Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Центр Післядипломної освіти
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Конвертація сканованих документів до формату PDF з використанням методів OCR
(тема)

Виконав:
студент 2 курсу, групи СШІздм-21-1
Ляубе О. О.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник к.т.н. доц. каф. ШІ Шевченко О.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Центр Післядипломної освіти
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Ляубе Олександрі Олексіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Конвертація сканованих документів до формату PDF з використанням методів OCR

затверджена наказом університету від 31 березня 20 23 р. № 73Стз

2. Термін подання студентом роботи до екзаменаційної комісії 23 травня 20 23 р.

3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел щодо методів конвертації сканованих документів до формату PDF з використанням методів OCR

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної області
2) Аналіз методів розпізнавання тексту
3) Обґрунтування обраних технологій
4) Розробка тематичного додатку
5) Експерименти по вдосконаленню

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	03.04.2023	Виконано
2	Аналіз предметної області	04.04.2023-10.04.2023	Виконано
3	Постановка завдання та узгодження з керівником	11.04.2023-12.04.2023	Виконано
4	Аналіз методів розпізнавання тексту	12.04.2023-18.04.2023	Виконано
5	Дослідження технологій	14.04.2023-18.04.2023	Виконано
6	Написання програмного додатку	19.04.2023-30.04.2023	Виконано
7	Експериментальні дослідження	30.04.2023-05.05.2023	Виконано
8	Написання пояснювальної записки	05.05.2023-08.05.2023	Виконано
9	Попередній захист		
10	Захист перед ЕК		

Дата видачі завдання 3 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. каф. ШІ Шевченко О.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 72 с., 13 рис., 1 дод., 17 джерел.

КОНВЕРТАЦІЯ ЗОБРАЖЕНЬ, МЕТОДИ OCR, НЕЙРОННІ МЕРЕЖІ, РОЗПІЗНАВАННЯ ТЕКСТУ, LATEX.

Об'єкт дослідження – алгоритми та методи екстракції тексту із зображень зі спотвореннями.

Предмет дослідження – методи покращення алгоритмів екстракції тексту із зображень та генерації PDF документів.

Мета роботи – використання вдосконалених методів розпізнавання тексту для оцифровування сканованих документів.

Методи дослідження – аналіз існуючих алгоритмів екстракції тексту із зображень, розробка алгоритму алгоритмів екстракції тексту із зображень та генерації PDF документів з отриманого тексту, програмна реалізація та проведення комп’ютерного експерименту, обробка та аналіз отриманих результатів. Під час виконання роботи проведений теоретичний аналіз літературних джерел щодо методів використання різних методів екстракції тексту із зображень зі спотвореннями.

ABSTRACT

Explanatory note: 72 p., 13 fig., 1 ann., 17 sourses.

IMAGE CONVERSION, LATEX, NEURAL NETWORKS, OCR METHODS, TEXT RECOGNITION,.

The object of research is algorithms and methods of extracting text from images with distortions.

The subject of research is methods of improving algorithms for extracting text from images and generating PDF documents.

The purpose of the work is to use advanced text recognition methods for digitizing scanned documents.

Research methods – analysis of existing algorithms for extracting text from images, development of an algorithm for extracting text from images and generating PDF documents from the received text, software implementation and computer experiment, processing and analysis of the obtained results. In the course of the work, a theoretical analysis of literary sources was carried out regarding the methods of using various methods of extracting text from images with distortions.

ЗМІСТ

ВСТУП

За останні кілька років, технологічний прогрес значно полегшив процес обробки і збереження документів. Однак, в той же час, він створив нові виклики для підприємств, які мають стикатися зі збільшенням кількості сканованих документів, а також з проблемами старих архівних документів. Проблема застарілого документообігу пов'язана з великою кількістю документів, які потрібно зберігати та обробляти. Старі документи можуть бути використані як важливі джерела інформації для прийняття рішень, а також як доказові матеріали у випадку юридичних справ. Проте, у випадку старих документів, які не були оцифровані, їх обробка та зберігання можуть стати важким завданням.

Оцифрування старих документів може викликати труднощі через різноманітність форматів документів, які можуть бути збережені на паперових носіях. Ці формати можуть відрізнятися за своєю якістю та зчитуваністю, що може затруднити процес оцифрування. Проте, застосування технологій оцифрування може зробити процес більш ефективним та продуктивним. Оцифрування старих документів може бути виконано шляхом сканування документів у відповідних форматах та подальшого застосування методів OCR для їх конвертації у формат PDF або інший електронний формат. Застосування сучасних технологій, таких як OCR, машинне навчання та штучний інтелект, може забезпечити ефективне та якісне оцифрування документів, зменшити витрати на їх обробку та покращити якість результатів. ШІ може бути застосований для оптимізації процесу обробки документів. Він може бути використаний для автоматизації процесу розпізнавання символів, що дозволяє значно зменшити час, необхідний для конвертації сканованих документів до формату PDF. ШІ також може допомогти відшукати та класифікувати документи, збережені на різних пристроях та мережах. Це забезпечує збільшення ефективності та зниження витрат, пов'язаних з обробкою

документів. Технології OCR та ШІ знаходяться у постійному розвитку, завдяки чому процес обробки та зберігання документів стає все швидшим та точнішим. На сьогоднішній день існує значна кількість різноманітних програмних засобів для OCR та ШІ, які можуть бути використані для оптимізації процесу обробки документів та забезпечення їх ефективного зберігання. Конвертація сканованих документів до формату PDF з використанням методів OCR – це важлива технологія, яка дозволяє забезпечити зручний доступ до документів у майбутньому.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Задача розпізнавання тексту в комп’ютерному зорі

Розпізнавання тексту (OCR) [1] – це процес перетворення сканованого або фотографованого документу в текстовий формат, який можна редагувати та обробляти на комп'ютері. Задача OCR полягає в тому, щоб зчитати кожен символ зображення та перетворити його у відповідний символ у текстовому форматі.

Задача OCR є дуже важливою у багатьох сферах, таких як бізнес, медицина, право, дослідження та інші. Використання OCR дозволяє зменшити час та витрати на обробку документів, забезпечити швидкий доступ до інформації та покращити точність обробки. Для розв'язання задачі OCR використовуються різні методи та технології комп'ютерного зору. Одним з найпоширеніших методів є метод зіставлення шаблонів, при якому зчитуються символи зображення та порівнюються зі зразками символів. Інші методи використовують нейронні мережі та машинне навчання для розпізнавання символів на зображенні.

Проте, задача OCR все ще має свої виклики та обмеження [2]. Одним з них є проблема розпізнавання рукописного тексту та тексту, написаного нестандартними шрифтами або на нестандартних мовах. Крім того, OCR може бути неефективним у випадку, якщо зображення містить шум, використання низької якості сканера або фотоапарата (рисунок 1.1). Для оптимізації процесу OCR можуть використовуватися різні підходи, такі як підготовка зображення, яка полягає в видаленні шуму та підсиленні контрасту, та оптимізація алгоритмів розпізнавання символів. Також досить часто використовують різні трансформації зображень, наприклад для вирівнювання перспективи на зображеннях, що були спотворені в результаті неякісного сканування.

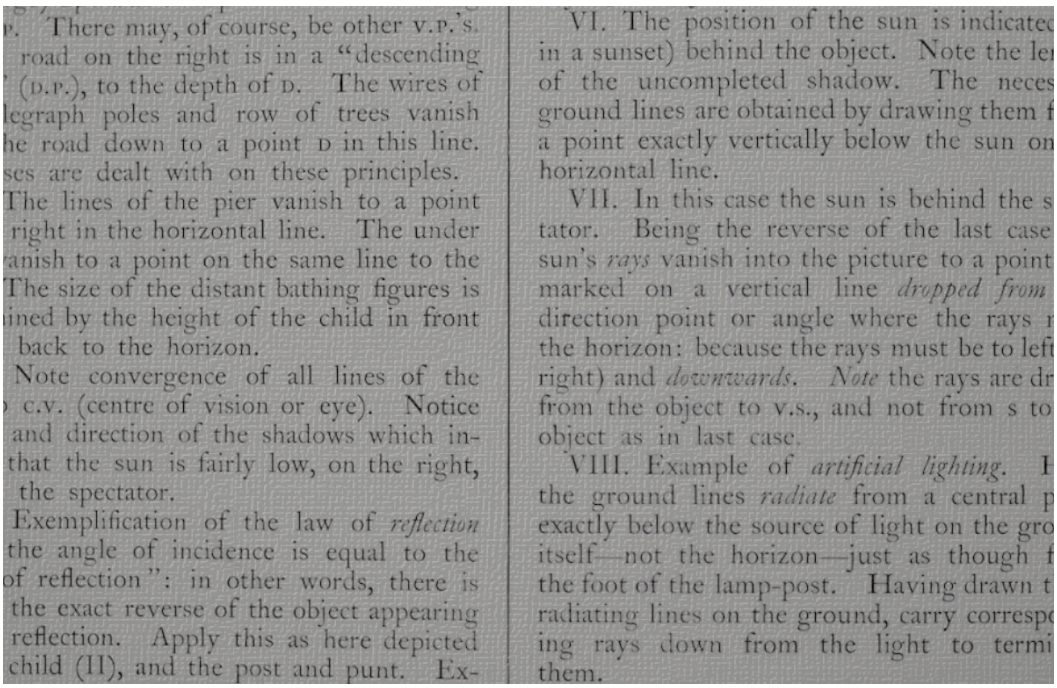


Рисунок 1.1 – Приклад фото з шумом і поганою контрасністю

1.2 Проблема розпізнавання рукописного тексту

Проблема розпізнавання рукописного тексту є однією з найбільш складних задач в галузі обробки природної мови та комп'ютерного зору. Незважаючи на значний прогрес в цій галузі, розпізнавання рукописного тексту залишається складним завданням через складність самого процесу.

Одна з основних причин, чому розпізнавання рукописного тексту є складною задачею, полягає у тому, що рукописний текст може бути написаний в різних стилях, розмірах та форматах, що робить його складним для аналізу та класифікації. Такі відмінності у почерку можуть впливати на точність та надійність процесу розпізнавання, особливо при використанні автоматизованих систем та програм. Тому, для досягнення оптимальних результатів у розпізнаванні почерку, необхідно враховувати індивідуальні особливості кожної людини та розробляти адаптивні алгоритми, що враховують широкий спектр можливих варіацій у почерку.Більшість людей має власний унікальний почерк, який може дуже відрізнятися від інших, і

це ускладнює процес розпізнавання рукописного тексту для комп'ютерних систем (рисунок 1.2).

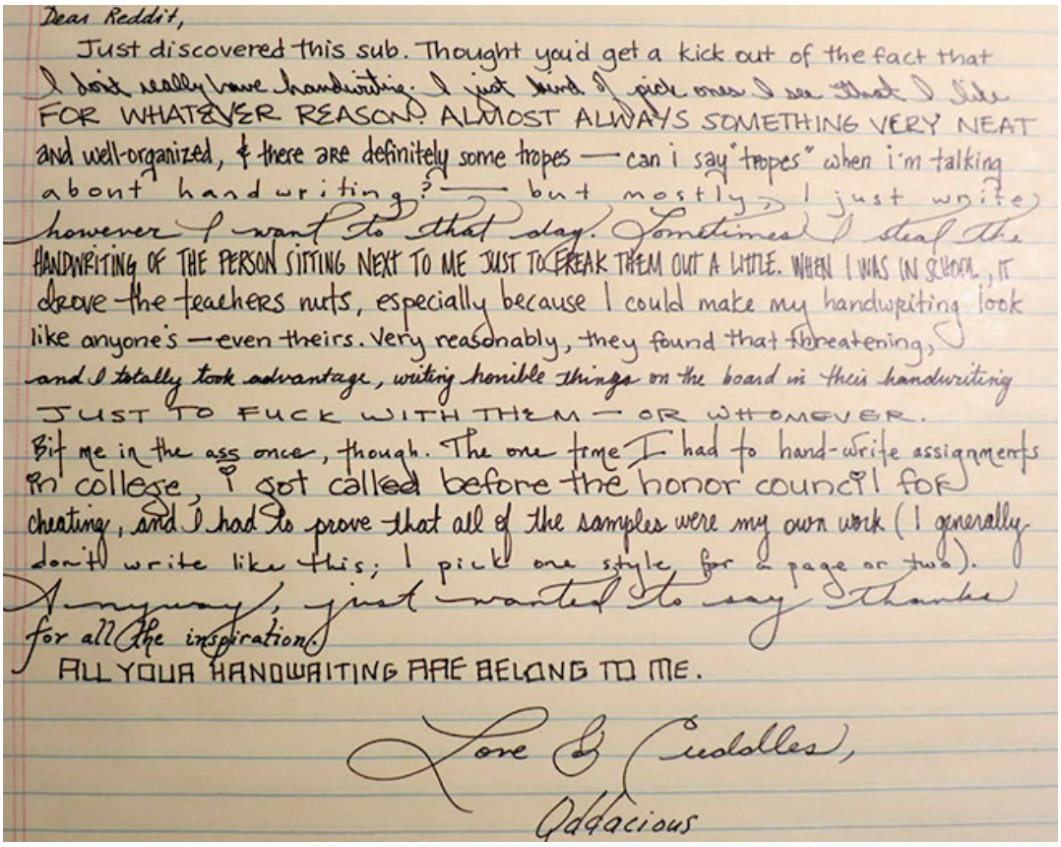


Рисунок 1.2 – Приклад фото різних почерків

Інші проблеми, які виникають при розпізнаванні рукописного тексту, включають шум та низьку контрастність, помилки у написанні та нетипові форми літер (рисунок 1.3). Ці проблеми можуть призвести до неправильного розпізнавання символів та слів, що може значно підірвати точність результатів. Також проблемою є складність аналізу контексту та зв'язків між словами та реченнями. Наприклад, деякі слова можуть мати декілька значень залежно від контексту, в якому вони вживаються.

Щоб вирішити ці проблеми, використовуються різні методи та алгоритми, такі як методи штучного інтелекту, нейронні мережі та машинне навчання. Одним з найбільш популярних підходів є використання нейронних мереж. Нейронні мережі можуть бути навчені розпізнавати

рукописний текст за допомогою великої кількості зразків рукописного письма. Для цього, спочатку, зображення рукописного тексту перетворюється на векторний формат, який може бути використаний як вхід для нейронної мережі. Після тренування нейронна мережа зможе розпізнавати рукописний текст на зображеннях.

