ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до магістерської роботи
на ступінь вищої освіти магістр

на тему: «**РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ТА ГРУПУВАННЯ УКРАЇНОМОВНИХ ДОКУМЕНТІВ НА ОСНОВІ НЕЙРОННИХ МЕРЕЖ»**

Виконала: студентка 6 курсу, групи ПДМ–61
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)

Гордієнко К.О.
(прізвище та ініціали)

Керівник Жебка В.В.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Київ –2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти -«Магістр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
Негоденко О.В.
"____" _____2020 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТЦІ

ГОРДІЄНКО КАТЕРИНІ ОЛЕКСАНДРІВНІ
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка інформаційної системи розпізнавання та групування україномовних документів на основі нейронних мереж»

Керівник роботи: Жебка Вікторія Вікторівна, к.т.н., доцент кафедри ІПЗ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «13» жовтня 2020 року №230.
2. Строк подання студентом роботи «24» грудня 2020 року
3. Вхідні дані до роботи
Методи обробки зображень;
Науково-технічна література з питань, пов'язаних з програмним забезпеченням щодо розпізнавання тексту з зображень;

4. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити).
4.1 Системи розпізнавання та вилучення текстової інформації з зображень.
4.2 Вимоги та оцінка якості системи.
4.3 Опис проектування системи.
4.4 Опис використаних технологій.

5. Перелік демонстраційного матеріалу (назва основних слайдів)

1. Актуальність проблеми
2. Існуюче програмне забезпечення
3. Передобробка зображень
4. Вилучення полів
5. Вилучення таблиць
6. Архітектура ПЗ
7. Інтерфейс розробленої системи

6. Дата видачі завдання «02» листопада 2020

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	02.11-07.11	
2	Вимоги до системи	08.11-10.11	
3	Створення та навчання моделі для вилучення полів	11.11-18.11	
4	Створення та навчання моделі для вилучення таблиць	19.11-28.11	
5	Концепція та архітектура програмного забезпечення	29.11-04.12	
6	Вступ, висновки, реферат	05.12-12.12	
7	Розробка обов‘язкових демонстраційних матеріалів	13.11-15.12	
8	Попередній захист роботи	16.12	
9	Здача роботи	24.12	

Студентка _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина магістерської роботи 72 с., 30 рис., 34 джерел.

РОЗПІЗНАВАННЯ ДОКУМЕНТІВ, НЕЙРОННІ МЕРЕЖІ, ДОДАТОК MICROSOFT FORM RECOGNIZER, ДОДАТОК GOOGLE FORM PARSER, ДОДАТОК AMAZON TEXTRACTS, ВЕБ ДОДАТОК

Об'єкт дослідження – розпізнавання та групування україномовних документів на основі нейронних мереж.

Предмет дослідження – додаток на основі моделі розпізнавання та групування україномовних документів на основі нейронних мереж.

Мета роботи – покращення процесу розпізнавання та групування україномовних документів за допомогою інформаційної системи розробленої на основі нейронних мереж.

Методи дослідження – методи розпізнавання та групування інформації, емпіричні методи.

У роботі проведено аналіз існуючих додатків, таких як Microsoft Form Recognizer, Google Form Parser, Amazon Textracts, Nanonets та ін. методів розпізнавання та вилучення даних представлені у роботі додатків та основному методу роботи таких додатків – використання нейронних мереж.

Загальною проблемою цих продуктів є необхідність у постійному з'єднанні з мережею інтернет, що може ставити небезпеку для збереження та обробки конфіденційних даних

Особливістю методу є можливість автоматичного розпізнавання документу, з можливістю внесенні змін у вилучені дані, та додання нових. Описано архітектуру та основні принципи розробки. За основу було взято технологію Python+Django з серверної сторони та JavaScript з клієнтської сторони для основного функціоналу.

Додаток працює з вже існуючими базами даних. В якості серверу баз даних було взято MongoDB та бібліотеку Mongoengine для взаємодії з нею.

Отже, розроблено та описано веб-додаток, завданням якого є розпізнавання та групування україномовних документів.

У якості вихідних даних є зображення з текстовими даними.

Даний додаток може бути використано у всіх сферах, які потребують автоматичного вилучення інформації з зображень і їх подальшої обробки.

Галузь використання – розпізнавання документів.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

NN– Neural Network

OCR – Optical Character Recognition

LSTM - Long short-term memory

SDK – software development kit

API - Application Programming Interface

PaaS – Platform as a Services

GCNs for VRDS – Graph Convolution for Multimodal Information Extraction for Visually Rich Documents

cGAN – Conditional Generative Adversarial Networks

MLP – Multilayer perceptron

ReLU – rectified linear unit

JSON – Java Script Object Notation

GUI – Guide User Interface

QA – Quality Assurance

ПЗ – програмне забезпечення

ORM – Object-Relation Mapping

СУБД – Система Управління Базами Даних

ВСТУП

Обґрунтування вибору теми та її актуальність: під час ведення документообігу або обробки даних фахівцям часто потрібно вручну вводити дані в інформаційні системи з паперу, або навіть з зображень, отриманих засобами електронної комунікації.

Для пришвидшення введення даних та їх обробки необхідно розробити інформаційну систему, яка автоматично буде вилучати необхідні дані з зображень, отриманих шляхом цифровізації або через способи електронної комунікації.

Ступінь вивчення проблеми: На даний момент існує досить багато продуктів, які займаються автоматичним вилученням тексту. Але, небагато за таких продуктів базується на технології нейронних мереж. До останніх належать Microsoft Form Recognizer, Google Document Parser, Amazon Textracts та Nanonets, але навіть серед представлених продуктів не всі підтримують вилучення україномовних даних. Усі ці продукти базуються також на хмарних технологіях, що не завжди підходить для обробки конфіденційних даних.

У зв’язку з цифровізацією суспільства, проблема автоматичного вилучення значень з електронних документів може виникати у підприємств будь-якої галузі. На даний час, для вилучення даних часто використовується робота звичайних операторів ПК, що займає багато часу. Через людський фактор, не виключається і наявність помилок у вилучених даних.

Існуючі системи вилучення даних дозволяють спростити роботу операторів, прискорюють час обробки документів, зменшують кількість помилок за рахунок відсутності людського фактору, проте не виключають їх повністю, через що присутність оператору все ще важлива.

Об’єктом дослідження є розпізнавання та групування україномовних документів за допомогою нейронних мереж

Предметом роботи інформаційна система для розпізнавання та групування україномовних документів на основі нейронних мереж

Метою роботи є пришвидшення вилучення даних з україномовних документів шляхом розпізнавання та групування їх.

Завданням роботи є розробка інформаційної системи, що дозволяє розпізнавати та групувати україномовні документи

Методика дослідження: Перш за все, потрібно було визначити правильну методику розпізнавання та класифікації документів, вибрати правильну архітектуру та стек технологій із забезпеченням коректної працездатності продукту, за можливості підтримки його розробником з метою ефективного додавання нового функціоналу або виправлення виявлених недоліків.

Враховуючи вимоги до специфіки програмного продукту, найкращим рішенням є веб-додаток, що має клієнт-серверну архітектуру, де клієнтська сторона відповідає за роботу та логіку програмного продукту, а серверна – за обчислення. Такий тип технології дає можливість використання складних обчислень, не призводячи до необхідності введення великої кількості даних.

Для клієнтської сторони, інформаційна система має включати в себе бібліотеку вже вилучених даних, та сторінку для обробки документів. Сторінка для обробки документі представлятиме з себе форму для введення/редагування даних та зображення документу що потребую обробки.

Результати обробки документів можна переглядати як всередині інформаційної системи, так і експортувати їх у зручні формати даних.

Наукова новизна роботи: Таким чином, наукова новизна полягає в створенні нейромережі, і на її основі створення інформаційної системи, що дозволить розпізнавати та групувати дані на україномовних документах

Практична значущість результатів: Даний продукт може бути використаний у всіх сферах діяльності, де потрібне вилучення даних з зображень отриманих шляхом цифровізації або через засоби електронної комунікації.

1 СИСТЕМИ РОЗПІЗНАВАННЯ ТА ВИЛУЧЕННЯ ТЕКСТОВОЇ ІНФОРМАЦІЇ З ЗОБРАЖЕНЬ

1.1 Azure Form Recognizer

Azure Form Recognizer – це хмарний продукт від компанії Microsoft. Azure Form Recognizer - це «когнітивний сервіс», який дозволяє створювати програмне забезпечення для автоматизованої обробки даних за допомогою технології машинного навчання.

Суть цього продукту в вилученні тексту, формуванні пар ключ-значення та автоматичне вилучення з документів таблиць – служба автоматично визначає структуровані дані формує зв’язки у вихідному файлі. Технологія дозволяє швидко та точно отримувати результати , що пристосовані до змісту конкретних документів.

Form Recognizer має власні моделі обробки документів, такі як чеки або візитівки, та готові макети API. Form Recognizer використовується через REST API або SDK, тому його неможливо використовувати як повноцінний продукт.

Можливі наступні варіанти використання Form Recognizer:

1. Кастомні моделі – оператор або програміст формує пари ключ-значення на оригінальному документі, технологія автоматично вилучає таблиці. В результаті роботи оператора створюється модель пристосована до необхідних документів, і в подальшому доступна до імплементації у будь-який програмний продукт через REST API та SDK.
2. Вбудовані моделі – вилучає дані із заздалегідь побудованих моделей. На даний момент доступні готові моделі квитанцій про продаж (чеків) та візитні картки. Доступні лише для повністю англомовних документів.
3. Макети API - витягує з документів структуру тексту та таблиці, а також їх обмежуючі координати.

Найпростішим і найточнішим методом розпізнання документів є використання кастомних моделей, за рахунок того що пари ключ-значення формуються оператором, і для вбудовування в створюване програмне забезпечення потрібні лише коди доступу до Cognitive Services, які формуються службами Azure.

Для навчання кастомної моделі необхідно лише 5 документів.

Кастомні моделі формуються за допомогою веб-додатку Sample Labeling Tool. Процес відбувається так:

1. Через служби Azure створюється проекту, група ресурсів, екземпляр Cognitive Service, та сховище.
2. У Sample Labeling Tool вказуються дані для з'єднання з Cognitive Service та сховищем
3. Додаються як мінімум 5 документів. Система проводить розпізнавання документів та відображає кордони розпізнаних слів на оригінальному документі. При цьому, орієнтація та нахил тексту на документах не мають значення.
4. Формується список ключів для визначення даних. Ключі потрібно формувати лише для звичайних полів, так як таблиця вилучається автоматично.
5. Кліками по словах на оригінальному зображенні оператор позначає дані, необхідні для вилучення поля.
6. Після вибору слів, кліком по ключу формується зв'язок ключ-значення
7. Кроки 4-6 повторюються мінімум для 5 документів
8. Виклик тренування моделі на основі навчених даних. В залежності від кількості навчених документів, час на тренування займає приблизно хвилину.
9. Тестування документів

Подальше використання моделі можливе через вбудовування моделі у інше програмне забезпечення використовуючи API, ключі до якого автоматично створюються службами Azure.

Azure Form Recognizer підтримує наступні мови – китайська (спрощена), датська, англійська (друкований та рукописний текст), французька, німецька, італійська, португальська та іспанська.

При спробі все-ж провести розпізнавання україномовних документів, будуть отримані наступні результати:

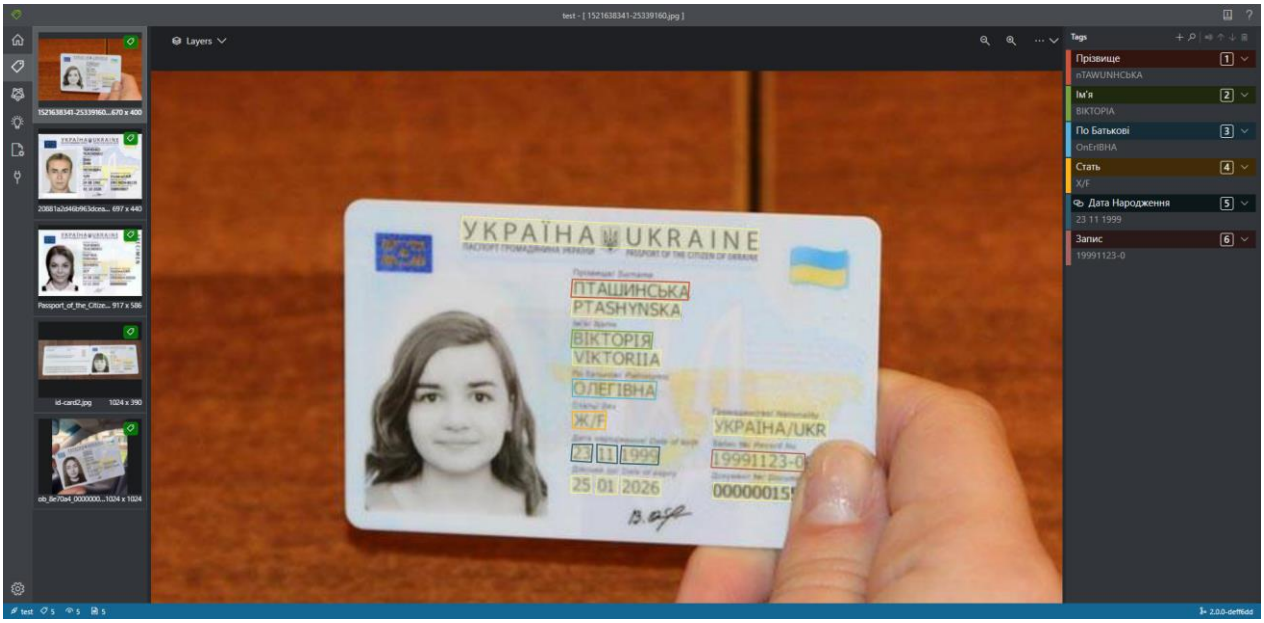


Рисунок 1.1 Вилучення даних з україномовних документів за допомогою Microsoft Form Recognizer на прикладі української ID-картки

Розглянемо приклад розпізнавання іншого зображення:

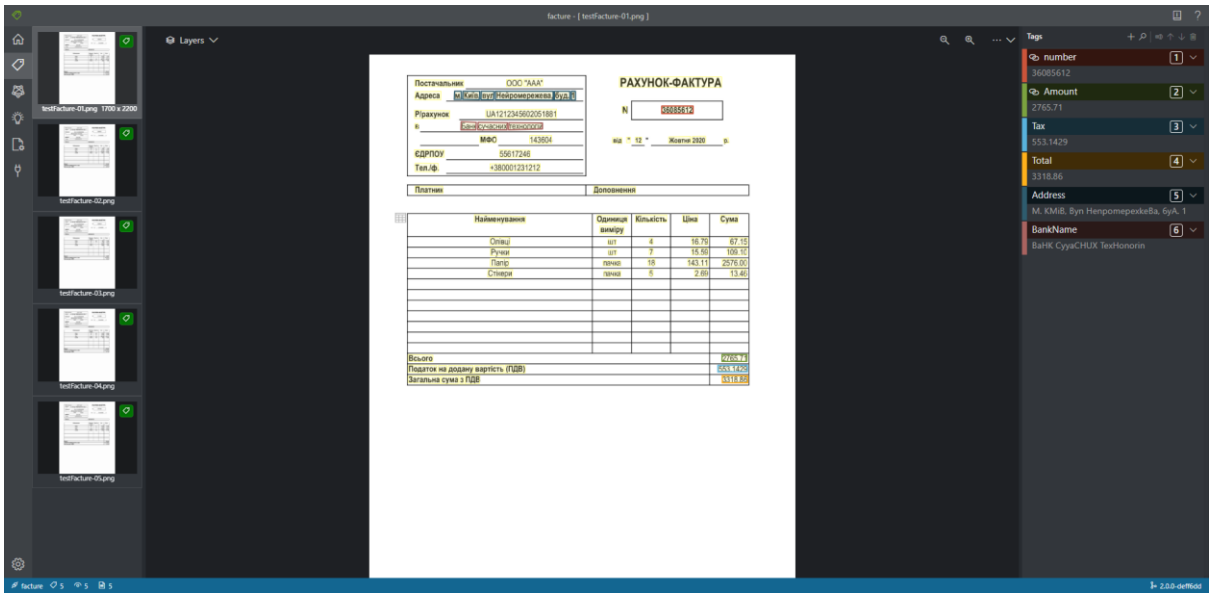


Рисунок 1.2 Вилучення даних з україномовних документів за допомогою Microsoft Form Recognizer на прикладі рахунку-фактури

Вилучення таблиці відбувається в автоматичному режимі, без втручання оператора в процесі навчання:

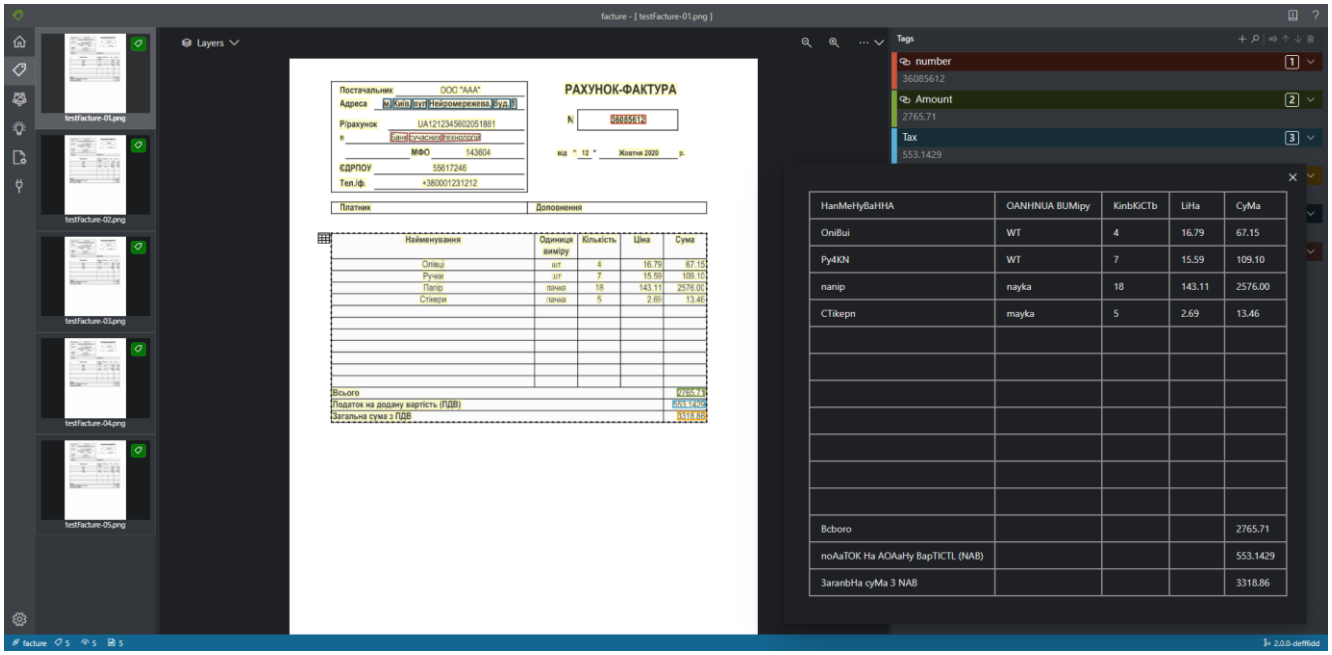


Рисунок 1.3 Вилучення таблиці з україномовного рахунку-фактури за допомогою Microsoft Form Recognizer

Як видно з рисунків 1.1, 1.2, 1.3, служба може навчатись на документах з україномовними значеннями, і навіть успішно вилучає числові значення, але не може розпізнати україномовний текст. Служба намагається накласти кириличні символи як латинські, і підбирає найбільш подібний аналог.

1.2 Google Document AI

Document AI може вилучати увесь текст з документу, аналізувати та вилучати текст файлів, в тому числі текст з неструктурованими даними (поля, вдати, відмітки, тощо) у формалізованих документах, або розбивати один документ з інформацією на декілька документів, керуючись логічною структурою цих документів.

Технологія може використовуватись лише онлайн, викликаючись через REST API, хмарну консоль, та через користувацькі бібліотеки, що можуть бути створені на мовах програмування Java, Node.js та Python.

Файли малих розмірів і з невеликою кількістю документів можливо опрацьовувати у повністю онлайн-режимі. Для великих файлів (але менше 2000 сторінок) використовується напів-оффлайн режим, який полягає в посиланні асинхронних запитів до екземпляру хмарних обчислень, а результат обробки документів зберігається у хмарному сховищі.

Система Document AI підтримує більше 200, які розподілені за наступними критеріями:

- Підтримувані мови – це мови, які являються пріоритетними для розробки, і ефективність яких періодично оцінюється. До множини таких мов входить і українська мова.
- Експериментальні мови – це мови які активно розробляються, але їх ефективність оцінюється не так регулярно, як для підтримуваних мов.
- Накладені мови – це ті мови, відображення яких підтримується, але всередині системи вони розглядаються як одна з підтримуваних мов, через їх подібність або ідентичний алфавіт. Наприклад, британська англійська підтримується системою, але розглядається як просто англійська. Або діалекти китайської мови, такі як кантонський, шанхайський або мандарин, які розпізнаються як путунхуа – офіційна китайська мова.

Form Parser – це технологія, що створена компанією Google, і є частиною технології Document AI.

Підсистема Form Parser має вже навчені моделі для певних форм документів – рахунків-фактур, чеків, американських податкових форм (W9, 1040, W2, 1099-MISC), позикових документів та форми 1003.

