Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

Розробка сервісу для оцифрування документів методом глибинного навчання

Текстова частина до курсової роботи
за спеціальністю „Комп'ютерні науки" - 122

Керівник курсової роботи
Кандидат фізико-математичних наук,
доцент Ющенко Ю.О.

_____
(підпис)
"____" _____ 2022 р.

Виконав студент МП-1
Дубчак О. Б.
"____" _____ 2022 р.

Київ 2022

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Зав.кафедри інформатики,

Доцент., к. ф.-м. н. С.С.Гороховський

(підпис)

„____”_____2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Дубчаку Олександру Богдановичу факультету інформатики 1-го курсу магістратури

ТЕМА Розробка сервісу для оцифрування документів методом глибинного навчання

Зміст ТЧ до курсової роботи:

1. Індивідуальне завдання
2. Календарний план
3. Вступ
4. Дослідження проблеми розпізнавання документів
5. Опис розробки
6. Висновки
7. Список літератури

Дата видачі „___” _____ 2020 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Календарний план виконання роботи:

Тема: Розробка сервісу для оцифрування документів методом глибинного навчання

№	Назва етапу	Термін виконання	Примітка
1.	Отримання теми курсової	13.10.2021	
2.	Пошук тематичної наукової літератури	11.12.2021	
3.	Ознайомлення з науковою літературою	28.01.2022	
4.	Створення і тренування моделей	17.02.2022	
5.	Створення та шаблонів	23.02.2022	
6.	Побудова графу на основі шаблонів	11.03.2022	
7.	Побудова графу на основі зображення	02.04.2022	
8.	Реалізація порівняння графів, та отримання результатів	26.04.2022	
9.	Внесення змін до роботи	09.06.2022	
10.	Перевірка роботи на плагіат	10.06.2022	

Зміст

Вступ

Однією з найперших проблем машинного навчання була проблема розпізнавання тексту. Дійсно, часто виникає потреба перетворити текст на папері, зображенні або інших предметах у щось, з чим можна взаємодіяти за допомогою техніки та програм. Наприклад, широко розповсюдженні камери на дорогах, що фіксують порушників, використовують системи розпізнавання тексту для знаходження номеру на автомобілях. Не можна не згадати і звичайне оцифрування книг, завдяки якому у можна прочитати величезну їх кількість прямо з екрану комп'ютера. Актуальність систем розпізнавання тексту завжди залишається високою, адже для них завжди можна знайти застосування. У даній роботі описується система для розпізнавання тексту на документах, а також виокремлення із цих даних тільки тих, які потрібні користувачеві. У першому розділі розглядаються проблеми розпізнавання тексту і документів як таких. Тут описано історію виникнення систем, головні проблеми при оцифруванні, а також наведено ідею для вирішення однієї із цих проблем. У другому розділі описується запропонований підхід до розпізнавання документів. Тут можна дізнатись про те з якими проблемами довелось стикнутись при розробці проекту, як їх вдалось вирішити, детально описано алгоритм для розпізнавання документу за шаблоном, а також результати роботи готового продукту. Головними результатами роботи є готовий сервіс для оцифрування документів, який може приймати будь-які документи, та відправляти назад структурований набір тих даних, які цікавлять користувача.

1. Дослідження проблеми розпізнавання документів

Проблема розпізнавання документів з’явилась тоді, коли людям знадобилось зберігати текст у цифровому форматі. Зберігання у такому вигляді дозволяє його редагувати, копіювати та розповсюджувати набагато ефективніше та швидше, ніж у папері. Проте перетворення тексту є складною задачею, якою почали займатись ще у попередньому столітті, та продовжують тепер.

1.1.Історія розпізнавання документів

Одними з найперших пристроїв для розпізнавання тесту були створені ще у двадцятих роках минулого століття. Так, австрійський винахідник Густав Таушек у 1928 розробив перший пристрій для розпізнавання тексту. Ця машина використовувала промені світла, направлені на слова, для подальшого порівняння їх із шаблонами та друкування. У 1951 американським вченим Девідом Шепардом було розроблено машину для розпізнавання всіх букв латинського алфавіту, написаних на друкарській машинці. У 1974 було створено першу машину, яка була здатна розпізнавати букви різних шрифтів. У 1992 компанія Apple випустила планшет MessagePad, який міг розпізнавати рукописний текст англійською мовою завдяки порівнянню написаних слів та слів у базі даних пристрою. У 2005 році найточніше на той момент ПЗ Tessaract було випущено у загальний доступ з відкритим кодом. У подальшому підтримувалось і спонсорувалось компанією Google. Таким чином, можна прослідкувати, що людство вже давно було зацікавлене у способах читання тексту машинами, та цей інтерес не згас [1] і до сьогодні.