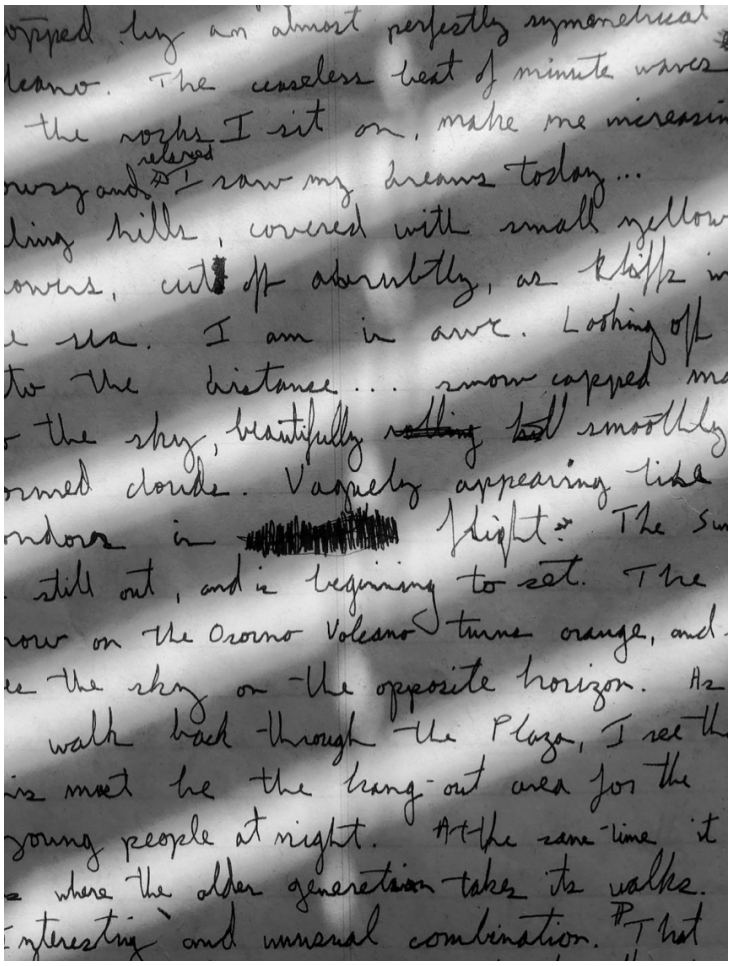


Рисунок 1.3 – Приклад нетипових форм літер

Окрім нейронних мереж, існують інші методи розпізнавання рукописного тексту, такі як шаблонний підхід та використання складових моделей. Шаблонний підхід використовує зразки рукописних символів та порівнює їх зі зображенням, що розпізнається. Використання складових моделей передбачає розбиття символів на складові частини, які можуть бути розпізнані окремо та зібрані разом для отримання повного символу.

Проте, незважаючи на ці досягнення, проблема розпізнавання рукописного тексту залишається відкритою і вимагає подальших досліджень та розробок, зокрема у зв'язку зі зростанням потреб у автоматизованому аналізі рукописних даних у різних сферах діяльності.

1.3 Формат даних PDF

Формат PDF (Portable Document Format) [3] є одним з найбільш популярних форматів електронних документів, цей формат файлу було створено в 1992 році компанією Adobe з метою створення документів, які можуть бути відкриті на будь-якому пристрої та зберігають оригінальний вигляд і форматування. З того часу PDF став визнаним стандартом у багатьох галузях, включаючи веб, друк та електронні книги.

Однією з найбільших переваг формату PDF є те, що він забезпечує збереження документів в оригінальному вигляді, без втрати форматування та без змін розміщення тексту та графіки на сторінках. Це дозволяє документам залишатися легко читабельними та зберігати свої властивості на будь-якому пристрої та програмному забезпеченні (рисунок 1.4). Крім того, PDF підтримує безпечне зберігання та передачу документів, включаючи захист від несанкціонованого доступу, редагування та друку. Файли PDF можуть бути зашифровані паролем, що забезпечує їх конфіденційність. Формат PDF також підтримує вбудовані гіперпосилання, відмітки, сторінкові замітки та інші інтерактивні елементи, що дозволяють користувачам ефективно взаємодіяти з документами. Також PDF дозволяє додавати метадані для кращої організації та керування документами. Важливою перевагою також є те, що PDF формат підтримується на більшості пристроїв та програмних забезпечень, включаючи операційні системи, браузери та програми для перегляду документів, тобто він є універсальним.

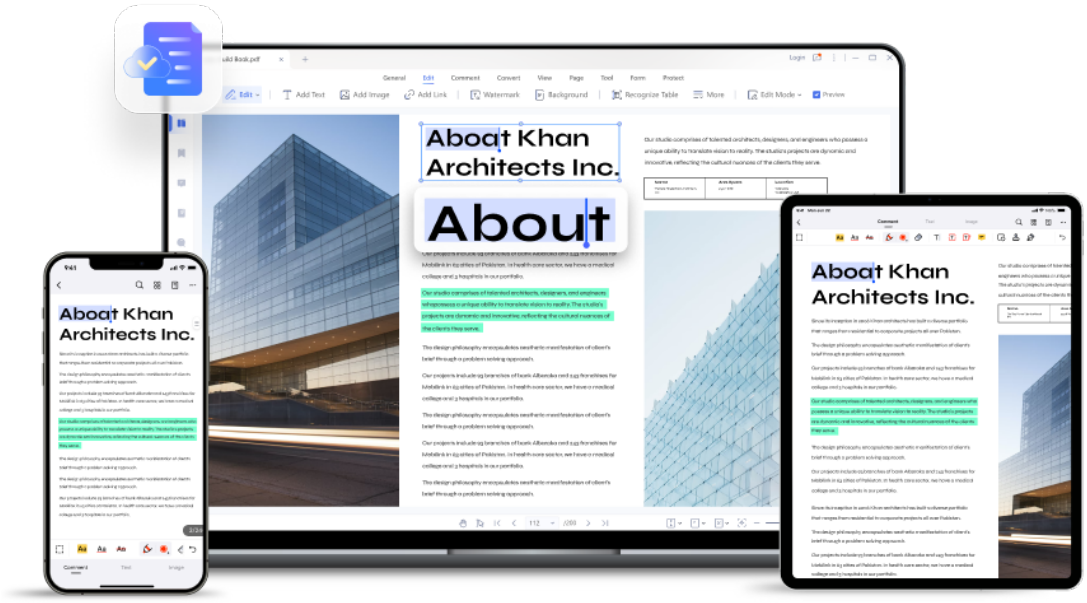


Рисунок 1.4 – Відображення PDF документу на різних пристроях

Незважаючи на багато переваг, формат PDF також має свої недоліки. Один з них – складність редагування. Оскільки PDF-файли зберігаються в форматі, який забезпечує збереження оригінального вигляду документу, редагування їх може бути складним. Крім того, виконання деяких операцій з PDF-файлами, таких як зміна макету або форматування тексту, може бути часом вимагає додаткових програм або ручної роботи, що займає багато часу та зусиль. Крім того, існують проблеми з читабельністю деяких PDF-файлів на мобільних пристроях, так як їх розмір може бути занадто великим для зручного читання на екрані смартфона або планшета.

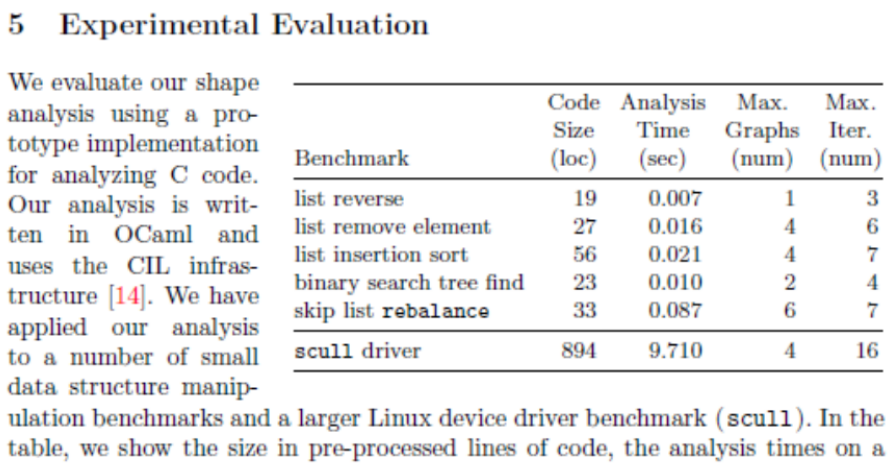
Отже, незважаючи на ці недоліки, формат PDF залишається одним з найбільш популярних форматів електронних документів. Його популярність полягає в тому, що він забезпечує стабільність форматування документу та підтримує безпеку даних, дозволяє створювати інтерактивні документи та додавати до них різноманітні мультимедійні елементи. Отже, формат PDF може використовуватись для різних цілей, в залежності від потреб користувачів. Наприклад, він ідеально підходить для електронної пошти, де важливо зберегти форматування документа і зробити його

доступним для перегляду на різних пристроях. Також PDF зручний для збереження та розповсюдження різних видів матеріалів, від документів та звітів до буклетів та презентацій. Крім того, цей формат є популярним серед видавництв, які використовують його для електронних книг та журналів. Усі ці цілі можуть бути досягнуті завдяки перевагам формату PDF.

1.4 Проблема розпізнавання тексту в сканованих документах

Проблема розпізнавання тексту в сканованих документах виникає при спробі перетворити фізичний документ в електронний формат. Незважаючи на технологічний прогрес в галузі оптичного розпізнавання символів, якість розпізнавання може залежати від різних факторів, таких як якість сканування, стиль написання тексту, наявність забруднень або пошкоджень на документі. Однією з основних проблем є розпізнавання рукописного тексту, оскільки він може містити неправильні форми літер, що робить його важким для оптичного розпізнавання. Крім того, складніші елементи, такі як таблиці, графіки та малюнки, можуть також ускладнювати процес розпізнавання (рисунок 1.5). Наприклад, таблиці мають складну структуру з рядками, стовпцями і комірками. Інформація в таблицях може бути організована в різні способи, що ускладнює процес розпізнавання. Особливо важко розпізнати таблиці з візуально складним форматуванням, об'єднаними комірками, зміщеними або обрізаними даними. Проблема розпізнавання тексту є особливо важливою в сфері бізнесу та юридичних послуг, де документація є ключовим елементом роботи. Неправильне розпізнавання тексту в сканованих документах може призвести до помилок, які можуть вплинути на ділові процеси та правові питання. Для розв’язання проблеми розпізнавання тексту в сканованих документах використовуються різні методи, включаючи використання програмного забезпечення OCR та машинного навчання. Однак, незважаючи на це, деякі документи можуть

вимагати ручної перевірки та корекції для досягнення найвищої якості розпізнавання.



5 Experimental Evaluation

We evaluate our shape analysis using a prototype implementation for analyzing C code. Our analysis is written in OCaml and uses the CIL infrastructure [14]. We have applied our analysis to a number of small data structure manipulation benchmarks and a larger Linux device driver benchmark (`scull`). In the table, we show the size in pre-processed lines of code, the analysis times on a

Benchmark	Code Size (loc)	Analysis Time (sec)	Max. Graphs (num)	Max. Iter. (num)
list reverse	19	0.007	1	3
list remove element	27	0.016	4	6
list insertion sort	56	0.021	4	7
binary search tree find	23	0.010	2	4
skip list `rebalance`	33	0.087	6	7
`scull` driver	894	9.710	4	16

Рисунок 1.5 – Текст з таблицею

Ще однією причиною виникнення проблем з точністю розпізнавання тексту являє собою низька якість документу. Це може включати в себе нерівномірне освітлення, затемнення, пошкодження документу або недостатню роздільну здатність сканера, а також низький контраст між текстом та фоном. Низька точність може призводити до помилок у розпізнаванні тексту, що може впливати на якість даних та документів. Також проблемою може стати розпізнавання мов. Вона полягає в тому, що сканований документ може містити тексти різних мов, але програмне забезпечення для розпізнавання тексту може бути обмеженим в розпізнаванні певних мов або відсутність підтримки декількох мов взагалі (рисунок 1.6). Крім того, багатомовний текст може містити проблеми зі синтаксисом, так як кожна мова має свої власні правила граматики, орфографії та пунктуації. Наприклад, якщо в документі міститься кілька мов, програмне забезпечення може стикається з проблемами визначення мови окремих частин тексту, а також зі змішуванням мов у одному реченні. Такі складнощі можуть призводити до

помилок в розпізнаванні та потребувати додаткового внесення корективів вручну.

Рисунок 1.6 – Приклад багатомовного документа

Для розпізнавання тексту на різних мовах існують спеціальні програмні засоби, які підтримують багато мов та використовують векторні моделі машинного навчання. Вони можуть автоматично визначати мову тексту та застосовувати відповідні алгоритми для розпізнавання. Навіть з використанням таких програмних засобів, розпізнавання мов в сканованих документах може бути складною проблемою. Крім того, надійність та точність розпізнавання тексту можуть залежати від наявності відповідних шрифтів та кодувань на комп'ютері. Якщо шрифт, використаний у документі, не має відповідного відображення на комп'ютері, то програма розпізнавання тексту може неправильно розпізнати текст. Розпізнавання тексту в сканованих документах може займати багато часу та ресурсу, особливо якщо документ містить велику кількість сторінок або складне форматування з графічними елементами.

Отже, незважаючи на ці проблеми, технології розпізнавання тексту з кожним роком стають все більш точними та надійними, бо розпізнавання тексту в сканованих документах є важливою задачею у багатьох галузях,

включаючи бізнес, медицину та правоохоронні органи. Використання спеціальних програм для розпізнавання тексту може значно полегшити та прискорити роботу з документами, а також зменшити кількість помилок у розпізнаванні тексту.

1.5 Постановка задачі

Дослідження має на меті вивчення та розробку методів конвертації сканованих документів до формату PDF з використанням технології оптичного розпізнавання символів (OCR). Основною задачею дослідження є покращення точності та ефективності процесу конвертації, забезпечення збереження оригінальної структури та форматування документів.

Дослідницька робота передбачає аналіз актуальних методів OCR та їх застосування в процесі конвертації сканованих документів до формату PDF. Окрема увага приділяється вдосконаленню алгоритмів розпізнавання символів, враховуючи особливості різних типів документів та можливих перешкод, таких як нерівномірне освітлення, шуми, спотворення тощо.

Також метою дослідження є розробка та вдосконалення алгоритмів автоматичного виявлення та виправлення помилок, що можуть виникнути під час процесу OCR. Це включає в себе виявлення та корекцію правопису, форматування тексту, відновлення пропущених або пошкоджених символів.

Крім того, в рамках дослідження будуть проведені порівняльний аналіз різних методів OCR, оцінка їх ефективності та вибір найбільш оптимальних підходів до конвертації сканованих документів до формату PDF. В результаті дослідження планується створення програмного рішення, яке забезпечить високу точність та надійність процесу конвертації, спрощення роботи з електронними документами та покращення продуктивності роботи з ними.