Загальний розпізнавач документів вилучає максимально всі дані документів. Для коректної роботи моделі потрібно 10 документів. Система автоматично формує пари ключ-значення та вилучає таблиці. Система автоматично повертає зображення у потрібне для розпізнавання положення.

При розпізнаванні україномовних документів через Form Parser можуть бути отримані наступні результати:

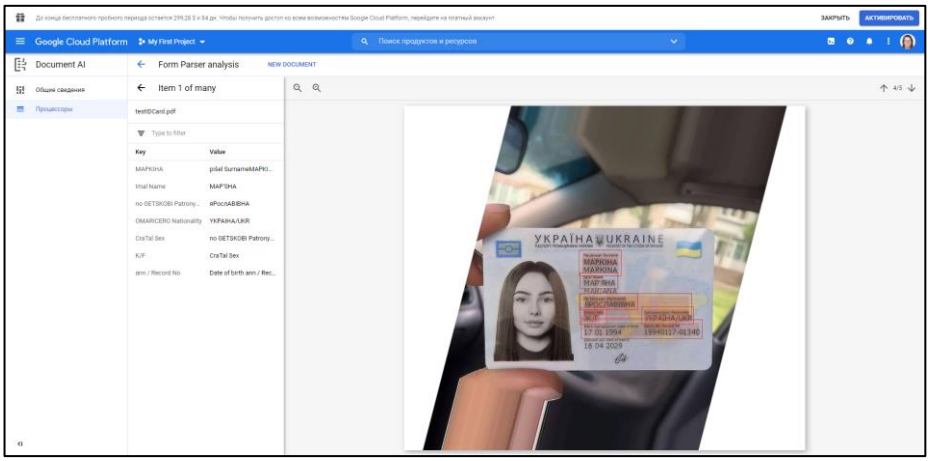


Рисунок 1.4 Вилучення даних з україномовних документів за допомогою Google Form Parser на прикладі української ID-картки

На рисунку 1.4 видно, що система досить погано працює з двомовними документами, такими як українська ID-картка. Наявні проблеми як найменуванні ключів, так і в самих вилучених даних.

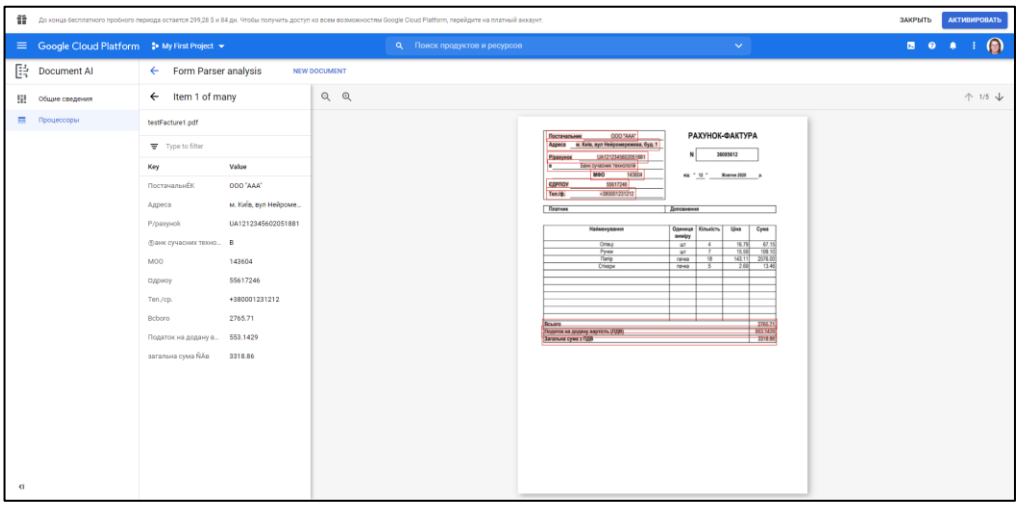


Рисунок 1.5 Вилучення даних з україномовних документів за допомогою Google Form Recognizer на прикладі рахунку-фактури

На рисунку 1.5 очевидно, що система вірно вилучає самі дані, але має проблеми в коректному розпізнаванні назв ключів.

В обох випадках, видно що система навіть на документах, що включає в себе одну мову, в деяких випадках намагається розпізнати латинські символи замість кирилиці.

1.3 Amazon Textracts

Textracts – це технологія, що була створена компанією Amazon. Служба дозволяє автоматично визначати пари ключ-значення та таблиці без втручання оператора. Якщо до документів потрібна особлива увага, тоді до системи під'єднується Amazon Augmented AI, котре самостійно визначає, чи потрібне втручання людини в розпізнавання тексту.

Служба Textracts підтримує лише латинський алфавіт та ASCI-символи. При застосуванні служби до україномовних документів, буде отриманий наступний результат:

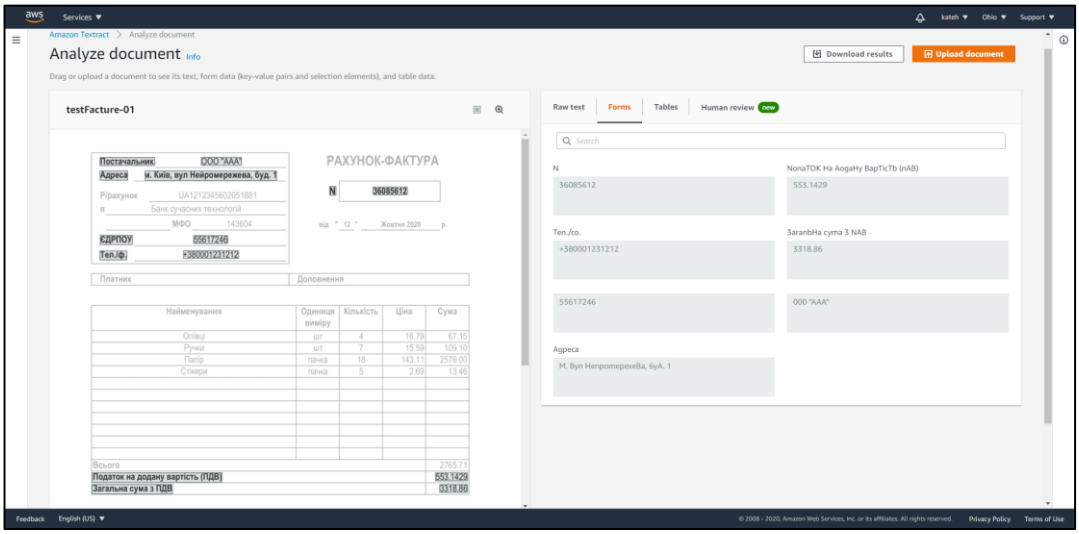


Рисунок 1.6 Вилучення даних з україномовних документів за допомогою Amazon Textracts на прикладі рахунку-фактури

Система автоматично визначає кордони стовпчиків таблиці, але не визначає імена стовпчиків таблиці

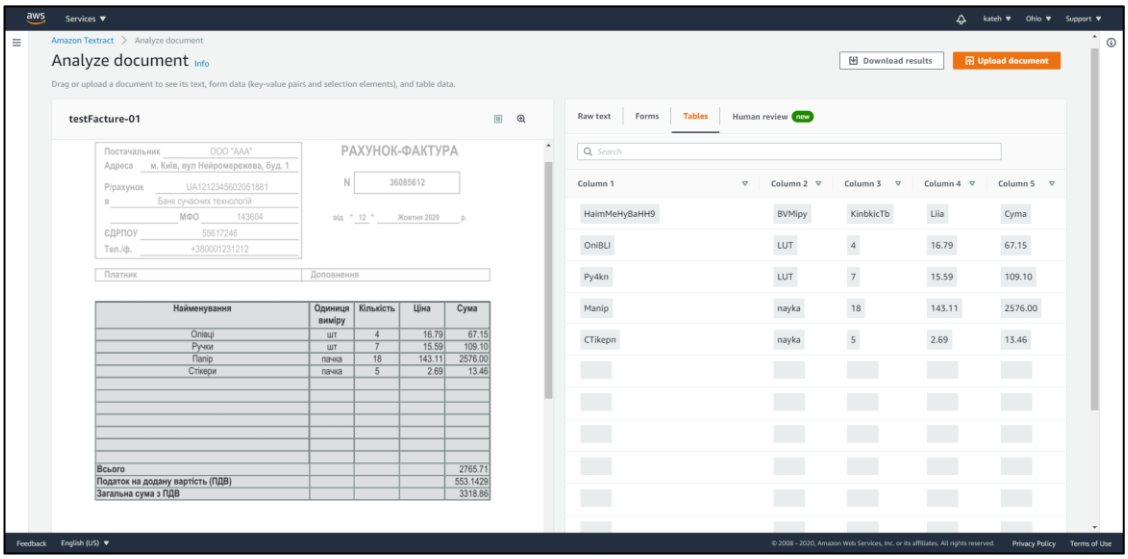


Рисунок 1.7 Вилучення таблиці з україномовного рахунку-фактури за допомогою Textracts

На рисунках 1.6, 1.7 видно що система шукає латинські аналоги символів кирилиці, і таким чином не може використовуватись для розпізнавання україномовних документів.

1.4 Nanonets

Nanonets – це веб-система глибокого навчання, що дозволяє розпізнавати, верифікувати та експортувати вилучені з документів дані.

В системі доступні такі попередньо-навчені OCR-моделі – рахунки-фактури, чеки, американські посвідчення водія, паспорти, ресторанне меню, резюме, автомобільні номерні знаки, показники лічильників та маркування морських вантажних контейнерів.

Система надає можливість самостійно навчити модель вилучати дані з власних документів.

Процес навчання відбувається наступним чином:

1. Завантажується мінімум 10 однотипних файлів. Чим більше файлів було завантажено – тим краще система навчається.
2. Визначаються ключі для розпізнання значень

3. На усіх завантажених документах оператор малює прямокутних на завантажених документах поверх значень що потрібні для вилучення.
4. Намальовані прямокутники оператор відносить до заданих ключів.
5. Навчання моделі, приблизно хвилин 20

Готову для навчання модель можна використовувати як в самій веб-системі, так і вбудувати до власного додатку використовуючи API. Дані розпізнаються лише онлайн.

Документи розпізнані у веб-додатку зберігаються у системі і доступні до перегляду і верифікації у будь-який момент. Дані, вилучені з документа, підсвічуються на оригінальному зображенні, і є можливість змістити зону розпізнавання. Для вилучених даних вказується рівень впевненості системи в розпізнаних символах Зображення не можуть автоматично або вручну повертатись, якщо електронний документ був отриманий з явними зміщеннями тексту, тому в систему потрібно завантажувати попередньо випрямлені зображення.

Система базується на двигуні OCR під назвою Tesseract, який підтримує українську мову, тому результати вилучення даних досить високої якості.

В доступних в системі заздалегідь навчених моделях для україномовних документів дані автоматично не вилучаються, і модель не донавчається. Проте є можливість самостійно вилучити дані з документів, призначивши їх до завбачливо заданих ключів.

Так як попередньо навчені моделі не орієнтовані на україномовні документи, в списку полів можна спостерігати поняття, які не можна застосовувати до документів, які використовувались для дослідження системи, наприклад – податок штату, який є в моделях рахунків-фактур, але не існує в українських рахунках-фактурах. В той же час, в системі може спостерігатись і недолік полів, як наприклад в моделі що розпізнає ID-картки відсутні «по батькові ».

Приклад вилучення даних з документу, з використанням не навченої попередньо моделі:

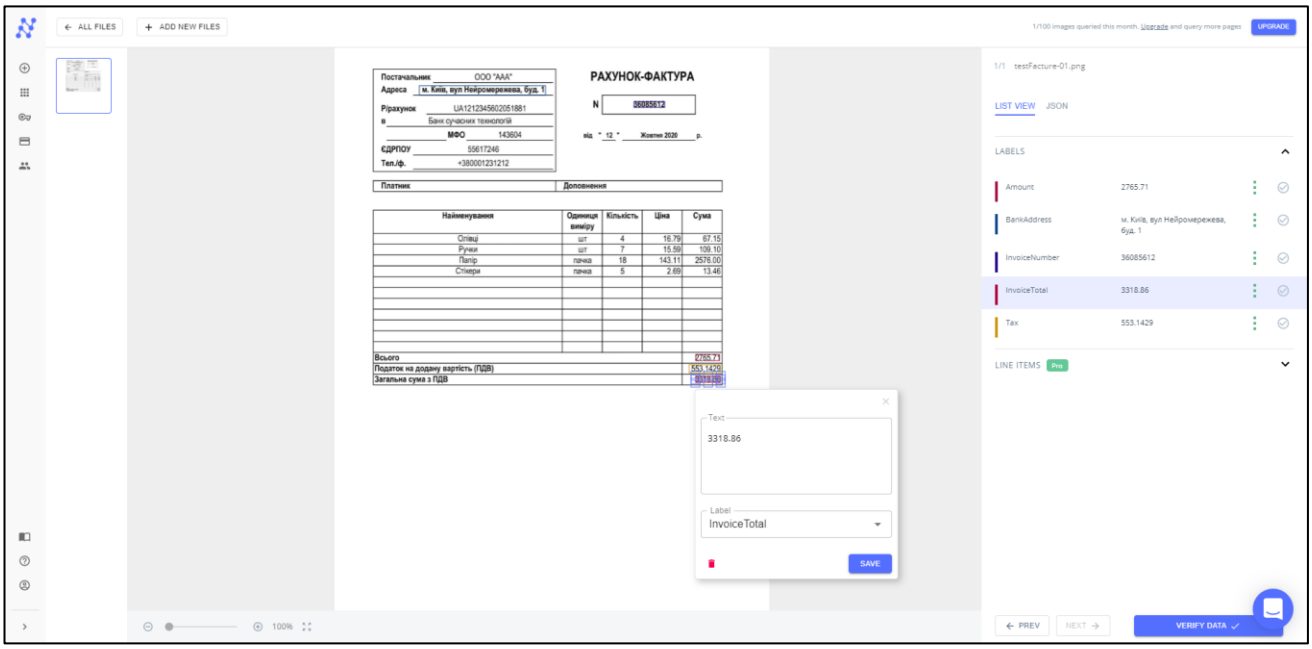


Рисунок 1.8 Вилучення даних з україномовних документів за допомогою Nanonets на прикладі рахунку-фактури

Для перевірки можливостей системи проведено навчання іншого типу документів, використовуючи при цьому зображення низької якості.

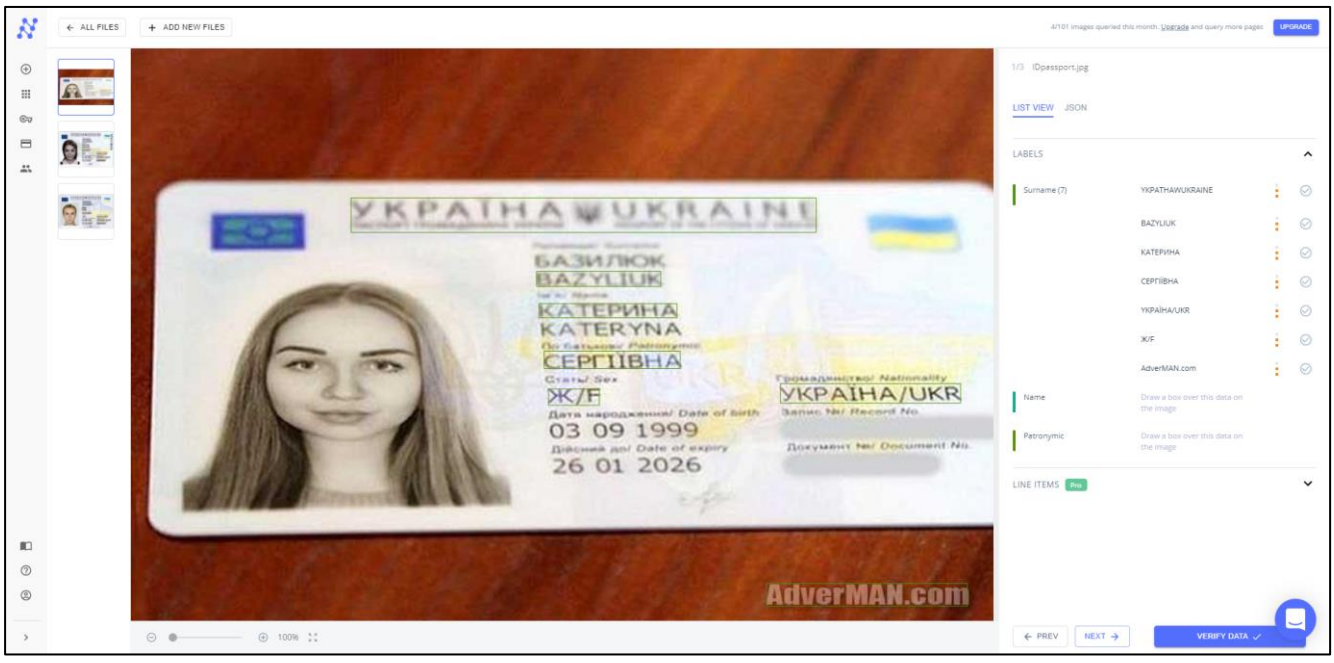


Рисунок 1.9 Вилучення даних з україномовних документів за допомогою Nanonets на прикладі української ID-картки

Як видно на рисунках 1.8 і 1.9, якщо на зображення немає викривлень, і документи високої якості – дані розпізнаються коректно. Якщо-ж на зображенні є нахили, то пари ключ-значення формуються некоректно, відсоток впевненості вилучених даних низький.

Висновки

В даному розділі були описані системи, що засновані на технологіях нейронних мереж та дозволяють автоматично вилучати дані з зображень отриманих за допомогою цифровізації або засобами електронної комунікації.

Проаналізувавши вищеописані системи, можна дійти до висновку що далеко не всі системи можуть самостійно вилучати дані, без попереднього навчання, а ті що можуть – не підтримують українську мову.

Результати дослідження систем можна сформувати у наступну таблицю:

Таблиця 1.1 – Порівняльна таблиця існуючих систем розпізнавання та вилучення даних з документів.

	Azure Form Recognizer	Google Documents AI	Amazon Textracts	Nanonets	Власна система
Підтримка української мови	Ні	Так	Ні	Так	Так
Використання хмарних обчислень	Так	Так	Так	Так	Ні
Можливість роботи без мережі інтернет	Ні	Частково	Ні	Ні	Так

Продовження таблиці 1.1.

Наявність модуля перевірки коректності вилучених даних	Ні	Ні	Так	Так	Так
Інтерфейс навчання нових типів документів	Так	Ні	Ні	Так	Так
Тип продукту	REST API	REST API	REST API	REST API та веб-інтерфейс з базовим функціоналом	Інформаційна система з веб-інтерфейсом

Виходячи з даних досліджень, важливим є створення прототипу інформаційної системи, котра зможе як в автоматичному, так і в ручному режимі вилучати текстові дані з зображень. Особливістю системи є підтримка вилучення даних з україномовних документів без необхідності постійного підключення до мережі інтернет

2 ВИМОГИ ТА ОЦІНКА ЯКОСТІ СИСТЕМИ

2.1 Обсяг та опис

Основною метою інформаційної системи є розпізнавання та автоматичне вилучення даних з україномовних документів з використанням нейронних мереж.

Додаток має складатись з сторінок обробки документів та бібліотеки зображень з вилученими даними.

Основною сторінкою для роботи з інформаційною системою має бути інтерфейс для розпізнавання зображень та опрацювання вилучених даних. Під опрацюванням мається на увазі перевірка на коректність вилучених символів та ручне вилучення невилучених даних.

Усі вилучені дані зберігаються у документоорієнтованій СУБД і мають можливість бути експортованими для подальшого використання в інших системах. Користувач системи має можливість опрацьовувати раніше опрацьовані та збережені зображення.

2.2 Загальний опис

Даний програмний продукт має на меті вирішення проблеми швидко вилучення даних з україномовних документів, які отримані шляхом оцифрування документів, або отримані через засоби електронної комунікації. Дана інформаційна система має бути розроблена як веб-додаток, для сумісності з будь-яким комп’ютерним обладнанням, яке буде використовуватись для взаємодії с системою.

Даний розділ містить специфікацію веб-додатку, де будуть описані основні функціональні та нефункціональні вимоги до системи.

2.2.1. Вимоги до безпеки

Інформаційна система з міркувань безпеки не має бути загальнодоступною, через що повинна існувати система автентифікації користувача.

Через те що веб-додаток заснований на клієнт-серверній архітектурі, всі точки доступи до серверу мають проводити валідацю користувача при виконання будь-якого запиту.

Валідація відбувається через відправку HTTP запитів під час взаємодії з веб-відображенням інформаційної системи.

2.2.2. Опис архітектури додатку

Інформаційна система має бути розроблена з максимальною модульністю – тобто, усі програмні частини повинні мати максимальну автономність.

Система має складатись з наступних модулів: автентифікатор, попередня обробка зображень, оптичне розпізнавання символів, вилучення даних, інтерфейс інформаційної системи, підключення до БД.

Автентифікатор. Автентифікація використовується для запобігання доступу несанкціонованого доступу до системи, так як сховище вилучених даних може містити конфіденційну інформацію. Функціонал модулі реалізований вбудованими засобами Django через HTTP запити.

Запити відбуваються на кожній веб-сторінці взаємодії з інформаційною системою. У випадку, якщо користувач не авторизований у системі, доступ до даних у нього відсутній. Такий підхід забезпечує достатню конфіденційність даних, і піддається масштабуванню правами та додатковими даними, що необхідні для користувача.