1.2.Проблема розпізнавання тексту

Розпізнавання тексту на фотографії є нетривіальною задачею, яка не є вирішеною до кінця і по сьогодні. Головними викликами з якими стикаються дослідники є велика варіативність написання одних і тих же букв. Фотографія тексту може бути повернута, обрізана, розмита. Сам текст може мати будь який шрифт, включаючи рукописний, він може розташовуватись будь-де на

фотографії, а також може бути написаний на різних мовах. Наразі існує багато підходів та засобів для оптичного розпізнавання тексту. Проте у комерційному використанні домінують наступні засоби:

- AWS Textract
- ExtractTable
- Tesseract
- Adobe Acrobat DC

Textract була розроблена компанією Amazon, та надає хмарні послуги розпізнавання, які вбудовані у їхню екосистему AWS. ExtractTable спеціалузється на структурованому тексті та надає схожі хмарні послуги з розпізнавання, проте вже як самодостатнє рішення. Tesseract це безкоштовне програмне забезпечення з відкритим кодом, яке підтримується компанією Google. Adobe Acrobat DC – це програма для перегляду PDF документів, яка також надає можливість розпізнавати текст. Усі ці програмні продукти мають свої переваги та недоліки, тому потрібно уважно аналізувати здатності кожної з них.

Наразі виділяють наступні фундаментальні проблеми у розпізнаванні тексту на зображеннях [2]:

- Локалізація тексту. Знаходження регіонів на зображенні, де ймовірно є текст з найменшою можливою площею фону.
- Верифікація тексту. Перевірка локалізованих ділянок на наявність в них тексту.
- Сегментація тексту. Включає в себе рядкову та символьну сегментацію. Рядкова сегментація розділяє ділянку з великою кількістю рядків у окремі ділянки з одним рядком у кожній. Символьна сегментація дозволяє розділити рядкові сегменти на регіони з окремими символами.
- Розпізнавання тексту. Перетворює ділянки з текстом на зображенні у звичайний текст, який зберігається у пам’яті комп’ютера як тип рядок(string).

- Обробка тексту засобами NLP(Natural language processing). Дозволяє обробити отриманий текст, та виправити в ньому помилки.

1.3.Проблема локалізації тексту

У багатьох випадках потрібно не лише розпізнати текст, а й знайти координати букв, слів, чи текстових блоків. Загалом для вирішення цієї проблеми вирізняють два підходи: класичні алгоритми машинного навчання та алгоритми глибинного навчання. [1] У першому випадку виділяють методи з ковзаючим вікном, згідно з яким, прямокутний регіон з фіксованою шириною та висотою «рухається» вздовж зображення з фіксованим кроком та на кожному кроці застосовується класифікатор, який визначає чи має задане вікно об'єкт; та методи зв’язаних компонентів, які виокремлюють ділянки з подібними властивостями(такими як колір, форма чи кути) для отримання кандидатів регіонів, які можуть бути текстом, після цього вони класифікуються за допомогою алгоритмів класифікації.

Для вирішення проблеми локалізації тексту методами глибинного навчання існують наступні підходи: координатно-регресійні(bounding box regression), сегментаційні, та гібридні. Координатно-регресійні методи працюють з текстом як зі звичайними об’єктами на зображенні, таким чином одразу отримуючи потрібні координати. Цей підхід вимагає тренування на великій кількості зображень, де вже позначені позиції текстових блоків. Проте недоліками таких підходів є проблеми з знаходженням мультиорієнтованого тексту, а також складнощі з групуванням текстових компонентів для отримання текстових рядків. Сегментаційні підходи намагаються семантично класифікувати текстові регіони на рівні пікселів. Ці методи зазвичай використовують повноз’єднані нейронні мережі для отримання текстових блоків, а потім отримують координати слів шляхом постпроцесингу. Гібридні методи використовують сегментаційні підходи для передбачення ймовірності появи тексту на зображенні, а також регресійні підходи для отримання координат тексту.

1.4.Проблема отримання потрібних даних з документу

Часто користувачам потрібно зберігати інформацію з документах у базі даних. У таких випадках зберігання всього оцифрованого тексту документу є нераціональним, оскільки він буде погано структурований, може займати багато місця, а також будуть складнощі з фільтрацією, та шуканими полями у тексті. Тому після оцифрування, текст потрібно обробити таким чином, щоб кожного разу отримувати ті дані, які потрібно, та в тому форматі, в якому їх можна буде зчитати. Для вирішення цієї проблеми можна окремо натренувати нейронну мережу по типу Yolo для розпізнавання лише потрібних полів у документів. Тоді кожне поле буде розпізнаватись як окремий клас, тому далі можна легко обробити результати нейронної мережі. Проте недоліком такого підходу є те, що для того, щоб додати якийсь новий тип документу для розпізнавання потрібно заново тренувати модель, а в деяких випадках навіть створювати нову. Отже такий підхід спрацює, якщо потрібно оцифровати лише один вид документу.

1.5.Запропонований підхід до вирішення проблеми

Для вирішення проблеми знаходження та розпізнавання тексту з документу було створено веб сервер, який приймає на вхід поля, які треба отримати в результаті, та зображення, з якого буде братись інформація. Такий підхід дозволяє виокремлювати саме потрібну частину інформації з документу, що зазвичай потрібно при оцифруванні документів. Система працює на двох методах локалізації та розпізнавання тексту – зв’язка моделі YOLO+tesseract та пропрієтарна розробка компанії Google – VisionAI. API серверу дозволяє обирати який з методів обирати користувачеві. Інформація про потрібні дані на зображенні береться за допомогою даних json формату, у якому вказуються потрібні поля, їх можливі значення, а також позиційні відношення між полями. Для порівняння даних із зображення та даних із шаблону будуються графи, за допомогою яких потрібні дані виокремлюються із графу зображення, та відправляються як відповідь у json форматі.