2 АНАЛІЗ МЕТОДІВ РОЗПІЗНАВАННЯ ТЕКСТУ

2.1 Існуючі підходи для розпізнавання тексту

Розпізнавання тексту є важливим етапом в багатьох сферах, включаючи наукові дослідження, бізнес та адміністративну діяльність. За останні роки було розроблено безліч інструментів для автоматичного розпізнавання тексту, кожен з яких має свої переваги та недоліки. В цьому розділі розглянуто кілька з найпопулярніших сервісів для розпізнавання тексту та проведено порівняльний аналіз їх можливостей. В цьому дослідженні розглянуто такі сервіси, як Tesseract OCR, Google Vision, ABBYY OCR, MMOCR та TrOCR. Існує багато підходів для розпізнавання тексту, які використовують різні технології та алгоритми для досягнення найкращих результатів. Найбільш поширеним з підходів є використання методів машинного навчання, які використовуються для створення моделей розпізнавання тексту. Інші підходи включають в себе методи, які базуються на використанні правил та шаблонів для визначення текстової інформації, а також методи, які комбінують у собі як машинне навчання, так і правила. Нижче наведено загальну схему алгоритму роботи OCR (рисунок 2.1).

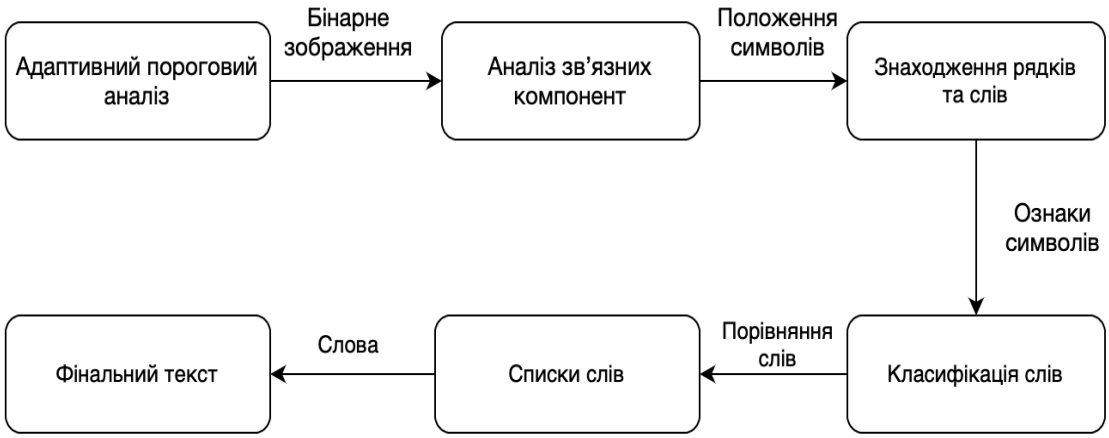


Рисунок 2.1 – Загальний алгоритм роботи OCR

Згідно схемі алгоритму роботи OCR, спочатку проводиться адаптивний пороговий аналіз. Це дозволяє відділити об’єкти на передньому плані від фону. Тобто в даній ситуації на цьому кроці алгоритми визначають що саме на зображенні являється текстом. Після цього за допомогою аналізу зв’язних компонент визначаються межі тексту та положення символів на зображенні, що на подальшому кроці, а саме знаходження рядків та слів, допомагає визначати як символи групуються в слова, а слова в речення. Після чого різними методами можуть бути отримані ознаки символів, що допомагають розпізнати чим саме являється цей символ – літерою, цифрою чи спеціальним символом. Після цього кроку алгоритм аналізує всю отриману вище інформацію і формує фінальний результат у вигляді тексту. Важливо зазначити, що вищеописаний алгоритм є приблизним та найбільш часто вживаним при вирішенні задач OCR. Деякі сервіси та інструменти, описані нижче, мають в основі інші алгоритми або модифікації вищезазначеного. Саме завдяки цим модифікаціям і змінам можна забезпечити покращення результату роботи алгоритму, а отже збільшити якість результуючого тексту.

2.1.1 Опис інструменту TesseractOCR

Одним з найбільш відомих програмних засобів для розпізнавання тексту є TesseractOCR [4]. Цей інструмент, який розробляється Google, використовує машинне навчання для розпізнавання тексту в сканованих документах, фотографіях або інших зображеннях. TesseractOCR базується на архітектурі нейронних мереж, що дозволяє досягати високої точності розпізнавання тексту. Він може бути використаний для перетворення нередагованих документів у редаговані, для зручності зміни інформації в документах, оптимізації пошуку в документі, а також для зберігання документа в цифровій формі. Крім того, TesseractOCR допомагає підвищити продуктивність, зменшити час та зусилля, що необхідні для обробки великої

кількості документів, які містять текст. Програма має досить високу точність розпізнавання тексту, яка може бути покращена за допомогою додаткової підготовки даних та настройки параметрів.

Одним із головних переваг TesseractOCR є те, що він безкоштовний та відкритий, що робить його доступним для використання в різних проектах та дослідженнях. Також TesseractOCR є досить простим у використанні та налаштуванні, що дозволяє швидко розпочати роботу з ним. Проте, як і у будь-якій технології, є й недоліки. Одним з них є низька швидкість обробки складних документів, таких як таблиці, графіки або фотографії з низькою якістю, а також при використанні TesseractOCR можуть бути помилки при розпізнаванні тексту зі складним форматуванням або зі спеціальними символами.

Ця технологія може бути корисною для розпізнавання рукописного тексту, що дозволяє редагувати та зберігати рукописні нотатки у цифровій формі. Але слід розуміти, що програма може мати певні обмеження у розпізнаванні рукописного тексту та тексту з поганою якістю сканування, тому для цих випадків можуть бути необхідні інші методи та інструменти. Важливо зазначити, що TesseractOCR не є універсальним рішенням для всіх видів задач з розпізнаванням тексту, і може потребувати додаткової обробки та налаштування в залежності від конкретного випадку застосування. Попри ці обмеження, для багатьох людей, особливо для студентів і викладачів, інструмент може бути корисним для зберігання і організації рукописних нотаток. Крім того, TesseractOCR може бути використаний у медичній сфері для розпізнавання рукописних рецептів або інших медичних документів.

Отже, TesseractOCR є потужним інструментом для розпізнавання тексту, який може бути використаний у різних сферах, що дозволяє підвищити продуктивність та зручність роботи з документами у цифровій формі.

2.1.2 Опис інструменту Google Vision

Google Vision надає можливість розпізнавання зображень та має додатковий функціонал для обробки цих зображень. Google Vision надає можливість розпізнавання об’єктів, облич, місцезнаходжень, маркерів, логотипів та інших елементів на зображеннях. Цей сервіс використовує найновіші алгоритми машинного навчання та штучного інтелекту для забезпечення точності та швидкості розпізнавання об’єктів. Загалом, Google Vision є потужним інструментом для розпізнавання зображень, який може бути використаний для різних задач, від автоматичної індексації фотографій до розпізнавання тексту на сканах документів. Завдяки поєднанню різних технологій, таких як машинне навчання, комп'ютерний зір та обробка природних мов, Google Vision може допомогти вирішити багато завдань, пов'язаних з обробкою зображень.

Основні функції Google Vision включають розпізнавання тексту на зображеннях, класифікацію зображень, розпізнавання емоцій на обличчях, виявлення об’єктів на зображеннях та автоматичне маркування зображень (рисунок 2.2). Розпізнавання тексту використовується для автоматичного виділення тексту зі зображення та його конвертації в електронний формат, таким чином дозволяючи проводити подальшу обробку тексту. Класифікація зображень дозволяє визначити тип зображення та його вміст. Розпізнавання емоцій на обличчях використовується для аналізу емоцій людей на зображеннях та дозволяє визначити настрій людини на фото. Виявлення об’єктів на зображеннях використовується для визначення їх типу та розміру, що може бути корисно для подальшої обробки цих зображень. Автоматичне маркування зображень дозволяє автоматично визначити основні елементи на зображенні, такі як текст, обличчя та інші об’єкти. Google Vision дозволяє використовувати свої API для інтеграції цієї послуги в веб-сайти, мобільні додатки та інші програмні продукти. Завдяки цьому користувачі можуть легко та швидко

використовувати функції цього додатку. За допомогою Vision API користувачі можуть також створювати власні моделі машинного навчання, що дозволяє адаптувати розпізнавання для конкретних потреб.

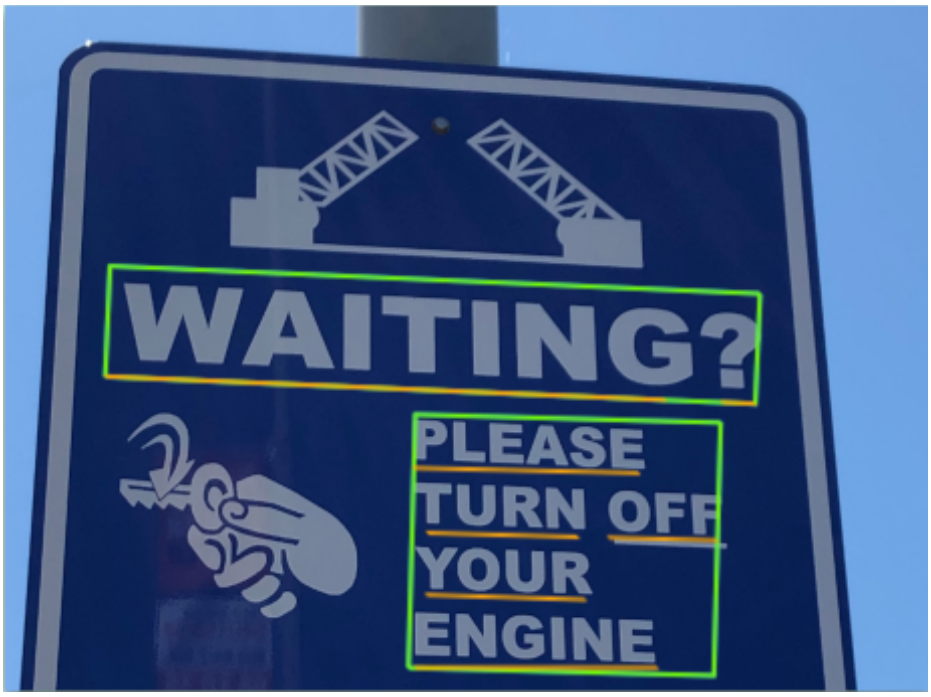


Рисунок 2.2 – Приклад автоматичного виявлення зон з текстом за допомогою Google Vision

Алгоритм розпізнавання тексту в Google Vision починається з обробки зображення та виявлення на ньому зон, де може міститись текст. Далі виконується сегментація зображення, що дозволяє виділити окремі символи та рядки тексту. Після цього виконується розпізнавання символів та їх обробка, щоб визначити, які символи складають окремі слова та речення. Також Google Vision може розпізнавати текст на різних мовах та відображати результати в різних форматах, включаючи текстовий вивід та графічний вивід у вигляді обробленого зображення.

Незважаючи на те, що Google Vision є потужним інструментом для розпізнавання тексту та інших елементів на зображеннях, він також має деякі обмеження. По-перше, використання Google Vision API вимагає

підключення до Інтернету, що означає, що ви не зможете використовувати його в офлайн-режимі. По-друге, залежно від обсягу даних, які потрібно обробити, вартість використання Google Vision API може бути високою. Крім того, API обмежує кількість запитів, які можна зробити за один раз, що може бути проблемою для великих проектів. По-третє, Google Vision API може бути обмежений у своїй точності розпізнавання в залежності від типу даних, які потрібно розпізнати. Наприклад, він може бути менш ефективним для розпізнавання рукописного тексту порівняно з друкованим. А також сервіс може мати проблеми з розпізнаванням складних нестандартних шрифтів. Крім цього, збір та передача даних за допомогою Google Vision API може порушувати приватність користувачів, тому слід бути обережним з використанням цього інструменту.

Google Vision API пропонує можливість розпізнавання тексту за допомогою двох методів: DOCUMENT_TEXT_DETECTION та HANDWRITING_DETECTION. Метод DOCUMENT_TEXT_DETECTION призначений для розпізнавання тексту на документах, що містять надрукований або машинописний текст. Однак, цей метод не гарантує точне розпізнавання рукописного тексту. Метод використовується для розпізнавання тексту з різноманітних документів, таких як звіти, книги, рекламні оголошення та інші документи з відомим шрифтом. Цей метод дозволяє знайти текст в документі, витягнути його та повернути у вигляді рядка символів. Одне з головних переваг даного методу полягає в тому, що він дозволяє розпізнавати текст, написаний багатьма мовами, що дозволяє здійснювати розпізнавання тексту в багатомовному середовищі.

Для розпізнавання рукописного тексту, Google Vision API використовує метод HANDWRITING_DETECTION. Цей метод дозволяє розпізнавати рукописний текст, написаний від руки на папері або на електронному пристрої, а також за допомогою методу можна розпізнати непопулярний рукописний шрифт. Однак, точність розпізнавання може залежати від якості зображення та якості написання рукописного тексту.

Загалом, розпізнавання рукописного тексту за допомогою Google Vision може бути корисним інструментом для автоматизації обробки документів та забезпечення більш ефективного робочого процесу. Однак, необхідно враховувати обмеження технології та можливість помилок в процесі розпізнавання. Даний метод виявляється більш складним у використанні порівняно з методом DOCUMENT_TEXT_DETECTION, особливо при розпізнаванні складного рукописного тексту.

Однією з головних проблем методу HANDWRITING_DETECTION являється те, що він потребує великої кількості даних для навчання. Це означає, що для досягнення високої точності розпізнавання необхідно мати достатньо великий набір даних зі зразками рукописного тексту, що може бути складним в реалізації. Також, іншою проблемою методу є відсутність підтримки деяких мов, особливо мов з ієрогліфічним письмом, таких як китайська або японська. Це обмежує можливості застосування цього методу для розпізнавання рукописного тексту в документах, що містять такі мови. Крім того, метод може мати проблеми з розпізнаванням рукописного тексту зі складним написанням, що може призвести до помилок в розпізнаванні.

Отже, HANDWRITING_DETECTION є потужним інструментом для розпізнавання рукописного тексту, але він має свої обмеження, зокрема в області навчання моделі та підтримки мов та складних написів.

2.1.3 Опис інструменту ABBYY OCR

Програмне забезпечення ABBYY OCR розроблено для розпізнавання тексту. Цей інструмент дозволяє розпізнавати текст на зображеннях та сканах документів. Технологія використовує різні методи та алгоритми для розпізнавання тексту, такі як зображення символів, сегментація тексту та розпізнавання контексту. Ці методи дозволяють досягати високої точності розпізнавання тексту, включаючи рукописний текст. ABBYY OCR може бути використаний для різних цілей, таких як автоматизація бізнес-

процесів, збереження документів у цифровому форматі, розпізнавання тексту в електронних книгах та журналах, переклад тексту та багато іншого. Технологія також має можливості для інтеграції з іншими програмними засобами, що дозволяє використовувати його у різних сферах. Однак, використання технології може бути складним у випадку низької якості зображень або в разі наявності складних макетів документів. Для досягнення максимальної точності розпізнавання тексту необхідно правильно налаштувати параметри програми та провести попередню обробку зображень.

