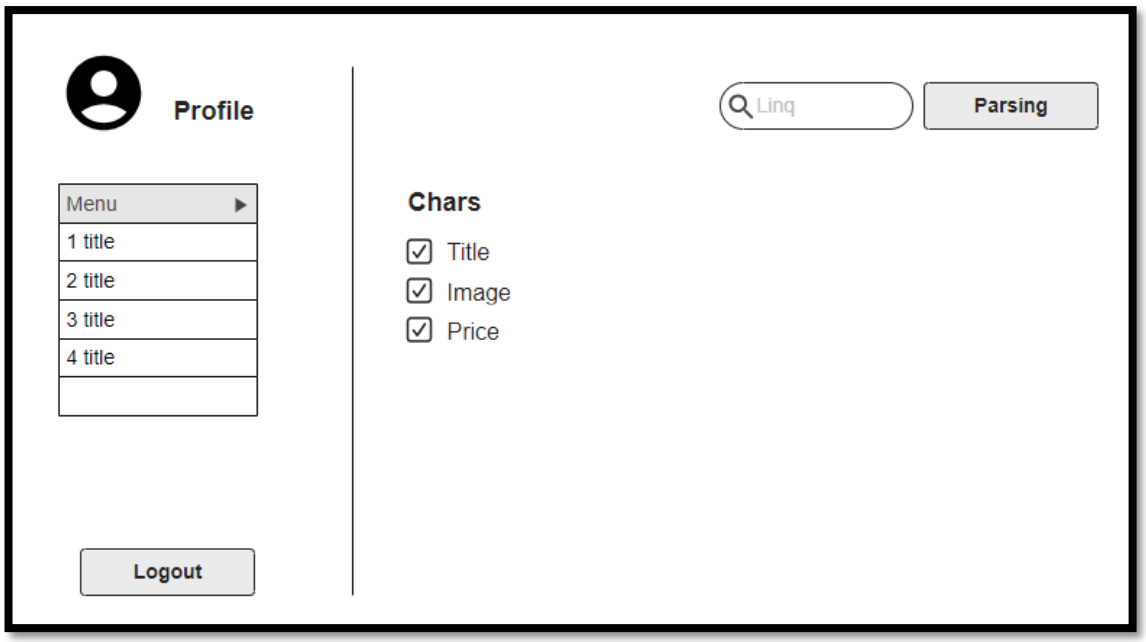

Рисунок 20 – Макет головного вікна

Форма введення «Link» відповідає за отримання адреси сайту. Форма «Char» відповідає за отримання параметрів для розпізнавання. При натисканні на кнопку «Parse», що відповідає за процес парсінгу даних, запускається анімація, яка діє доти, доки відбувається парсінг даних.

Після зробленого парсінгу користувач отримує файл у потрібному форматі та переходить на сторінку основного вікна.

2.9 Проектування бази даних

Для того, щоб надати дані користувачу як результат, слід робити запис цих даних до бази даних. Для зберігання інформації про користувачів та сайти було реалізовано базу даних, яка складається з двох таблиць: «User» і «Site».

У таблиці 1 представлені поля таблиці «User», яка відповідає за зберігання інформації про користувача.

Таблиця 1 – Таблиця «User»

№	Назва столбця	Тип даних	Призначення
1	id	Int	Первиний ключ
2	email	Varchar	Адрес поштової скрині користувача
3	password	Varchar	Зашифрований пароль користувача
4	name	Varchar	Нік користувача

Таким чином, таблиця «User» зберігає інформацію про авторизованого користувача, а також його «email» та «password» для авторизації.

У таблиці 2 представлена таблиця «Site», яка містить такі поля та типи даних, пов'язані з веб-сторінками інтернет-магазинів.

Таблиця 2 – Таблиця «Site»

№	Назва столбця	Тип даних	Призначення
1	id	Int	Первиний ключ
2	site_name	Varchar	Посилання на веб-сторінку
3	title	Varchar	Назва товару
4	domain	Varchar	Домен веб-сторінки
5	depth	Int	Глибина веб-сторінки
6	price	Varchar	Ціна товару
7	img	Varchar	Фото товару
8	user_id	int	Зовнішній ключ до таблиці «User»

Таким чином, таблиця «Site» зберігає інформацію про веб-сторінку, яка була класифікована як картка товару, а також інформацію про те, ким було запущено парсінг. Дані таблиці пов'язані з допомогою ID користувача (user.id і site.user_id).

На рисунку 21 представлена схема бази даних, в яку записуються дані після того, як користувач розпочав процес парсінгу.