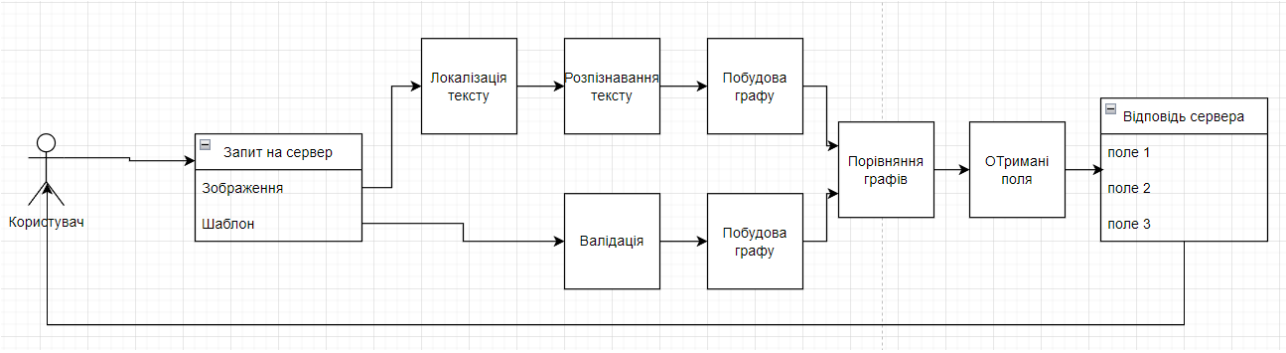

Рисунок 1 Діаграма підходу

2. Опис розробки

2.1.Структура проекту

Проект складається із трьох частин – локалізація та розпізнавання тексту, побудова графів залежності, та веб-сервер. Для локалізації тексту було використано модель YOLO [3], а також було зібрано та оброблено датасет розміром у 250 зображень документів різних видів. Розпізнавання тексту відбувається за допомогою програми Tesseract OCR [4]. Побудова та взаємодія з графами описана у подальших розділах. Веб сервер слугує як простий спосіб взаємодії з користувачем.

2.2.Локалізація текстових блоків

2.2.1. Модель «You look only once»

2.2.1.1. Опис моделі

YOLO [3] – це одна з найпопулярніших моделей глибокого навчання для розпізнавання об’єктів. Назва розшифровується як «You look only once», адже на відміну від попередніх підходів, Yolo не потребує перегляду зображення кілька разів, а з одного проходу дозволяє виявляти об’єкти. Завдяки тому, що вона виконує одночасне знаходження координат об’єктів, а також точність передбачення цих об’єктів, модель є гарно оптимізованою та дозволяє їх розпізнавати у реальному часі. Вже було випущено п’ять ітерацій Yolo, кожна з яких покращує результати попередньої. Так, на рисунку 2 зображено точність передостанньої версії, у якій видно що четверта версія значно випереджає третю в плані точності, а також є найкращим варіантом коли потрібно водночас швидкість розпізнавання, а також точність.

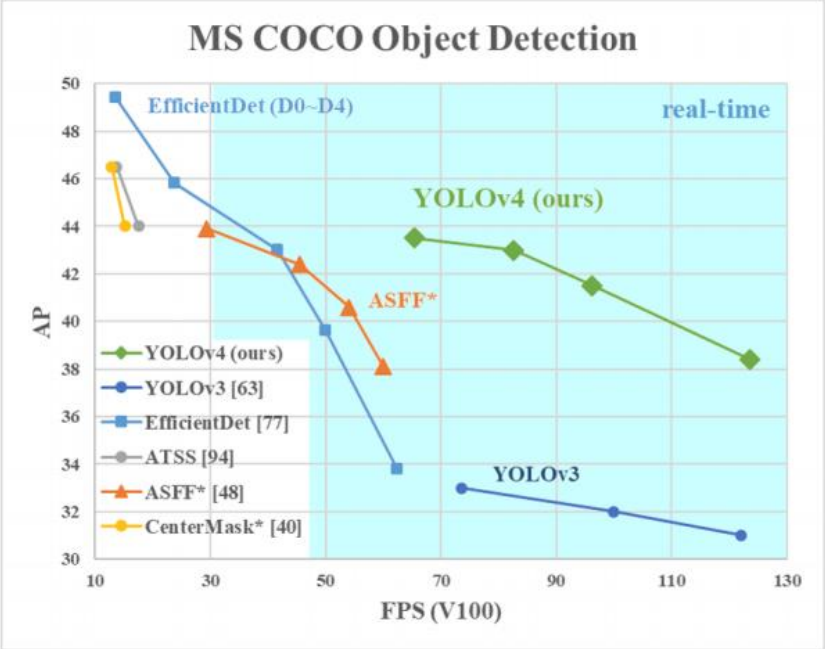

Рисунок 2
Точність YOLOv4 в порівнянні з іншими моделями для розпізнавання об'єктів на зображенні

Через вище описані переваги було обрано саме цю модель для локалізації тексту на зображенні.