Перевагами є те, що ABBYY OCR забезпечує високу точність розпізнавання тексту, завдяки використанню технологій, які дають можливість розпізнавати текст з високою швидкістю і точністю, навіть у випадках, коли вихідний текст складний або має складну структуру. Також ABBYY OCR підтримує розпізнавання тексту в різних мовах та скриптах, що дозволяє застосовувати це програмне забезпечення для розпізнавання тексту в документах з різними мовами та мовними комбінаціями. Також перевагою є висока швидкість роботи, що дозволяє ефективно застосовувати його для обробки великого обсягу даних, а ще зручний та інтуїтивно зрозумілий інтерфейс користувача, що дозволяє легко використовувати програму навіть для користувачів з мінімальним досвідом у цій галузі. Важливо зазначити здатність працювати з різними типами документів, включаючи скановані зображення та PDF-файли, що дозволяє застосовувати цей інструмент для обробки різних типів документів та даних. Усі ці переваги дозволяють ABBYY OCR бути ефективним інструментом для розпізнавання тексту, який може бути застосований у багатьох галузях, включаючи фінанси, медицину, юридичні послуги та інші.

ABBYY OCR, як і будь-який інший OCR-додаток, має свої недоліки. Одним з головних недоліків є вартість продукту, яка є значно вищою порівняно з іншими OCR-додатками на ринку. Крім того, процес налаштування ABBYY OCR може бути дещо складним, зокрема, для

невисококваліфікованих користувачів. Ще одним недоліком є те, що ABBYY OCR може мати проблеми з розпізнаванням тексту зі складною структурою, такою як таблиці або списки. Крім того, може бути складно отримати задовільний результат розпізнавання тексту в документах з різними мовами, що вимагає встановлення додаткових мовних пакетів. До інших недоліків можна віднести потребу в наявності достатньо потужного обладнання для оптимальної роботи програмного забезпечення, а також те, що підтримка деяких форматів файлів, таких як скановані зображення в векторних форматах, може бути обмеженою. Незважаючи на ці недоліки, ABBYY OCR залишається одним з провідних OCR-додатків на ринку завдяки своїм передовим технологіям і здатності до точного розпізнавання тексту з високою швидкістю.

2.1.4 Опис інструменту MMOCR

MMOCR є бібліотекою з відкритим вихідним кодом для розпізнавання тексту, яка базується на PyTorch. Вона пропонує високу швидкість роботи та високу точність розпізнавання тексту для різних типів документів, таких як скановані зображення, фотографії або екрани комп'ютерів. Бібліотека підтримує розпізнавання тексту різних мов, включаючи азіатські мови зі складними символами, такі як китайська, японська та корейська. Вона використовує глибоке навчання та інші техніки обробки зображень для розпізнавання тексту в різних форматах, включаючи рукописний та друкуваний текст. Алгоритми MMOCR працюють в кілька етапів (рисунок 2.3). Спочатку відбувається передобробка зображення для видалення шуму та інших недоліків. Далі застосовуються алгоритми виявлення тексту та визначення його границь. Потім виконується розпізнавання символів та відновлення повного тексту зображення.

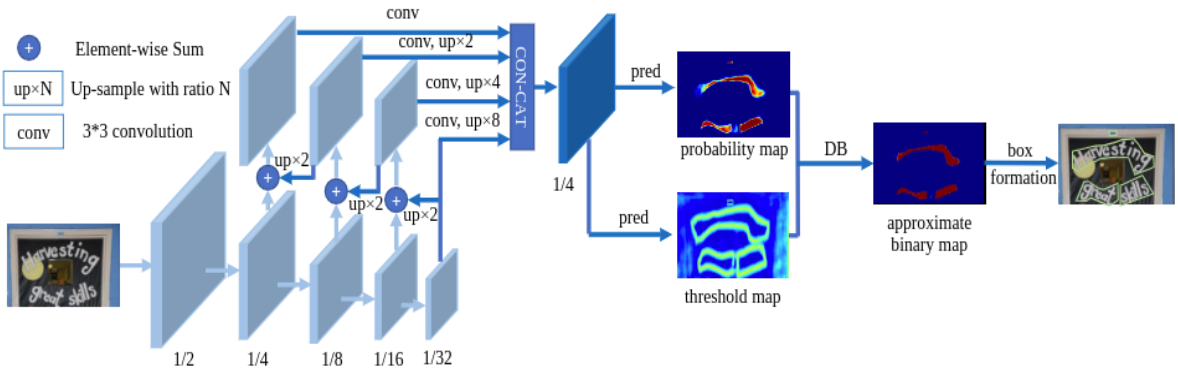


Рисунок 2.3 – Схема алгоритму роботи MMOCR

Однією з основних переваг MMOCR є високий рівень точності розпізнавання тексту. Бібліотека використовує новітні алгоритми та моделі глибокого навчання, що дозволяє досягати високих результатів на різних типах документів та мовах. Крім того, MMOCR є досить швидкою і ефективною технологією, що робить її привабливим варіантом для використання у різних проектах, які вимагають обробки великої кількості даних. Ще однією перевагою MMOCR є відкритий вихідний код, що дозволяє розробникам додатково налаштовувати та покращувати функціонал бібліотеки відповідно до їх потреб. Бібліотека також має документацію та приклади коду, що полегшує розуміння та використання. Ще однією перевагою є підтримка різних форматів даних, включаючи JPEG, PNG та PDF, що робить бібліотеку універсальним рішенням для розпізнавання тексту в різних типах документів та зображень.

Хоча MMOCR має багато переваг, він також має кілька недоліків. Перш за все, на відміну від ABBYY OCR, який має більш розвинуту технологію розпізнавання рукописного тексту, MMOCR не здатний або має обмежену можливість розпізнавання рукописного тексту. Крім того, на даний момент MMOCR не має готових моделей для деяких мов, що може стати перешкодою для використання його в певних міжнародних проектах. Крім того, навчання MMOCR вимагає достатньо великої кількості даних, що

може бути проблемою в деяких випадках. Нарешті, як і більшість інших OCR-додатків, MMOCR може здійснювати помилки при розпізнаванні тексту, що може призвести до неправильної інтерпретації документа або неправильної обробки даних.

2.1.5 Опис інструменту TrOCR

TrOCR [5] є відкритим програмним забезпеченням для розпізнавання тексту, розробленим на базі архітектури трансформерів. Використовуючи моделі, навчені на великих обсягах даних, TrOCR може ефективно розпізнавати текст в зображеннях та сканованих документах. Однією з ключових особливостей TrOCR є використання двоетапної стратегії розпізнавання, яка включає детектування областей з текстом та розпізнавання самого тексту. Перший етап використовує модель об'єктного визначення, яка визначає області [6], де знаходиться текст. Після цього, другий етап використовує модель розпізнавання тексту для отримання текстового вмісту в цих областях (рисунок 2.4).

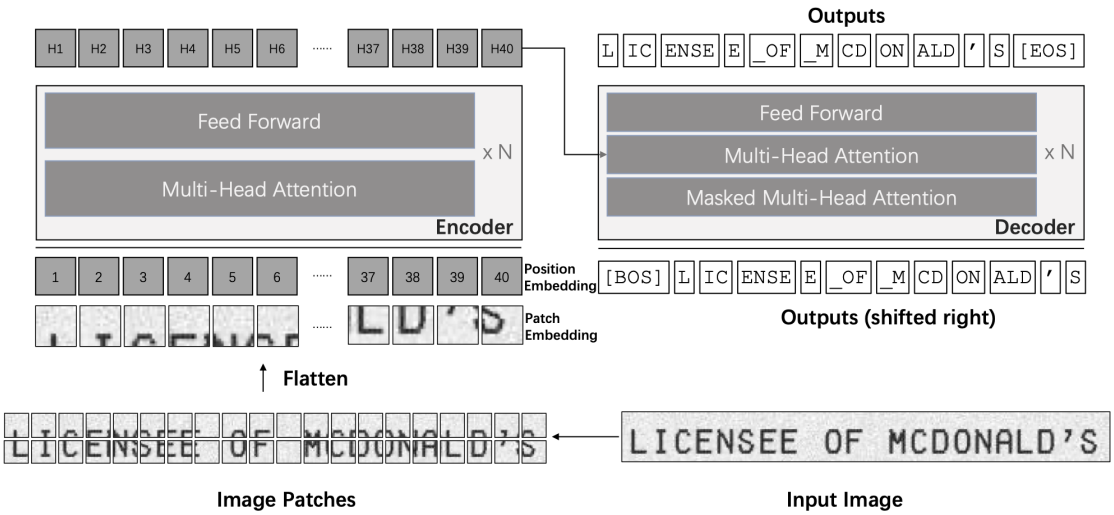


Рисунок 2.4 – Схема алгоритму роботи TrOCR

TrOCR також використовує передові техніки обробки зображень, такі як попередній аналіз зображень, аугментація даних та нормалізація зображень, щоб забезпечити якісне розпізнавання тексту в складних умовах, таких як низька якість зображень або шум. Крім того, TrOCR може бути легко адаптований до різних мов та діалектів, завдяки можливості тренування власних моделей на власних даних. Це робить його універсальним інструментом для розпізнавання тексту для різних завдань та проектів. Перевагою даного програмного забезпечення є те, що завдяки застосуванню глибоких нейронних мереж TrOCR може досягати високої точності розпізнавання тексту, навіть при роботі зі складними зображеннями, які містять різні типи шрифтів та розмірів. Також TrOCR є досить швидким та ефективним інструментом. Застосування оптимізованих алгоритмів дозволяє прискорити процес розпізнавання тексту на великих обсягах даних. Важливо зазначити, що TrOCR був розроблений з використанням передових технологій, що забезпечує його надійність та стабільність. Завдяки цьому TrOCR може бути використаний у великих проектах та системах. Але TrOCR є досить новим методом розпізнавання тексту та все ще перебуває у стадії розробки, тому на даний момент він має кілька недоліків. По-перше, порівняно з деякими іншими методами розпізнавання тексту, TrOCR ще не має великої бази даних, що ускладнює розпізнавання тексту з високим рівнем точності. По-друге, TrOCR залежить від умов освітлення та фону, що може призводити до невірної розпізнавання тексту. Крім того, TrOCR має відкритий вихідний код, що може бути перевагою для розробників, але потенційно ставити під загрозу конфіденційність даних.

2.2 Висновок щодо існуючих підходів для розпізнавання тексту

Усі сервіси OCR мають свої переваги та недоліки [8]. TesseractOCR є відкритим та безкоштовним, але потребує додаткового підготовчого етапу

для досягнення оптимальних результатів. Google Vision має потужний функціонал та добре працює з текстом, але може бути дорогим для використання на великому масштабі. ABBYY OCR має хорошу точність розпізнавання, але також може бути дорогим та не має відкритого коду. MMOCR та TrOCR є відкритими та безкоштовними, але їхні результати можуть бути менш точними порівняно з комерційними аналогами.

Отже, для подальшої розробки OCR рішення буде використано сервіс, який має здатність до маштабування, безкоштовний доступ та легко зрозумілий для розробників.

3 ОБҐРУНТУВАННЯ ОБРАНИХ ТЕХНОЛОГІЙ

3.1 Обґрунтування обраної мови програмування

Основним принципом Python є читабельність коду та простота його написання, що робить її популярною серед початківців і досвідчених програмістів. Python відомий своєю універсальністю та широким спектром застосувань. Він використовується для розробки веб-додатків, наукових обчислень, штучного інтелекту, обробки даних, автоматизації завдань, розробки ігор та багато іншого. Ця мова програмування має велику і активну спільноту користувачів, що дозволяє легко знайти допомогу та ресурси для вивчення мови. Однією з основних переваг Python є його простота та лаконічність. Він має чистий синтаксис, який дозволяє писати зрозумілий код з меншими зусиллями. Python також підтримує об'єктно-орієнтоване та функціональне програмування, що дозволяє розробникам використовувати різні підходи в залежності від потреб проекту.

Python має велику стандартну бібліотеку, що включає в себе багато корисних модулів та функцій для різних завдань. Крім того, існує також велика кількість сторонніх бібліотек, що розширюють можливості мови. Це дозволяє розробникам використовувати готові рішення та прискорює процес розробки. Також програми, написані на цій мові, можуть працювати на різних операційних системах, таких як Windows, macOS та Linux, без змін вихідного коду. Python є мовою програмування, яка має динамічний тип даних і автоматичне керування пам'яттю, що робить його досить простим у використанні. Ця мова має широку підтримку відкритих стандартів і протоколів, таких як HTTP, XML, JSON, і може взаємодіяти з іншими мовами програмування.

Python також пропонує велику кількість розширень та інструментів для розробки, таких як різноманітні фреймворки, бібліотеки і пакети. Python також має простий синтаксис, який допомагає зрозуміти код і знижує

ймовірність помилок. Це особливо корисно для початківців, а також сприяє швидкому розвитку проекту.

Використання Python в проекті з OCR має кілька переваг, які роблять його привабливим вибором. По-перше, це широкий вибір бібліотек та фреймворків. Ця мова програмування має розширену екосистему бібліотек та фреймворків, які спеціалізуються на обробці зображень та розпізнаванні тексту. Наприклад, OpenCV, PyTesseract, EasyOCR і багато інших. Ці інструменти роблять розробку OCR-додатків більш ефективною та зручною. Також Python відомий своєю зрозумілою синтаксичною структурою, що дозволяє швидко розробляти програми. Він має просту синтаксичну граматику, лаконічний код та велику кількість документації, що робить його дуже доступним для початківців і досвідчених розробників. Варто зазначити й широкі можливості машинного навчання, бо Python має багато бібліотек для машинного навчання, таких як TensorFlow, PyTorch, scikit-learn, Keras, що дозволяє використовувати сучасні алгоритми для покращення роботи OCR-додатків. Це дозволяє використовувати глибоке навчання та нейронні мережі для покращення точності розпізнавання тексту. Не меншою перевагою є й активна спільнота розробників, яка постійно співпрацює, обмінюється ідеями, надає підтримку та дозволяє отримувати швидкі відповіді на різноманітні питання, а також сприяє постійному розвитку та вдосконаленню інструментів для розпізнавання тексту в Python. І наостанок, Python має здатність легко інтегруватися з іншими технологіями та сервісами, такими як бази даних, веб-сервери, веб-фреймворки та інші.

Узагалі, використання Python в проекті з використанням OCR надає широкий спектр можливостей та переваг. Він поєднує простоту використання, багатий вибір інструментів та бібліотек, високу продуктивність та широку підтримку спільнотою. Таким чином, Python є потужним інструментом для розробки ефективних та точних систем розпізнавання тексту з використанням OCR.

3.2 Фреймворк для задач комп'ютерного зору OpenCV

OpenCV надає широкі можливості для обробки зображень та аналізу даних в реальному часі. OpenCV є одним з найпопулярніших фреймворків для роботи з задачами комп'ютерного зору та обробки зображень, і використовується в різних галузях, включаючи робототехніку, автомобільну промисловість, медицину, безпеку, відеоспостереження та багато іншого.