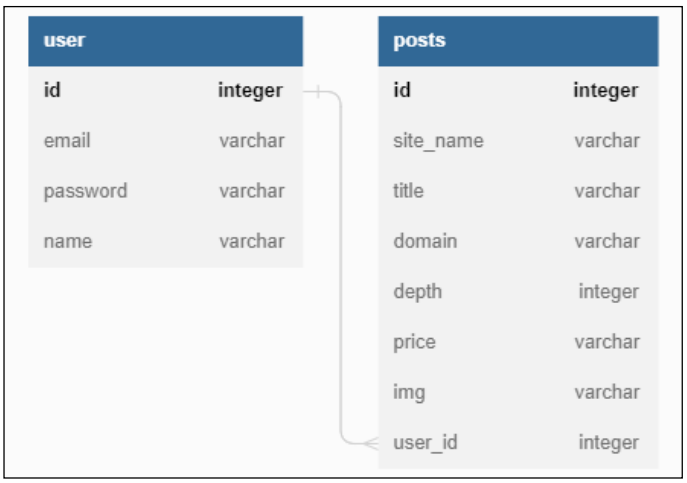

Рисунок 21 – Структура БД

Висновки з другого розділу

У цьому розділі було визначено функціональні та нефункціональні вимоги, на основі яких були спроектовані алгоритми нейронних мереж, вирішальні задачі класифікації та детектування, розглянуто існуючі архітектури нейронних мереж для вирішення задач класифікації та детектування.

У діаграмі компонентів були наведені основні програмні модулі, які відповідають за певні етапи роботи системи та містять архітектуру та алгоритми нейронних мереж.

Для зберігання даних про користувачів системи, а також даних, отриманих у процесі парсінгу, була спроектована база даних, що складається з двох таблиць «Site» і «User».

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Вибір програмних засобів реалізації системи

Веб-скрапінг – дуже потужний метод, який можна використовувати для отримання даних з Інтернету. Ця практика стала для компаній одним з найпопулярніших способів збору інформації про своїх конкурентів, клієнтів та ринок загалом.

Коли справа доходить до парсингу веб-даних, Python є найпопулярнішою мовою серед безлічі інструментів, бібліотек та фреймворків, доступних для парсингу веб-даних. Але є сильні суперники, популярність яких з роками зростає.

Для реалізації програмної частини системи була обрана високорівнева мова Python 3.10, тому що ця мова володіє великою кількістю наявних бібліотек, що підключаються, володіє низьким порогом входу. Розробка велася в хмарному сервісі Google Colab, а також у кросплатформовому інтегрованому середовищі програмування Python Py-Charm.

Ці засоби були вибрані, тому що Google Colab надає можливість редагування коду онлайн, запуск окремих осередків під час навчання, а PyCharm надає просту організацію проектів і є найпопулярнішим рішенням при виборі редактора коду. Для реалізації компонентів були використані найбільш популярні бібліотеки мови Python, які надають функціонал для навчання та добутку даних: tensorflow, numpy, scikitlearn, pytorch, scipy, scikit-image, flask.

Для реалізації бази даних була обрана система управління базами даних SQLite, тому що вона має високу швидкість, надійна, має малий розмір, а також зміна даних доступна за допомогою методів, що надаються flask.

3.2 Реалізація нейронної мережі для класифікації

Для реалізації нейронної мережі для класифікації зображень веб-сторінок інтернет-магазину була використана бібліотека Keras, тому що це простий у використанні API, за допомогою якого можна створювати моделі нейронних мереж. У процесі навчання було використано різні параметри, і після експериментів було обрано варіант із найкращими показниками.

Для оптимізації параметрів моделі було обрано оптимізатор Adam зі швидкістю навчання 0,003, функціями активації було обрано лінійний випрямляч (англ. ReLU) для відбору найбільш значущих ознак зображення та Softmax для отримання ймовірностей класів на вихідному шарі. Функція втрат була обрана категоріальна кросентропія (англ. categorical crossentropy), а метрика точність (англ. precision) для того, щоб збільшити вірогідність вірно відповідального класу.

Під час навчання кількість епох визначалося наступним правилом. Спочатку було 50 епох, але також використовувалися функції зворотних викликів (англ. callbacks): Early Stopping, ModelCheckpoint, через що кількість епох було зменшено. Дані функції дозволяють відстежувати поведінку функції втрат і зупиняти процес навчання при досягненні певної умови. Код створення нейронної мережі для класифікації наведено у лістингу 1.

Усього було зібрано 15000 зображень, кількість навчальних прикладів (англ. batch_size) склала 128, як компроміс між ефективністю та швидкістю навчання.

