МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ

(підпис)

15 грудня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук,

освітньо-професійної програми «Інформатика»

на тему: «Інформаційна технологія обробки рахунків-фактур з використанням оптичного розпізнавання символів для інтеграції з Microsoft Dynamics 365 Business Central»

здобувача групи ІН.м-22 Аніщенка Олега Олексійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Олег АНІЩЕНКО

(підпис)

Керівник

асистентка кафедри комп'ютерних наук,

к.ф.-м.н. Ольга ШУТИЛЄВА _____

(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН.м-22 Аніщенка Олега Олексійовича

1. Тема роботи: «Інформаційна технологія обробки рахунків-фактур з використанням оптичного розпізнавання символів для інтеграції з Microsoft Dynamics 365 Business Central» затверджую наказом по СумДУ від *«06» грудня 2023 р. № 1412-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 15 грудня 2023 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1)Аналіз проблеми предметної області, постановка й формулювання завдань дослідження. 2).Огляд та аналіз теоретичних питань. 3). Проєктування та розробка інформаційної технології 4). Аналіз результатів*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 2023 р.

Завдання прийняв до виконання _____ (підпис) Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Визначення предмету дослідження, актуальності та постановка задачі*	06.11.2023-13.11.2023	
2	*Виконання огляду та аналізу теоретичних питань*	14.11.2023-21.11.2023	
3	*Проєктування та розробка інформаційної технології*	22.11.2023-06.12.2023	
4	*Аналіз результатів розробки інформаційної технології*	07.12.2023-08.12.2023	
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*	08.12.2023-17.12.2023	

Здобувач вищої освіти _____ (підпис) Керівник _____ (підпис)

АНОТАЦІЯ

Записка: 50 стор., 30 рис., 1 додаток, 1 табл., 20 використаних джерел.

Обґрунтування актуальності теми роботи – тема кваліфікаційної роботи є актуальною, оскільки автоматизація обробки рахунків-фактур є необхідною для кожного бухгалтера для зменшення вірогідності помилки та збереження часу.

Об'єкт дослідження – процес проєктування інформаційної системи обробки рахунків-фактур за допомогою оптичного розпізнавання символів.

Мета роботи – розробка інформаційної системи для автоматизації заповнення рахунку-фактури для інтеграції з Business Central.

Методи дослідження – методи порівняння моделей для класифікації та локалізації зображень, аналіз технології оптичного розпізнавання символів, аналіз літератури.

Результати – проаналізовано можливі моделі для навчання та технології для оптичного розпізнавання символів. Розроблена інформаційна технологія обробки рахунків-фактур за допомогою оптичного розпізнавання символів, а також інтегрована в систему Microsoft Dynamics 365 Business Central.

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ, ОБРОБКА РАХУНКІВ-ФАКТУР,
BUSINESS CENTRAL, OCR, PYTHON

ЗМІСТ

ВСТУП

Поява нових інформаційних технологій значним чином покращує життя багатьох людей. Автоматизація інтегрована в велику кількість систем, що полегшують життя, зменшують ризики помилку та економлять гроші. До прикладу, майже щодня, бухгалтеру потрібно вести звітність по покупкам, і для цього йому потрібно взяти рахунок-фактуру, відкрити програму, створити запис необхідного типу та власноруч вносити з дані з кожного отриманого документу. Уявімо, що йому потрібно перенести сотню документів. Робити це вручну не лише довго та важко, а ще й небезпечно, оскільки роблячи таку монотонну роботу, швидко втрачається фокус, що може призвести до невідповідності даних в системі.

Така ситуація є ідеальним прикладом того, які саме дії можна автоматизувати. Автоматизація перенесення даних з рахунків-фактур в системи обліку є актуальною бо це значно знижує час для обробки рахунків-фактур, дозволяючи бухгалтерам та фінансовим аналітикам зосередитися на більш складних та стратегічних завданнях. Автоматизовані системи зменшують ризик помилок у даних, що підвищує якість обліку та фінансової звітності. Більше того, зменшення часу на обробку рахунків-фактур та помилок при введенні даних допомагає знизити загальні операційні витрати компанії.

Таким чином, автоматизація процесу перенесення даних з рахунків-фактур є ключовим елементом для підвищення ефективності, зниження витрат та покращення точності фінансового обліку в сучасному бізнесі.

Якщо поєднати всі аргументи, наведені вище, можна зрозуміти актуальність автоматизації дій бухгалтера за допомогою інформаційної технології обробки рахунків-фактур з використанням оптичного розпізнавання символів.