Основні переваги використання OpenCV включають те, що OpenCV має велику кількість функцій та алгоритмів для обробки зображень, включаючи фільтрацію, морфологічні операції, детектування країв, виявлення об'єктів, вимірювання, розпізнавання обличчя та багато іншого. OpenCV також підтримує роботу з веб-камерами, відеозаписом та відтворенням відео. Це дозволяє розробникам ефективно вирішувати широкий спектр завдань комп'ютерного зору. Також, OpenCV визначається кросплатформеністю, тобто підтримує різні операційні системи, включаючи Windows, Linux, macOS, Android та iOS. Це дозволяє розробникам реалізовувати свої проекти на різних платформах з використанням єдиного фреймворку. OpenCV має простий та зрозумілий інтерфейс програмування, що полегшує розробку. Важливою перевагою також є підтримка різних мов програмування. OpenCV пропонує інтерфейси для використання з багатьма мовами програмування, зокрема Python, C++, Java та іншими. Це дає розробникам можливість використовувати OpenCV у своїх проектах на мові програмування, яка найбільш підходить для їхніх потреб і навичок. Фреймворк має велику спільноту та якісну документацію. Активна спільнота розробників постійно вносить внески до фреймворку та надає підтримку іншим користувачам. Крім того, OpenCV має багато наочних прикладів, документацію та ресурси, що полегшують навчання та використання фреймворку. OpenCV підтримує можливість створення власних алгоритмів та додаткових функцій, що дозволяє розробникам

налаштовувати та розширювати фреймворк згідно зі своїми потребами. Даний фреймворк є проектом з відкритим вихідним кодом, що дає можливість розробникам використовувати, модифікувати та розповсюджувати його без обмежень. Це сприяє спільній роботі, обміну знаннями та вдосконаленню фреймворку.

В цілому, використання OpenCV у проекті з комп'ютерним зором має безліч переваг, включаючи широкий функціонал, кросплатформеність, ефективність, легкість використання, підтримку різних мов програмування, активну спільноту та відкритий вихідний код. Враховуючи ці фактори, OpenCV стає потужним інструментом для розв'язання завдань комп'ютерного зору та обробки зображень.

Загалом, OpenCV є незамінним інструментом у галузі комп'ютерного зору та обробки зображень. Вона має численні застосування і користується популярністю серед розробників з різних галузей, включаючи комп'ютерний зір, робототехніку, медичне зображення, відеоаналітику, розпізнавання облич та багато іншого. OpenCV може бути використаний в широкому спектрі проектів, пов'язаних з комп'ютерним зором, обробкою зображень та аналізом даних. Наприклад, OpenCV є чудовим вибором для проекту з розпізнавання облич, бо фреймворк надає алгоритми для виявлення облич, розпізнавання особливостей та аналізу облич. Це може бути використано в системах безпеки, соціальних медіа, системах розпізнавання емоцій тощо. Також можна його використовувати у відеоаналітиці, бо OpenCV дозволяє виконувати аналіз відео, включаючи виявлення руху, слідування об'єктів, вимірювання швидкості, визначення траєкторії тощо. Це корисно для систем відеоспостереження, аналізу поведінки, розпізнавання дій тощо. Фреймворк може бути використано в роботі з робототехнікою, а саме для програмування та керування роботами з вбудованими камерами. Він дозволяє роботам розпізнавати об'єкти, уникати перешкод, виконувати навігацію та багато іншого. Також OpenCV надає інструменти для обробки та аналізу медичних зображень, таких як

рентгенограми, знімки МРТ тощо. Це може бути використано для виявлення патологій, сегментації зображень, вимірювання параметрів тощо. Крім того, фреймворк можна використати в розробци додатків для мобільних пристроїв, бо OpenCV підтримує платформи Android та iOS, що дозволяє розробникам створювати додатки для розпізнавання облич, аналізу зображень та інші комп'ютернозорові завдання. Це лише кілька прикладів проектів, для яких може бути використаний OpenCV.

Незважаючи на широкий функціонал та популярність, у OpenCV також є деякі недоліки, які варто враховувати. Наприклад, складність використання, через велику кількість функцій та алгоритмів, що може призвести до складнощів у їх розумінні та використанні. Для новачків це може виявитись викликом, і може знадобитись час та зусилля для освоєння бібліотеки. Також слід зазначити відсутність підтримки глибокого навчання. У порівнянні з деякими сучасними фреймворками, OpenCV не має вбудованої підтримки для розробки моделей глибокого навчання. Це означає, що для роботи з нейромережами потрібно використовувати інші бібліотеки або фреймворки. Відсутність активного оновлення також можна віднести до недоліків. Хоча OpenCV має велику спільноту користувачів, не завжди є регулярні оновлення бібліотеки. Це може вплинути на наявність нових функцій, виправлення помилок та підтримку нових технологій. Важливим є те, що OpenCV може вимагати значних обсягів пам'яті та обчислювальних ресурсів, особливо при обробці великих зображень або відео. Це може стати проблемою на обмежених пристроях з обмеженими ресурсами. І наостанок, недоліком також є відсутність розширеної підтримки для деяких задач: В деяких випадках OpenCV може не мати готових рішень для деяких специфічних задач комп'ютерного зору або обробки зображень. У таких випадках може знадобитись додаткове налаштування або використання інших бібліотек.

Однак, враховуючи широкий функціонал та популярність OpenCV, було обрано використовувати цей фреймворк декількох причин. По-перше,

OpenCV надає потужні інструменти для обробки зображень, що є ключовим етапом у процесі OCR. Вона має багато функцій та алгоритмів, таких як зміна розміру, фільтрація, сегментація та виявлення країв, які можна використовувати для покращення якості зображень перед розпізнаванням. По-друге, OpenCV є кросплатформеним фреймворком, що означає, що він підтримує різні операційні системи, такі як Windows, Linux, macOS, Android та iOS. Це робить його універсальним вибором. Крім того, OpenCV має велику та активну спільноту користувачів, що сприяє наявності багатьох ресурсів, документації та прикладів. Це допомагає швидше розібратися з функціями та отримати допомогу у вирішенні проблем.

Таким чином, вибір фреймворку OpenCV обґрунтований його потужними функціями обробки зображень, кросплатформеною підтримкою та наявністю великої спільноти користувачів. Використання OpenCV дозволить розробникам ефективно реалізувати функціонал OCR та досягти бажаних результатів у проекті.

3.3 Мова розмітки LaTeX

LaTeX [7] є мовою розмітки, яка широко використовується для створення наукових та технічних документів. LaTeX надає потужні можливості для структурування, форматування та компонування документів різного типу, таких як академічні статті, книги, презентації, тези, листи та багато іншого. Також LaTeX надає розширені можливості типографічного форматування, що дозволяє створювати професійно виглядаючі документи. Він автоматично вирівнює текст, встановлює оптимальні пробіли між словами та рядками, керує розташуванням заголовків, списків, таблиць та інших елементів.

Одна з головних переваг LaTeX полягає в його здатності до автоматичного керування нумерацією розділів, заголовками, таблицями змісту та посиланнями. Це забезпечує послідовність індексів і зручну

навігацію по документу. LaTeX дозволяє точно контролювати розмітку документа. Ви можете налаштовувати розміри сторінки, відступи, колонки, номера сторінок, заголовки розділів та підрозділів, змінювати шрифти, кольори та інші параметри. Це дозволяє створювати документи, які відповідають специфічним вимогам форматування. Ще одна важлива перевага LaTeX – це його висока якість набору математичних формул. Математичні формули в LaTeX виглядають професійно і легко змінюються, оскільки вони представлені у вигляді текстового коду (рисунок 3.1). LaTeX має потужні засоби для створення складних математичних рівнянь, символів, матриць, символів грецького алфавіту та багато іншого. Він забезпечує відмінну роздільну здатність, збереження стилю та високу якість друку для математичних виразів, що робить його незамінним інструментом для наукових публікацій та досліджень. Крім того, LaTeX є відкритим і безкоштовним програмним забезпеченням, що означає, що він доступний для використання всім бажаючим без обмежень або плати за ліцензію. Його поширена популярність серед науковців і технічних спеціалістів. Також LaTeX дозволяє використовувати програмування для створення власних команд та макросів, що полегшує автоматизацію та повторне використання коду. Крім того, LaTeX підтримує багато додаткових пакетів, які додають нові функціональні можливості і зручні інструменти для роботи зі специфічними типами документів.

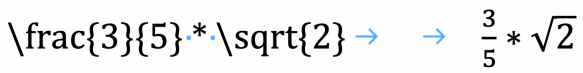


Рисунок 3.1 – Приклад написання формули та її фінальне відображення

Однак, важливо відзначити, що LaTeX вимагає від користувача знання його синтаксису та структури документації, що може виявитися

викликом для новачків. Також, порівняно з деякими іншими текстовими процесорами, LaTeX може бути менш інтерактивним, оскільки його основна мета – це створення професійних та якісних документів, а не швидке редагування на місці. LaTeX побудований на основі текстового редактора, що означає, що ви повинні редагувати документи у вигляді текстового файлу. Він не надає візуального редактора, що може бути не дуже зручно для користувачів, звиклих до візуального інтерфейсу. Слід зазначити, що хоча LaTeX дозволяє точне контролювання розмітки документа, він може бути обмежений у варіантах макету. Існують певні обмеження щодо налаштування деяких параметрів, таких як розташування плаваючих об'єктів або гнучкість вирівнювання тексту. Також незважаючи на те, що LaTeX надає потужні можливості для створення документів, він може бути менш зручним для роботи з іншими форматами, такими як редагування HTML-сторінок або обробка тексту у форматі Word. Іноді потрібно використовувати додаткові інструменти або конвертувати документи, щоб працювати з іншими форматами.

LaTeX використовується для написання документів у вигляді коду з розміткою і структурованим текстом. Коли LaTeX-файл компілюється, він перетворюється на PDF-файл, який відображає кінцевий вигляд документа. Під час компіляції LaTeX використовує свої внутрішній механізми для обробки розмітки, обчислення залежностей і форматування тексту. В результаті отримується проміжний файл, який називається DVI (Device Independent) файл. Цей DVI-файл не є прямим PDF, але містить інформацію про структуру та вигляд документа. Для отримання кінцевого PDF-файлу DVI-файл потрібно конвертувати. Це виконується за допомогою спеціальних утиліт у поєднанні зі сторонніми програмами для перетворення DVI в PDF. Ці утиліти зчитують DVI-файл, розбирають його та створюють відповідний PDF-файл, який відображає остаточний вигляд документа з усіма форматуваннями, зображеннями, таблицями, посиланнями та іншими елементами. Отже, LaTeX використовується для створення документів у

вигляді LaTeX-коду, а згодом цей код компілюється і конвертується в PDF-формат. PDF-файл, отриманий з компіляції LaTeX, є високоякісним інтерактивним документом, який може бути легко переглянутий, роздрукований і поширений без необхідності використання вихідного LaTeX-коду. Узагалі, LaTeX є потужним інструментом для створення складних наукових та технічних документів. Він забезпечує високу якість відформатованого вигляду, особливо для математичних формул, і має широкі можливості для структурування та форматування документів. Його безкоштовність та наявність великої спільноти користувачів роблять його популярним вибором серед науковців, дослідників та письменників, які зосереджені на якості та професійному вигляді своїх документів

3.4 Бібліотека img2table

Бібліотека img2table є інструментом, який спеціалізується на розпізнаванні таблиць зображень і конвертації їх у структурований формат даних. Вона дозволяє витягувати таблиці зі сканованих документів, фотографій або інших зображень і перетворювати їх у таблиці, які можна легко обробляти та аналізувати.

Однією з головних переваг бібліотеки img2table є її здатність працювати з різними типами таблиць та різними мовами. Вона підтримує таблиці з різними розмірами, форматами і структурами, включаючи таблиці з об'єднаними комірками, заголовками, підзаголовками та багатошаровими таблицями. Крім того, вона має підтримку різних мов, що дозволяє розпізнавати текст у таблицях написаний на різних мовах. Іншою важливою перевагою бібліотеки img2table є її простота в використанні. Вона надає зручний API, який дозволяє легко інтегрувати її функціональність у власні проекти. Завдяки цьому, розпізнавання таблиць стає простим завданням навіть для користувачів з обмеженим досвідом у сфері комп'ютерного зору або розпізнавання образів.

Однак, варто враховувати, що бібліотека img2table також має деякі обмеження. Вона може стикатися з труднощами у розпізнаванні таблиць з високим ступенем складності, таких як таблиці зі складними геометричними структурами або великою кількістю об'єднаних комірок. Крім того, вона може давати некоректні результати у випадку низької якості зображень. Деякі складні форматування, такі як ті, що включають графіки або таблиці зі складними стилями, також можуть бути складними для коректного розпізнавання за допомогою бібліотеки img2table. Крім того, важливо зазначити, що бібліотека img2table має обмежені можливості оптимізації та налаштування. Наприклад, вона може не мати вбудованих інструментів для виправлення помилок розпізнавання або ручного визначення структури таблиць. Це може бути проблематично, якщо виявляються помилки або потрібно внести корективи у розпізнані таблиці.

Отже, перед використанням бібліотеки img2table варто врахувати ці недоліки і зробити оцінку їх впливу на конкретний проект. Враховуючи її простоту використання та загальну ефективність, бібліотека img2table може бути цінним інструментом для багатьох завдань розпізнавання таблиць зображень, але слід усвідомлювати її обмеження та потребу у додаткових ресурсах для обробки складних випадків.

Бібліотека img2table може взаємодіяти з Tesseract OCR для поліпшення процесу розпізнавання тексту в таблицях зображень. При розпізнаванні тексту за допомогою Tesseract OCR, зображення таблиці може бути передано в бібліотеку img2table для подальшої обробки та витягування даних. img2table може виконувати попередню обробку зображення, таку як видалення шуму або покращення контрастності, що допомагає покращити точність розпізнавання тексту. Після обробки зображення бібліотека img2table використовує алгоритми та методи для витягування текстових даних з таблиці.

Отримані результати можуть бути подані у вигляді структурованої таблиці або інших форматів, що легко обробляються іншими програмами

або алгоритмами. Таким чином, взаємодія між бібліотекою img2table та Tesseract OCR дозволяє досягти кращої точності та ефективності розпізнавання тексту в таблицях зображень. Важливо зазначити, що обидва інструменти, img2table і Tesseract OCR, мають свої особливості та обмеження. Оптимальні результати можуть бути досягнуті завдяки належному налаштуванню параметрів та використанню підходящих методів обробки зображень та розпізнавання тексту.

Загалом, взаємодія між бібліотекою img2table та Tesseract OCR відкриває нові можливості для розпізнавання тексту в таблицях зображень, покращує точність та поліпшує якість отриманих результатів

3.5 Обґрунтування обраного OCR інструменту

Після проведення аналізу різних OCR-інструментів, було прийняте рішення вибрати Tesseract OCR для подальшого використання в проекті. По-перше, Tesseract OCR є одним з найпопулярніших та найширше використовуваних OCR-інструментів у спільноті розпізнавання тексту. Він має велику кількість користувачів і активну підтримку спільноти розробників, що сприяє постійному вдосконаленню і розвитку інструменту. По-друге, Tesseract OCR надає якісні результати розпізнавання тексту, особливо при роботі з друкованим текстом. Він демонструє високу точність і швидкість розпізнавання, що є важливими факторами для ефективного використання OCR-інструменту в проекті. Крім того, Tesseract OCR має відкритий вихідний код, що дозволяє легко налаштовувати та розширювати функціональність інструменту залежно від потреб проекту. Це забезпечує гнучкість і можливість використовувати Tesseract OCR у різних сценаріях із зручністю. Таким чином, в результаті аналізу було встановлено, що Tesseract OCR має потрібну комбінацію переваг – популярність, надійність і розширюваність, що робить його найкращим вибором для використання в нашому проекті з OCR.