Лістинг 1 – Створення нейронної мережі для класифікації

```
resnet_model = Sequential()
pretrained_model=
    ResNet50(include_top=False,
    input_shape=(331, 331, 3),
  pooling='max',
    classes=2,
```

```
    weights='ima
    genet')

for layer in pretrained_model.layers:
layer.trainable = False

resnet_model.add(pretrained_model)
resnet_model.add(Flatten())
resnet_model.add(Dense(512, activation='relu'))
resnet_model.add(Dense(84,  activation='relu'))
resnet_model.add(Dense(2,activation='softmax'))
resnet_model.compile(optimizer=Adam(lr=0.003),
loss='categorical_crossen-  tropy',   metrics=[precision_m])
resnet_model.load_weights('project/weights/resnetfinal331.h5
')
```

Вхід моделі є зображенням, переведеним у тензор розмірністю (331, 331, 3). Вихід моделі повертає два значення – ймовірність, з якою вхідні дані належать до одного з двох класів.

Навчання моделі проводилося на хмарній платформі Google Colab із наступними технічними характеристиками Nvidia Tesla T4, ОЗУ 12 Гб/16 Гб, HDD 500 Гб. Фрагмент коду навчання подано на лістингу 2.

Лістинг 2 – Код навчання класифікаційної моделі:

```
history = resnet_model.fit(
x_train_img_np,y_train_np,
validation_data=(x_test_img_np, y_test_np),epochs=10,
verbose=1, callbacks=[model_checkpoint_callback])
```

3.3 Реалізація нейронної мережі для детектування

Для реалізації нейронної мережі для детектування об'єктів на веб-сторінці інтернет-магазину була використана бібліотека PyTorch, тому що завдання детектування є ресурсомісткою, але PyTorch дозволяє переміщати обчислення певного виразу на відеокарту і далі повертати результат назад на процесор. У процесі навчання було використано різні параметри, і після цього було відібрано найкращу модель.

Модель нейронної мережі складається з кількох частин: для візуального представлення вхідних даних вибрано згорткову нейронну мережу, для відбору даних з DOM-дерева було обрано Graph Attention Layer.

На лістингу 3 представлений код створення згорткової нейронної мережі у складі моделі детектування, а створення GraphAttentionLayer представлено на лістингу 4.

Лістинг 3 – Код створення згорткової нейронної мережі:

```
##### REPRESENTATION NETWORK (RN) #####
self.convnet = torchvision.models.resnet18(weights=True)
modules = list(self.convnet.children())[:-5] # remove last few layers!
```

Видаляємо останні п’ять слоїв:

```
self.convnet = nn.Sequential(*modules)

_imgs = torch.autograd.Variable(torch.Tensor(1, 3, img_H, img_H))
_conv_feat = self.convnet(_imgs)
_convnet_output_size = _conv_feat.shape # [1, C, H, W]
spatial_scale = _convnet_output_size[2] / img_H
self.roi_pool = torchvision.ops.RoIPool(roi_output_size, spatial_scale) self.n_visual_feat = (
_convnet_output_size[1] * roi_output_size[0] * roi_output_size[1]
)
self.n_feat = self.n_visual_feat + self.bbox_hidden_dim + self.n_additional
_feat

if self.bbox_hidden_dim > 0:
self.bbox_feat_encoder = nn.Sequential( nn.Linear(5, self.bbox_hidden_dim), nn.BatchNorm1d(self.bbox_hidden_dim), nn.ReLU(),
)
if self.n_additional_feat > 0:
self.bn_additional_feat = nn.BatchNorm1d(self.n_additional_feat) else:
self.bn_additional_feat = lambda x: x ##### GRAPH ATTENTION LATER (GAT) #####
if self.use_context:
self.gat = GraphAttentionLayer(self.n_feat, self.hidden_dim) ##### FC LAYERS #####
```

```
     self.n_total_feat    =    self.n_feat    +    self.hidden_dim
self.decoder     =     nn.Sequential(     nn.Dropout(drop_prob),
nn.Linear(self.n_total_feat,                    self.n_total_feat),
nn.BatchNorm1d(self.n_total_feat),
     nn.ReLU(), nn.Dropout(drop_prob),
     nn.Linear(self.n_total_feat, self.n_classes),
     )
```

Лістінг 4 – Код створення Graph Attention Layer:

```
     self.in_features = in_features self.hidden_dim = hidden_dim

     self.W_i   =   nn.Linear(self.in_features,   self.hidden_dim,
bias=False)      self.W_j      =      nn.Linear(self.in_features,
self.hidden_dim, bias=False)

     self.attention_layer  =  nn.Linear(2  *  self.hidden_dim,  1)
self.leakyrelu = nn.LeakyReLU(alpha)
```

На вхід моделі подається зображення розмірністю (1280, 1280, 3) та DOM-подання HTML-документа. Вихід моделі є зображення з зазначеними елементами і текстове представлення самих елементів.

3.4 Реалізація алгоритму отримання та обробки даних

Для того, щоб розпочати процес парсингу користувачеві потрібно заповнити форму введення адреси сайту, вибрати характеристики для запису у файл та натиснути на кнопку «Parsing!».