2.2.1.2. Опис датасету

Для тренування моделі було власноруч зібрано датасет, який складається зі 250 зображень документів. Після цього кожне зображення було оброблене, промарковане, та підготовлене для подачі до нейронної мережі.

2.2.1.2.1. Збір даних

Фотографії документів було зібрано шляхом ручного пошуку доступних зображень за допомогою пошукової системи Google. Зображення поділяються на фотографії паспортів, водійських прав, закордонних паспортів різних країн, та інших видів документів. Загальний розмір датасету склав приблизно 250 зображень. Серед фотографій приблизно 32% складали закордонні паспорти, 20% паспорти України, 26% фотографії водійських посвідчень, 17% фотографії паспортів Росії, решту – інші документи. Всі зображення були великі за розміром, для того щоб мати змогу розпізнавати дрібний текст.

2.2.1.2.2. Маркування

Для того, щоб нейронна мережа могла розпізнавати об'єкти на зображенні, їй спочатку потрібно показати де саме ці об'єкти знаходяться. Для цього вже

існує велика кількість готових рішень, проте було обрано Visual Object Tagging Tool – застосунку від Microsoft, який поєднує в собі простоту і зручність у використанні, та швидкість роботи. На рисунку 3 можна побачити приклад маркування для паспорту. На зображенні було виділено окремі семантичні блоки, які репрезентували поля в документі.

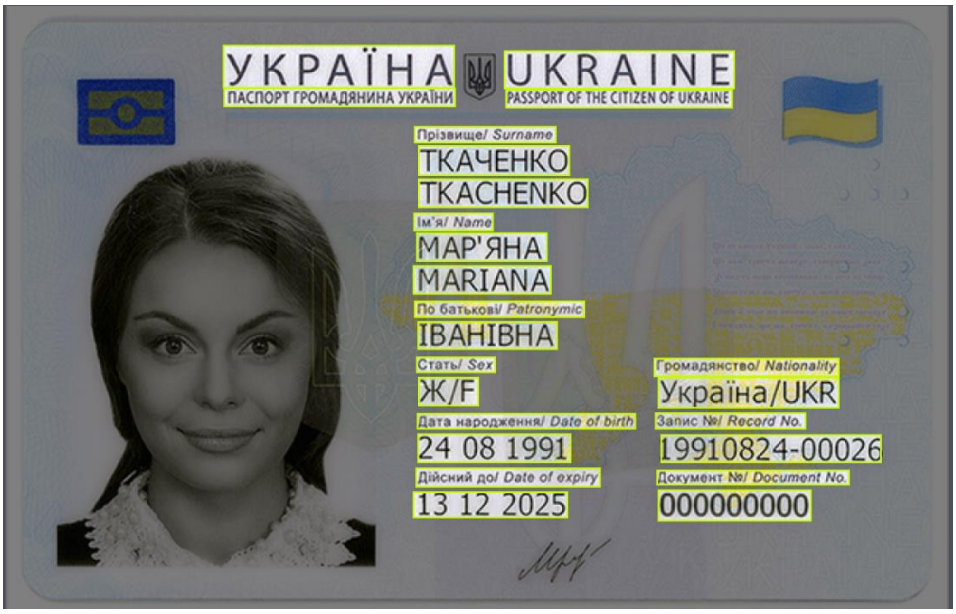

Рисунок 3Приклад маркування паспорту

2.2.1.2.3. Постпроцесинг датасету

Для того, щоб збільшити кількість зображень у датасеті та щоб покращити відсоток точності при розпізнавання, до датасету було застосовано методи препроцесингу зображень у датасеті. Для початку зображення автоорієнтували таким чином, щоб текст розташовувався горизонтально. Потім, для розпізнавання маленького тексту, зображення поділялось горизонтальною лінією навпіл, отримуючи з нього два окремих зображення. Після цього до кожного з цих зображень застосовувались методи аугментації даних, для збільшення розміру датасету. Для кожного з цих методів створюється окреме зображення із його застуванням. Спочатку для зображень було додано повороти на 90 градусів вліво та вправо, а також перевертання. Для кожного зображення також було додано випадкове значення повороту від -5 до 5 градусів. Потім для зображень було додано гаусове розмиття, та випадкові шуми. Застосування саме таких комбінацій постпроцесингу зображення надало найкращі результати

у виявленні тексту на фотографії. Після цього датасет поділявся наступним чином: 87% - тренувальний сет, 9% - валідаційний сет, 4% - тестовий сет.