4 РЕАЛІЗАЦІЯ ТЕМАТИЧНОГО АЛГОРИТМУ

Відповідно до зазначеного у минулих розділах, для успішної реалізації сервісу для конвертації сканованих документів до формату PDF необхідно реалізувати швидкодіючий та точний алгоритм екстракції тексту з зображень, враховуючи вірогідність зустріти скани низької якості або із спотвореннями. Далі буде описано процес створення інструменту для екстракції тексту з зображень а також алгоритмів, необхідних для покращення результатів основної задачі.

4.1 Алгоритм екстракції тексту

Як зазначалося вище, для коректної роботи екстрактору тексту необхідно врахувати усі можливі види спотворень на сканованих зображеннях для того, щоб запобігти некоректного розпізнавання тексту. При розпізнаванні тексту на сканованих зображеннях необхідно враховувати широкий спектр можливих спотворень, які можуть впливати на якість розпізнавання. Деякі з цих спотворень включають неоднорідне освітлення, затемнення, тіні, бліки, розмитість, нахил, спотворення перспективи, шум та інші недоліки, які можуть виникати під час сканування або фотографування документів.

Неоднорідне освітлення може створювати нерівномірність яскравості і контрасту на зображенні, що ускладнює виявлення та розпізнавання тексту. Затемнення та тіні можуть перекривати деякі частини тексту, що знижує точність розпізнавання. Бліки, викликані відбиванням світла, також можуть спотворити зображення і ускладнити його обробку. Інші види спотворень, такі як розмитість, нахил та спотворення перспективи, можуть виникати під час руху камери або некоректного позиціонування документів. Розмитість може знижувати роздільну здатність тексту, ускладнюючи його розпізнавання. Нахил та спотворення перспективи можуть спотворювати

геометричну структуру тексту, що може призводити до неправильного розпізнавання символів та слів.

Врахування всіх цих можливих видів спотворень на сканованих зображеннях є важливим для запобігання некоректного розпізнавання тексту. Цього можна досягти шляхом застосування певних методів і технік обробки зображень перед подачею їх на вхід OCR-системи. Недостатня увага до цих факторів може призвести до помилкових результатів OCR, що може майже повністю підривати корисність системи. Наприклад, навіть невеликий зсув на зображенні може призвести до того, що OCR помилково розпізнає букву «O» як «Q». Тому, для досягнення найкращих результатів OCR, необхідно бути відповідальним і враховувати всі можливі спотворення при обробці зображення.

Як зазначалося в пункті 3.4, для подальшої роботи було обрано Tesseract OCR для подальшого використання в проекті. Python має два головні модулі для роботи з Tesseract OCR – це pytesseract [8] і tesseocr. Для використання у проекті було обрано tesseOCR через те, що він має кращі показники швидкості виконання. Це обумовлено тим, що даний модуль може використовувати більше потоків процесора для розпізнавання тексту, що дозволяє розподіляти завдання між різними ядрами процесора та зменшувати час обробки. Також tesseOCR може використовувати більш оптимізований алгоритм розпізнавання тексту, що дозволяє досягати більшої швидкості обробки зображень. Також tesseOCR має оптимізовану обробку зображень з використанням методів обрізання, налаштування контрастності та якості зображень, що знижує час обробки зображень та покращує точність розпізнавання. Також важливо зазначити, що даний модуль має більш ефективний механізм кешування результатів, що дозволяє зберігати результати розпізнавання та використовувати їх для наступних запитів, замість повторної обробки тих самих зображень. Це дозволяє зменшити час обробки та збільшити продуктивність при використанні більшої кількості зображень. Отже, зважаючи на ці фактори, технічні

особливості та оптимізація, що містяться в tesseOCR, дозволяють досягати більшої швидкості обробки зображень та розпізнавання тексту, що робить його більш ефективним та швидким в порівнянні з Pytesseract.

Програмний код реалізації процесу розпізнавання тексту зі сканованих сторінок PDF за допомогою OCR наведено в лістингу 4.1.

Лістинг 4.1 – Програмний код реалізації процесу розпізнавання тексту зі сканованих сторінок

```
def extract_text_from_pdf(images: List[np.ndarray]) ->
List[List[namedtuple]]:
    """
    Extract text from PDF pages using OCR.
    :param images: List of pages converted to images.
    :return: List of lists of namedtuples with data for each text
extracted (text, bbox, confidence).
    """
    page_text = namedtuple('Page_text', ['high_confidence',
'low_confidence'])
    data = []
    boxes_all = []
    with PyTessBaseAPI() as api:
        for image in tqdm(images):
            api.SetImage(image)
            b2e = []
            low_conf = []
            boxes = api.GetComponentImages(RIL.TEXTLINE, True)
            boxes_all.append(boxes)
            for im, box, _, _ in boxes:
            api.SetRectangle(box['x'], box['y'], box['w'], box['h'])
                ocr_result = api.GetUTF8Text()
                conf = api.MeanTextConf()
                element = ELEMENT(ocr_result.replace('\n', ' ') +
'\n', list(box.values()), conf)
```

Продовження лістингу 4.1

```
                if conf > 90:
                    b2e.append(element)
                else:
                    low_conf.append(element)
            data.append(page_text(b2e, low_conf))
    extract_text_from_pdf.boxes_all = boxes_all
        return data
```

Основна особливість функції полягає у тому, що вона відбирає фрагменти тексту, що були розпізнані з низькою точністю. Це в майбутньому дозволить сфокусувати більше уваги саме на проблемних фрагментах.

4.1.1 Вирівнювання координатної сітки зображення

У процесі роботи з документами, особливо зі сканованими зображеннями, однією з основних проблем є вирівнювання координатних сіток для ефективного розпізнавання тексту. В традиційних документах, наприклад, у форматі А4, розташування тексту та інших елементів зазвичай має стандартизований вигляд. Однак, у випадку сканованих документів, що можуть мати будь-яке розширення та розміри, координатні сітки можуть бути спотворені або неправильно вирівняні.

Це викликає необхідність вирівнювання координатних сіток для забезпечення точного розпізнавання тексту. Вирівнювання полягає у встановленні правильних координат для кожного елемента тексту або графічного об'єкта на зображенні. Це може включати визначення положення тексту, прямокутників, ліній, таблиць тощо. Процес вирівнювання координатних сіток може включати в себе використання алгоритмів обробки зображень та комп'ютерного зору. Ці алгоритми аналізують скановане зображення, визначають його орієнтацію, розміри, відстані між елементами, а також виявляють спотворення або перекручення. На основі

цих даних вирівнюються координатні сітки, щоб забезпечити правильну позицію кожного елементу.

В лістингу 4.2 наведено програмний код для вирівнювання координатної сітки. Наразі допускається, що на зображенні відсутній нахил тексту, тобто немає необхідності вирівнювати перспективу. Для коректного позиціювання майбутнього тексту використовується стандартне розширення A4 у піксельному форматі, орієнтація портретна, а саме 595x842 пікселя.

Лістинг 4.2 – Програмний код для вирівнювання координатної сітки

```
def transform_cords(elements: List[List[namedtuple]]) ->
List[List[namedtuple]]:
    """
    Transform bboxes of each element as if it was A4
coordinates.
    :param elements: Elements detected on each page.
    :return: List of elements per-page with coords transformed.
    """
    w_alpha = 595 / w
    h_alpha = 842 / h
    new_elements = []
    for page_elements in elements:
        new_el = []
        for element in page_elements:
            x, y, width, height = element.bbox
            new_box = [int(x * w_alpha), int(y * h_alpha),
int(width * w_alpha), int(height * h_alpha)]
            new_el.append(element._replace(bbox=new_box))
        new_elements.append(new_el)
         return new_elements
```

4.1.2 Використання апскейлінгу зображення для покращення результатів розпізнавання текстів

Апскейлінг [9] – це процес збільшення роздільної здатності зображення з метою поліпшення якості та чіткості зображення. Використання апскейлінгу може бути корисним при роботі з OCR, оскільки він може покращити розпізнавання тексту з низької якості зображення.

Один з прикладів використання апскейлінгу для покращення роботи OCR полягає в збільшенні роздільної здатності зображення для поліпшення якості зображення. Якщо зображення має низьку роздільну здатність, текст може бути нерозпізнаним або розпізнаним з помилками, що ускладнює роботу OCR. Застосування апскейлінгу для збільшення роздільної здатності зображення може покращити якість зображення та зробити текст більш чітким для розпізнавання OCR. Інший приклад використання апскейлінгу – це зменшення шуму на зображенні. Шум може виникати на зображеннях з різних причин, таких як низька якість зображення або погане освітлення. Застосування апскейлінгу для зменшення шуму може покращити якість зображення та зробити текст більш чітким для розпізнавання OCR. Отже, апскейлінг може бути корисним для покращення роботи OCR, оскільки він може покращити якість та чіткість зображення, що сприяє точнішому розпізнаванню тексту. Програмний код функції апскейлінгу зображення наведено в лістингу 4.3.

Лістинг 4.3 – Програмний код функції апскейлінгу зображення

```
def upscale(image):
    """
    Upscales the given image to update quality.
    :param image: Image to upscale.
    :return: Upscaled image.
    """
```

Продовження лістингу 4.3

```
    sr = dnn_sr.DnnSuperResImpl_create()
    path = "models/EDSR/EDSR_x3.pb"
    model_name = os.path.basename(path).split("_")[0].lower()
    model_scale = int(path.split("_x")[-1].split(".")[0])
    print(f"Using model: {model_name} with scale factor:
{model_scale}")
    sr.readModel(path)
    sr.setModel(model_name, model_scale)
    result = sr.upsample(image)
         return result
```

Використання апскейлінгу дозволяє суттєво покращити результати розпізнавання тексту, проте іноді виникають ситуації, коли помилка в розпізнаванні стається через некоректно розпізнані границі тексту. В даній кваліфікаційній роботі пропонується простий, і в той час ефективний, метод вирішення цієї проблеми.

4.1.3 Метод коригування границь тексту

В ситуаціях, коли границі тексту були розпізнані невірно, часто виникають помилки розпізнавання окремих символів. Наприклад, модель OCR визначає нижню границю слова по нижній границі більшості літер, але слово містить літеру, нижня границя якої знаходиться нижче, ніж у інших. В українському алфавіті, до прикладу, такими літерами можуть бути: р, ф, у, тощо. В результаті некоректного розпізнавання границі, частини цих літер можуть бути обрізані, і як результат, літери будуть розпізнані некоректно. Отже, використовуючи фрагменти тексту, які модель розпізнала найменш точно (лістинг 4.1), можна зробити допущення, що саме в цих фрагментах були некоректно розпізнані границі. Для вирішення цієї

проблеми застосовується метод штучного розширення границь фрагменту і примусового фокусування уваги моделі на цьому фрагменті. Програмний код методу штучного розширення границь фрагменту і примусового фокусування уваги моделі на цьому фрагменті наведено у лістингу 4.4.

Лістинг 4.4 – Програмний код методу штучного розширення границь фрагменту

```
def update_text(image: np.ndarray, bbox: List[int], idents:
Tuple[int, int, int, int], upscale_im: bool = False) ->
Tuple[str, float]:
    """
    Update text based on bbox and idents.
    :param image: Numpy array of image data.
    :param bbox: Bounding box coordinates.
    :param idents: Up, down, left, right pixel shifts.
    :param upscale_im: Whether to upscale image.
    :return: Tuple of updated text and its confidence.
    """
    minus_y, plus_y, minus_x, plus_x = idents
```

Продовження лістингу 4.4

```
    ocv_im = np.array(image)
    new_imm = ocv_im[(bbox[1] - minus_y):(bbox[1] + bbox[3] +
plus_y), (bbox[0] - minus_x):(bbox[0] + bbox[2] + plus_x)]
    if upscale_im:
        new_imm = upscale(new_imm)
    img_bytes = new_imm.tobytes()
    with PyTessBaseAPI() as api:
        api.SetImageBytes(img_bytes,
                          new_imm.shape[1],
                          new_imm.shape[0],
                          3,
```

Продовження лістингу 4.4

```
                        3 * new_imm.shape[1])
        boxes = api.GetComponentImages(RIL.TEXTLINE, True)
        new_text = api.GetUTF8Text()
        conf = api.MeanTextConf()
         return new_text, conf
```

Дана функція складається з двох частин: в першій вирізається фрагмент зображення з текстом, який був погано розпізнаний на першому кроці, таким чином, що початкові межі цього тексту розширюються на фіксовані значення, які передаються як параметр функції. В другій частині ця частина зображення піддається апскейлінгу, проте для оптимізації роботи алгоритму було вирішено зробити цю частину опціональною, що контролюється параметром функції upscale.Це зроблено для того, щоб скоротити час, що витрачається на обробку документів, оскільки апскейлінг досить ресурсо- та часовитратна операція.

4.1.4 Загальний алгоритм екстракції тексту

В попередніх функціях було описано усі процедури, необхідні для екстракції тексту з зображень та поступового покращення результатів. В лістингу 4.5 буде представлено функцію, яка об’єднує всі описані вище процеси в єдиний конвеєр по екстракції тексту.

Лістинг 4.5 – Функція, яка об’єднує всі описані вище процеси в єдиний конвеєр по екстракції тексту

```
def process_pages(n_bboxes: List, images: List) -> List[List]:
    """
    Process OCR text on each page of a document based on high
and low confidence bounding boxes.
```

Продовження лістингу 4.5

```
    :param n_bboxes: List of namedtuples containing bounding
boxes for high and low confidence OCR results.
    :param images: List of images for each page of the
document.
    :return: List of OCR text for each page, where low
confidence OCR text has been updated if necessary.
    """
    final = []
    for idx, page_data in enumerate(n_bboxes):
        print(f'Processing page: {idx + 1}')
        new_text = []
        new_text.extend(page_data.high_confidence)
        ntb = []
        for lc in page_data.low_confidence:
            updated = update_text(images[idx], lc.bbox, (5, 0,
0, 0))
            if updated[1] > 70:

ntb.append(lc._replace(text=updated[0].replace('\n', ' ') +
'\n'))
            else:
                updated = update_text(images[idx], lc.bbox,
(10, 10, 10, 10), True)

ntb.append(lc._replace(text=updated[0].replace('\n', ' ') +
'\n'))
        new_text.extend(ntb)
        final.append(new_text)
         return final
```

На рисунку 4.1 представлено алгоритм, який пояснює принцип роботи наведеної функції. Функція генерації PDF документів працює з вхідним текстом, який може бути розпізнаний з низькою точністю (рисунок 4.2).