Для отримання посилань на веб-сторінки інтернет-магазину рекурсивним методом здійснюється прохід веб-сторінками.

Спочатку робиться запит до головної сторінки, з якої збираються всі потрібні посилання: ті, яких не було, і ті, які є внутрішніми. Потрібні посилання записуються до списку для подальшого застосування до цих посилань передобробки та передачі в нейронні мережі.

Після цього відбуваються ітерації за списком потрібних посилань. Алгоритм отримує посилання, робить скріншот веб-сторінки, класифікує її і,

якщо вона відповідного класу, то посилання потрапляє до списку для детектування. На лістингу 5 подано код алгоритму ітерації за списком необхідних посилань.

Лістинг 5 – Алгоритм ітерації за списком потрібних посилань:

```
for link in links:
try:
elems = detect(link, screen=domain) # if price_checked:
price = elems[0]

# if title_checked:
title = elems[1]

# if photo_checked:
img = elems[2]
depth = urlparse(link).path.count('/')

new_site = Site(title=title, price=price,
img=img, depth=depth, domain=domain, site_name=link
)

db.session.add(new_site) db.session.commit()
```

Після попадання до списку для детектування робиться новий скріншот веб-сторінки іншої дозволу для того, щоб подати зображення в нейронну мережу для детектування об'єктів також подається DOM-дерево веб-сторінки.

Після цього розпізнані елементи записуються до бази даних. На рисунку 22 подано фрагмент записів у базі даних.

https://posudavdom.com/product/...	Набор стаканов для виски хрусталь Bohemia ...	posudavdom.com	2
https://posudavdom.com//blog/novosti/	Новости	posudavdom.com	4
https://posudavdom.com/product/85-1805-blyud...	Блюдо для 8 яиц фарфор lefard Sunday 85-180...	posudavdom.com	2
https://posudavdom.com/product/189-337-tarelk...	Тарелка закусочная фарфор Lefard Ажур Ёлка ...	posudavdom.com	2
https://posudavdom.com/product/...	Ваза для цветов хрусталь Bohemia LISBOA ...	posudavdom.com	2

Рисунок 22 – Фрагмент запису у БД

Реалізація алгоритму отримання даних:

```
def get_valid_links(domain, amount=0): valid_links = []
structure = []

links = [] links_to_visit = [] try:
r = requests.get(domain, headers=header) except MissingSchema:
flash('Пустое поле')
return valid_links, structure

if r.status_code != 200:
raise Exception('Это 404') soup = BeautifulSoup(r.content)
a_nodes = soup.findAll("a", href=True)

for i in a_nodes: a_node_str = i['href']

if a_node_str[
a_node_str.rfind('.'):].lower() in file_exten- sions: #
```

Прибираємо з адресу непотрібне розширення:

```
    continue
    a_node_str = delete_anchors(a_node_str)

    links.append(a_node_str) links_to_visit.append(a_node_str)

    links = set(links)
    links = (list(links))

    links_to_visit = set(links_to_visit)
    links_to_visit = (list(links_to_visit))

    parent = 'Головна' parent_link = domain
    child_links = links_to_visit.copy()

    [structure.append([parent, par-
    ent_link, child_link]) for child_link in child_links] while
links_to_visit:
    add_to_links_to_visit = []
    for i in links_to_visit: link = i
    if len(link) == 0 or 'tel:' in link: continue
```

Наступний крок: слід зробити відносне посилання абсолютним:

```
    if (domain not in link) or ('http' not in link): if link[0]
!= '/':
    link = '/' + link link = do-
    main + link #

    try:
```

```
     r     =     requests.get(link,     headers=header)     except
LocationParseError as e:
     continue
     except ConnectTimeout: continue
     if r.status_code != 200: # Можно еще подумать над редиректами
continue

     soup = BeautifulSoup(r.content)
     parent = soup.select('h1')[0].decode_contents() parent_link
= link

     pred, color = is_product(link) # Классификация на 2 класса if
pred:
     print(link) valid_links.append(link) continue

     if (len(valid_links) >= amount) and (amount > 0): return
valid_links, structure
     a_nodes = soup.findAll("a", href=True) for j in a_nodes:
     a_node_str = j['href']
```

Позбавляємось від адрес з непотрібним розширенням та прибираємо якорні посилання:

```
     if a_node_str[a_node_str.rfind( '.'):].lower() in file_exten-
     sions:
     continue
     a_node_str = delete_an- chors(a_node_str)
     structure.append([parent,   parent_link,   a_node_str])   if
a_node_str in links:
     continue

     links.append(a_node_str)
add_to_links_to_visit.append(a_node_str)

     links_to_visit = add_to_links_to_visit # print(valid_links)
     return valid_links, structure
```

3.5 Реалізація інтерфейсу користувача

На рисунку 23 представлено інтерфейс основного вікна програми, що містить поле введення адреси сайту. У даному випадку сайт «www.englishhome/kitchen» зчитується додатком для аналізу даних (опис та фото товарів).