Попередня обробка зображень. Попередня обробка зображень відповідає за підготовку зображень перед вилученням тексту двигуном оптичного розпізнавання. Даний модуль має виконувати наступні дії:

- конвертація зображення у формат PNG;
- визначення кордонів зображення;
- визначення повороту зображення;
- кадрування і вирівнювання зображення;

- переведення зображення у відтінки сірого;
- бінаризація зображення;
- видалення шумів;

Для коректного вилучення даних з документів, однієї з стадій попередньої обробки зображення є визначення кордонів зображення та відділення його від можливого фону. Робиться це задля усунення можливості визначення двигуном оптичного розпізнавання символів фальшивих символів з фону зображення.

Відділивши зображення від фону, визначається правильна орієнтація зображенні. У випадку коли орієнтація попередньо не визначена, двигун оптичного розпізнавання символів витрачає більше часу на роботу, або повертає невірно визначені символи

Більшість двигунів оптичного розпізнавання символів дають найкращі результати при опрацюванні чорно-білих зображень. Для того щоб отримати чорно-біле зображення, проводиться переведення оригінального кольорового зображення у відтінки сірого з використанням схеми RGBtoGray.

Далі, отримане зображення у відтінках сірого проходить через процес адаптивної бінаризації зображення за методом Гаусса.

Видалення шумів прибирає зайві точки на зображенні, таким чином зменшуючи навантаження та пришвидшуючи роботу двигуна оптичного розпізнавання символів.

Оптичне розпізнавання символів. Оптичне розпізнавання застосовує OCR двигун під назвою Tesseract, який вилучає увесь наявний на попередньо обробленому зображенні текст.

Вилучення даних. Завданням модуля є вилучення даних з зображення за допомогою GCNs for VRDS для звичайних даних та CGAN в комбінації з генетичними алгоритмами для таблиць.

Послідовність використання модулів. Послідовність модулів, які приймають участь у інформаційній системі показана на рис. 2.1.

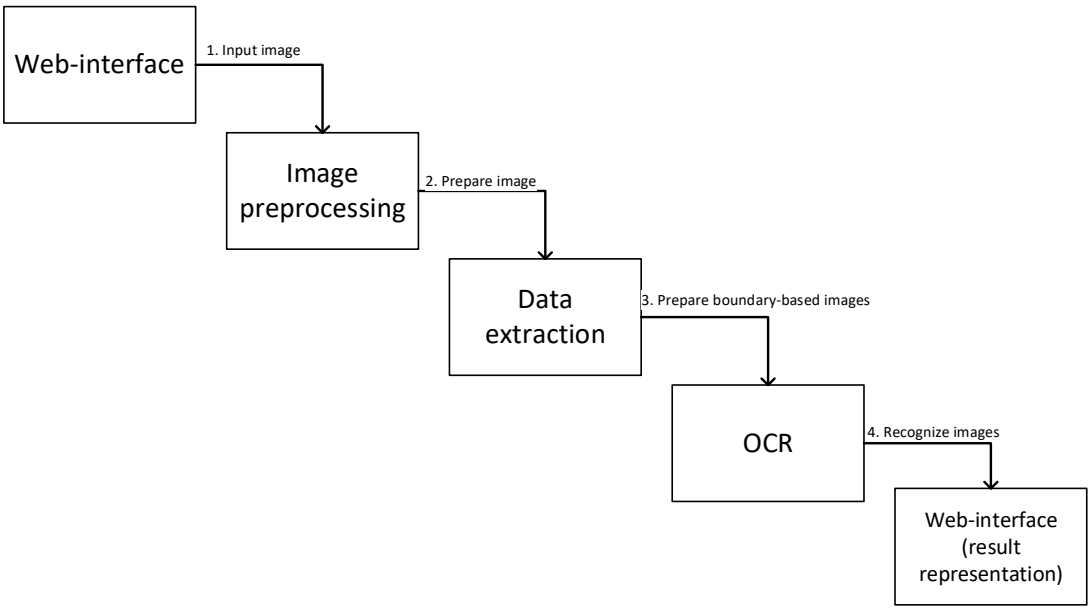


Рисунок 2.1 – Діаграма послідовності робочих модулів

Інтерфейс інформаційної системи. Під інтерфейсом мається на увазі веб-частина інформаційної системи, що призначена для взаємодії з користувачем. Інтерфейс використовується для завантаження зображення в інформаційну систему для подальшої обробки, відображення результатів опрацювання зображення та доступ до раніше опрацьованих документів. В інтерфейсі є прихована сторінка для адміністратора, задля роботи з користувачами.

2.2.3. Операційне середовище

Дана інформаційна система є веб-додатком Django, а отже ділиться на дві частини – клієнт і сервер.

Клієнтська версія системи вимагає підключення до мережі (локальної або мережі інтернет, в залежності від розміщення серверної частини) та використання будь-якого браузеру, який може підтримувати сучасні версії JavaScript та CSS. На даний час такими браузерами є, наприклад, Google Chrome, Mozilla Firefox або Microsoft Edge.

Серверна сторона має бути розміщена на PaaS («платформа як послуга»), або може використовуватись на локальному веб-середовищі. У випадку неможливості використання PaaS, функціонал буде працювати з обмеженнями. Локальний веб-сервер можна розміщувати на Windows-системах, будь-яких варіаціях Unix-систем та OS X.

2.2.4. Обмеження на розробку та реалізацію

Інформаційна система побудована як Web-додаток на основі платформи Django. Також використовується Mongoengine.

Версія клієнта використовує мову Java Script. Версія серверу використовує мову Python версії 3.9. Використовуються бази даних SQLite та MongoDB.

Взаємозв’язок між клієнтом та сервером відбувається через HTTP запити. Всі запити відбуваються за допомогою вбудованого механізму Django.

Платформа бази даних SQLite не вимагає запуску серверу, зв'язок з нею відбуваються через вбудований в Django коннектор. Платформа для бази даних MongoDB знаходиться на сервері MongoDB, зв’язок з котрим відбувається через Mongoengine.

2.3 Вимоги до зовнішнього інтерфейсу

Даний підрозділ містить опис всіх можливостей з'єднання з системою. В розділі надаються прототипи інтерфейсу користувача, опис апаратних і програмних інтерфейсів.

2.3.1. Інтерфейси користувача

Однією з головних умов використання системи є необхідність авторизації, для чого використовується сторінка логіну (рис. 2.2), де користувач вводить облікові дані.

Якщо у користувача відсутні логін та пароль від системи, або є необхідність у відновленні облікових даних, користувач надсилає запит до адміністратора системи з тієї-ж сторінки авторизації.

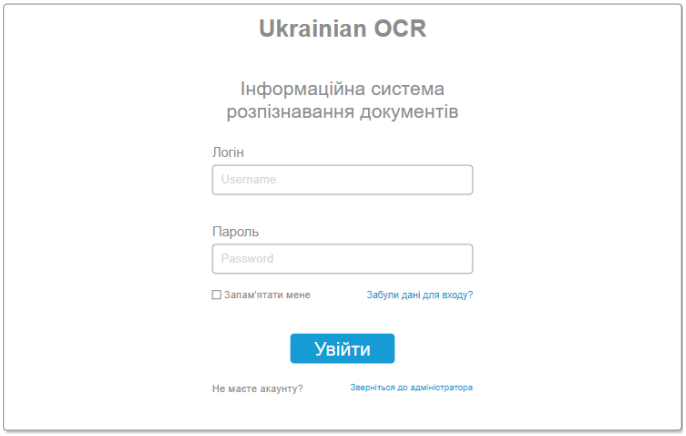


Рисунок 2.2 – Вікно авторизації

Після вводу облікових даних, за умови їх коректності, користувачу відкривається сторінка з колекціями опрацьованих документів, згрупованих відповідно до їх структури (рис. 2.3). На даний момент в системі є лише два типи документів, для яких навчені моделі. З системи можна переглянути усі зображення з колекції, або видалити всю колекцію.

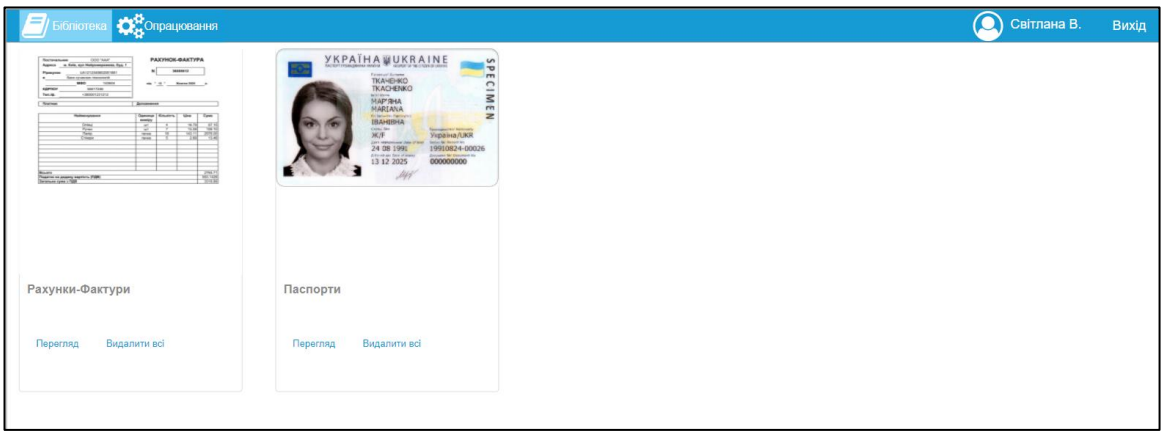


Рисунок 2.3 – Сторінка бібліотеки з колекціями опрацьованих документів

Під час перегляду змісту колекції, користувачу будуть відображені усі цифрові зображення, структура яких відповідає загальній структурі колекції документу. На цій сторінці користувач має можливість експортувати вилучені дані з опрацьованого документу, відредагувати опрацьований документ, або видалити його (рис. 2.4).

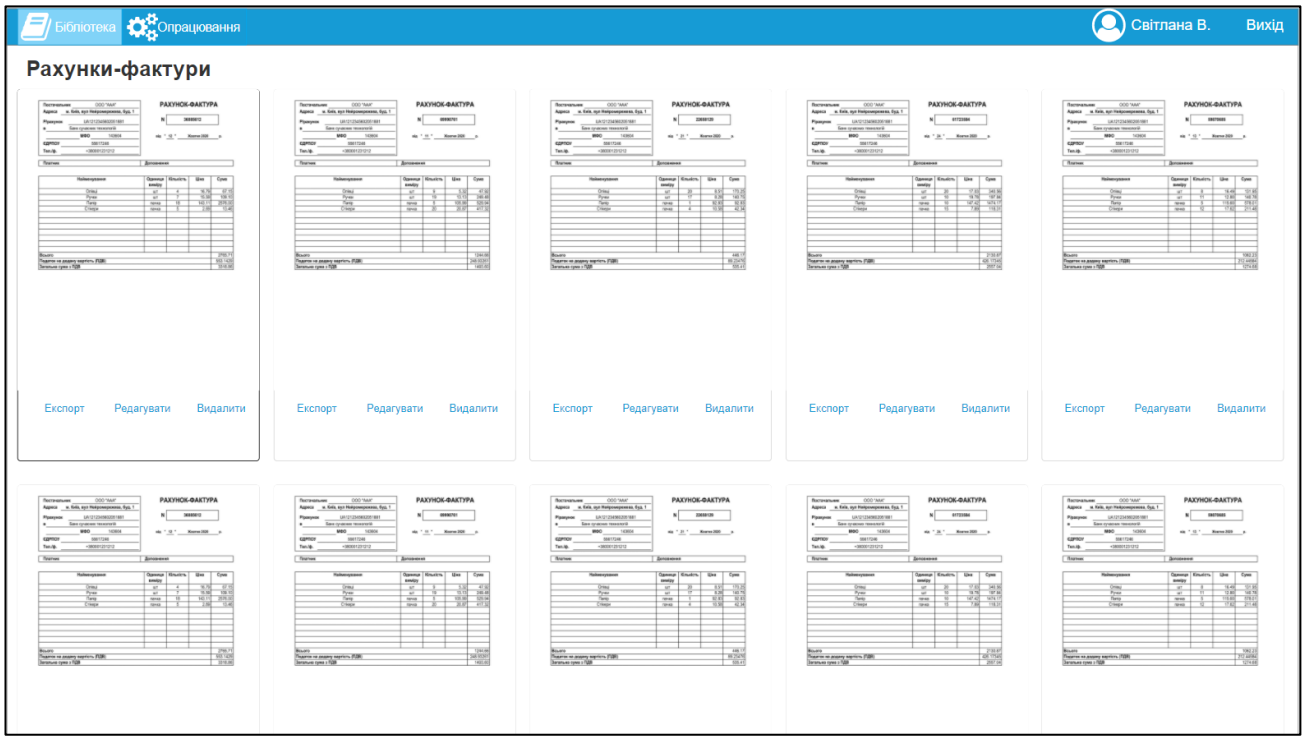


Рисунок 2.4 – Сторінка перегляду змісту колекції документів

Для розпізнавання нового документу, користувач через меню, що розташоване зверху кожної сторінки переходить до сторінок опрацювання документів. Перш за все, користувач має завантажити у систему зображення, яке необхідно опрацювати (рис. 2.5).

Рисунок 2.5 – Сторінка завантаження зображення

Після того як користувач завантажив у інформаційну систему зображення, система опрацьовує його, і по завершенню усіх внутрішніх процесів повертає

користувачу результат. На сторінці з вилученими даними (рис. 2.6) користувач має можливість внести зміни в вилучені поля (відредагувати, змінити ключ або видалити), додати власні поля, переобробити документ та визначити таблицю.

Якщо дані вилучись не з того місця, користувач має можливість виділити прямокутну частину зображення, де є необхідні дані. Після цього, дані автоматично будуть розпізнані і перенесені у вибране користувачем поле.

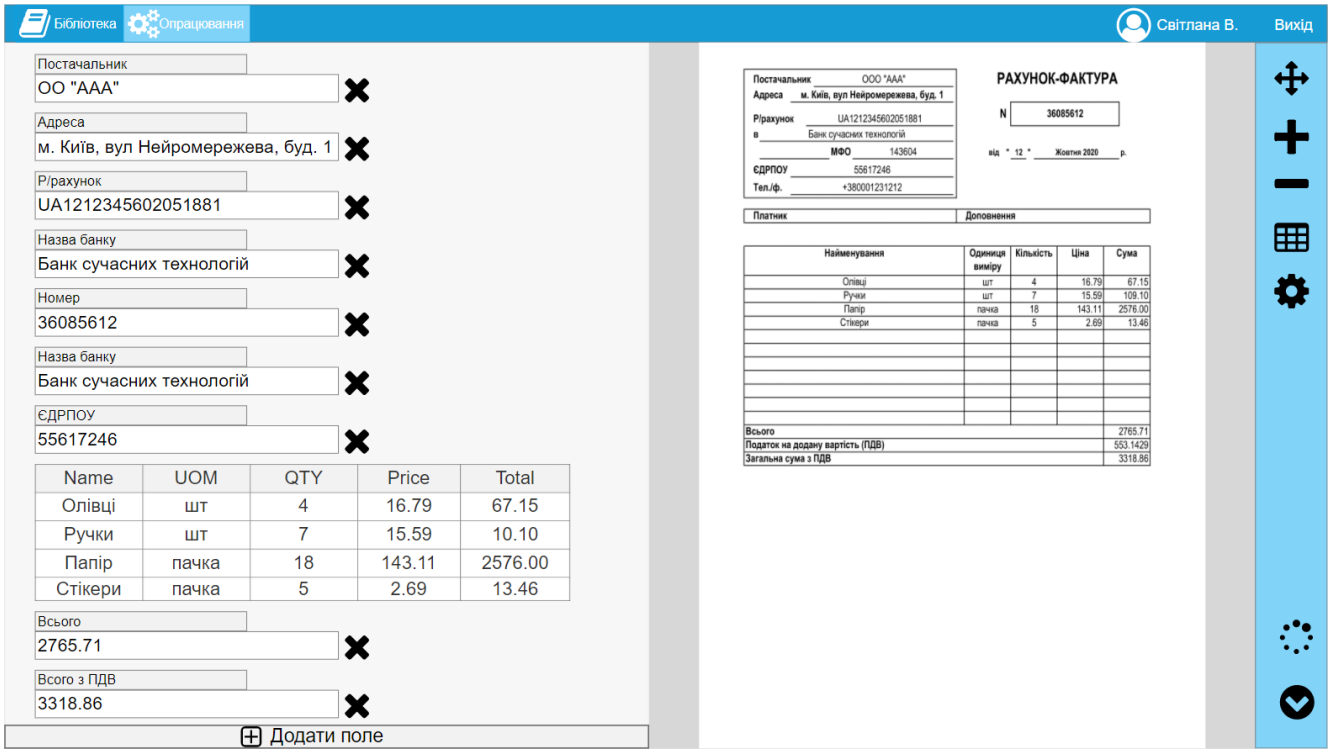


Рисунок 2.6 - Інтерфейс роботи з вилученими даними

Для полегшення коригування даних, в тому випадку коли місце їх вилучення є коректним, але самі символи вилучились неправильно, або є сумніви у правильності їх вилучення, на сторінці обробки зображення викликається вікно (рис. 2.7), що дозволяє провести перевірку вилучених символів.

Вікно дозволяє наскрізно проходити всі поля, де є символи з низьким рівнем впевненості у їх вилученні, та за необхідності дозволяє редагувати їх та підвереджувати правильно вилучені дані. У вікні підсвічуються ті символи, в яких низький поріг впевненості і правильності їх визначення.

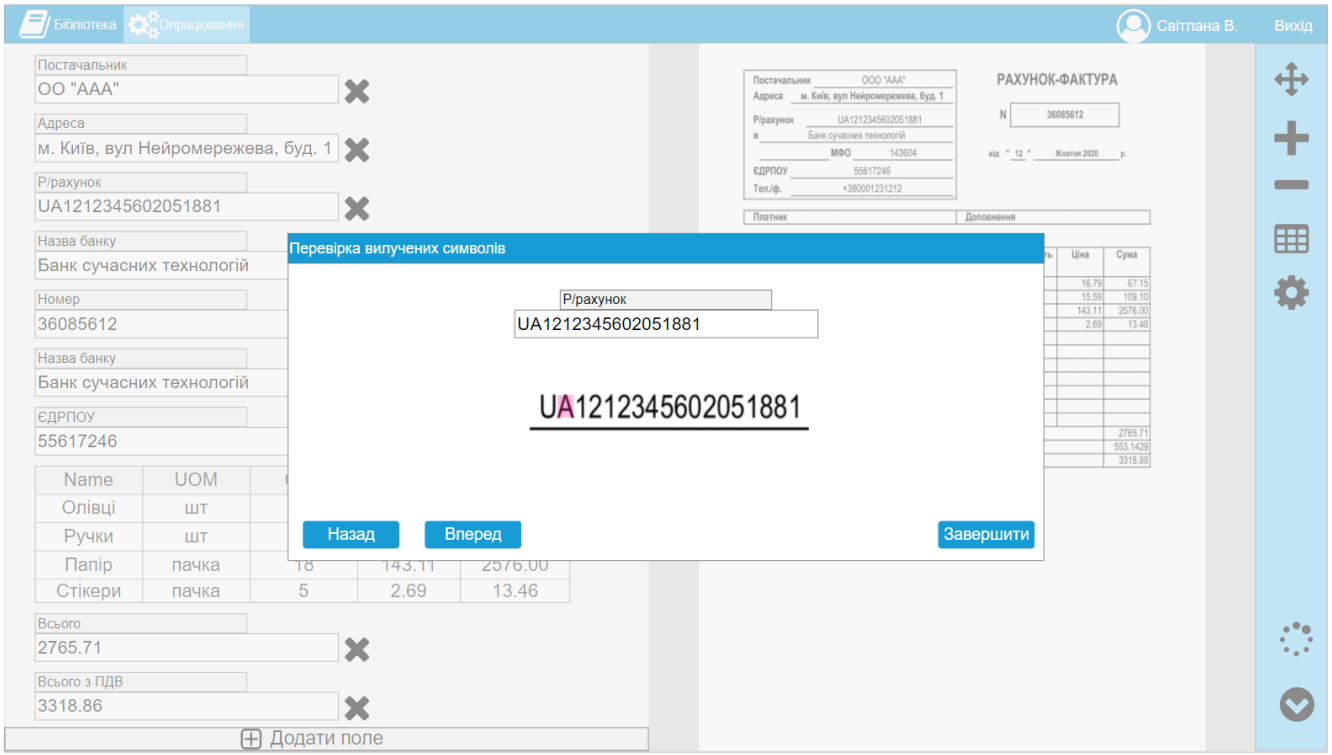


Рисунок 2.7 – Вікно перевірки вилучених символів

Можливість керувати користувачами, які матимуть доступ до системи, є лише у адміністратора системи, котрий через приховану сторінку (рис. 2.8) може керувати списком користувачів.