Для вдосконалення якості розпізнавання тексту використовується метод розширення границь. Спочатку нижня границя тексту розширюється на 5 пікселів вниз, щоб включити можливі нижні елементи тексту, які можуть бути пропущені при початковому розпізнаванні. Після цього перевіряється точність розпізнавання тексту. Якщо точність менше 70%, що свідчить про низьку якість розпізнавання, виконується подальше розширення границь на 10 пікселів з кожного боку. Це дозволяє включити більше контексту і може поліпшити розпізнавання тексту. Після розширення границь застосовується апскейлінг, що дозволяє збільшити розмір тексту для отримання більш точних результатів розпізнавання.

У випадку, коли точність розпізнавання тексту після попередніх операцій перевищує 70%, розширення границь та апскейлінг не застосовуються. Вміст тексту оновлюється без змін границь, оскільки розпізнавання вже було задовільним. Це дозволяє зекономити час обробки та ресурси. Після оновлення погано розпізнаного тексту функція застосовує модель корекції правопису. Це важливий крок, який допомагає виправити можливі помилки розпізнавання та покращити остаточний результат. Модель корекції правопису аналізує текст і вносить відповідні зміни для виправлення помилок. Використання моделі корекції правопису є надзвичайно важливим кроком у процесі покращення точності розпізнавання тексту. Проте, необхідно усвідомлювати, що жодна модель не є повністю ідеальною і може викликати певні помилки та особливості. Кожна модель корекції правопису має свої обмеження та особливості. Вони базуються на різних алгоритмах та наборах правил, що впливає на їхню ефективність і точність. Деякі моделі можуть краще впоратися з певними типами помилок або мовними особливостями, тоді як інші можуть бути менш ефективними.

Також варто враховувати, що жодна модель не може гарантувати 100% коректність виправлення правописних помилок. Іноді вони можуть пропустити деякі помилки або, навпаки, внести зайві зміни до правильних слів. Тому, при використанні моделі корекції правопису, важливо бути обережним і перевіряти результати, особливо при обробці великого обсягу тексту. Рекомендується проводити додаткову перевірку та редагування тексту для виправлення можливих помилок, особливо у важливих документах або наукових текстах, де точність має велике значення. Як результат, маємо покращений текст, який буде застосовуватися для подальшої генерації PDF документа.

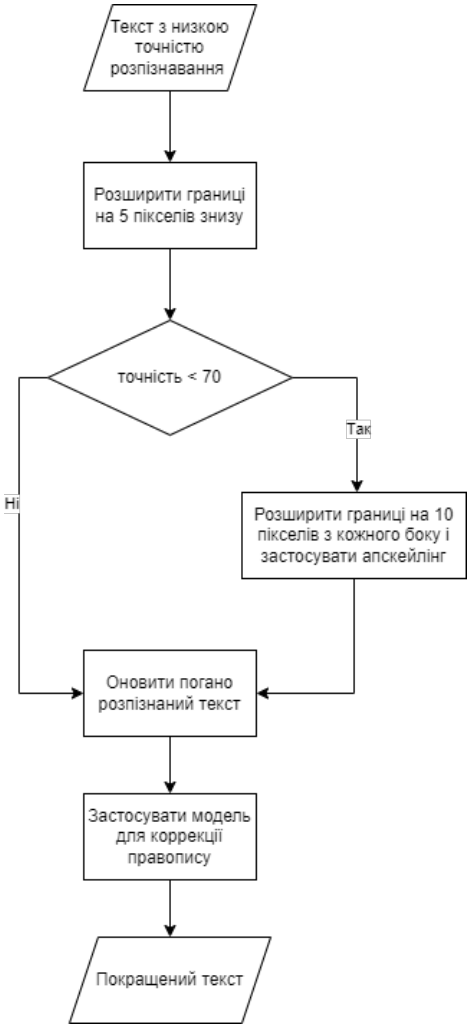


Рисунок 4.1 – Алгоритм роботи функції

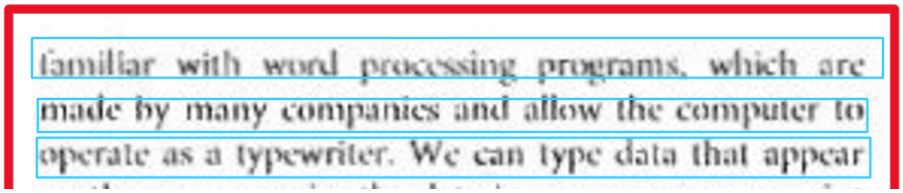


Рисунок 4.2 – Приклад тексту, який був розпізнаний з низькою точністю

4.1.5 Видалення шуму з зображення

Видалення шуму з зображення є важливою задачею в обробці зображень, оскільки шум може вплинути на якість та чіткість зображення. Шум може бути результатом різних факторів, таких як невідповідна освітленість, низька якість обладнання або стиснення зображень. Існує кілька методів видалення шуму з зображень, залежно від типу шуму та характеристик зображення. Наприклад, фільтри згладжування, такі як фільтр Гауса або медіанний фільтр, використовуються для зменшення шуму за рахунок розмиття пікселів. Ці фільтри допомагають згладити неправильність у значеннях пікселів, що сприяє зменшенню шуму. Ще існують статистичні методи використовуються для моделювання шуму та видалення його на основі статистичних характеристик зображення. Наприклад, алгоритм видалення шуму за допомогою адаптивного фільтра може використовувати статистику пікселів у визначеному околі для зменшення шуму та збереження деталей зображення. Ще нейромережі, зокрема глибокі нейронні мережі, стали популярними для видалення шуму з зображень. Ці мережі навчаються на великій кількості шумних та чистих зображень, щоб встановити зв'язок між шумом та його очищенням. Після навчання мережі можна застосовувати для видалення шуму з нових зображень. Використання глибоких нейронних мереж дозволяє досягти високої ефективності видалення шуму та збереження деталей зображення.

Ці моделі можуть розпізнавати шумові шаблони та ефективно їх прибирати, покращуючи якість зображення.

При видаленні шуму з зображень важливо забезпечити баланс між зменшенням шуму та збереженням деталей. Занадто агресивне видалення шуму може призвести до втрати важливих деталей та розмиття зображення, тоді як недостатньо ефективне видалення може не забезпечити достатньої якості. Окрім технічного видалення шуму, також можна застосовувати методи післяобробки, які поліпшують якість зображення. Усі ці методи видалення шуму мають на меті поліпшити якість зображення шляхом позбавлення його від шуму та збереження деталей та текстур. Вибір конкретного методу залежить від типу шуму, характеристик зображення та вимог до якості результату.

4.1.6 Морфологічні трансформації в OpenCV

Морфологічні трансформації [10] є важливою складовою обробки зображень і використовуються для зміни форми, розміру та структури об'єктів на зображенні. В бібліотеці OpenCV доступні різноманітні функції для виконання морфологічних операцій. Дві основні морфологічні операції, що підтримуються в OpenCV, – це розширення (dilation) та ерозія (erosion). Розширення розширює контури об'єктів на зображенні, використовуючи ядро або шаблон, і дозволяє заповнити порожнини та з'єднати розрізані об'єкти. Ерозія, навпаки, зменшує контури об'єктів шляхом зменшення їхнього розміру та видалення дрібних деталей.

У OpenCV для виконання морфологічних трансформацій використовуються два основних елементи: ядро і операційний елемент. Ядро – це невелика матриця або структура, яка задає форму операції. Вона переміщується по всьому зображенню, і для кожного пікселя зображення виконується операція, використовуючи значення пікселів, що потрапляють у ядро. Операційний елемент – це саме ядро, до якого додана інформація

про центр, що використовується для визначення точки прикладання ядра до зображення. Окрім розширення та ерозії, OpenCV також підтримує інші морфологічні операції, такі як відкриття (opening), закриття (closing), морфологічне відмикання (morphological gradient) та інші. Ці операції виконуються за допомогою комбінацій розширення та ерозії, що дозволяє досягти більш складних результатів обробки зображень. Наприклад, відкриття – це послідовне застосування ерозії та розширення, яке дозволяє видалити малий шум та замкнути малі розриви в контурі об'єктів. Закриття, навпаки, використовує послідовність розширення та ерозії для заповнення порожнин та з'єднання розрізаних об'єктів. Для використання морфологічних трансформацій в OpenCV потрібно вказати вхідне зображення та операційний елемент. Операційний елемент може бути вказаний як попередньо визначена матриця, вбудована форма або створена користувачем. За допомогою цих операцій можна здійснювати фільтрацію зображень, видалення шуму, розширення та ерозію об'єктів, виокремлення контурів та багато іншого.

Морфологічні трансформації є потужним інструментом обробки зображень, який дозволяє виконувати різні операції для покращення якості зображень та виділення потрібних об'єктів. Застосування цих трансформацій у поєднанні з іншими методами обробки зображень може привести до отримання бажаних результатів у різних задачах, таких як сегментація об'єктів, виявлення контурів, видалення шуму та інше.

4.2 Модель BART

Модель BART [11] (Bidirectional and AutoRegressive Transformers) є однією з передових моделей у сфері корекції правопису та генерації тексту. Вона базується на трансформерній архітектурі і використовує методи зворотнього та авторегресивного моделювання. BART використовується для завдань, пов'язаних з автоматичною корекцією правопису [12],

включаючи виправлення орфографічних помилок, граматичних недоліків та інших видів текстових недоліків. Ця модель може бути застосована до різних типів тексту, включаючи новини, статті, листування тощо. Основна особливість моделі BART полягає у тому, що вона може працювати як у режимі зворотного моделювання (розпізнавання помилок), так і у режимі авторегресії (генерація виправленого тексту). У режимі зворотного моделювання, модель отримує вхідний текст з помилками та спробує знайти та виправити помилки. У режимі авторегресії, модель генерує виправлений текст, використовуючи контекст та правила граматики.

Одним з основних переваг BART є те, що вона може використовуватися в поєднанні з іншими моделями та алгоритмами для покращення результатів корекції правопису. Також перевагою моделі BART є її гнучкість та універсальність. Вона може бути застосована для різноманітних завдань генерації тексту, таких як стиснення тексту, переклад, розпізнавання мови та багато інших. Також, модель BART здатна ефективно вирішувати завдання корекції правопису завдяки використанню механізмів автокодування та декодування, що дозволяє їй "запам'ятовувати" та узагальнювати інформацію з попередніх текстових послідовностей. Також вона здатна самостійно навчатися на великих обсягах тексту, що дозволяє покращувати її якість з часом. Важливо зазначити, що модель BART є частково переносною на завдання розпізнавання тексту та конвертації в PDF формат, оскільки вона може розуміти структуру та зміст тексту. Це дозволяє використовувати її для автоматичного вирішення проблеми некоректного форматування та граматичних помилок, що покращує якість конвертації документів. А також, модель BART є відносно швидкою та легкою в розгортанні, що дозволяє її застосовувати в різних сценаріях та на різних пристроях. Вона також здатна працювати з різними мовами та складними мовними конструкціями, що дозволяє їй застосовуватися для розпізнавання та корекції тексту на різних рівнях складності.

Окрім своїх переваг, модель BART також має деякі недоліки, які варто враховувати. Один з основних недоліків моделі BART полягає в її великому розмірі та складнощах з обробкою. Вона має значну кількість параметрів, що потребує великих обчислювальних ресурсів для тренування та використання. Це може бути обмеженням для застосування моделі на пристроях з обмеженими ресурсами. Крім того, модель BART має тенденцію до генерування надмірно складних речень або дублювання інформації, особливо при тривіальних або неоднозначних вхідних даних. Це може призводити до збоїв в корекції правопису та зменшувати читабельність результатів. Такі проблеми потребують додаткової обробки та фільтрації результатів моделі, щоб забезпечити точність та прийнятну якість виходу. Наступним недоліком моделі BART є її обмежена здатність до розуміння контексту, особливо в складних або многозначних текстах. Вона може показувати недостатню чутливість до семантичних нюансів, що може призводити до неточностей або неправильностей у корекції правопису. Такі проблеми можуть виникати у випадках, коли контекст важко визначити або коли є потреба в глибшому семантичному розумінні.

Враховуючи ці недоліки, важливо бути уважним при використанні моделі BART для корекції правопису. Вона може забезпечувати високу якість результату, але потребує додаткової уваги та обробки для вирішення можливих проблем. Розуміння обмежень та недоліків моделі BART допоможе забезпечити ефективне та точне використання її в контексті конкретних завдань корекції правопису.

4.3 Генерація документів програмним методом

Проблема генерації PDF документів програмним методом [13] є актуальною та широко обговорюваною у сучасному інформаційному середовищі. Цей процес включає перетворення різноманітних даних, таких як текст, графіка, таблиці та інші об'єкти, в структурований формат PDF,

який забезпечує зручну і надійну обробку та обмін даними. Проте, при програмній генерації PDF документів з'являються деякі виклики, які можуть суттєво вплинути на якість та надійність створених документів.

Однією з основних проблем [14] є збереження форматування та структури даних при перетворенні в PDF формат. При відсутності відповідного алгоритму або некоректній обробці даних, можуть виникати проблеми зі збереженням кольорів, шрифтів, розміщенням об'єктів та іншими елементами документа. Це може призвести до втрати важливої інформації та зниження зрозумілості документа для користувача. Іншою поширеною проблемою є підтримка різних типів об'єктів та їх відображення в PDF документах. В залежності від використаного програмного середовища та бібліотек, може бути обмежена підтримка деяких графічних форматів, векторних зображень, шрифтів або інших об'єктів. Це може призвести до некоректного відображення або втрати даних під час генерації PDF документів.

Крім того, ефективна обробка великих обсягів даних та швидка генерація PDF документів є ще однією важливою проблемою. Залежно від складності та розміру вихідних даних, застосування програмних методів для генерації PDF документів може бути часо- та ресурсоємким завданням. Обробка великої кількості даних, включаючи текст, зображення, таблиці та інші елементи, може призвести до значного збільшення часу виконання процесу генерації. А також, неправильна оптимізація алгоритмів та використання неефективних методів можуть призвести до зайвих навантажень на системні ресурси, такі як процесор та оперативна пам'ять, що може вплинути на швидкість та продуктивність програми.

Ще однією проблемою, пов'язаною з генерацією PDF документів програмним методом, є забезпечення безпеки та захисту інформації. PDF документи можуть містити конфіденційні дані, такі як особисті відомості, банківські реквізити тощо. Забезпечення високого рівня безпеки, такого як шифрування даних, контроль доступу та підписи, є важливим аспектом у

процесі генерації PDF документів. Неналежна реалізація заходів безпеки може вести до витоку конфіденційної інформації та порушення правил захисту даних.