2.2.1.3. Тренування моделі

Для тренування моделі Yolo було використано хмарний сервіс Google Colab. Він надає безкоштовні обчислювальні ресурси для тренування моделей машинного навчання. Навчання відбувалось з використанням вже натренованих на датасеті MS COCO коефіцієнтах на відеокарті Nvidia Tesla T4, та продовжувалось впродовж десяти годин. У результаті найкраща точність моделі складала 90.74%, що і зображено на рисунку

```
(next mAP calculation at 3500 iterations)

Tensor Cores are used. Last accuracy mAP@0.5 = 90.04 %, best = 90.74 %
3492: 14.435822, 12.028745 avg loss, 0.000100 rate, 10.036527 seconds, 167616 images, 1.198904 hours left
Loaded: 0.000036 seconds
```

Рисунок 4 Результат тренування моделі

2.2.1.4. Результати роботи моделі

Ціллю роботи даної моделі є пошук текстових блоків, які виокремлені від інших. І з цим завданням вона справляється. На рисунку 2 наведено приклад роботи моделі. Можна побачити, що слова, дати та елементи, які знаходяться поряд один з одним, але далеко від інших слів виокремлені окремо.

Рисунок 5 Результат роботи моделі Yolo

2.2.2. Модель Google Vision AI

Для варіативності та для порівняння було також використано модель від Google, Vision AI.

2.2.2.1. Опис моделі

Модель Google Vision AI розташована у хмарі та за допомогою API надає послуги з розпізнавання обличь, емоцій, об’єктів, тексту, та багато іншого. Розповсюджується у екосистемі Google cloud за додаткову плату, проте перші 1000 запитів на місяць є безкоштовними.

2.2.2.2. Результати роботи моделі

Модель точно розпізнає координати текстових ділянок. Приклади її роботи можна бачити на рисунку 3. Якщо порівнювати її із Yolo, то було розпізнано більше тексту. Проте структуризація тексту не зовсім відповідає потребам сервісу, тому після розпізнавання також потрібен етап постпроцесингу для об’єднання семантично однорідних текстових ділянок в одну. Наприклад, у паспорті блок «ім’я /Name» було розпізнано як три блоки «ім'я», «/», «Name» у подальшому вони мають об’єднатись в один, зберігаючи координати.

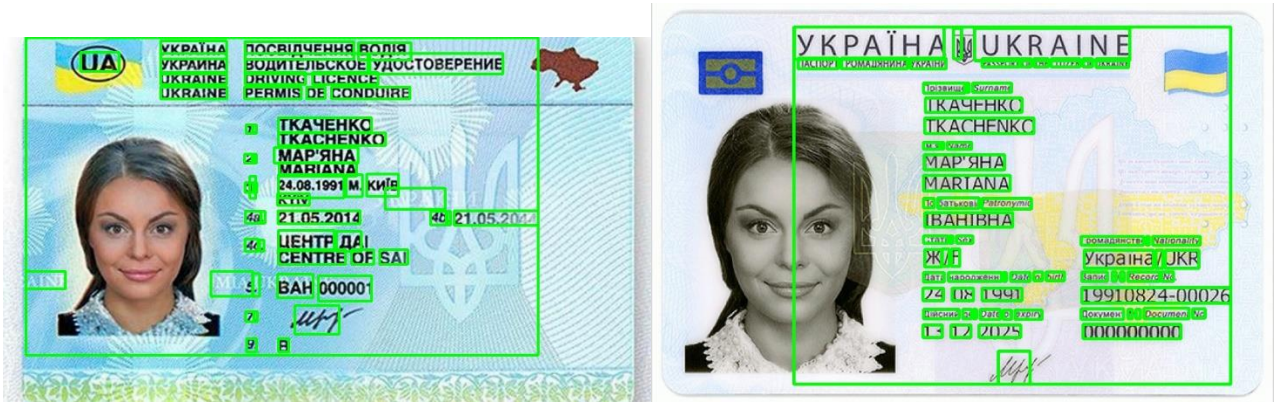

Рисунок 6 результати роботи моделі Google Vision AI

2.3.Оцифровування тексту

2.3.1. Tesseract

Для розпізнавання тексту було обрано Tesseract-OCR [4] – потужну OCR бібліотеку з відкритим кодом. Вона використовує підхід, який складається із двох кроків: спочатку йде пошук букв та символів на зображенні; далі для символів, у яких модель не впевнена, відбувається заміна символів, які могли б підходити за контекстом чи словом. Ця бібліотека може розпізнавати до 100 мов без додаткового тренування, проте, для уникнення помилок, бажано вказувати на якій саме мові написано документ.

2.4.Графове порівняння

Часто цільовому користувачеві не потрібно знати весь оцифрований текст, або йому потрібно отримати структуровані дані. У такому випадку постає проблема виокремлення потрібних даних із зображення.

2.4.1. Загальний опис підходу

Для того щоб вирішити проблему знаходження потрібних даних на зображенні було запропоновано використовувати графові репрезентації даних документів. Загальний підхід побудови графів полягає в тому, що часто на документах є незмінні поля, такі як наприклад «ім'я», «дата народження» чи інші. Ці поля можна використовувати, при виокремленні потрібних елементів тексту, адже можна точно знати, що під полем «ім'я» мусить бути дійсне ім'я власника паспорта. Таким чином, знаючи розташування деяких елементів на документі, можна побудувати граф залежностей полів, де кожен вузол має 8 зв'язків для кожного напрямку – верх, низ, ліво, право, верх-право, верх-ліво, низ-право, низ-ліво. На рисунку 5 зображено ілюстративний приклад побудованого графу для деяких полів паспорту громадянина України.