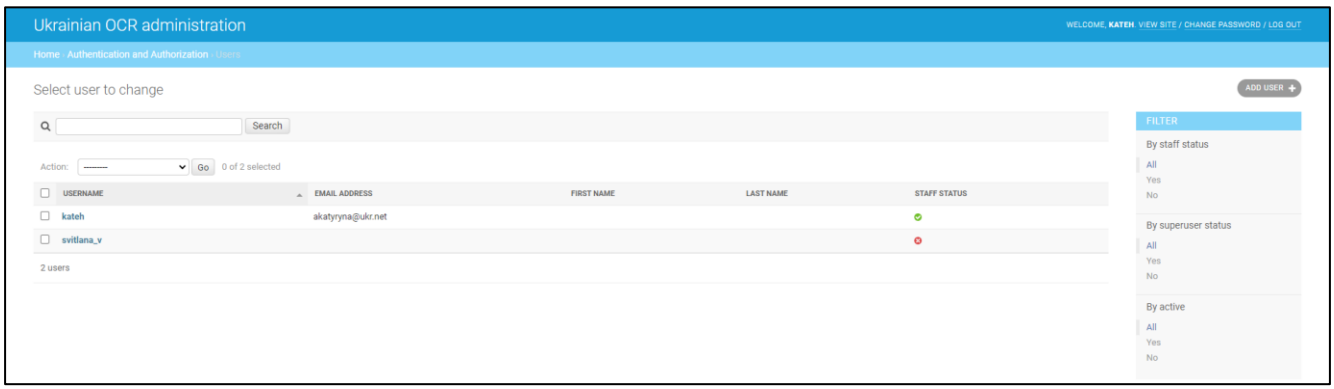


Рисунок 2.8 - Прихована сторінка управління користувачами

Обліковими даними користувача в першу чергу є логін та пароль, які вводить адміністратор системи при створенні користувача. Сторінка створення користувача показана на рисунку 2.9.

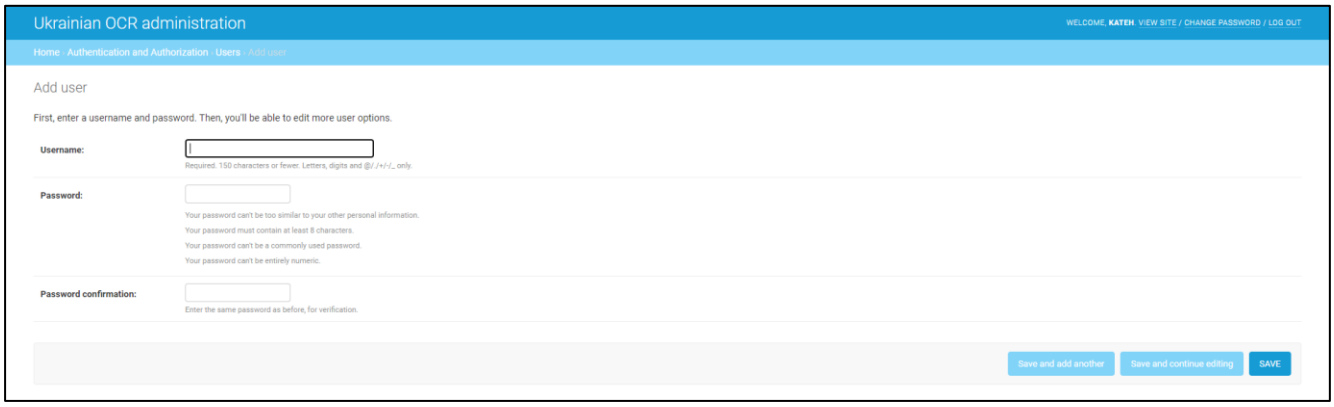


Рисунок 2.9 - Сторінка створення користувача системи

Після створення основних облікових даних користувача, адміністратор системи має можливість заповнити додаткові дані користувача, такі як прізвище, ім’я, електронну адресу та додати права адміністратора в систему. Сторінка введення додаткових даних показана на рисунку 2.10.

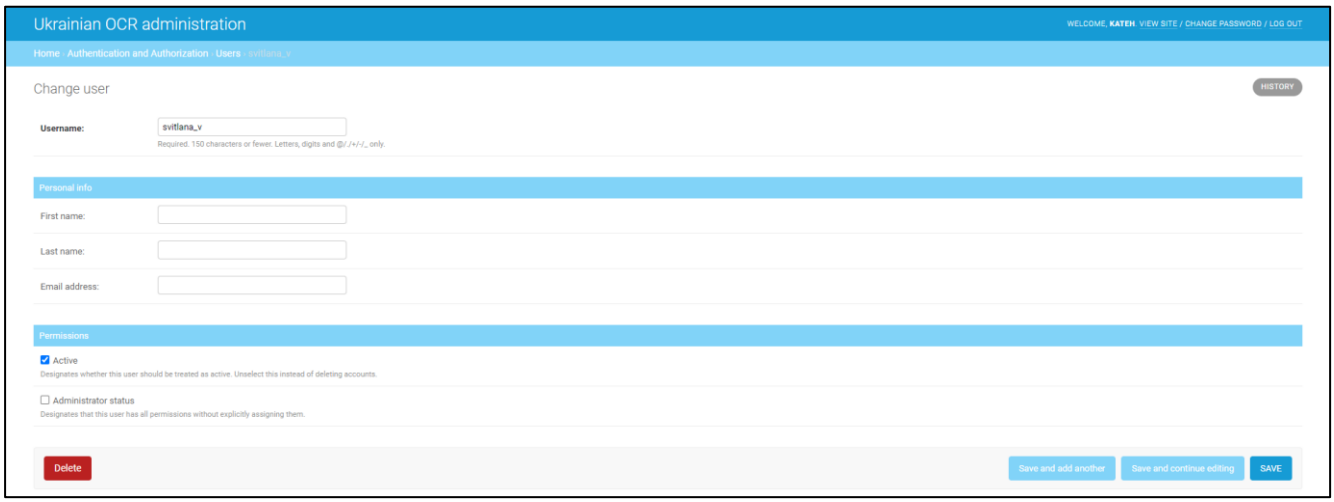


Рисунок 2.10 - Сторінка введення додаткових даних користувача

Таким чином, адміністратор має можливість керувати доступом до інформаційної системи, що є одним з методів збереження конфіденційної інформації.

2.3.2. Апаратні інтерфейси

Поточна інформаційна система не пов’язана з жодним апаратним інтерфейсом. Така система може бути розширена зв’язком з обладнанням, необхідним для оцифрування зображень.

2.3.3. Програмні інтерфейси

Клієнтська частина програми вимагає браузери з підтримкою мови JavaScript.

Серверна сторона повинна мати Django development server, або розташовуватись на серверах PaaS і мати MongoDB сервер для зберігання баз даних.

Також, потребує встановлення Python з версією 3.9.

2.4 Нефункціональні вимоги

Даний підрозділ містить опис нефункціональних вимог до інформаційної системи.

2.4.1. Вимоги до безпеки

Інформаційна система повинна мати обмеження у авторизації , тому таку систему може використовувати лише авторизований користувач.

Клієнтська сторона на вхід відсилає лише вхідне зображення, після чого отримує результат. Під результатом мається на увазі список полів та вилучених даних для цих полів. Усі операції щодо обробки зображень та вилучення інформації з зображень повинні виконуватись на серверній стороні, і не мають бути доступні клієнту.

Для безпеки авторизації, паролі повинні зберігатись в зашифрованому вигляді, з використанням алгоритму PBKDF2 та хешуванням SHA256, таким чином, ніхто не може отримати доступ до даних що є наявними у інформаційній системі.

2.4.2. Характеристики якості інформаційної системи

Інформаційна система має відповідати таким характеристикам:

- правильність. Система повинна правильно зберігати вилучені з зображення дані, так як вони можуть містити цінну для користувача інформацію.

Також, ці дані важливі, через те що використовуються для донавчання нейронних мереж.

• настроюваність/розширюваність Додаток повинен мати таку архітектуру, щоб його можна було розширити для подальших розробок, як на стороні клієнта, наприклад пошук документів по певному значення, або пошук дублікатів зображення, що зможе зекономити час користувачу, так і на стороні обслуговування, такі як поділ модулів, можливість зміни типу нейронної мережі на іншу, або додавання нових нейронних мереж для підвищення якості вилучення даних. Також слід розробити високі абстрактні принципи зберігання даних, щоб розширити їх за допомогою більшої кількості даних.

• простота / автоматизм: Клієнтська програма повинна спростити процес обробки зображення та вилучення з них даних, мінімізувати необхідність внесення даних оператором у ручному режимі, а при невідворотності таких – не перебериват процес таким явищем як оновлення сторінки.

Даний розділ містить опис принципів та процесів оцінки якості додатку. Будуть розглянуті, як підходи до ручного тестування так і пропозиції до автоматизації описаного процесу.

2.5 Тестування та оцінка якості програмного продукту

Великий наголос на постачанні якісного програмного забезпечення зробив тестування програмного забезпечення невід’ємною процедурою розробки програмного забезпечення.

У наш час, хоча роль тестування на перший погляд може здатися не такою значною, що особливо стосується людей, які не знайомі з життєвим циклом програмного забезпечення, процес тестування програмного забезпечення є невід'ємною частиною розробки.

Існує багато різновидів тестування, але загалом їх можна виділи в 2 групи – ручне та автоматизоване.

2.5.1. Ручне тестування

Методи випробувань у ручному тестування можна класифікувати як «чорний ящик», «білий ящик» або як ті, що засновані на досвіді [8].

Методи тестування на чорному ящику (їх також називають поведінковими методами) засновані на аналізі відповідної тестової основи (наприклад, офіційні документи вимог, технічні умови, випадки використання, історії користувачів або бізнес-процеси). Ці методи застосовуються як для функціонального, так і для нефункціонального тестування. Методи тестування в чорній скриньці концентруються на входах і виходах тестового об'єкта без посилання на його внутрішню структуру.

Тестові методи білої скриньки (також називаються структурними методами) засновані на аналізі архітектури, детального проекту, внутрішньої структури або коду тестового об'єкта. На відміну від методів тестування чорного ящику, методи тестування білого ящику концентруються на структурі та обробці всередині тестового об'єкта.

Тестові методи, що базуються на досвіді, використовують досвід розробників, тестувальників та користувачів для розробки, впровадження та виконання тестів. Ці методи часто поєднують з методами тестування на чорного ящику та білого ящику.

Однією з важливих частин методик досвіду є тестування на основі контрольних списків.

Під час тестування на основі контрольного списку тестувальники розробляють, впроваджують та виконують тести, щоб охопити умови тестування, знайдені в контрольному списку. В рамках аналізу тестувальники створюють новий контрольний список або розширюють існуючий контрольний список, але тестувальники можуть також використовувати існуючий контрольний список без змін. Такі контрольні списки можна складати на основі досвіду, знань про те, що важливо для користувача, або розуміння того, чому і як програмне забезпечення виходить з ладу.

2.5.2. Автоматизоване тестування

Автоматизоване тестування програмного забезпечення – вид тестування, що застосовує програмні засоби для виконання тестів та перевірки результатів виконання. Даний вид допомагає скоротити час тестування, спростити сам процес та зменшити кількість помилок при тестуванні у зв'язку з відсутністю людського фактору [10].

Існують такі види автоматизацій, як:

- GUI-автоматизація;
- модульне тестування.

GUI-автоматизація – вигляд автоматизації, коли тестується інтерфейс користувача.

Відбувається за рахунок таких технік:

• запис/відтворення – під час ручного тестування записуються дії тестувальника, а потім інструмент просто їх відтворює. Дана техніка дозволяє виконувати тести без прямої участі людину у процесі. Такі тести можна виконувати протягом тривалого часу. Це збільшує продуктивність та усуває необхідність повторення одноманітних ручних дій тестувальника.

Такий підхід має недолік, через те що кожна зміна інтерфейсу вимагає перезапису дії, тому негативно відображається на ефективності процесу;

• сценарії. Дана техніка прийшла на зміну вищеописаної. Суть її полягає в мовах програмування, що були спеціально розроблені для опису сценаріїв. Розробкою даного типу мають займатися програмісти, які працюють окремо від тестувальників. Такі програмісти ж і запускають автоматизовані тести;

• data-driven testing – методологія, особливістю якої є виконання скриптів на основі даних, що зберігаються в сховищі;

• keyword-based testing – методологія, особливістю якої є поділ процесу на 2 етапи: планування та реалізацію;

• модульне тестування (unit testing) – метод тестування програмного забезпечення, що полягає в окремому тестуванні кожної функції та методу програми. Найчастіше пишеться та виконується програмістами, як перед додаванням нового функціоналу ПЗ, так і під час інтегрування його у вже існуючий.

Для автоматизації тестування існує професія, що має назву Інженер-автоматизатор забезпечення якості (QA Automation Engineer). Його обов'язки полягають у створенні автоматичних скриптів, які будуть перевіряти роботу програми.

Визначити рентабельність введення автоматизованого тестування можна за рахунок методу ROI [11] – фінансового коефіцієнту, що ілюструє рівень витрат на ручне тестування програмного продукту щодо прибутковості та довго тривалості проекту.

Умовно кажучи, якщо проект має існувати більше аніж 6 місяців, то доцільно використовувати саме автоматизацію, в іншому випадку потрібно використовувати ручне тестування.

GUI-автоматизація. Оскільки описаний додаток є веб-додатком, написаним мовою Python, з використанням веб-фреймворку Django найкращим варіантом використання є проект Selenium, що має серію програмних бібліотек для автоматизації тестування під назвою Selenium Web Driver, метою якої є роботи з браузерами.

В рамках даного драйвера розробляються драйвери для таких браузерів, як Chrome, Internet Explorer, Safari, Opera та ін. Web Driver також розробляться для таких мов, як Java, .Net, Ruby, C#, JavaScript, PHP, Perl та Haskell [12].

Модульне тестування. Для тестування програмної частини коду, написаної мовою Python використовується бібліотека, що має назву PyTest.

PyTest - це заснована на Python середу тестування, яка використовується для написання і виконання тестових кодів. В даний час служби REST PyTest в основному використовуються для тестування API, хоча є можливість використання

PyTest для написання простих і складних тестів, тобто є можливість писати код для тестування API, бази даних, призначеного для користувача інтерфейсу і т. д.

Переваги PyTest полягають в наступному[13]:

- PyTest може виконувати кілька тестів паралельно, що скорочує час виконання набору тестів;
- У PyTest є власний спосіб автоматичного визначення тестового файлу і тестових функцій, якщо це не вказано явно;
- PyTest дозволяє нам припустити підмножина тестів під час виконання;
- PyTest дозволяє нам запускати підмножина всього набору тестів;
- PyTest є безкоштовним і відкритим вихідним кодом;
- Завдяки простому синтаксису, PyTest дуже простий для запуску.

PyTest не має власного інтерфейсу, тому може використовуватись з будь-якої IDE, яка може виконувати файли з кодом на Python.

Висновки

В даному розділі були описані вимоги до системи.

Були представлені внутрішні та зовнішні технічні вимоги. Опираючись на описані дані можна уявляти побудову, як архітектуру користувацького інтерфейсу, так і логічних шарів, та шарів роботи з даними.

Виходячи з таких вимог, інформаційна повинна бути організована як клієнт-серверний веб-додаток з взаємопов'язаною базою даних. Всі зовнішні модулі та бібліотеки також були визначені.

Математична модель нейронних мереж має також обчислювати результат, використовуючи лише свої знання, що були отримані під час навчання, і на основі цього опрацьовувати нові документи.

3 ОПИС ПРОЕКТУВАННЯ СИСТЕМИ

3.1 Методи вилучення даних

Враховуючу мету даної інформаційної системи, має бути обраний спосіб вилучення даних з цифрового зображення, не потребуючи попередньої обробки зі сторони користувача, але залишаючи йому можливість внести корективи у випадку невірного вилучення даних.

Традиційним підходом до вилучення чітко-структурованих форм є створення певного шаблону або комплексу правил, які застосовуються до вхідного зображення. Вилучення даних за шаблонним підходом полягає в необхідності оператором або програмістом попередньо самостійно розбивати цифрове зображення на певні секції, в яких можуть вилучитися необхідні дані. Такі налаштування можуть працювати швидко і з високою якістю вилучення даних, допоки в структурі документа не виникнуть зміни.

Рисунок 3.1 – Приклади рахунків-фактур

На рис. 3.1 зображені фрагменти рахунків-фактур, у яких виділені дати виставлення рахунків. З зображень видно, що при вилученні дат з використанням шаблонів і правил можуть виникнути проблеми, так як дати мають різний формат відображення, місцезнаходження відносно інших елементів. Ще однією проблемо

при пошуку дати виставлення рахунку є те, що в документі можуть бути ще сутності з датами, але які не є саме датою виставлення рахунку.

Інший підхід до вилучення даних – це використання попередньо-навчених лінгвістичних моделей, таких як Word2Vec, Bert, Fastex та Glove, що побудовані на основі найпростіших нейронних мереж. Для використання таких моделей, необхідно попередньо вилучити увесь наявний на цифровому зображенні текст, і передати на вхід до нейромережі.

Схожий підхід – це використання рекурентних нейронних мереж, наприклад LSTM. Як і у випадку з попередньо-навченими лінгвістичними моделями, потрібно попередньо провести оптичне розпізнавання символів з документу.

Для вилучення таблиць з документів часто використовується метод знизу-вгору. Спочатку визначаються кордони на рівні пікселів для комірок таблиці, орієнтуючись на обмежувальні лінії та відступи. Після цього, кордони поєднуються в цілісну таблицю. Такий підхід ігнорує інформацію щодо структури таблиці вищого рівня.

Інший варіант визначення таблиці – використання ієрархічної агломеративної кластеризації для групування інформації на рівні макету слів, що використовується згодом для створення таблиці з набору слів, які можуть належати одному стовпчику таблиці. Це підхід «знизу», тобто графічний макет таблиці ставиться у вищий пріоритет.

Коли приймається на віру те, що таблиця в документі є лише довільним графічним поданням невидимою структури даних – таких підхід називається «зверху вниз». При такому підході, логічна та графічна форми повністю незалежні одне від одного. Підхід починається з припущення, що існує певна логічна структура даних, яку необхідно відновити, при цьому, пошук правильної логічної структури таблиці відокремлюється від графічної структури таблиці, але остання використовується для пошуку даних.

3.2 Вибір конфігурації нейронної мережі для вилучення текстових даних

З усіх можливих варіантів було вибране використання згортки графів для мультимодального вилучення даних для візуально-насичених документів (GCNs for VRDs).

Будь-який документ зі своєю структурує є візуально насиченим. Стандартні підходи, такі як LSM, BiLSTM-CRF використовують лише текстові послідовності, що не розкриває всіх можливостей роботи з чітко-структурованими документами.

Семантична структура документу визначається не лише тестовими даними, але і візуальними функціями, такими як таблична структура, макет, розмір шрифту, тощо.

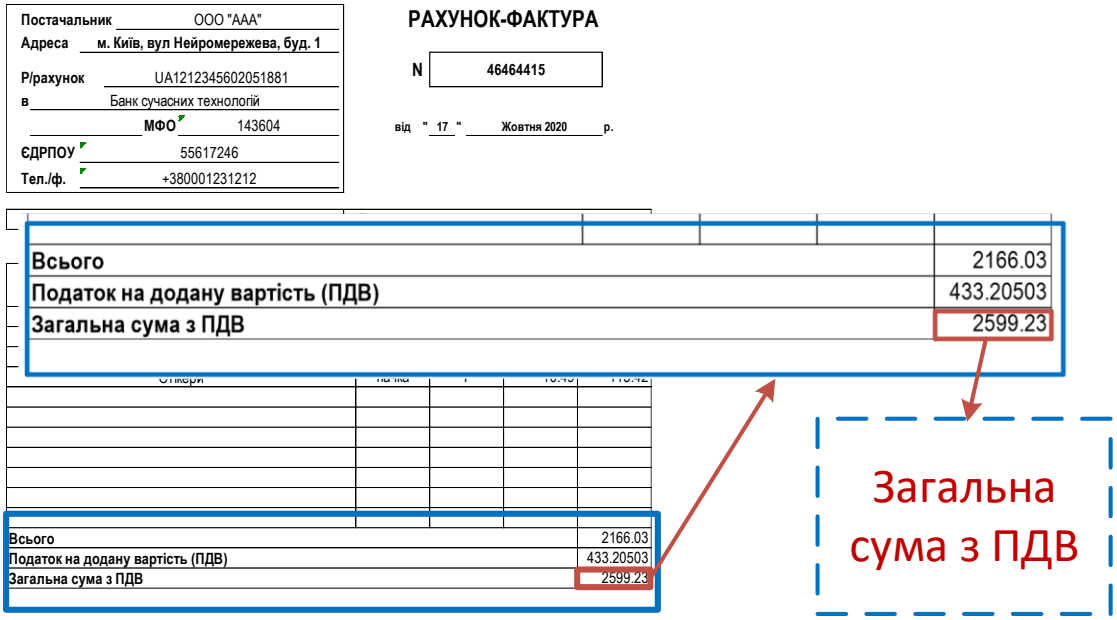


Рисунок 3.2 – Приклад візуально-насиченого документу з чіткою структурою, та приклад сутності для вилучення

Документи зі структурою можна репрезентувати як граф текстових сегментів, де кожен текстовий сегмент складається з його місцезнаходження і тексту всередині нього. Положення тексту визначається чотирма координатами, що створюють обмежуючу рамку тексту.

Для того, щоб вилучити дані з документу, спочатку обчислюються вкладення графу для кожного текстового сегменту в документі використовуючи згортання графів. Вкладення графу – це зміст поточного текстового сегменту, в якому операція згортання поєднує текстові і візуальні особливості вмісту зображення. Після цього, вкладення графу комбінуються з текстовим вкладення, після чого відправляється у LSTM для подальшого вилучення інформації. В даній інформаційній системі, роль LSTM відіграє OCR-двигун під назвою Tesseract.