Для виконання даного завдання, потрібно чітко визначити цілі, провести аналіз існуючих технологій та алгоритмів, виявити їхні переваги та недоліки та створити своє ідеальне рішення цієї проблеми.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Сучасний стан та тенденції обробки рахунків-фактур

Рахунок-фактура – це документ, що інформує покупця про товари та послуги, які він планує купувати. Він зазвичай складається з номера рахунку, дати його виставлення, дату до якої потрібно сплатити, контакти компанії постачальника або вендора, опис товарів або послуг, їхня ціна та загальна вартість. Також зазвичай такі рахунки містять реквізити, по яким потрібно сплатити.

Обробка рахунків-фактур є звичною справою для бухгалтера, оскільки щодня компанії можуть надходити такі рахунки від постачальників послуг або товарів. Сам процес обробки даних рахунків полягає в тому, що бухгалтер має перенести відомості з рахунку-фактури до головного бухгалтерської книги для коректного ведення звітності.

Сьогодні для ведення звітності використовується різне програмне забезпечення, яке дає змогу вносити необхідні проводки в систему, автоматичне створення балансуючих проводок, комбінування даних для квартальних та річних звітів та багато інших функцій, які полегшують життя бухгалтерам, зменшують можливість помилки та відмінностей в фінансовій інформації.

Основними застосунками для ведення обліку зараз є Odoo, Xero, FreshBooks та інші. Всі вони дозволяють створювати всі необхідні проводки, формувати звіти, підтримують інтеграції з банками та поштовими компаніями.

Microsoft Dynamic 365 Business Central (далі Business Central) це комплексне рішення для малого та середнього бізнесу, перша версія якого вийшла ще в далекому 1983 році під назвою «PC Plus». Дане програмне забезпечення, на початку, застосовувалося лише в деяких країнах Європи, але в 2002 році компанію придбала Microsoft та зробила його доступним майже по всьому світу.

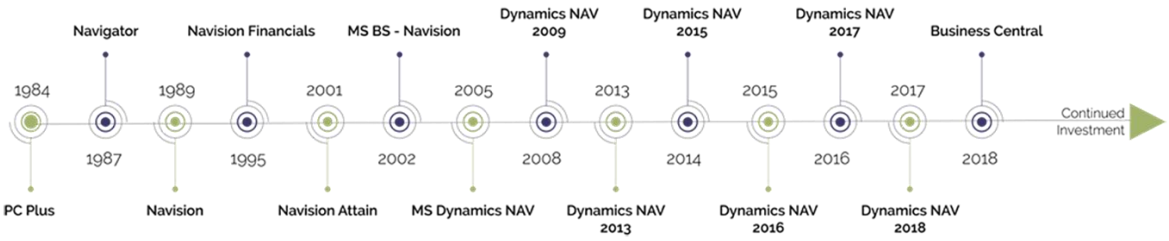

Рисунок 1.1 – Історія розвитку Business Central [1]

Однією з ключових особливостей Business Central є можливість використовувати цей застосунок для більшості потреб бізнесу, а також спрощення та прискорення ведення бізнесу. Business Central це не просто бухгалтерія, а ще й контроль ланцюга постачання, ведення кадрів та закупівлі. Він також містить інтегрований штучний інтелект для написання опису для товарів, використовуючи передові маркетингові тренди. Також має інтеграції з Excel, що спрощує введення інформації в систему.