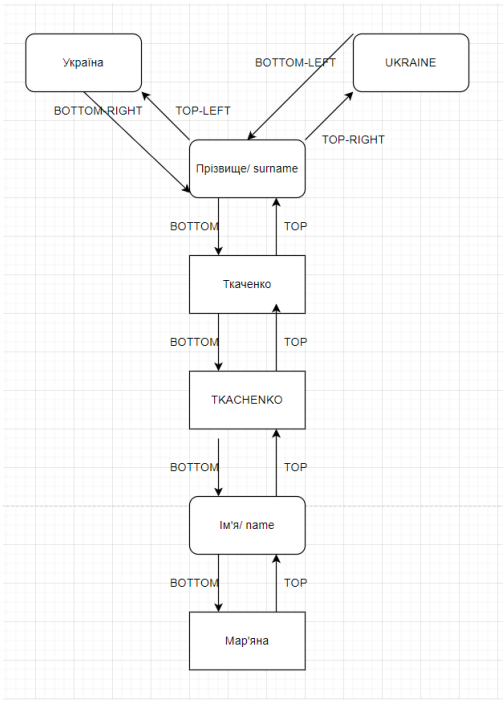

Рисунок 7Ілюстративний приклад графу залежностей між полями

2.4.2. Побудова графу на основі json шаблону

Для того, щоб вказати потрібні дані, користувач має надіслати їх у форматі json. Ці дані мають бути строго структурованими, інакше система не зможе побудувати граф на їх основі. У шаблоні вирізняють наступні типи об’єктів – якорі та значення. Якорі – це незмінні поля у документі, їхній текст залишається однаковим на будь-яких одинакових типах документів. Об’єкти значення ж це ті поля, які цікавлять користувача, та які потрібно розпізнати. У додатку 1 наведено простий приклад шаблону. Спочатку користувач має вказати назву документу, а також мови, які є присутні у документі. Далі він створює об’єкти, де вказує їхні властивості. Айді слугують для унікальної репрезентації кожного вузла графу. Назва вузла слугує як підказка до того, як вузол може називатись. У випадку полів-значень воно також використовується для побудови назв шуканих полів при створенні відповідді сервера. Тип може набувати значень anchor або value, та дозволяє вказувати користувачеві якорі та поля-значення. Для того, аби знайти потрібні вузли на зображенні використовується поле regex. У випадку якоря воно є незмінним та просто надає інформацію про те, який текст має бути у якорі; у випадку поля-значення воно показує те, яким може бути поле, та на що воно схоже. У полі position вказуються позиційні залежності між полями у документі. Для якорів це поле непотрібно заповнювати, адже на зображенні якорі знаходяться за допомогою їхнього тексту. Для полів-значень потрібно вказувати айді залежних об’єктів та їхнє місцезнаходження відносно поточного об’єкта.

Перед побудовою графу потрібно провести валідацію введеного користувачем шаблону. Перевіряється унікальність всіх айді, правильність введення позицій, правильність структури шаблону, а також наявність всіх полів.

Для побудови графу та взаємодії з ними було створено класи DocumentGraph, Node та JSONTemplate. Клас Node репрезентує звичайний вузол у графі, та методи для отримання потрібного сусіднього вузла за позицією, створення зв’язку з іншим вузлом, а також отримання списку сусідів за їх позиціями. Клас DocumentGraph зберігає вузли у форматі списку, надає можливість їх пошуку за айді, та можливість базової взаємодії з графом. Клас JSONTemplate відповідає за побудову графу на основі шаблону, а також за

порівняння з графом зображення. Щоб побудувати граф на основі шаблону, спочатку ітеративно створюють екземпляри класу Node із даними об'єктів шаблону, потім присвоюють позиційні залежності між полями. Оскільки граф є двозв'язним, то при присвоєнні сусідньому вузлу позиції, сусідній вузол створює залежність з даним вузлом з інвертованою позицією. Наприклад, якщо для поля-якоря присвоїти залежність BOTTOM-інше поле, то для цього поля також створиться залежність TOP-початковий якір.

2.4.3. Побудова графу на основі локалізацій тексту

Побудова графу на основі зображення вимагає отримання координат текстових блоків, а також розпізнавання тексту в них. Для абстракції текстових блоків було створено клас Rectangle, у якому зберігаються верхній лівий кут прямокутника, нижній правий, а також методи для роботи з геометрією прямокутників на зображенні. Оскільки нейронна мережа Yolo повертає прямокутники у вигляді точки у центрі, довжини і ширини прямокутники, було також створено підклас YoloRectangle, який наслідується від Rectangle, проте має інший конструктор. Аналогічно, Google Vision AI повертає чотири точки, які є чотирикутником, проте для того, щоб зробити з них прямокутник було створено підклас GVisionRectangle.