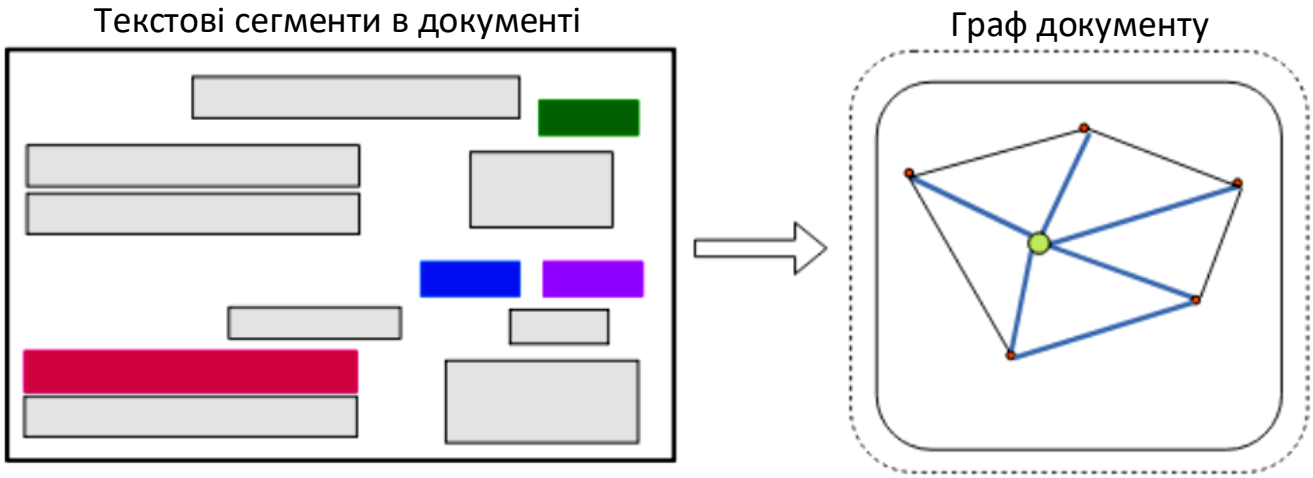


Рисунок 3.3 – Перетворення зображення на граф документу

Математично, документ D це кортеж (T,E), де $T = \{t_1, t_2, \cdots, t_n\}$, $t_i \in T$ – це множина з n текстових полів, $R = \{r_{i1}, r_{i2}, \cdots, r_{in}\}$, $r_{ij} \in R$ – множина з ребер, а $E = T \times R \times T$ – це множина направлених ребер виду (t_i, r_{ij}, t_j), де $t_i, t_j \in T$ та $r_{ij} \in R$.

3.2.1. Вилучення функцій

Для вузла t_i обчислюється вкладення вузла $\mathbf{t}_i$ використовуючи одношарову Bi-LSTM для вилучення з сегменту функцій текстового змісту. Вкладення ребер між вузлами t_i, і t_j визначається наступним чином:

$$r_{ij} = \left[x_{ij}, y_{ij}, \frac{w_i}{h_i}, \frac{h_j}{h_i}, \frac{w_j}{h_i}\right], \tag{3.1}$$

де x_{ij} та y_{ij} – горизонтальні та вертикальні відстані між двома текстовими полями відповідно, а w_i та h_i це ширина та висота відповідного текстового поля. Третє, четверте, і п'яте значення вкладення - це відношення сторін вузла t_i, відносна висота та ширина вузла t_i відповідно. Вбудування вузлів кодує текстові елементи, тоді як вбудовування країв насамперед представляє візуальні особливості.

3.2.2. Згортання графу

Графічна згортка застосовується для обчислення візуального вбудовування текстових сегментів у графік. Визначається згортання на триплетах вузлових осередків (t_i, r_{ij}, t_j), а не для поодиноких вузлів. Для вузла t_i витягуються функції $\boldsymbol{h}_{ij}$ для кожного сусіда t_j, використовуючи багатошарову мережу перцептронів (MLP).

$$h_{ij} = g\left(\boldsymbol{t}_i, \boldsymbol{r}_{ij}, \boldsymbol{t}_j\right) = \mathrm{MLP}([\boldsymbol{t}_i || \boldsymbol{r}_{ij} || \boldsymbol{t}_j]), \tag{3.2}$$

де || - операція конкатенації.

Вихідне вбудовування $\boldsymbol{t}'_i$ шару для вузла t_i обчислюється за наступною формулою:

$$t'_i = \sigma\left(\sum_{j\in\{1,\ldots,n\}} \alpha_{ij}\mathbf{h}_{ij}\right), \tag{3.3}$$

де α_{ij} – це attention-коефіцієнт, та σ – це функція активації.

Механізм attention виглядає так:

$$a_{ij} = \frac{\exp(\mathrm{LeakyReLU}(w_a^T \mathrm{h}_{ij}))}{\sum_{j\in\{1,\ldots,n\}} \exp(\mathrm{LeakyReLU}(w_a^T \mathrm{h}_{ij}))}, \tag{3.4}$$

де w_a – це вектор ваги спільної уваги. Використовується функція активації LeakyReLU, щоб збільшити «контраст» коефіцієнтів уваги.

Вихід для вбудовування краю шару згортки графіків визначається як:

$$\boldsymbol{r}'_{ij} = \mathrm{MLP}(h_{ij}). \tag{3.5}$$

Виходи $\boldsymbol{t}'_i$, $\boldsymbol{r}'_{ij}$ подаються як входи на наступний рівень згортки графіків (як обчислюється у рівнянні 3.2) або мережеві модулі для подальших завдань.

Графічно, вся процедура зображена рисунку 3.4:

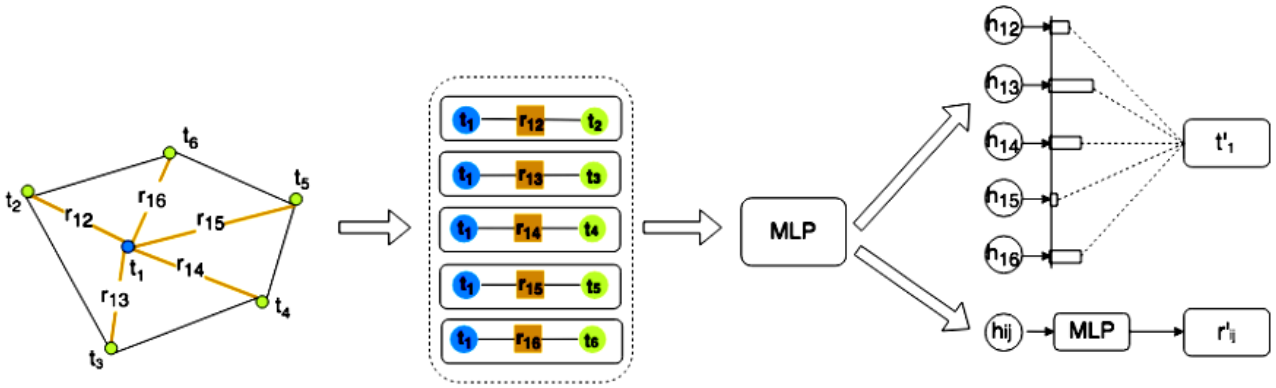


Рисунок 3.4 – Згортання графу документа

Результати роботи нейромережі передаються до двигуна розпізнавання Tesseract, для подальшого розпізнавання символів.

3.3 Вибір конфігурації нейронної мережі для вилучення табличних даних

Для вилучення таблиць був обрани підхід «зверху вниз». Як уже згадувалось раніше, в документах є важливим не лише його логічна складова, а і графічна структура. Підхід «зверху вниз» дозволяє орієнтувати на обидві частини цього компоненту.

Такий підхід можна організувати у два етапи:

1. Переведення вхідного зображення, що містить таблицю, у абстракту стандартну форму «скелету», що відображає контури пікселів для визначення приблизного розташування комірок і меж таблиці, але не враховуючи фактичний вміст таблиці;

2. Оптимізується відповідність прихованих структур даних кандидату до сформованого зображення скелету, використовуючи мір відстані між кожним кандидатом і скелетом.

Графічно наступний процес зображений на рисунку 3.5:

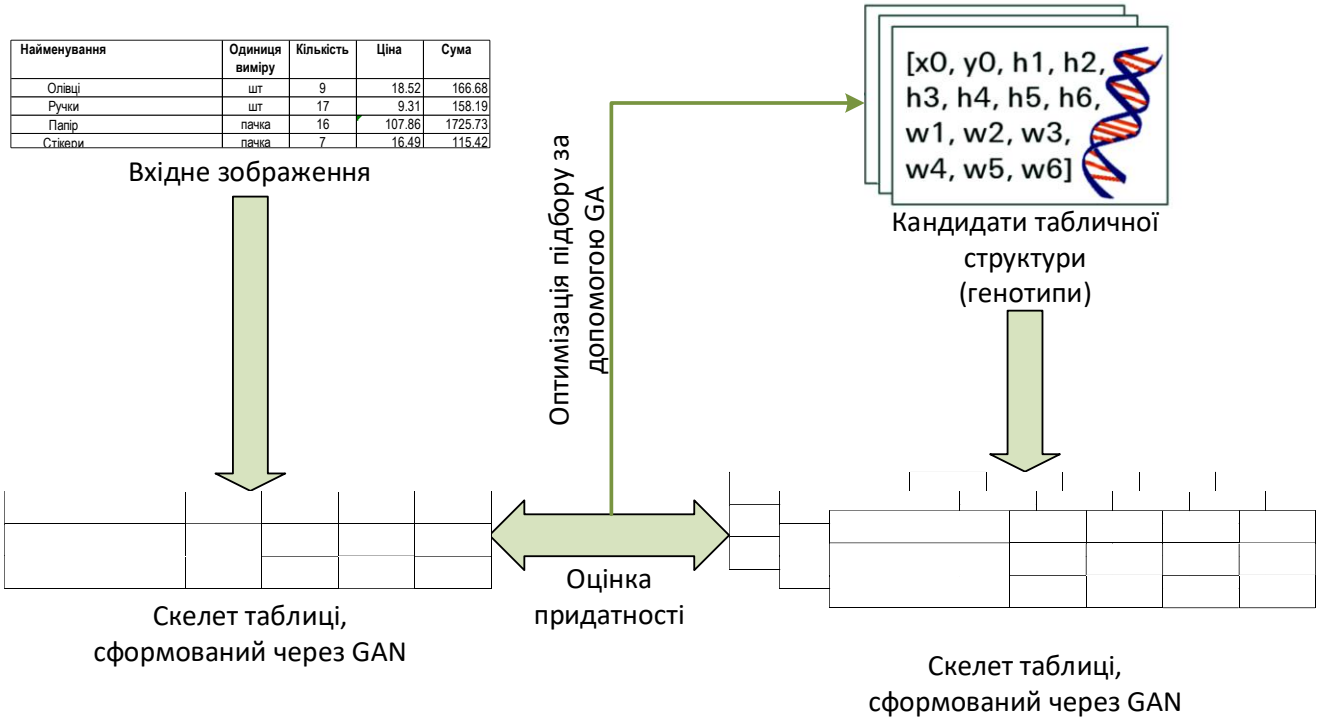


Рисунок 3.5 - Загальна схема обробки таблиці

Після того як визначиться найкраща відповідність, дані вилучаються з зображення таблиці та зберігаються у виявленій структурі, після чого передаються на опрацювання OCR-двигуном, в нашому випадку – Tesseract.

3.3.1. cGAN

Умовно-генеративна змагальна мережа (cGAN) використовує дві нейромережі : *генератор*, що вчиться генерувати реалістичні «фальш-кандидати», для того щоб ввести в оману *дискримінатор*, що вчиться виявляти штучно-сформовані входи з набору реальних прикладів ведення.

В першому кроці гаданого вище підходу для вилучення таблиці, cGAN навчається перекладати зображення у стандартизовані «скелети», що позначають межі комірок таблиці

Архітектура cGAN. Для спрощення моделі cGAN та оптимізації часу обробки зображення, виконується зменшення роздільної здатності зображення до розміру 256x256.

cGAN апроксимує відображення від вхідного зображення *x* та вектора випадкового шуму *z* до вихідного зображення *y*, так що $y = G(x, z)$. Генератор *G* навчений виробляти вихідні дані, які неможливо відрізнити від цільових вихідних зображень за допомогою дискримінатора *D*, який навчений для виявлення "фальшивих" зображень.

Генератор *G* – це мережа кодера-декодера, так що вхідне зображення передається через ряд шарів, що поступово знижують дискретизацію, до вузького місця, де процес змінюється. Для передачі достатньої інформації шарам декодування використовується архітектура U-Net (один з різновидів згорткової нейронної мережі) із пропусками з'єднань з трансляцією зображення-до-зображення через умовно-змагальну мережу, а пропуск з'єднання додається між кожним шаром *i* та *n* , де *n* – загальна кількість шарів, а *i* – номер шару в кодувальнику. Далі, слідом за умовно-змагальною мережею, випадковий шум *z* подається лише у формі відсіву, що застосовується на декількох шарах генератора як під час навчання, так і під час випробувань.

Дискримінатор *D* робить два зображення – вхідне зображення (цільове), або вихідне зображення з генератора, після чого визначає ймовірність того, що друге зображення генерується генератором чи ні (тобто зображення є реальним). Для дискримінатора *D* у cGAN використовується штрафна архітектура PatchGAN, яка накладає штраф на структуру вихідного зображення в масштабі частини зображення. Зокрема, дискримінатор класифікує, чи є кожна частина зображення *N* x *N* на зображенні у справжнім чи фальшивим (значення N = 70), так що $D(x, y)$ дорівнює *M* x *M* матриці, що містить ймовірності того, що частини зображення в *y* були реальними (значення *M* = 30), тобто представник цільового зображення із заданим вхідним зображенням *x*. Такий дискримінатор трактує зображення як випадкове поле Маркова, припускаючи незалежність між пікселями, розділеними більше, ніж діаметр частина зображення, і розуміється як форма втрати текстури.

Оптимізація та формування висновків у cGAN. Параметри cGAN вибираються шляхом оптимізації наступних об’єктивних функцій щодо генератора *G* та дискримінатора *D*:

$$\min_G \left[\max_D L_{cGAN}(G, D) + \lambda L_{L1}(G)\right], \quad (3.6)$$

де L_{L1}- це L1 – відстань між зображенням, що генерується генератором *G* та вихідним зображенням, усередненим по навчальному набору, λ - постійна вага (λ = 100), а L_{cGAN} виражає ціль cGAN для створення зображень, які не можуть бути дискримінованими з реальних зображень:

$$L_{cGAN}(G, D) = E_{x,y}[\log D\,(x, y)] + E_{x,y}[\log(1 - D\,(x, G(x, z)))]. \quad (3.7)$$

Генератор G використовується з відсівом та "нормалізацією випадків" під час генерації висновків.

3.3.2. Генетичний алгоритм

Генетичні алгоритми (GA) використовують популяційний підхід для вибірки простору пошуку можливих рішень і для підняття градієнта до оптимуму. Основне представлення рішення-кандидата (розміри та зміщення рядків і стовпців таблиці) кодується як вектор чисел, позначений як генотип , який декодується у фенотип (візуалізоване зображення таблиці). Потім особи у популяції оцінюються за допомогою функції придатності *f*, що діє на фенотиповий показник. У нашому випадку f забезпечується компонентом дискримінації cGAN, який повертає відстань між відображеним індивідуальним зображенням та цільовим зображенням. Парам осіб з вищою фізичною формою (менші відстані до цільового зображення) імовірнісно підбираються, їх генотипи рекомбінуються, щоб створити нові рішення для нащадків, що містять корисні особливості обох батьків - наприклад, тих що поєднують кількість стовпців від одного з батьків із кількістю рядків від другого батька.

Оператор генетичної мутації додатково підтримує різноманітність у популяції шляхом випадкового зміни значень на швидкість μ для пошуку сусіднього простору розчину. Таким чином, селективний тиск діє на рівні фенотипу

(порівняння між наданими розчинами-кандидатами та цільовим скелетом, сформованим cGAN), щоб оптимізувати представлення основного генотипу.

Похідний скелет таблиці параметризує цільову функцію, яка визначає придатність структури таблиці, щоб забезпечити її оптимізацію за допомогою генетичного алгоритму.

Структура таблиці описується як набір наступних чисел, визначених як генотип таблиці:

1) таблиця потужності кількість рядків n і кількість стовпців m;
2) координати лівого верхнього кута таблиці $\{ x_0 , y_0\}$;
3) вектор висот рядків $\{h_1, \dots , h_n\}$, $h_i \geq 0$;
4) вектор ширини стовпців $\{w_1, \dots , w_m \}$, $w_i \geq 0$.

Кількість рядків n та кількість стовпчиків m задані апріорно і виражають очікувану максимальну потужність таблиці. Для конкретної таблиці, коли кількість рядків і стовпців менше, ніж n і m відповідно, для генотипу таблиці відповідні висоти рядків h_i та ширини стовпців w_i встановлюються рівними нулю. Генотип представляю собою просту двовимірну таблиці у випадку коли злиття комірок таблиці вимкнено. Клас таблиці може бути додатково розширено за рахунок введення індикаторного масиву клітин злиття в таблицю генотипу:

$$\{c_{ij}\}_{(i,j=1)}^{(n,m)}, \tag{3.8}$$

де c_{ij} буде визначається як унікальний номер , який вказує на те, який тип клітини злиття (якщо такі є) необхідно для осередку в i-му рядку та j-му стовпці.

Цільові функції оптимізації генотипу. Придатність таблиці визначається лінійно пропорційно значенню цільової функції для таблиці, тому, ніж таблиця, що встановлює, має краще значення цільової функції. Деякі цільові функції тестуються для визначення відповідності між згенерованим cGAN «скелетом» таблиці і створеним кандидатом в таблиці, фенотипу u , які представляють оптимізований генотип таблиці:

1) Максимізація ймовірності фенотипу кандидатів таблиці *u* на те що вони є вірними, згідно дискримінатору cGAN:

$$\max_{u}[\log D\,(x,u)]\,; \tag{3.9}$$

2) Мінімізація дистанції L_1 відповідно згенерованому cGAN зображенню та кандидата в зображення на піксельному рівні:

$$\min_{u}|G(x,z)-u|_{L1}\,; \tag{3.10}$$

3) Максимізація зваженої різниці, що виникає в результаті обчислень (3.9) і (3.10) подібної до (3.6):

$$\max_{u}[\log D\,(x,u)-\lambda|G(x,z)-u|_{L1}]\,; \tag{3.11}$$

4) Мінімізація частки не білих пікселів, що не узгоджуються між cGAN та зображенням кандидатом обчислюється наступним чином:

$$\min_{u}\frac{|G(x,z)-u|_{L1}}{|1-u|_{L1}\cdot|1-G(x,z)|_{L1}}; \tag{3.12}$$

де пікселі вихідного зображення з cGAN *G(x, z)* та фенотипу таблиці кандидатів масштабуються до значень від 0 до 1, причому 0 відповідає чорному пікселю, а 1 відповідає білому пікселю. Цільова функція (3.12) коливається від 0 до 1, так що найкращий збіг відповідає 0, а найгірша здогадка відповідає 1.

3.4 Tesseract OCR

Tesseract OCR – це вільно-розповсюджуваний двигун оптичного розпізнавання символів. Ця система OCR була розроблена у 1985 році компанією Hewlett Packard. На даний момент система належить Google

Система працює за наступним принципом:

1. Базова передобробка зображення, виділення контурів символів
2. Аналіз макету сторінки
   2.1. Аналіз ліній через LSTM (для деяких мов, починаючи з версії 4.0)
   2.2. Формування розпізнаних символів у слова
3. Двопрохідний аналіз зображення

3.1. Спроба розпізнати кожне слово по черзі. Кожне задовільне слово передається в адаптивний класифікатор, через що текст внизу сторінки розпізнається точніше

3.2. Друга спроба розпізнати слова, які були погано розпізнані, через брак знань в адаптивному класифікаторі

4. Нормалізація символів по X-висоті

5. Нормалізація нечітких пробілів

6. Нормалізація біграмми слів

Схема роботи двигуна оптичного розпізнавання символів Tesseract зображена на рисунку 3.6.

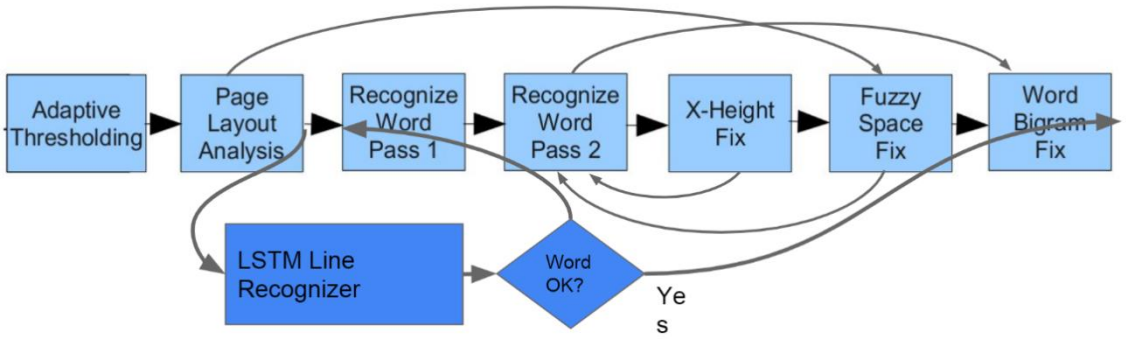


Рисунок 3.6 Схема роботи Tesseract OCR

Перевагою цього OCR є попередньо навчені моделі даних для кожної підтримуваної мови, в тому числі української, які поділяються на:

- Великі, з якісним вилученням даних, але дуже повільні
- Малі, з гіршим вилученням даних але більш швидкодіючі

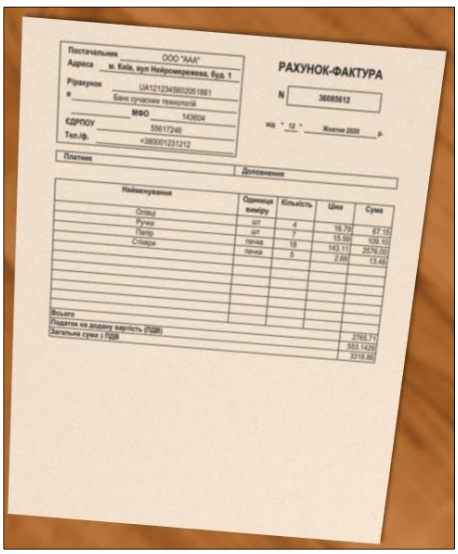
За рахунок вільного розповсюдження, в систему є можливість додавати свої шрифти та словники, на основі яких можна покращити вилучення даних під кожний конкретний випадок.