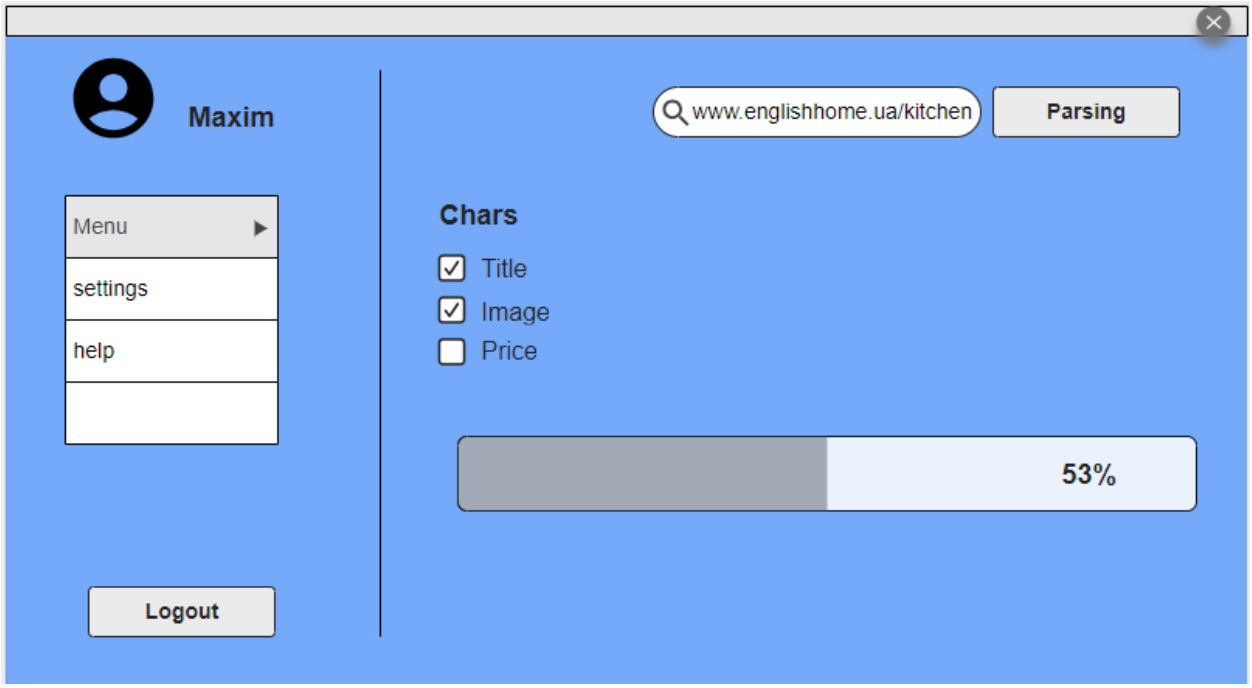

Рисунок 23 – Інтерфейс головного вікна

Інтерфейс програми було реалізовано з допомогою HTML, CSS. Анімації в додатку були реалізовані за допомогою JavaScript, зокрема jQuery. jQuery – набір функцій JavaScript для взаємодії JavaS-cript та HTML. Інтерфейс основного вікна програми складається з меню та блоку введення даних.

Пункт меню Data Extract відповідає за основне вікно програми, яке надає графічний інтерфейс введення даних, вибору необхідних характеристик. Блок введення даних складається з форми введення адреси посилання, що є HTML-тегом «form», кнопки «Parsing», що є HTML-тегом button та форми вибору характеристик товару, що є HTML-тегом checkbox.

Висновки з третього розділу

Відповідно до завдання роботи були зібрані необхідні дані для навчання нейронних мереж. Крім цього, були навчені та протестовані нейронні мережі, що обробляють зображення та текстовий вигляд HTML-документа, за

допомогою яких з'являється можливість класифікації сторінок веб-сайту та детектування необхідних характеристик даних веб-сайтів.

Відповідно до макету інтерфейсу головного вікна веб-додатка було реалізовано веб-інтерфейс для представлення веб-додатку, який містить такі необхідні форми, як форма введення адреси сайту, форма вибору необхідних характеристик.

4 ТЕСТУВАННЯ СИСТЕМИ

Тестування програмного забезпечення (ТПО) – це процес перевірки та оцінки якості ПЗ з метою виявлення помилок, дефектів та проблем. Метою тестування є переконатися, що ПЗ працює правильно, відповідає вимогам та очікуванням користувачів, а також забезпечує надійність, безпеку та ефективність роботи.