Узагальнюючи, генерація PDF документів програмним методом стикається з рядом проблем, включаючи збереження форматування та структури даних, підтримку різних типів об'єктів, ефективну обробку великих обсягів даних та забезпечення безпеки і захисту інформації. Для успішного вирішення цих проблем необхідно використовувати ефективні алгоритми, оптимізовані методи та застосовувати сучасні підходи до генерації PDF документів, забезпечуючи надійність, якість та швидкість роботи. Вдосконалення процесу генерації PDF документів є важливим завданням, що дозволить забезпечити якісну та надійну обробку даних, збереження форматування та структури, ефективну роботу з великими обсягами інформації та забезпечення безпеки даних.

5 ЕКСПЕРИМЕНТИ ПО ВДОСКОНАЛЕННЮ

5.1 Заміна Tesseract OCR на trOCR

У рамках проведення експерименту та порівняння результатів, замість Tesseract OCR було використано trOCR. Перехід до використання trOCR був зумовлений його спеціалізацією на розпізнаванні рукописного тексту, що відкриває нові можливості для точного аналізу такого типу даних.

Використання trOCR дозволило провести детальні порівняльні дослідження, оцінити його ефективність та визначити його переваги порівняно з Tesseract OCR. При проведенні експерименту були використані однакові набори документів для обох систем з метою об'єктивного порівняння їх результатів. При використанні trOCR вдалося отримати більш точні результати розпізнавання рукописного тексту, порівняно з Tesseract OCR. Модель trOCR показала високу точність і здатність розпізнавати різноманітні стилі рукописного почерку. Це робить його корисним інструментом для завдань, де рукописний текст є ключовим. А також, це дало змогу зрозуміти особливості роботи цієї моделі, виявити її переваги та недоліки в контексті конкретної задачі конвертації сканованих документів до формату PDF з використанням методів OCR.

Отже, використання trOCR у експерименті дало можливість здійснити об'єктивне порівняння з Tesseract OCR і оцінити його ефективність для розпізнавання рукописного тексту. Такий підхід розширює спектр можливостей у використанні OCR-технологій та дозволяє підібрати оптимальний інструмент для конкретної задачі.

Після здійснення порівняння можна зробити висновок, що обидві системи мають свої переваги та обмеження, які варто враховувати при виборі підходящого рішення. Наприклад, Tesseract OCR є однією з найпопулярніших та широко використовуваних систем OCR. Вона відома своєю високою швидкістю та продуктивністю, що робить її ідеальним

варіантом для обробки великих обсягів тексту. Tesseract OCR також підтримує розпізнавання тексту в різних мовах та має широкий спектр функціональних можливостей. Вона постійно оновлюється та вдосконалюється спільнотою розробників, що сприяє її надійності та якості розпізнавання. trOCR, з іншого боку, володіє унікальною можливістю розпізнавання рукописного тексту. Це дозволяє їй опрацьовувати документи, що містять рукописні записи, які не підлягають легкому автоматичному розпізнаванню. trOCR використовує складні алгоритми та машинне навчання для досягнення високої точності розпізнавання рукописного тексту. Рукописний текст може бути складним завданням для OCR систем, оскільки він може містити незвичайні форми символів та погано читані написи.

Однак, слід зазначити, що trOCR може працювати повільніше порівняно з Tesseract OCR. Розпізнавання рукописного тексту вимагає більш складних алгоритмів та обчислювальних ресурсів, що може призвести до більшого часу обробки. Це може становити проблему, особливо якщо важливою вимогою є швидкість опрацювання документів. Тому, якщо час є критичним фактором, і рукописний текст не є пріоритетним, Tesseract OCR може бути більш ефективним варіантом завдяки його швидкості та добрій продуктивності.

Щодо якості розпізнавання, обидві системи мають високу точність, але на неї може впливати різноманітність стилів письма та якості сканування документів. Tesseract OCR добре працює з чітким друкуваним текстом, але може зазнавати певних труднощів у випадку з рукописним текстом або текстом низької якості. З іншого боку, trOCR спеціалізується на розпізнаванні рукописного тексту, що дозволяє йому досягти високої точності в цьому напрямку, але його ефективність може залежати від зрозумілості та чіткості рукописного почерку.

Отже, при порівнянні trOCR та Tesseract OCR варто враховувати їх основні переваги та обмеження. Tesseract OCR є швидшим і ефективним для

обробки великих обсягів тексту та друкуваного тексту, забезпечуючи загальну високу якість розпізнавання. trOCR, натомість, є корисним для розпізнавання рукописного тексту, що дає йому перевагу у випадках, коли це є важливим. Остаточний вибір між цими системами повинен залежати від конкретних потреб та вимог проекту, враховуючи фактори швидкості, якості розпізнавання та типів тексту, які потрібно обробити.

5.2 Випробування додаткового функціоналу

5.2.1 Знаходження зображень на сканах документів

Знаходження зображень на сканах документів є одним із завдань обробки зображень, яке полягає в автоматичному виявленні та виділенні графічних елементів на скані. Цей процес має на меті виявити присутність зображень у документі, визначити їх межі та виділити їх для подальшого використання або обробки. Алгоритми знаходження зображень на сканах документів зазвичай базуються на комп'ютерному зору та аналізі зображень. Ці алгоритми використовують різні методи, такі як виявлення контурів, аналіз яскравості та текстурних характеристик, машинне навчання та нейромережі, для автоматичного виділення зображень на скані.

Процес знаходження зображень може бути реалізований у двох основних кроках: перша стадія – виявлення потенційних зображень на скані, а друга стадія – підтвердження чи відкидання знайдених областей як справжніх зображень. У першій стадії застосовуються різні алгоритми для виявлення потенційних областей, що містять зображення, на основі характеристик зображення. У другій стадії використовуються додаткові критерії та алгоритми для підтвердження або відкидання знайдених областей.

Знаходження зображень на сканах документів має різні практичні застосування. Наприклад, у сфері автоматизованого розпізнавання

документів (OCR), цей процес може бути використаний для автоматичного виділення фотографій або ілюстрацій зі сканів текстових документів. У медичній сфері знаходження зображень може використовуватись для виділення медичних зображень, таких як рентгенограми або знімки КТ, для подальшого аналізу та обробки. Також знаходження зображень може бути використано для створення зображень-переглядів, які відображають зразки документів, що містять зображення, для подальшого навчання моделей розпізнавання або класифікації.

За допомогою знаходження зображень на сканах документів можна автоматизувати процеси обробки і аналізу документів, що містять графічні елементи. Це дозволяє покращити ефективність роботи з документами, зменшити зусилля, потрібні для ручної обробки, та забезпечити більш точні результати. Знаходження зображень на сканах документів є важливим етапом в процесі обробки та аналізу документів, особливо у випадках, коли важлива інформація міститься саме на зображеннях, таких як фотографії, діаграми або графіки.

5.2.2 Виправлення перспективи сканів документів

Виправлення перспективи на сканах документів є важливим кроком у процесі обробки та аналізу документів. Перспектива може виникати через кутове спотворення зображень, яке зазвичай виникає під час фотографування або сканування документів під кутом. Це може спотворювати форму та пропорції об'єктів на зображенні та ускладнювати подальшу обробку та аналіз.

Для виправлення перспективи на сканах документів використовуються різні алгоритми та методи. Один з найпоширеніших методів – це геометрична трансформація, відома як перетворення афінної матриці. Цей метод дозволяє змінити перспективу зображення, зміщуючи його точки та розтягуючи його залежно від потреби.

Процес виправлення перспективи включає декілька кроків. Спочатку знаходяться ключові точки або розпізнаються області зображення, які можуть вказувати на перспективні спотворення. Наступним кроком є обчислення афінної матриці перетворення та її застосування до зображення для виправлення перспективи. Це може включати зсув, поворот та масштабування зображення залежно від знайдених параметрів перспективи. Зображення може бути обрізано або розтягнуто, щоб відновити правильну форму та пропорції об'єктів. Виправлення перспективи на сканах документів допомагає забезпечити правильну геометрію та пропорції об'єктів на скані документу, що полегшує подальшу обробку та аналіз. В результаті, відновлення правильної геометрії зображення допомагає усунути спотворення та деформацію тексту, що може виникати через некоректне положення камери або сканера під час створення документа.

Один з популярних алгоритмів для виправлення перспективи на сканах документів – це алгоритм прямокутної перспективи. Він базується на знаходженні прямокутників або квадратів, що мають фіксовані розміри або відомі пропорції. За допомогою цих прямокутників алгоритм визначає кути перспективи та виконує перетворення, щоб виправити спотворення та відновити прямі лінії та прямокутні форми. Після виправлення перспективи можна застосувати додаткові обробки, які покращують зображення перед подальшою обробкою тексту. Наприклад, можна виконати фільтрацію шуму, покращити контрастність або збалансувати кольори. Ці кроки допомагають зробити текст більш чітким та зрозумілим для наступного етапу розпізнавання. Виправлення перспективи на сканах документів важливо для отримання якісних результатів розпізнавання тексту. Воно допомагає усунути спотворення, зберегти форму об'єктів та покращити зрозумілість текстової інформації. Це важливий крок у процесі обробки документів, який сприяє точності та надійності розпізнавання тексту на сканах документів.

ВИСНОВКИ

В ході виконання дипломної роботи було проведено порівняння різних інструментів для конвертації сканованих документів до формату PDF з використанням методів OCR. Були порівняні такі популярні інструменти, як Tesseract OCR, Google Vision, ABBYY OCR, MMOCR та TrOCR. Для порівняння були враховані критерії точності розпізнавання тексту, швидкості обробки, можливостей розпізнавання рукописного тексту та інші. Також, рамках дипломної роботи було реалізовано ефективний алгоритм екстракції тексту зі сканованих документів з використанням таких технік, як апскейлінг зображень, морфологічні трансформації та видалення шуму з зображення. Ці методи значно покращують якість розпізнавання тексту та допомагають отримати більш точні результати.

Крім того, для поліпшення результатів розпізнавання тексту були використані моделі для корекції правопису. Ці моделі допомагають уникнути можливих помилок та покращити фінальний результат розпізнавання. За допомогою бібліотеки PyLatex було реалізовано відтворення отриманого тексту у форматі PDF. Це дозволяє зручно зберігати та поділитися результатами розпізнавання тексту. Додатково, було проведене порівняння роботи Tesseract OCR та trOCR, інструменту, який вміє розпізнавати рукописний текст, але працює трохи повільніше. Ці експерименти дали можливість зрозуміти переваги та обмеження кожного з цих інструментів та з'ясувати, що використання trOCR може бути більш доцільним для певних завдань, де потрібно розпізнати рукописний текст, а Tesseract OCR забезпечить кращі результати при розпізнаванні великого об’єму документів з печатним шрифтом.

За допомогою експериментів було також досліджено знаходження зображень на сканах документів та виправлення перспективи сканів. Ці етапи виявилися дуже важливими для покращення якості сканів та поліпшення їх естетичного вигляду. Знаходження зображень допомагає

виділити графічні елементи на документах, такі як фотографії або діаграми, що може бути корисно для подальшої обробки та інтерпретації. Виправлення перспективи зображень дозволяє усунути спотворення, що виникає внаслідок неправильного положення документу під час його сканування.

В результаті виконаної дипломної роботи було досягнуто високої точності розпізнавання тексту зі сканованих документів та його ефективного відтворення у форматі PDF. Проведені експерименти з різними інструментами та методиками дозволили виявити переваги та недоліки кожного підходу. Використання trOCR для розпізнавання рукописного тексту, апскейлінгу зображень, морфологічних трансформацій та моделей корекції правопису показали суттєве покращення якості результатів. Отримані результати підтверджують перспективність розробленого алгоритму та підходу для конвертації сканованих документів до формату PDF з використанням методів OCR. Цей проект має потенціал для подальшого розвитку та вдосконалення, зокрема шляхом вдосконалення алгоритмів розпізнавання тексту, оптимізації швидкодії та розширення функціональності для вирішення різноманітних завдань обробки документів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Lakshmanan V., Görner M., Gillard R. Practical Machine Learning for Computer Vision: End-To-End Machine Learning for Images. O'Reilly Media, Incorporated, 2021. 482 p.

2. Optical Character Recognition (OCR) with Document AI (Python) | Google Codelabs. *Google Codelabs*. URL: https://codelabs.developers.google.com/codelabs/docai-ocr-python#0 (дата звернення: 01.03.2023).

3. Hanna K. T. What is Portable Document Format (PDF)?. *WhatIs.com*. URL: https://www.techtarget.com/whatis/definition/Portable-Document-Format-PDF (дата звернення: 03.03.2023).

4. Using Tesseract OCR with Python - PyImageSearch. *PyImageSearch*. URL: https://pyimagesearch.com/2017/07/10/using-tesseract-ocr-python/(дата звернення: 14.03.2023).

5. TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models. *arXiv.org*. URL: https://arxiv.org/abs/2109.10282 (дата звернення: 15.03.2023).

6. Papers with Code - TrOCR Explained. *The latest in Machine Learning | Papers With Code*. URL: https://paperswithcode.com/method/trocr (дата звернення: 23.03.2023).

7. Kopka H. Guide to LATEX. 4th ed. Boston : Addison-Wesley, 2004. 597 p.

8. Zelic F. Tesseract OCR in Python with Pytesseract andOpenCV. *Nanonets AI & Machine Learning Blog*. URL: https://nanonets.com/blog/ocr-with-tesseract/ (дата звернення: 26.03.2023).

9. OpenCV Super Resolution with Deep Learning - PyImageSearch. *PyImageSearch*.

URL: https://pyimagesearch.com/2020/11/09/opencv-super-resolution-with-deep-learning/ (дата звернення: 01.04.2023).

10. Improving the quality of the output. *tessdoc*. URL: https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html (дата звернення: 05.04.2023).

11. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv.org*. URL: https://arxiv.org/abs/1910.13461 (дата звернення: 17.04.2023).

12. Mansar Y. Automatic Grammar and Spelling Correction with PyTorch–Part 1: A Baseline. *Medium*. URL: https://towardsdatascience.com/automatic-grammar-and-spelling-correction-with-pytorch-part-1-a-baseline-d97b7202de74 (дата звернення: 19.04.2023).

13 How to Create PDF in Python: A Comprehensive Guide. *Aspose Blog*. URL: https://blog.aspose.com/pdf/create-pdf-files-in-python/(дата звернення: 20.04.2023).

14. Khorasani M. How to Generate Automated PDF Documents with Python. *Medium*. URL: https://towardsdatascience.com/how-to-generate-automated-pdf-documents-with-python-55981f4d9e3(дата звернення: 21.04.2023).

15. Bengio Y., Courville A., Goodfellow I. Deep Learning. MIT Press, 2016. 800 p.

16. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, Incorporated, 2022.

17. S P. Extract Data From PDF: 5 PDF Data Extraction Methods. *Nanonets AI & Machine Learning Blog*. URL: https://nanonets.com/blog/extract-data-from-pdf/(дата звернення: 24.04.2023).

ДОДАТОК А

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	_72_ с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Ляубе О.О.			Конвертація сканованих документів до формату PDF з використанням методів OCR	Шифр групи	Код напр./спец.
Перевірив		ШевченкоО.Ю				СШІздм-21-1	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Філатов В.О.					