Недоліком цього OCR є необхідність підготовки зображень, для покращення якості вилучення даних, не зважаючи на те що сам двигун також проводить передобробку зображень. Tesseract може коректно вилучати дані з зображень, якщо

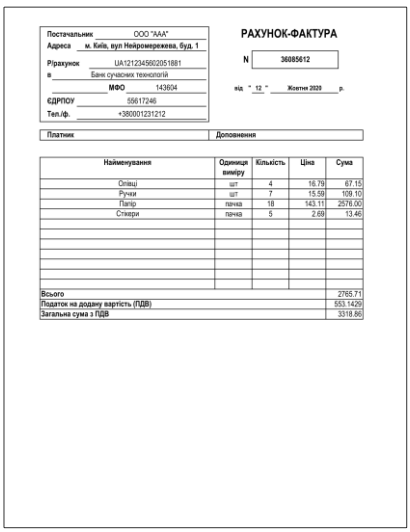
вони не мають зайвих шумів, а у тексту немає суттєвих викривлень. Інакше, дані можуть бути вилучені з великим відсотком помилок.

3.5 Попередня обробка зображень

Як було сказано раніше, для покращення роботи OCR-двигуна, потрібно проходити стадію попередньої обробки зображення. Існують випадки, коли двигун не може розпізнати текст, або знаходить «псевдо»-символи у тій частині зображення, яка не відноситься до оброблюваного документу. Часто таке може трапитись з документами, які отримані шляхом фотографування на камеру смартфону, а не через сканування документу. Приклад такого документу, та документу що придатний до вилучення даних наявний на рисунку 3.7.



а) Оригінальне зображення



Постачальник ООО "ААА"
Адреса м. Київ, вул Нейромережева, буд. 1
Р/рахунок UA121234560205188
в Банк сучасних технологій
МФО 143604
ЄДРПОУ 55617246
Тел./ф. +380001231212

РАХУНОК-ФАКТУРА
N 36085612
від " 12 " Жовтня 2020 р.

Платник | Доповнення

Найменування	Одиниця виміру	Кількість	Ціна	Сума
Олівці	шт	4	16.79	67.15
Ручки	шт	7	15.59	109.10
Папір	пачка	18	143.11	2576.00
Стікери	пачка	5	2.69	13.46
Всього				2765.71
Податок на додану вартість (ПДВ)				553.1429
Загальна сума з ПДВ				3318.86

б) Приклад зображення, придатного до вилучення даних

Рисунок 3.7 Приклад зображення, відносно яких може виникнути необхідність вилучення даних

Основними етапами є:

- вирівнювання і кадрування зображення;
- переведення зображення у відтінки сірого;
- бінаризація зображення;
- видалення шумів;

Розглянемо ці етапи детальніше.

3.5.1. Вирівнювання і кадрування зображення.

Якщо розглянути рисунок 3.7, з лівої сторони ми бачимо зображення, де документ, з якого може виникнути необхідність вилучити дані знаходить під нахилом, прямокутні пропорції втрачено, присутній зайвий фон.

Для того щоб прибрати усі вищеописані недоліки, які можуть вплинути на результати розпізнавання, проводиться наступний ряд операцій:

- пошук максимальної висоти та ширини зображення;
- пошук кутів зображення;
- створення нових приблизних контурів;
- кадрування зображення і приведення до пропорцій правильного прямокутника;

Графічно увесь процес і результат зображені на рисунку 3.8.

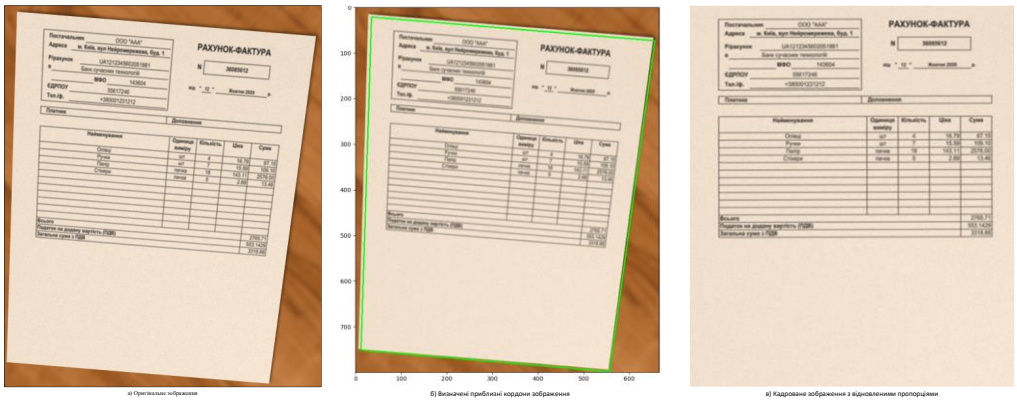


Рисунок 3.8 – Етапи вирівнювання та кадрування зображення

3.5.2. Бінаризація зображення та видалення шумів

Під бінаризацією зображення мається на увазі процес перетворення зображення з сірих кольорів в чорно-біле зображення. Таким чином, з 256 відтінків сірого кольору залишається лише 2 відтінки - чорний та білий. Інколи це називається встановленням порогу зображення.

Якщо вхідне зображення кольорове, перед бінаризацією проводиться приведення зображення до відтінків сірого.

Видалення шумів – це прибирання тих пікселів з зображення, які не відносяться до жодної «групи» пікселів, і по суті являються зайвою інформацією.

Видалення шумів поділяється на звичайне, , адаптивне, та Гауссове.

Звичайне видалення шумів полягає в тому, що для кожного пікселю накладається одне і те саме порогове значення. Якщо значення пікселя менше порогового – тоді він прибирається, інакше виставляється максимальне порогове значення. У вигляді формули можна зобразити так:

$$\mathrm{dst}(\mathrm{x},\mathrm{y}) = \begin{cases} \mathrm{maxval} & \text{if } src(x,y) > \mathrm{thresh} \\ 0 & \text{otherwise} \end{cases}. \qquad (3.13)$$

Адаптивне видалення шумів найчастіше за все, використовується на зображеннях з різним освітленням в різних областях, через те що отримуються різні порогові значення для областей з різним освітленням, що в сумі дає кращий результат. У вигляді формули. Порогове значення являє собою середнє значення площі околу точки, мінус певне константне значення.

Різновидом адаптивного видалення шумів є Гауссова порогізація. При використанні такого видалення шумів, пороговим значенням стає гауссо-взважена сума значень околу, мінус певне константне значення.

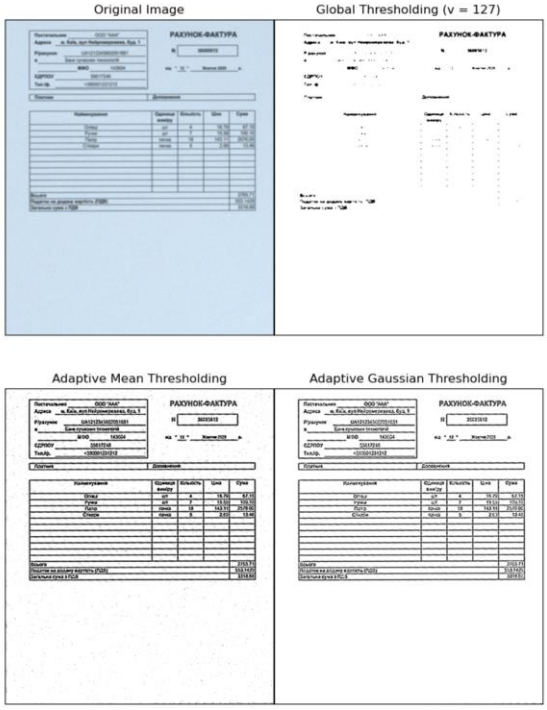


Рисунок 3.9 – Порівняння різних методів прибирання шуму

Як видно з рисунку 3.9, найкраще за все для зображень з текстом працює адаптивне прибирання шуму за методом Гауса.

3.6 Архітектура системи

Відповідно до вимог, інформаційна система побудована як веб-додаток, з використанням клієнт-серверної архітектури.

Так як від системи вимагається не тільки опрацювання інформації, а і збереження даних, у систему включений модуль баз даних. Усі модулі працюють лише за своїм призначенням і можуть взаємодіяти з іншими модулями інформаційної системи.

3.7 Модулі

В даному розділі описані архітектура інформаційної системи та призначення модулів програми.

3.7.1. Серверний модуль

У клієнтському модулі відбувається опис частини веб-інтерфейсу, взаємодія з клієнтською частиною, управління доступом до системи, операції з базою даних з користувачами і відправка даних до модуля БД.

Основним завданням модуля є зв'язок з усіма іншими компонентами інформаційної системи, маршрутизація запитів до системи, управління доступом до системи.

3.7.2. Модуль вилучення даних

У цьому модулі відбувається попередня обробка зображень, вилучення пар ключ-значення та вилучення значень з таблиць, що присутні на зображенні. Зображення, які опрацьовуються цим модулем потрапляють з клієнтської частини через модуль сервера.

3.7.3. Модуль веб-інтерфейсу

Модуль забезпечує відображення інтерфейсу, описаного у клієнтському модулі, відправку запитів до серверної частини і відображення результатів опрацювання запитів з клієнтської частини.

Через веб-інтерфейс відбувається вся робота з користувачем, така як авторизація, завантаження зображення, яке потребує вилучення даних, відображення результатів опрацювання зображення.

3.7.4. Модуль бази даних

Під модулем бази даних мається на увазі сховище для оцифрованих зображень та вилучених з них даних. Також, у базі даних зберігаються множина можливих варіантів значень ключів, які дозволяють відносити різні документи до одного типу документів.

Використовується документоорієнтована база даних, що використовує структуру дерева. Це дозволяє навіть при складні структурі бази легко знаходити шлях шуканих даних.

В базі даних визначені наступні колекції:

- колекція з документами, де зберігається вилучені пари ключ-значення, таблиці та саме зображення:
  - колекція полів, де зберігається ключ та власне значення;
  - колекція таблиць, де зберігаються табличні колонки, в яких відповідно, зберігається колекція табличних колонок та їх значень;
- колекція з можливими варіантами ключів, за якими можна віднести те, що вилучені дані відносяться до певного типу документу.

Колекції не пов’язані між собою зв’язками, пошук співпадінь між варіантами ключів та вилученими даними відбувається програмно. Таким чином забезпечується можливість розширення вже існуючих навчених моделей, не вносячи зміни в уже вилучені дані.

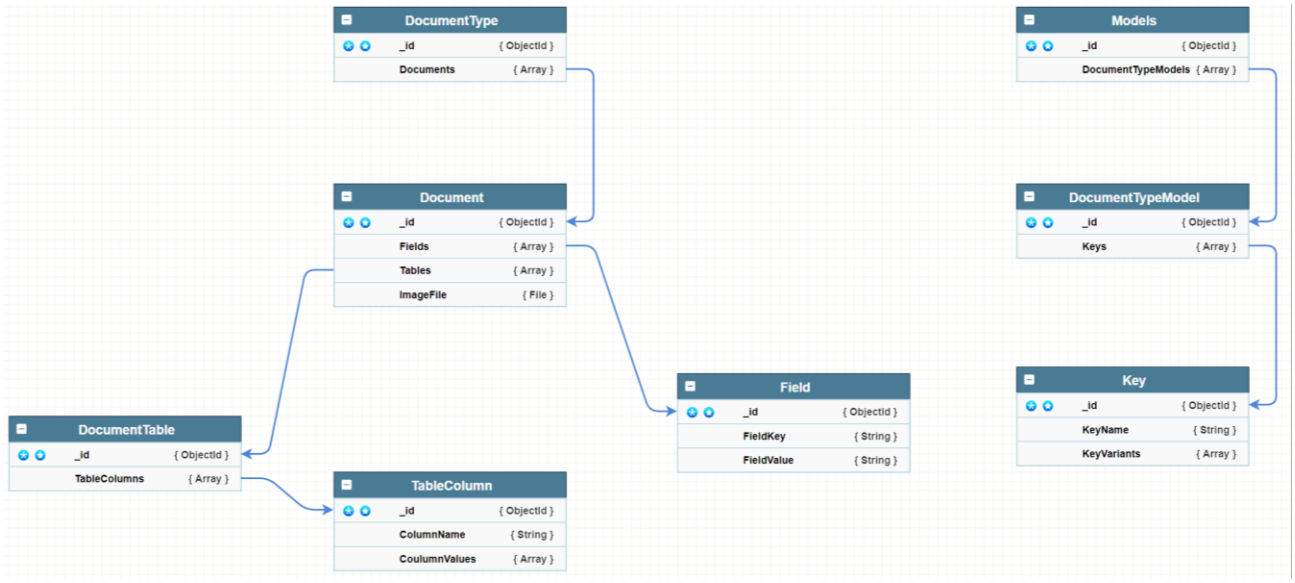
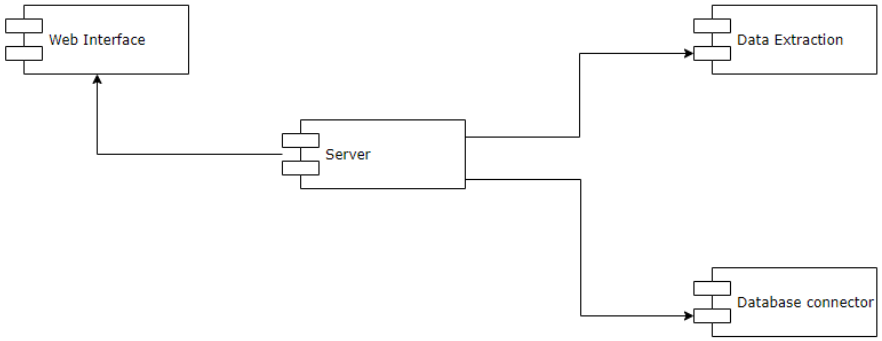


Рисунок 3.10 - EER діаграма бази даних

EER-діаграму бази даних можна зобразити наступним чином (рис.3.10).

3.8 Діаграма компонентів інформаційної системи

Діаграму компонентів можна зобразити наступим чином:



1

Рисунок 3.11 - Діаграма компонентів інформаційної системи

Як видно з рисунку 3.11, взаємодія модулів інформаційної системи відбувається через серверний модуль. Таким чином, модулі існують незалежно одне від одного, і не є повністю взаємопов’язаними.

Висновки

У цьому розділі були дані математичне відображення нейронних мереж, їх загальні принципи роботи, та загальна архітектура інформаційної системи.

Архітектура є розширюваною і гнучкою завдяки поділу системи на окремі модулі, які мають свою власну роль і власну логіку. Існують також різні зовнішні бібліотеки, які спрощують код і роблять його більш читабельним, отже - підтримуваним. Ці характеристики та описані принципи дозволяють враховувати специфічну логіку, що використовується в цій системі.

Математичне представлення нейронних мереж показує загальний принцип їх роботи.

Рівень точності роботи є прийнятним, вибрані формули, вбудовані в основу прорахунків, забезпечують нейронні мережі конкретними цифрами з точністю, що допускається для таких нейронних мереж.

4 ОПИС ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

4.1 Платформи, які використовуються

В даному розділі розміщений опис платформ, що використовувались при розробці інформаційної системи.

4.1.1. Python

Python - це мова програмування високого рівня, розроблена для того, щоб її було легко читати та легко реалізовувати. Вона є відкритою , що означає, що її можна використовувати навіть для комерційних програм. Python може працювати на системах Mac, Windows та Unix, а також може бути перенесеним на віртуальні машини Java та .NET.

Python вважається мовою сценаріїв, і часто використовується для створення веб- додатків та динамічного веб-вмісту. Він підтримується низкою програм 2D та 3D-зображень, що дозволяє користувачам створювати власні плагіни та розширення за допомогою Python. Також, Python часто використовується тоді, коли є необхідність використовувати нейронні мережі, глибоке або машинне навчання.

Сценарії, написані на Python (файли .PY), можуть бути збережені як складені програми (файли .PYC), які часто використовуються як модулі програмування, на які можуть посилатися інші програми Python.

Python є мовою, що використовує інтерпретатор, замість компілятора. Це дозволяє динамічно керувати пам’яттю і виконує порядково необхідний код, на відміну від компільованих програм, де спочатку всі інструкції аналізуються, транслюються в машинний код і лише після цього виконуються. У цьому підході є і свої недоліки, наприклад по помилку в коді можна дізнатись лише під час виконання конкретного рядку коду.

Python підтримує динамічну типізацію, тобто, тип змінної визначається лише під час виконання. З базових типів слід зазначити підтримку цілих чисел довільної довжини і комплексних чисел. Python має багату бібліотеку для роботи з рядками, зокрема, кодованими в Unicode. З колекцій Python підтримує кортежі (tuples), списки (масиви), словники (асоціативні масиви) і від версії 2.4, множини. Система

класів підтримує множинне успадкування і метапрограмування. Будь-який тип, включаючи базові, входить до системи класів, й за необхідності можливе успадкування навіть від базових типів [18]. Але такий підхід також може викликати в роботі програми

4.1.2. Django

Django - це безкоштовний високорівневий фреймворк веб-додатків із відкритим кодом, написаний на Python. Веб-фреймворк - це набір компонентів, який допомагає швидше та простіше розробляти веб-сайти.

Коли створюється веб-сайт або веб-додаток, часто потрібний певний набір компонентів, як-то наприклад спосіб обробки автентифікації користувачів (реєстрація, вхід, вихід), панель управління веб-сайтом, форми, спосіб завантаження файлів тощо.

Django використовує архітектуру MTV (model- template -view), що є однією з варіацій MVC (model-view-controller).

В MTV-підході до розробки:

- «M» визначено для «Моделі» (Model), шару доступу до даних. Цей шар знає все про дані: як отримати до них доступ, як перевірити їх, як з ними працювати і як дані пов'язані між собою.
- «T» визначено для «Шаблона» (Template), шару представлення даних. Цей шар приймає рішення щодо подання даних, як і що має відображатися на сторінці або в іншому типі документа.
- «V» визначено для «Уявлення» (View), шару бізнес-логіки. Цей шар містить логіку, як отримувати доступ до моделей і застосовувати відповідний шаблон. Розглядається як посередник між моделями і шаблонами.

Деякі можливості Django [20]:

- ORM, API доступу до БД з підтримкою транзакцій[6]
- вбудований інтерфейс адміністратора, з уже наявними перекладами на більшість мов
- диспетчер URL на основі регулярних виразів

- розширювана система шаблонів з тегами та наслідуванням
- система кешування
- інтернаціоналізація
- архітектура застосунків, що підключаються, які можна встановлювати на будь-які Django-сайти
- «generic views» - шаблони функцій контролерів
- авторизація та аутентифікація, підключення зовнішніх модулів аутентифікації: LDAP, OpenID та ін.
- система фільтрів («middleware») для побудови додаткових обробників запитів, наприклад включені в дистрибутив фільтри для кешування, стиснення, нормалізації URL і підтримки анонімних сесій
- бібліотека для роботи з формами (наслідування, побудова форм за існуючою моделлю БД)
- вбудована автоматична документація по тегам шаблонів та моделям даних, доступна через адміністративний застосунок

Різні компоненти фреймворку між собою пов'язані слабо, тому достатньо будь-яку частину замінити на аналогічну.

4.1.3. ORM та Mongoengine

Об’єктно-реляційні карти (ORM) - це бібліотека коду, яка автоматизує передачу даних, що зберігаються в таблицях реляційних баз даних, в об’єкти, які частіше використовуються в коді програми.

ORM забезпечують абстракцію високого рівня на реляційній базі даних, що дозволяє розробнику писати код Python замість запитів для створення, читання, оновлення та видалення даних та схем у своїй базі даних. Розробники можуть використовувати мову програмування, яка їм комфортна, для роботи з базою даних замість того, щоб писати оператори SQL або використовувати збережені процедури.

Так як у додатку використовується документоорієнтована СУБД MongoDB, , замість звичайного Python ORM використовується Mongoengine.

MongoEngine – це Object Document Mapper (ODM), тобто механізм, який автоматизую передачу даних від документів, що зберігаються в документоорієнтованій БД до об’єктів. Цей механізм створений спеціально для взаємодії між Python та Mongo DB. Абстракція, яку надає MongoEngine, базується на класах, тому всі створені вами моделі є класами.

4.1.4. MongoDB

MongoDB — документоорієнтована система управління базами даних з відкритим вихідним кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СУБД, функціональними і зручними у формуванні запитів.

MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі зміни і додавання даних в БД, може працювати відповідно до парадигми Map/Reduce, підтримує реплікацію і побудову відмовостійких конфігурацій. У MongoDB є вбудовані засоби із забезпечення шардінгу (розподіл набору даних по серверах на основі певного ключа), комбінуючи який з реплікацією даних можна побудувати горизонтально масштабований кластер зберігання, в якому відсутня єдина точка відмови (збій будь-якого вузла не позначається на роботі БД), підтримується автоматичне відновлення після збою і перенесення навантаження з вузла, який вийшов з ладу. Розширення кластера або перетворення одного сервера на кластер проводиться без зупинки роботи БД простим додаванням нових машин.

Основні можливості MongoDB:

- Документоорієнтоване сховище (проста та потужна JSON-подібна схема даних);
- Досить гнучка мова для формування запитів;
- Динамічні запити;

- Повна підтримка індексів;
- Профілювання запитів;
- Швидкі оновлення «на місці»;
- Ефективне зберігання бінарних даних великих обсягів, наприклад, фото та відео;
- Журналювання операцій, що модифікують дані в БД;
- Підтримка відмовостійкості і масштабованості: асинхронна реплікація, набір реплік і шардінг;
- Може працювати відповідно до парадигми MapReduce (це модель розподілених обчислень, яка використовується в технологіях Big Data для паралельних обчислень над дуже великими (до декількох петабайт) наборами даних в комп'ютерних кластерах, і фреймворк для обчислення розподілених задач на вузлах (node) кластера);

Як видно з можливостей, MongoDB є розвиненою, потужною і стійкою системою для сховища, яка за рахунок своєї структури краще за все підходить для обробки документів.

4.2 Модулі, що використовуються в роботі інформаційної системи

В даному розділі розміщений опис модулів, що використовувались при розробці інформаційної системи.

4.2.1. TensorFlow

TensorFlow - це фреймворк з відкритим кодом, розроблений дослідниками Google для запуску машинного навчання, глибокого навчання та інших навантажень із статистичної та прогнозної аналітики. Подібно до інших платформ, він розроблений для впорядкування процесу розробки та запуску супутніх додатків аналітики для користувачів, таких як спеціалістів в дослідженні даних, статистиків та спеціалістів в створенні моделей прогнозування.

Програмне забезпечення TensorFlow обробляє набори даних, які розміщені як обчислювальні вузли у вигляді графіків. Ребра, що з'єднують вузли в графіку,

можуть представляти багатовимірні вектори або матриці, створюючи так звані тензори. Оскільки програми TensorFlow використовують архітектуру потоку даних, яка працює з узагальненими проміжними результатами обчислень, вони особливо відкриті для дуже масштабних додатків паралельної обробки, а нейронні мережі є загальним прикладом.

Структура включає набори API високого та низького рівня. Рекомендується використовувати високорівневі мови програмування, коли це можливо, для спрощення розробки конвеєра даних та програмування програм. Однак знання того, як використовувати API низького рівня - під назвою TensorFlow Core - може бути цінним для експериментів та налагодження програм, це також дає користувачам "розумову модель" внутрішнього функціонування технології машинного навчання.

Додатки TensorFlow можуть працювати як на звичайних центральних процесорах, так і на більш високопродуктивних графічних процесорах (GPU) , а також на власних процесорах Google для обробки тензорів (TPU), які є спеціальними пристроями, спеціально розробленими для прискорення завдань TensorFlow.

4.2.2. Keras

Keras – це бібліотека нейромережевих компонентів з відкритим кодом, написана на Python. Keras може працювати на TensorFlow , Theano , PlaidML та інших. Бібліотека була розроблена, щоб бути модульною та зручною для користувача, однак спочатку вона почалася в рамках дослідницького проекту з відкритою нейро-електронною інтелектуальною операційною системою (скорочено ONEIROS).

Платформа з відкритим кодом Keras, що складається з бібліотеки часто використовуваних компонентів машинного навчання, включаючи цілі, функції активації та оптимізатори, також пропонує підтримку періодичних та згорткових нейронних мереж . Крім того, Keras пропонує розробку мобільної платформи для користувачів, які мають намір впровадити моделі глибокого навчання на смартфонах, як iOS, так і Android.

4.2.3. PyTorch

PyTorch - це бібліотека машинного навчання для Python, яка використовується в основному для обробки природної мови. PyTorch використовує модуль Autograd для обчислення автоматичної диференціації. Коротше кажучи, реєстратор детально описує, які операції виконуються, а потім відтворює його для синтезу градієнтів. Це економить час на розробку нейронних мереж, оскільки диференціація даних виконується швидко на прямому проході. Оптимальний пакет PyTorch дозволяє користувачеві визначити оптимізатор, який автоматично оновлюватиме ваги. Однак, коли користувачі хочуть створити власну модель, вони можуть скористатися перевагами nn.module PyTorch. Враховуючи різні модулі, PyTorch дозволяє реалізовувати різні типи шарів, такі як згорткові шари, періодичні шари, і лінійні шари, серед інших.

4.3 Робота с початковим зображенням

4.3.1. OpenCV

OpenCV (Open Source Computer Vision Library) - це бібліотека програмного забезпечення для комп'ютерного зору та машинного навчання [31]. OpenCV був побудований для забезпечення загальної інфраструктури для програм комп'ютерного зору та прискорення використання машинного сприйняття в комерційних продуктах.

Бібліотека має понад 2500 оптимізованих алгоритмів, що включає в себе повний набір як класичних, так і найсучасніших алгоритмів комп’ютерного зору та машинного навчання. Ці алгоритми можна використовувати для виявлення та розпізнавання обличь, ідентифікації об'єктів, класифікації людських дій у відео, відстеження рухів камери, відстеження рухомих об'єктів, вилучення 3D-моделей об'єктів, створення 3D-хмар точок зі стереокамер, зшивання зображень для отримання високої роздільної здатності зображення цілої сцени, знайти подібні зображення з бази даних зображень, видалити червоні очі із зображень, зроблених за допомогою спалаху, стежити за рухами очей, розпізнавати декорації та встановлювати маркери, щоб накласти їх на доповнену реальність тощо. Бібліотека

широко використовується у компаніях, дослідницьких групах та державних органах.

OpenCV має інтерфейси C ++, Python, Java та MATLAB та підтримує Windows, Linux, Android та Mac OS. OpenCV здебільшого використовується для «програм зору» в реальному часі та використовує переваги інструкцій MMX та SSE, коли вони доступні. Зараз розробляються повнофункціональні інтерфейси CUDA та OpenCL . Існує понад 500 алгоритмів і приблизно в 10 разів більше функцій, які складають або підтримують ці алгоритми. OpenCV написаний на мові C ++ і має шаблонний інтерфейс, який безперебійно працює з контейнерами STL.

4.3.2. NymPy

NumPy - це основний пакет для наукових обчислень у Python. Це бібліотека Python, яка забезпечує використання багатовимірних масивів, різних похідних від масивів об’єктів (наприклад, масковані масиви та матриці) та асортимент підпрограм для швидких операцій над масивами, включаючи математичні, логічні, маніпуляції з фігурами, сортування, вибір, введення/виведення , дискретні перетворення Фур'є, основна лінійна алгебра, основні статистичні операції, випадкове моделювання та багато іншого [32].

Існує кілька важливих відмінностей між масивами NumPy та стандартними послідовностями Python [32]:

- Масиви NumPy мають фіксований розмір при створенні, на відміну від списків Python (які можуть динамічно зростати). Зміна розміру масиву створить новий масив і видалить оригінал.

- Всі елементи масиву NumPy повинні мати один і той же тип даних, і, отже, вони матимуть однаковий розмір у пам'яті. Виняток: можна мати масиви об’єктів (Python, включаючи NumPy), що дозволяє мати масиви різних за розміром елементів.

- Масиви NumPy сприяють вдосконаленим математичним та іншим типам операцій над великою кількістю даних. Як правило, такі операції виконуються

ефективніше і з меншим кодом, ніж це можливо за допомогою вбудованих послідовностей Python.

• Зростаюче безліч науково-математичних пакетів, заснованих на Python, використовують масиви NumPy; хоча вони, як правило, підтримують введення послідовності Python, вони перетворюють такі входи в масиви NumPy перед обробкою, і вони часто виводять масиви NumPy. Іншими словами, для ефективного використання більшої частини (можливо, навіть більшості) сучасного науково-математичного програмного забезпечення на базі Python, просто знання того, як використовувати вбудовані типи послідовностей Python, недостатньо - потрібно також знати, як використовувати масиви NumPy.

4.4 Нейронні мережі

В даному розділі будуть дані основні відомості про типи нейронних мереж, що використовуються в інформаційній системі.

4.4.1. GCN

Графічна нейронна мережа, також відома як Graph Convolutional Network (GCN), є методом класифікації зображень. GCN виконує згортку на графіку, а не на зображенні, що складається з пікселів.

Подібно до того, як CNN прагне отримати найважливішу інформацію із зображення для класифікації зображення, GCN передає фільтр через графік, шукаючи основні вершини та ребра, які можуть допомогти класифікувати вузли в графі.

GCN можна використовувати для наступних цілей:

• генерування прогнозів у фізичних системах;

• класифікація зображень;

• прогнозування спільноти;

• аналіз молекулярної структури;

• дослідження операцій та комбінаторна оптимізація.

4.4.2. cGAN

Умовна генеративна змагальна мережа (cGAN) - це продовження генеративної змагальної мережі (GAN), яка використовується як основа машинного навчання для навчання генеративних моделей.

cGAN - це метод глибокого навчання, де застосовується умовна установка, що означає, що і генератор, і дискримінатор обумовлені якоюсь допоміжною інформацією, такою як мітки класів або дані з інших способів. Як результат, ідеальна модель може навчитися мультимодальному відображенню від входів до виходів, подаючи їй різну контекстну інформацію.

На даний момент, сфера застосування cGAN наступна:

- синтез зображень в зображення;
- синтез тексту до зображення;
- генерація відео;
- генерація обличчя з атрибутами шуму;
- формування тіньових карт;
- завдання комп’ютерного зору, чутливі до різноманітності.

Висновки

В цьому розділі описані різні платформи, модулі та технології, які застосовувались при створенні інформаційної системи.

Можливість використання готових різноманітних модулів та платформ дозволяє значно спростити процес розробки . Кожен із принципів реалізації був організований відповідно до принципів та архітектурних рішень, описаних у розділі 3 із вимог, описаних у розділі 2.

ВИСНОВКИ

У даній роботі описані основині принципи розробки інформаційної системи, здатної до автоматичного вилучення ключових даних і таблиць з україномовних документів, отриманих шляхом цифровізації або шляхом електронної комунікації.

1. Проаналізовано існуючі комерційні рішення, направлені на вирішення проблеми вилучення даних, визначені їх переваги та недоліки.

Аналіз параметрів та підходів розроблених методів вилучених даних стало підґрунтям для створення чіткого технічного завдання, що закладене в основу розробки описаного програмного продукту.

2. Вибрані конфігурації нейронних мереж, які найкраще за все підходять для вирішення поставленої задачі;

3. Розроблене технічне завдання, що містить базові визначення, вимоги до інтерфейсу, функціональні та нефункціональні вимоги до додатку;

4. 4. Описана архітектура системи модулів, кожен з яких має свої принципи та зону відповідальності, що забезпечує гнучкість та розширюваність програмного продукту. Автономність модулів дає можливість перевикористання коду;

5. В основі додатку реалізовані технології нейронних мереж та машинного навчання;

6. Шляхом використання спеціальних коефіцієнтів вирішена проблема спрощення введення вхідних даних;

7. В основі принципів проектування системи додатку використані широковживані фреймворки та модулі, чим додається можливість авторизації, гнучкості в налаштуваннях для основного функціоналу клієнтської сторони та забезпечується подальша можливість покращення системи.

Перевагою програмного продукту є те, що він дозволяє швидко та ефективно вилучити всі не обхідні дані, майже без участі оператора.

Перелік посилань

1) Smith, R.. (2007). An Overview of the Tesseract OCR Engine. Ninth International Conference on Document Analysis and Recognition (ICDAR 2007). 2. 629 - 633. 10.1109/ICDAR.2007.4376991.

2) Azure vs AWS vs GCP (Part 2: Form Recognizers). Режим доступу - https://cazton.com/blogs/executive/form-recognition-azure-aws-gcp. (дата звернення: 02.11.2020).

3) Smith, R.. (2016). Architecture And Data Structures. Tutorial on Tesseract at DAS2016. Режим доступу: https://github.com/tesseract-ocr/docs/blob/master/das_tutorial2016/2ArchitectureAndDataStructures.pdf
Дата звернення: 03.11.2020

4) Amazon Textract Developer Guide. Режим доступу - https://docs.aws.amazon.com/textract/latest/dg/what-is.html. Дата звернення: 04.11.2020

5) Document AI Documentation. Режим доступу: https://cloud.google.com/document-ai/docs. (дата звернення: 04.11.2020)

6) Nanonets. Режим доступу: https://nanonets.com/

7) Form Recognizer documentation. Режим доступу: https://docs.microsoft.com/en-us/azure/cognitive-services/form-recognizer/

8) ISTQB – Standard Glossary of Terms used in Software Testing Version 3.2, All Terms – 67 pages.

9) Автоматизоване тестування – Вікіпедія. Режим доступу: https://uk.wikipedia.org/wiki/Автоматизоване_тестування.

10) ROI – Вікіпедія. Режим доступу: https://uk.wikipedia.org/wiki/ROI.

11) Що таке Selenium? | Україномовний туторіал по Selenium Web Driver – Habr. Режим доступу: https://www.quality-assurance-group.com/shho-take-selenium/.

12) PyTest Tutorial: What is, Install, Fixture, Assertions Режим доступу: https://www.guru99.com/pytest-tutorial.html

13) Liu, Xiaojing & Gao, Feiyu & Zhang, Qiong & Zhao, Huasha. (2019). Graph Convolution for Multimodal Information Extraction from Visually Rich Documents. 32-39. 10.18653/v1/N19-2005.

14) Eric Medvet, Alberto Bartoli, and Giorgio Davanzo. 2011. A probabilistic approach to printed document understanding. International Journal on Document Analysis and Recognition (IJDAR), 14(4):335–347.

15) Petar Velickovi ˇ c, Guillem Cucurull, Arantxa Casanova, ´ Adriana Romero, Pietro Lio, and Yoshua Bengio. ` 2017. Graph attention networks. arXiv preprint arXiv:1710.10903.

16) Vine, Nataliya & Zeigenfuse, Matthew & Rowan, Mark. (2019). Extracting Tables from Documents using Conditional Generative Adversarial Networks and Genetic Algorithms. 1-8. 10.1109/IJCNN.2019.8851886.

17) Patel, Shreeshiv. (2020). Abstractive Information Extraction from Scanned Invoices (AIESI). Режим доступу: https://www.researchgate.net/publication/341615804_Abstractive_Information_Extraction_from_Scanned_Invoices_AIESI

18) Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-toimage translation with conditional adversarial networks. In 2017 IEEEConference on Computer Vision and Pattern Recognition (CVPR), pages5967–5976. IEEE, 2017.

19) Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In European Conference on Computer Vision, pages 702–716. Springer, 2016.

20) Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022, 2016

21) Image Thresholding. OpenCV documentation. [Електронний ресурс] Режим доступу: https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html

22) Christensson, Per. "Python Definition." TechTerms. Sharpened Productions, 15 June 2010. [Електронний ресурс] Режим доступу: https://techterms.com/definition/python

23) Python – Вікіпедія. Режим доступу: https://uk.wikipedia.org/wiki/Python Дата звернення: 11.12.2020.

24) Nigel George. «Build a Website With Django 3: A complete introduction to Django 3». GNW Independent Publishing, December 20, 2019

25) Django – Вікіпедія. Режим доступу: https://uk.wikipedia.org/wiki/Django Дата звернення: 11.12.2020.

26) Introduction to MongoDB and Python. [Електронний ресурс] Режим доступу: https://realpython.com/introduction-to-mongodb-and-python/#mongodb

27) MongoDB – Вікіпедія. Режим доступу: https://uk.wikipedia.org/wiki/MongoDB Дата звернення: 11.12.2020.

28) TensorFlow – Techtarget- Search Data Management [Інтернет-портал]. – Режим доступу: https://searchdatamanagement.techtarget.com/definition/TensorFlow (дата звернення 11.12.2020). – Назва з екрану.

29) What is Keras? – DeepAI [Інтернет-портал]. – Режим доступу: https://deepai.org/machine-learning-glossary-and-terms/keras (дата звернення 11.12.2020). – Назва з екрану.

30) PyTorch – DeepAI [Інтернет-портал]. – Режим доступу: https://deepai.org/machine-learning-glossary-and-terms/pytorch (дата звернення 11.12.2020). – Назва з екрану.

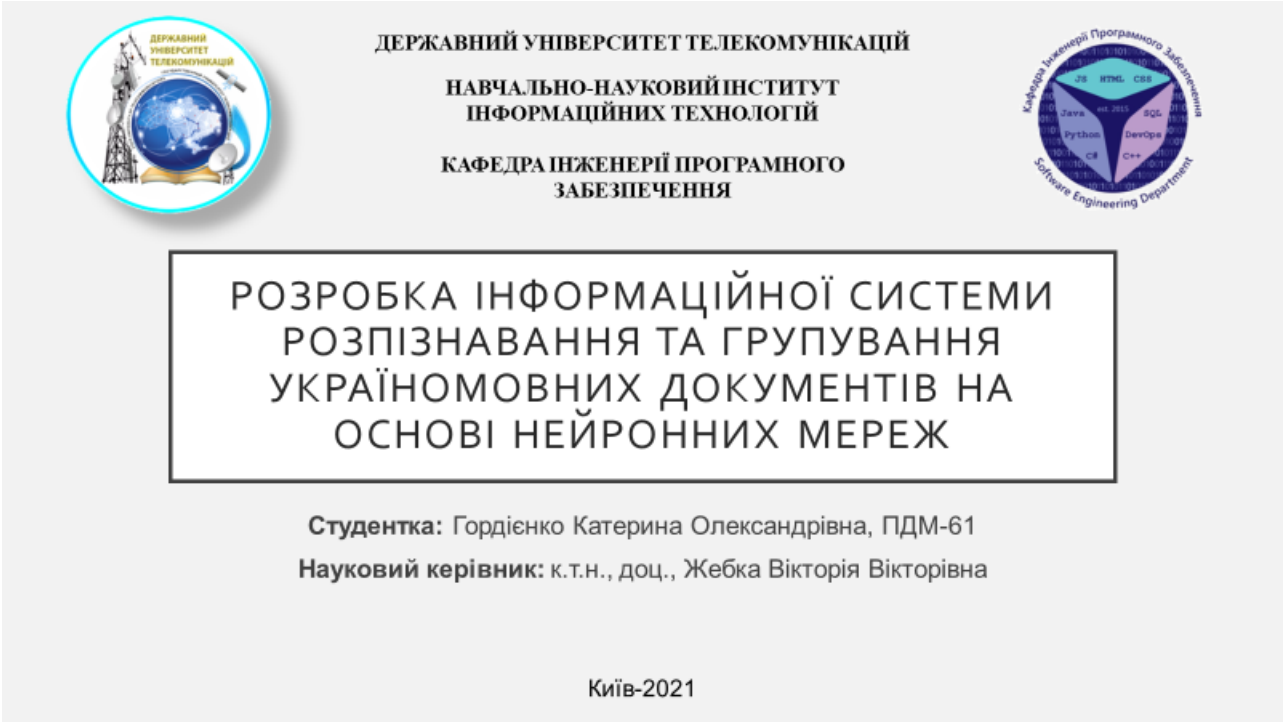
31) About – OpenCV [Електронний ресурс]. – Режим доступу: https://opencv.org/about/ (дата звернення 11.12.2020). – Назва з екрану.

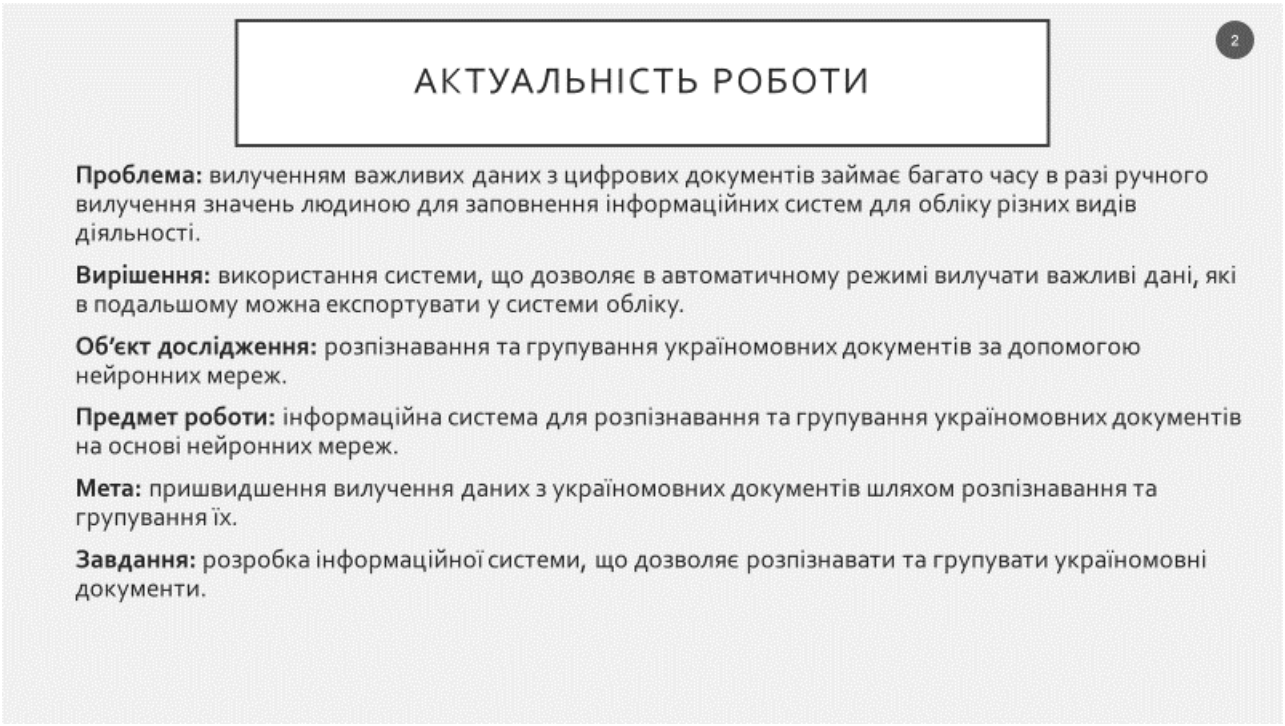
32) What is NymPy? – NumPy [Електронний ресурс]. – Режим доступу: https://numpy.org/doc/stable/user/whatisnumpy.html (дата звернення 11.12.2020). – Назва з екрану.