Основні елементи процесу тестування – планування тестування, розробка тестових кейсів та сценаріїв, виконання тестів, аналіз результатів та звітність. Важливо також враховувати та поєднувати різні типи тестування, такі як функціональне, навантажувальне та ін., а також використовувати автоматизацію тестування для підвищення ефективності та повторюваності процесу.

Для тестування системи потрібно провести функціональне тестування, а також тестування та оцінка якості нейронних мереж.

4.1 Функціональне тестування

Функціональне тестування програмного забезпечення є важливою частиною будь-якої процедури тестування програмного забезпечення. Якщо все зробити правильно з першого разу, це допоможе уникнути дорогого та трудомісткого ремонту надалі та зберегти клієнтів задоволеними. Функціональне тестування перевіряє відповідність вимог проектованої системи.

При проведенні функціонального тестування ви шукаєте будь-які прогалини, помилки або те, що немає у вимогах до програмного забезпечення або додатку. Різниця між системним та функціональним тестуванням полягає в тому, що при системному тестуванні тестується вся система, а при функціональному – лише окремі функції.

Таблиця 3 – Функціональне тестування

№	Назва тесту	Кроки	Очікуваний результат	Успіх тесту
1	Відправлення пустої строки	1. Перейти у розділ детектування. 2. Відправити пусту строку у полі адреси сайту 3. Натиснути «Parsing».	Повідомлення щодо некоректного вводу даних	Да
2	Відправлення Повної строки	1. Перейти у розділ детектування. 2. Відправити заповлену строку у полі адреси сайту 3. Натиснути «Parsing».	Починається алгоритм класіфікації сторінки і детеклування даних	Да
3	Відправлення неіснуючого сайту	1. Перейти у розділ детектування. 2. Відправити неіснуючу адресу 3. Натиснути «Parsing».	Повідомлення щодо помилкового посилання	Да
4	Перевірка роботи кнопки «Parsing».	У головному меню натиснути кнопку «Parsing».	Початок анімації роботи системи	Да
5	Перевірка роботоздатності модуля класифікації та детектування	У головному меню натиснути кнопку «Parsing».	Після натискання кнопки йде запуск процесів класифікації та детектування	Да

4.2 Оцінка якості роботи нейронних мереж

Для оцінки якості роботи нейронної мережі для класифікації сторінок веб-сайту було сформовано валідаційну вибірку, що складається з зображень веб-сторінок, які не увійшли до навчальної вибірки.

Модель для класифікації сторінок веб-сайту досягла за метрикою значення 0.9433 на валідаційній вибірці. Ця метрика показує, наскільки вірна модель під час прогнозування веб-сторінки сайту (визначає точність передбачень). На рисунку 24 представлені показники навчання нейронної мережі для класифікації сторінок веб-сайту, які були отримані при оцінці результатів на навчальній вибірці та валідаційній вибірці, також на малюнку представлені показники помилок для кожної з вибірок відповідно.

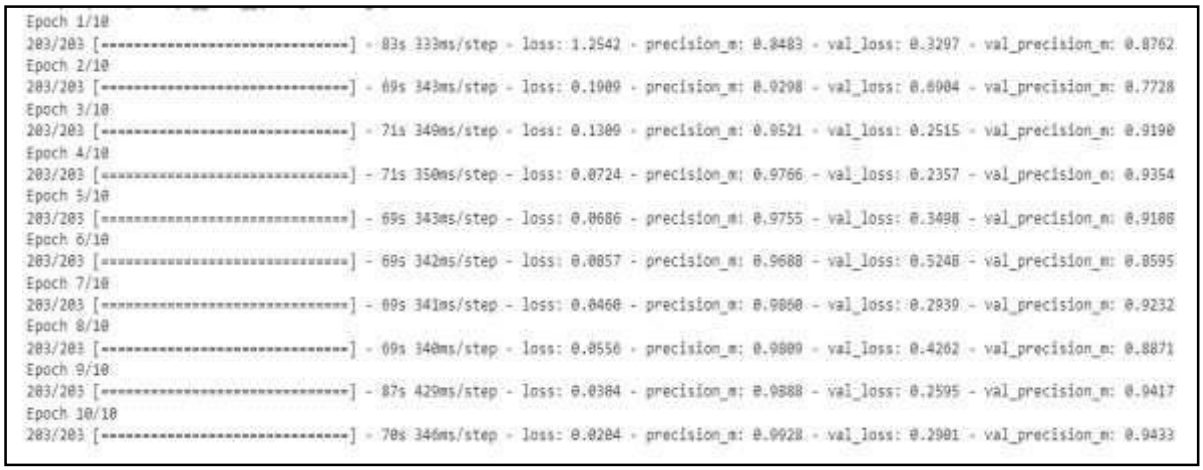

Рисунок 24 – Показники навчання нейронної мережі для класифікації сторінок веб-сайту

Під час навчання нейронних мереж використовувалася крос-валідація. Таким чином, дані розбилися на п'ять частин, у яких для навчання моделі використовувалися чотири частини, а для валідації одна частина.