Перед тим, як розпочати розпізнавання тексту, має відбутись підготовка зображення. Спочатку зображення має бути вирівняно відносно горизонтальної осі. Це було досягнуто за допомогою Tesseract-OCR. Далі створюється дві копії зображення – одна для розпізнавання Yolo, інша для розпізнавання тексту. Для локалізації тексту ніяких додаткового редагування зображення вже не потрібно, тому його одразу передають у нейронну мережу. Отримавши результати, і відсіявши текстові блоки з малим відсотком впевненості, можна переходити до розпізнавання тексту. Для цього копію зображення для розпізнавання обробляють, щоб отримати вищу точність. Спочатку зображення перетворюють у чорнобіле. Далі, для прибирання шумів з зображення застосовують медіанне розмиття, яке заміняє всі пікселі навколо ядра медіанними значеннями, що допомагає зберігати контури тексту чіткими, та водночас позбутись частини шумів. Після цього зображення бінаризують за допомогою адаптивного порогу,

який ставить кожному пікселю значення 0 або 255 в залежності від значення цього порогу, причому він обчислюється окремо для кожних регіонів зображення. Після того як зображення було підготовано, обираються координати текстових блоків з попереднього кроку, та обрізаються у регіони зацікавленості з обробленого зображення. Далі Tesseract-OCR розпізнає текст, та виводить результати.

Отримавши координати текстових регіонів, та їх значення, можна розпочинати побудову графу залежностей полів для зображення. Для цього для кожного розпізнаного регіону, в якому присутній текст будується екземпляр класу YoloRectangle, у який передаються координати та текст. Після цього за допомогою класу RectangleDocumentGraph будується граф. Оскільки із зображення вже відомо всі можливі вузли, потрібно лише розставити між ними позиційні залежності. Вони присвоюються наступним чином:

1. Спочатку потрібно дізнатись попарні відстані між кожним прямокутником. Вони обраховуються як евклідові відстані між найближчими точками цих прямоктників, та збергіються у матриці розміром NxN, де N – це кількість регіонів.
2. Далі для кожного регіону потрібно знайти найближчих сусідів. Це робиться за допомогою знаходження сортованих індексів у підмножині матриці відстаней таким чином, щоб кожен елемент масиву був індексом елемента у матриці відстаней, де чим ближчий елемент до початку масиву, тим ближчий елемент за цим індексом до шуканого прямокутника.
3. Отримавши найближчих сусідів потрібно знайти відносне положення прямокутника відносно шуканого. Це відбувається за допомогою знаходження кута між горизонтальною віссю та вектором, який утворюється із двох лівих верхніх точок прямокутника. Для того, щоб знайти положення виходячи з кута було розділено 360 градусів на 8 рівних секторів як зображено на рисунку 6. Порівнявши отриманий кут з даними секторів можна легко визначити положення прямокутників відносно один одного.

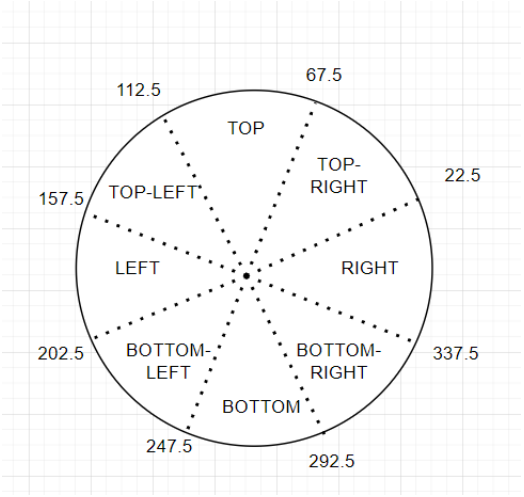

Рисунок 8 розподіл кутів між прямокутниками, та відповідне положення

4. Маючи положення прямокутників, залишається лише встановити відповідності між вузлами. Для цього у відповідний атрибут вузла зберігається посилання на інший вузол, а також аналогічно робиться з іншим вузлом, але вже з інвертованим положенням.
5. Для того, щоб посилання на вузли не переписувались декілька разів, кожної ітерації йде перевірка на наявність вузла у положенні. Таким чином кожен вузол графа буде зв’язаний лише з найближчими сусідами

2.4.4. Отримання інформації з графу за шаблоном

Основною метою сервісу є отримання потрібної та структурованої інформації з зображення. Оскільки вже було побудовано графи для шаблону та зображення, можна розпочинати процес виокремлення даних.

2.4.4.1. Пошук якорів

Порівняння графів розпочинається із пошуку якорів на графі зображення. Ці поля є незмінними для шаблону та зображення, тому вони мають бути розпізнаними з самого початку. Для початку потрібно знайти всі якорі, присутні у графі шаблону. Це робиться простою фільтрацією всіх вузлів з подальшим порівнянням їх типів. Отримавши всі якорі на шаблоні, потрібно знайти їм відповідники на зображенні. Оскільки з шаблону відомо текст, який має бути присутнім на цих вузлах, можна скористатись відстанню Левенштейна. Для знаходження ймовірних кандидатів якорів на зображенні будуєтьсья матриця

відстаней розміром NxM, де N це кількість якорів у шаблоні, а M – це кількість вузлів у графі зображення. Відстані обчислюються наступним чином: якщо відношення відстані левенштейна та максимальної довжини рядка більше за заданий поріг – то вважаємо що елементи не є подібними. Після отримання матриці відстаней можна для кожного якоря з шаблону співставити вузли зображення з найменшою відстанню між ними. Таким чином, отримавши якорі на зображенні, можна переходити до наступного кроку та почати розпізнавати значення.