33) Graph Convolutional Networks – missinglink.ai [Електронний ресурс]. – Режим доступу: https://missinglink.ai/guides/convolutional-neural-networks/graph-convolutional-networks/ (дата звернення 11.12.2020). – Назва з екрану.

34) Conditional generative adversarial network (cGAN) – Golden [Електронний ресурс]. – Режим доступу: https://golden.com/wiki/Conditional_generative_adversarial_network_(cGAN) (дата звернення 11.12.2020). – Назва з екрану.

Додаток А. Демонстраційні матеріали

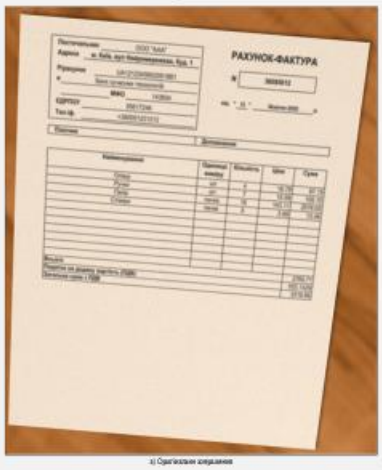




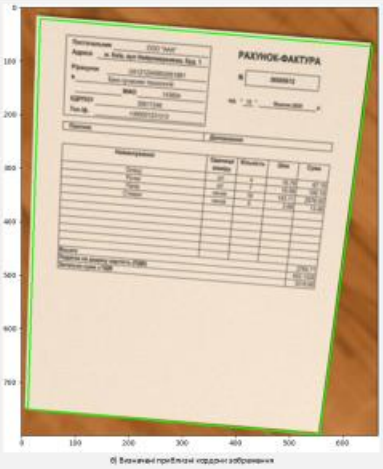
ІСНУЮЧЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Логотип та назва		Тип	Недоліки	Переваги
ABBYY FlexiCapture	Abby FlexiCapture	Вилучають дані на основі заздалегідь заданих шаблонів та правил	При необхідності вилучити дані для документу незнайомого типу потрібно створювати нові шаблони або правила для опрацювання таких документів	Досить висока якість вилучення даних для документів, до яких вже були створені відповідні налаштування. Можливість контролю даних
	Intellix by DocuWare			
dA	docAlpha by Artsyl			
G	Google Form Parser	Хмарні системи, що автоматично створюють пари ключ-значення	Має проблеми з коректним вилученням назв ключів. Потрібно мінімум 5 документів для вилучення даних	Повністю автоматично визначають ключі та їх значення, без втручання оператора
	Amazon Textracts		Не підтримує українську мову. Замість вилучення кириличних символів заміняє їх на латинські аналоги	
	Microsoft Azure Form Recognizer	Хмарні системи, в яких необхідну створювати ключі та самостійно навчати моделі, або використовувати вже навчені		Необхідно лише 5 зображень для навчання моделей. Автоматичне вилучення таблиці
Nanonets	Nanonets		Не розпізнає документи з нахилом у зображенні. В перенавчених моделях є поля, які не застосовуються в україномовних документах	Підтримка української мови. Має базовий інтерфейс коригування даних.

ЕТАПИ ВИРІВНЮВАННЯ ЗОБРАЖЕННЯ



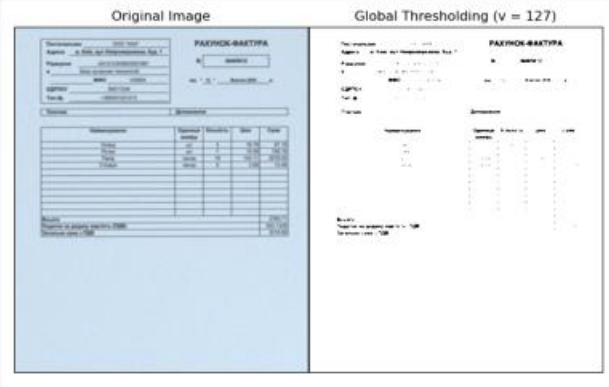
а) Оригінальне зображення

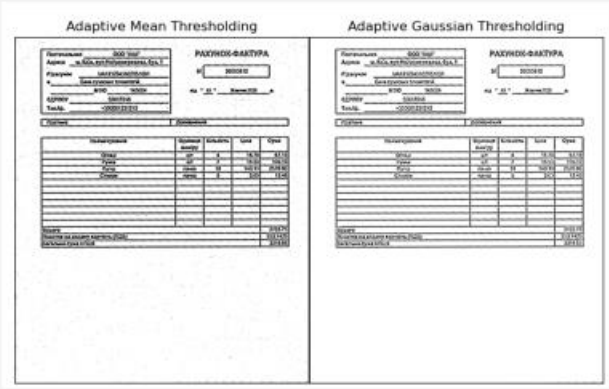


б) Визначені проблемні координати зображення

в) Кадроване зображення з відновленими пропорціями

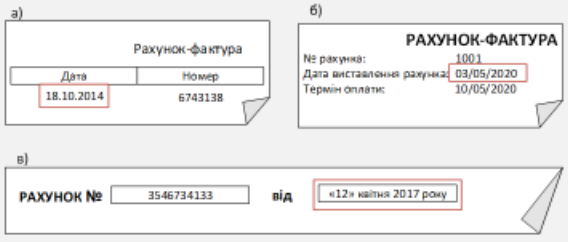
ПОКРАЩЕННЯ ЗОБРАЖЕННЯ



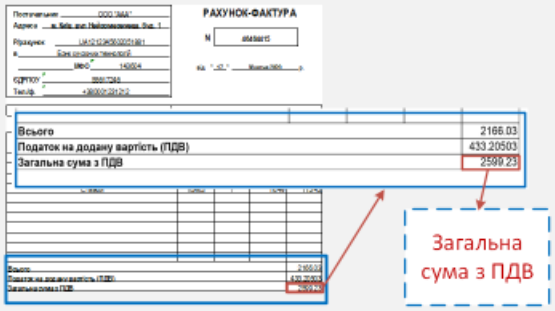


ВИЛУЧЕННЯ ПОЛІВ

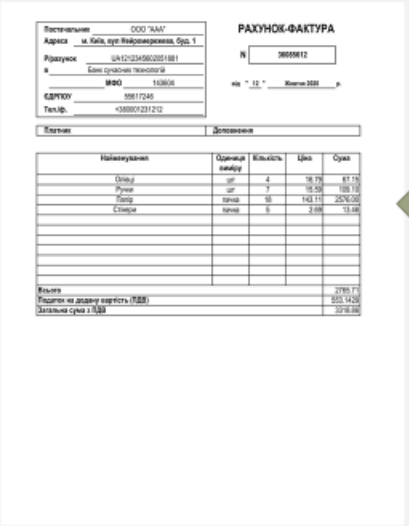
ПРИКЛАД РІЗНИХ ФОРМАТУВАНЬ В ДОКУМЕНТІ

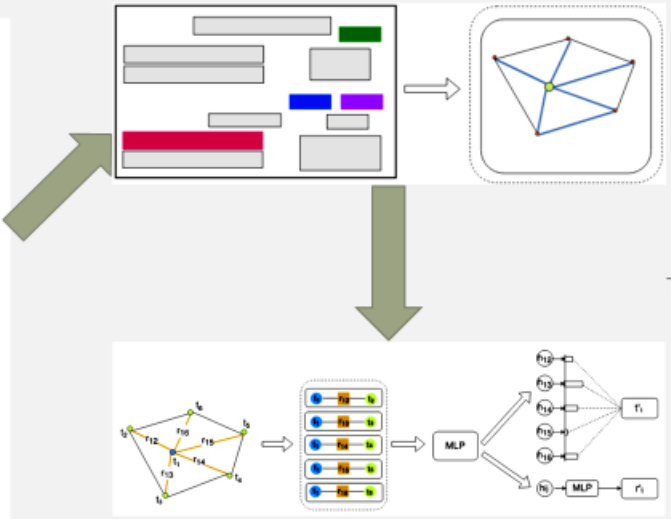


СУТНІСТЬ ДЛЯ ВИЛУЧЕННЯ



ВИЛУЧЕННЯ ПОЛІВ





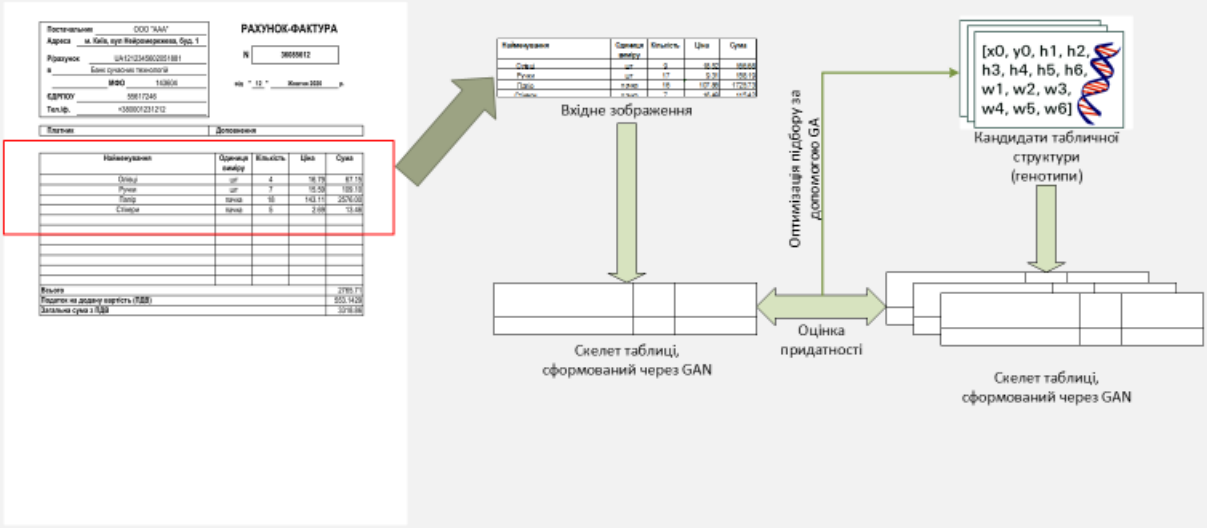
Tesseract OCR

МАТЕМАТИЧНЕ ОБҐРУНТУВАННЯ ВИЛУЧЕННЯ ПОЛІВ

Математично, документ D це кортеж (T,E), де $T = \{t_1, t_2, \cdots, t_n\}$, $t_i \in T$ – це множина з n текстових полів, $R = \{r_{i1}, r_{i2}, \cdots, r_{in}\}$, $r_{ij} \in R$ – множина з ребер, а $E = T \times R \times T$ – це множина направлених ребер виду (t_i, r_{ij}, t_j), де $t_i, t_j \in T$ та $r_{ij} \in R$.

Призначення	**Формула**
Вкладення ребер між вузлами t_i і t_j	$r_{ij} = \left[x_{ij}, y_{ij}, \frac{w_i}{h_i}, \frac{h_j}{h_i}, \frac{w_j}{h_i}\right],$
Згортання графу	$h_{ij} = g(\boldsymbol{t}_i, \boldsymbol{r}_{ij}, \boldsymbol{t}_j) = \mathrm{MLP}([\boldsymbol{t}_i \| \boldsymbol{r}_{ij} \| \boldsymbol{t}_j])$
Вихідне вбудовування $\boldsymbol{t'}_i$ шару для вузла t_i	$t'_i = \sigma\left(\sum_{j\in\{1,\ldots,n\}} \alpha_{ij}\mathbf{h}_{ij}\right),$
Attention-механізм	$a_{ij} = \frac{\exp(\mathrm{LeakyReLU}(w_a^T \mathrm{h}_{ij}))}{\sum_{j\in\{1,\ldots,n\}} \exp(\mathrm{LeakyReLU}(w_a^T \mathrm{h}_{ij}))}$
Вихід для вбудовування краю шару згортки графіків	$\boldsymbol{r'}_{ij} = \mathrm{MLP}(h_{ij})$

ВИЛУЧЕННЯ ТАБЛИЦІ



МАТЕМАТИЧНЕ ОБҐРУНТУВАННЯ ВИЛУЧЕННЯ ТАБЛИЦЬ

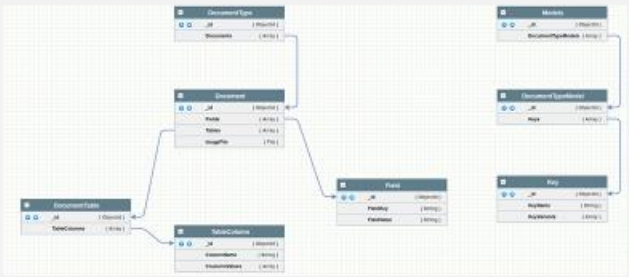
Структура таблиці описується як набір наступних чисел, визначених як генотип таблиці:

1) таблиця потужності кількість рядків n і кількість стовпців m;
2) координати лівого верхнього кута таблиці $\{x_0, y_0\}$;
3) вектор висот рядків $\{h_1, \dots, h_n\}$, $h_i \geq 0$;
4) вектор ширини стовпців $\{w_1, \dots, w_m\}$, $w_j \geq 0$.

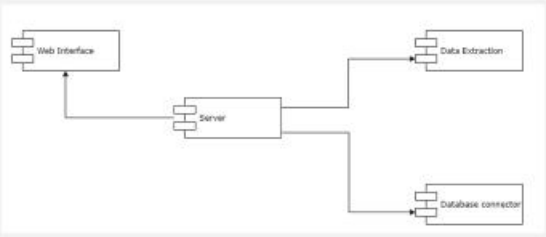
Призначення	**Формула**
Оптимізація та формування висновків у cGAN	$\min_G \left[\max_D L_{cGAN}(G,D) + \lambda L_{L1}(G)\right]$
Ціль cGAN для створення зображень, які не можуть бути дискримінованими з реальних зображень	$L_{cGAN}(G,D) = E_{x,y}[\log D(x,y)] + E_{x,y}[\log(1 - D(x, G(x,z)))].$
Максимізація ймовірності фенотипу кандидатів таблиці u на те що вони є вірними	$\max_u [\log D(x,u)]$;
Мінімізація дистанції L_1 для фенотипу-зображення і реального	$\min_u \lvert G(x,z) - u\rvert_{L1}$;
Максимізація зваженої різниці, що виникає в результаті обчислень	$\max_u [\log D(x,u) - \lambda \lvert G(x,z) - u\rvert_{L1}]$;
Мінімізація частки не білих пікселів, що не узгоджуються між cGAN та зображенням кандидатом	$\min_u \frac{\lvert G(x,z) - u\rvert_{L1}}{\lvert 1 - u\rvert_{L1} \cdot \lvert 1 - G(x,z)\rvert_{L1}}$;

АРХІТЕКТУРА ПЗ

АРХІТЕКТУРА БД



АРХІТЕКТУРА ДОДАТКУ



ЗАСОБИ ДЛЯ РОЗРОБКИ ПЗ

Мови та фреймворки

python

django

mongoDB.

HTML CSS JS

Tesseract OCR

NumPy

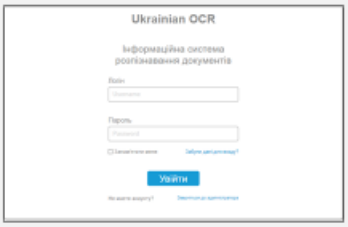
OpenCV

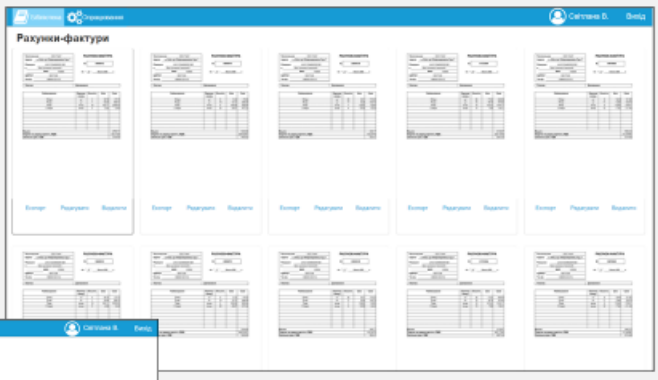
TensorFlow

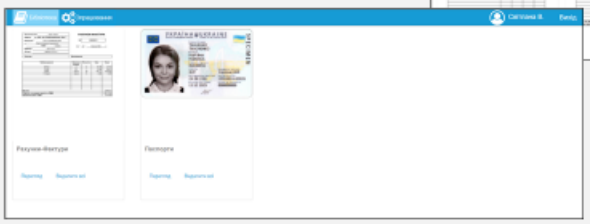
Keras

ІНТЕРФЕЙС РОЗРОБЛЕНОЇ СИСТЕМИ

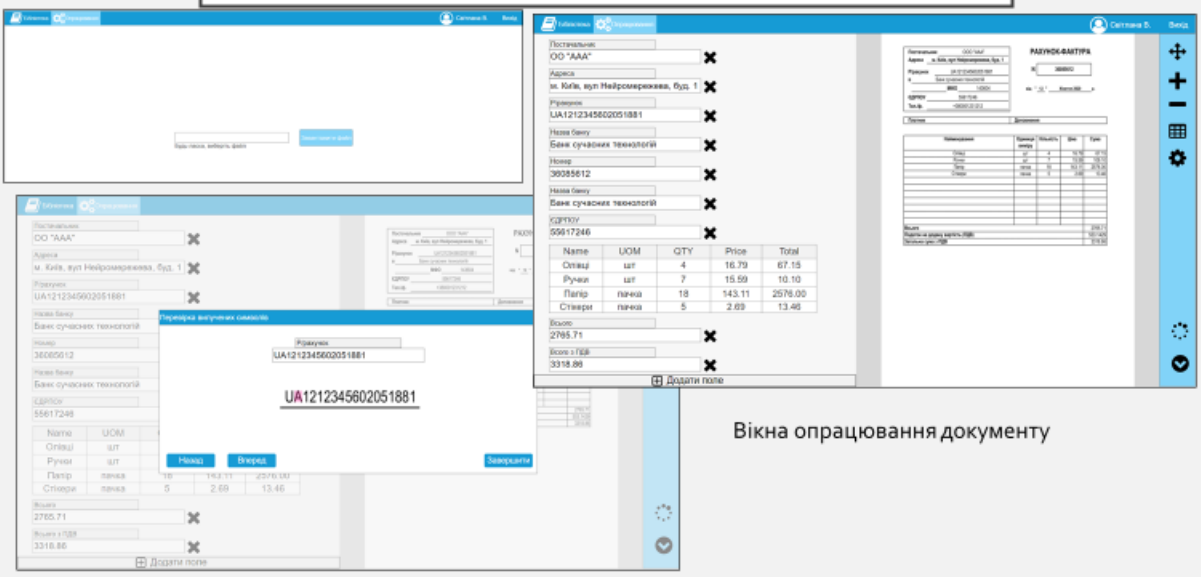
Вікно авторизації та сховище опрацьованих документів







ІНТЕРФЕЙС РОЗРОБЛЕНОЇ СИСТЕМИ



Вікна опрацювання документу

АПРОБАЦІЯ

За матеріалами опубліковані тези:

- Гордієнко К. О. Дослідження інтелектуальних систем вилучення даних з цифрових документів. // Загально-університетська, науково-технічна конференція «Проблеми комп'ютерної інженерії»;

та стаття:

- К. О. Гордієнко, В.В. Жебка «Дослідження систем розпізнавання тексту та вилучення даних з україномовних документів» , Журнал "Зв'язок", №6, 2020, Київ.

ВИСНОВКИ

- Проаналізовано існуючі моделі та методи програного моделювання, направлені на вирішення аналогічних проблем
- Вибрані існуючі моделі, які найбільше відповідає завданню роботи. Моделі вдосконалені в бік опрацювання україномовних документів.
- Описана архітектура системи модулів, кожен з яких має свої принципи та зону відповідальності, що забезпечує гнучкість та розширюваність програмного продукту. Автономність модулів дає можливість перевикористання коду.
- В основі принципів проектування системи додатку використані широковживані технології та бібліотеки.
- Розроблено прототип веб-додатку, що дозволяє розпізнавати та вилучати дані з україномовних документів.