Модель для детектування сторінок сайту на валідаційній вибірці досягла за метрикою mAP значення 97.3. Ця метрика показує наскільки чітко виділяються межі на цих скріншотах. На рисунку 25 представлені результати навчання моделі для детектування залежно від кількості сусідніх елементів основних характеристик: ціна, назва, фотографія.

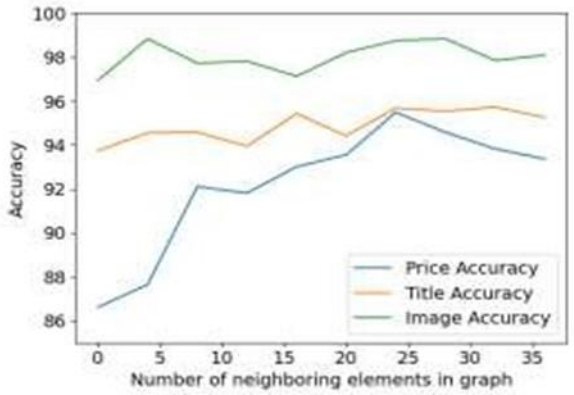

Рисунок 25 – Результати навчання моделі для детектування

4.3 Тестування роботи нейронних мереж

Для оцінки правильності роботи мережі слід подати на вхід невідому модель веб-сторінку та оцінити те, який клас їй буде присвоєно і те, які блоки будуть виявлені на ній.

На рисунку 26 представлено розпізнану як картку товару веб-сторінку. На цьому зображенні можна побачити три блоки: зелений, синій, червоний. Дані блоки відповідають характеристикам парсингу. На рис. 27 представлені дані, отримані з розпізнаної веб-сторінки: посилання, назва, домен, глибина, ціна.

Точність роботи моделі багато в чому визначається веб-сайтом, поданим на вхід. Однією з причин падіння чи збільшення точності є різні види верстok. Подолати цю проблему можна збільшенням навчальних прикладів, тому що збільшиться кількість даних для крос-валідації, а також з'являться навчальні приклади, які будуть містити інформацію або ознаки, відмінні від тих, що виявлялися раніше.