2.4.4.2. Виокремлення значень

Значення отримують шляхом співставлення вже відомих взаємно відповідних вузлів на шаблоні та зображенні. Для цього для кожного відомого елемента у графі зображень потрібно знайти його відповідник у графі зображень. Отримавши їх, потрібно виокремити всі залежності цього вузла з графу шаблону. Потім для кожної такої залежності потрібно знайти відповідну з вузла графу зображення. Наприклад, якщо у якоря шаблону «ім’я» є залежність BOTTOM – «name_ukr», тоді у відповідного вузла якоря фотографії знаходиться залежність BOTTOM, яка виявляється вузлом «Мар’яна». Таким чином, знайшовши відповідний за позицією елемент графу зображення, його порівнюють на відповідність згідно з регулярним виразом, і якщо її було знайдено – то елемент додається як розпізнаний. Після знаходження нових розпізнаних елементів на графі зображення, аналогічний пошук залежностей повторюється.

2.4.4.3. Обробка значень

Знайшовши взаємно відповідні вузли на фотографії, можна готувати результат роботи програми. Для цього можна додати семантичну обробку тексту за допомогою NLP, заміну латинських букв на кирилицю та навпаки, або інші способи покращити результат тексту. Після цього готується словник із розпізнаними полями, де ключ – назва вузла у графі шаблону, а значення – це текстове значення відповідного вузла у графі зображення.

2.5.Приклади роботи

Після оцифрування паспорт з рисунку 5, та з шаблоном, наведеним у додатку, було отримано наступні дані:

{
'surname_ukr': 'ТКАЧЕНКО',
'name_ukr': "МАР'ЯНА",
'sex_value': 'Ж/F',
'expires_in_value': '13 12 2025',
'nationality_value': 'Україна /(ШКЕ',
'document_number_value': '000000000'
}

Використавши той самий шаблон, але вже фотографію власного паспорта було отримано такі дані:

{
'surname_ukr': 'ДУБЧАК',
'surname_eng': 'DUBCHAK',
'name_ukr': 'ОЛЕКСАНДР',
'date_birth_value': '07 02 2000',
'expires_in_value': '17 03 2026',
'name_eng': 'OLEKSANDR'
}

Можна побачити, що вдалось отримати не всі дані з зображення. Це може бути пов’язано з тим, що було неправильно вказано регулярний вираз, неправильно вказані залежності між полями, або просто не вдалось цифрувати текст у тих регіонах. Таким чином система намагається розпізнати якнайбільше полів, та повертає всі знайдені. Тому структура відповіді однакових типів документів може відрізнятись, адже не всі поля вдається віднайти на зображенні.

2.6. Подальші способи покращення роботи системи

Для покращення роботи системи потрібно орієнтуватись на три важливі аспекти – локалізація, розпізнавання, та порівняння. Для оптимізації локалізації

тексту потрібно покращити роботу нейронної мережі. Можна спробувати інші види моделей, або покращити датасет вже існуючої. Це можна зробити, збільшивши кількість фотографій документів, збільшивши кількість видів документів, та додавши більше шарів аугментації та постобробки зображень у датасеті. Покращення розпізнавання тексту можна добитися шляхом дотреновування моделі Tesseract OCR на фотографіях з датасету. Це допоможе моделі краще вирізняти та розпізнавати текст на документах.

Висновки

Результатом роботи є програмний продукт, що може розпізнавати дані на зображенні згідно із шаблоном, у якому вказані потрібні для користувача поля, а також зв’язки між ними. Під час його написання було отримано значний досвід у роботі нейронних мереж, що розпізнають об’єкти на зображенні, у роботі систем, що розпізнають текст на зображенні, а також досвід проектування та взаємодії з графовими структурами даних.

Список літератури

[1] Raisi, Zobeir & Naiel, Mohamed & Fieguth, Paul & Wardell, Steven & Zelek, John., «Text Detection and Recognition in the Wild: A Review.,» 2020.

[2] Xiaoxue Chen, Lianwen Jin, Yuanzhi Zhu, Canjie Luo, Tianwei Wang, «Text Recognition in the Wild: A Survey,» *Association for Computing Machinery*, 2020.

[3] J. Redmon, S. Divvala, R. Girshick та A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection, 2016.

[4] «Tesseract OCR,» [Онлайновий]. Available: https://tesseract-ocr.github.io.

Додатки

1. Простий приклад json шаблону

```
{
  "document": "citizen_passport",
  "language": ["ukr", "rus", "eng"],
  "fields": [
    {
      "id": 1,
      "name": "surname",
      "type": "anchor",
      "regex": "прізвище/ surname",
      "position": []
    },
    {
      "id": 2,
      "name": "surname_ukr",
      "type": "value",
      "regex": "[а-яА-Я''ґєїҐЄЇъЪ-]+",
      "position": [
        {
          "dep_id": 1,
          "pos": "TOP"
        }
      ]
    }
}
```