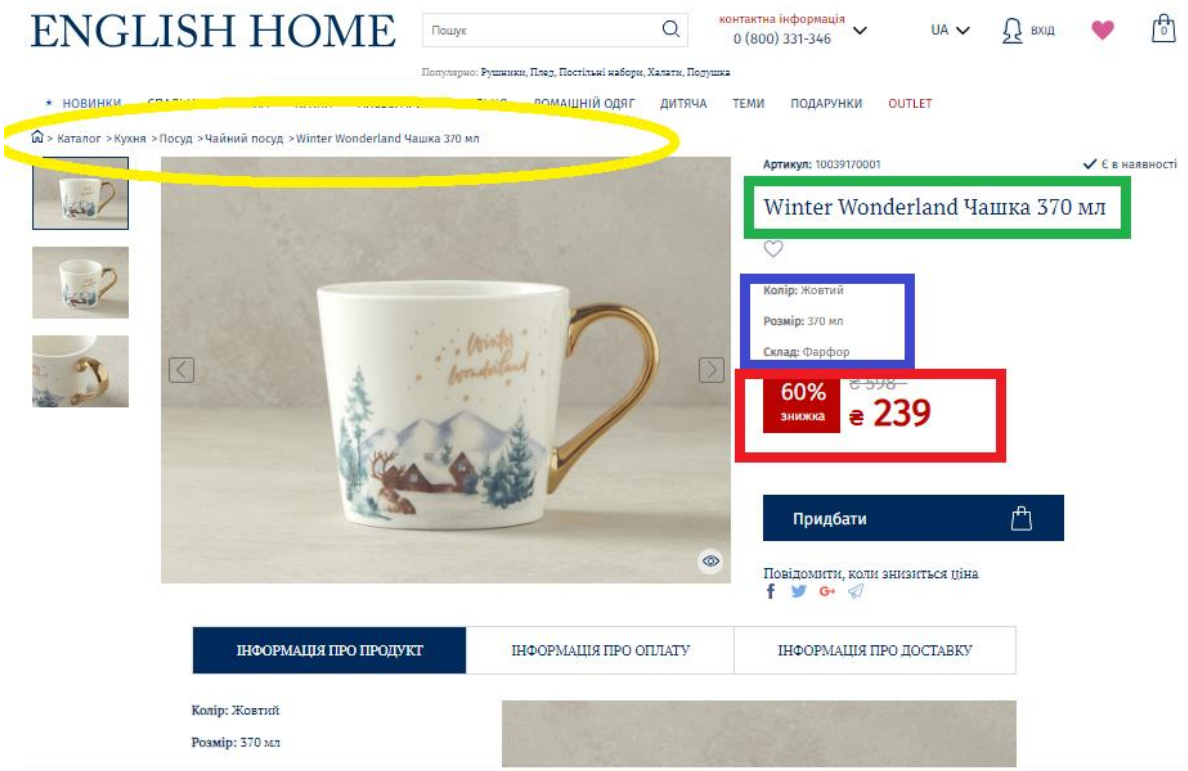

Рисунок 26 – Веб-сторінка для парсінгу даних

2	Winter Wonderland Чашка 370 мл	239	
3	Karlina Набір серветок 38 см 2 шт	<span= class “product_price_prefix” >	

Рисунок 27 – Детектування необхідних даних

Результати функціонального тестування та тестування нейронних мереж збігаються з очікуваними. Функціональне тестування та тестування нейронних мереж пройдено успішно.

ВИСНОВКИ

В результаті виконання кваліфікаційної магістерської роботи було проведено проектування та розробку системи для детектування об'єктів з використанням нейронних мереж. При виконанні роботи досліджені технології розробки, проаналізовано методи реалізації поставленої мети, а також оглянуті нейронні мережі, які можуть використовуватись для вирішення таких задач.

Під час проектування було досліджено шляхи парсінгу веб-даних у мережі Інтернет, проблеми які виникають. Визначено актуальність задачі, роль програми-парсера. Досліджено інструментарій, за допомогою якого можна реалізувати дане завдання, а також проаналізовано варіанти синтаксичного розбору веб-сторінок, імпорту та експорту даних.

В результаті виконано програмну реалізацію веб-системи для детектування. Крім навчання нейронної мережі, було здійснено тестування програми. Практична цінність роботи полягає в тому, що створена система забезпечує зручний засіб для збору даних за певними категоріями за короткий час для користувача.

Як подальший розвиток буде розробка та покращення даного веб-додатку, а саме додавання можливість вивантаження даних у потрібний формат та ресурс, покращення якості роботи нейронних мереж.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Neural Networks for Applied Sciences and Engineering From Fundamentals to Complex Pattern Recognition. By Sandhya Samarasinghe. 594 Pages. ISBN 9780849333750.
2. Neural Network Design (2nd Edition). Martin T. Hagan, Howard B. Demuth, Mark H. Beale. ISBN-13: 978-0-9717321-1-7
3. Convolutional Neural Networks, Explained. URL: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939 (дата звернення 12.09.2023)
4. Parse. URL: https://developer.mozilla.org/en-US/docs/Glossary/Parse (дата звернення 12.09.2023)
5. Object Detection. URL: https://huggingface.co/tasks/object-detection (дата звернення 25.09.2023)
6. BeautifulSoup vs. Scrapy: Which Is Better For Web Scraping? URL: https://scrapingrobot.com/blog/beautifulsoup-vs-scrapy/ (дата звернення 11.10.2023)
7. 15 Best Web Scraping Tools in 2023 to Extract Online Data. URL: https://popupsmart.com/blog/web-scraping-tools (дата звернення 11.10.2023)
8. Turn unorganized web content into well-structured datasets. URL: https://www.diggernaut.com/ (дата звернення 11.10.2023)
9. An Overview on Multilayer Perceptron (MLP). URL: https://www.simplilearn.com/tutorials/deep-learning-tutorial/multilayer-perceptron (дата звернення 16.10.2023)
10. Hopfield Networks: Neural Memory Machines. URL: https://towardsdatascience.com/hopfield-networks-neural-memory-machines-4c94be821073 (дата звернення 17.10.2023)

11. Models for Object Detection. URL: https://medium.com/neuronio/understanding-convnets-cnn-712f2afe4dd3 (дата звернення 17.10.2023)
12. What are convolutional neural networks? URL: https://www.ibm.com/topics/convolutional-neural-networks (дата звернення 20.10.2023)
13. The Complete Guide To Understand IDEF Diagram. URL: https://www.edrawmax.com/article/the-complete-guide-to-understand-idef-diagram.html (дата звернення 24.10.2023)
14. What is Use Case Diagram? URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/;WWWSESSIONID=9A865EF47832494EB7ADC0C356F9BDF4.www1 (дата звернення 24.10.2023)
15. Webpage Object Detection on CoVA. URL: https://paperswithcode.com/sota/webpage-object-detection-on-cova (дата звернення 15.11.2023)