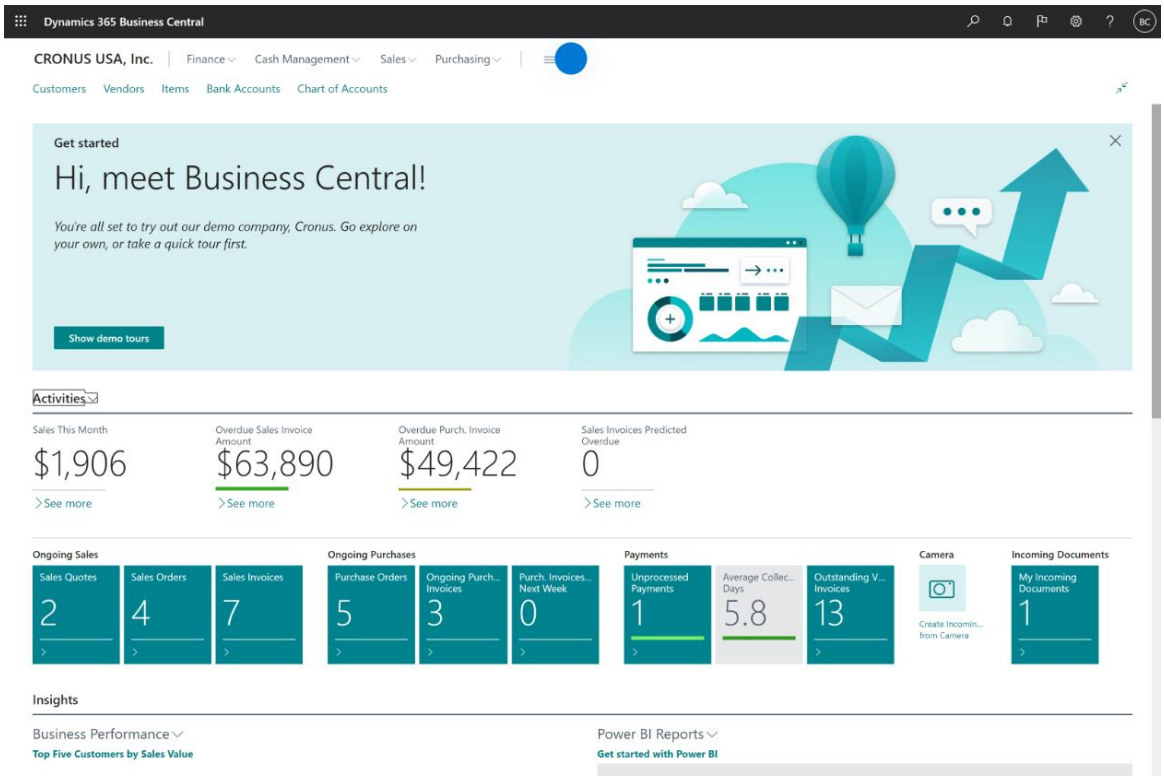

Рисунок 1.2 – Основна сторінка Business Central

Більше того, це програмне забезпечення є розширюваним, що дає змогу додавати необхідний функціонал власноруч та покращувати існуючий. Розробка відбувається за допомогою мови програмування AL.

Проте, стандартний функціонал жодного з перелічених застосунків, включаючи Business Central, не має можливості автоматично оброблювати рахунки-фактури, без інтеграції зі сторонніми сервісами.

Вищезазначена проблема завжди була ключовим стимулом, що стимулюють розробку інструментів для автоматичної обробки документів, в тому числі рахунків-фактур. Розвиток машинного навчання та нейронних мереж значно полегшив та пришвидшив розробку та інтеграцію розпізнавання ключової інформації на документах, для їхньої подальшої обробки.

1.1.1 Машинне навчання

Машинне навчання та штучний інтелект займають значущу роль в питаннях, пов’язаних з покращенням бізнесу, оскільки це покращує ефективність маркетингу, оперативну ефективність, зменшує витрати та економить дорогоцінний час.

Машинне навчання – це галузь штучного інтелекту , яка фокусується на побудові систем, здатних вчитися та приймати рішення на основі даних. На відміну від традиційного програмування, де люди явно визначають правила, машинне навчання дозволяє комп'ютерам аналізувати і вчитися на даних, покращуючи їх продуктивність з часом без прямого втручання людини.[2]

Можна виділити декілька кроків, які описують процес машинного навчання:

- збір даних;
- попередня обробка даних;
- вибір моделі;
- тренування;
- тестування та перевірка.

1.1.2 Збір даних

Етап «Збір даних» є чи не найважливішим, оскільки визначає ефективність та точність моделі. Вибір джерел даних залежить від проблеми, яку потрібно вирішити. Дані можуть надходити з різноманітних джерел, як-от сенсори, логи веб-сайтів, бази даних, опитування користувачів тощо. Це можуть бути як вже готові набори даних, так і нові, створені власноруч. Також існує багато сервісів, які генерують синтетичні дані. Тут також потрібно правильно обрати місце, де будуть зберігатися дані, для легкого доступу до них в процесі навчання.

Отже, збір даних – це процес збору, вимірювання та аналізу точних даних з різних відповідних джерел, щоб знайти відповіді на проблеми дослідження, відповісти на питання, оцінити результати та спрогнозувати тенденції та ймовірності [3]. Точний збір даних необхідний для прийняття обґрунтованих бізнес-рішень, забезпечення якості та збереження цілісності досліджень. Під час збору даних дослідники повинні визначити типи даних, джерела даних та які методи використовуються.

1.1.3 Попередня обробка даних

Попередня обробка даних в машинному навчанні є критичним кроком, який значно впливає на ефективність моделей. Вона передбачає перетворення необроблених даних у чистий, зрозумілий формат, придатний для побудови та навчання моделей машинного навчання. Процес складний, охоплює різні методи і стратегії.

Перш за все, потрібно очистити дані, якщо вони некоректні або відсутні. Відсутні значення можуть бути заповнені за допомогою середніх або медіанних значень, відкинуті зовсім, або замінені на нуль.

1.1.4 Вибір моделі

Вибір правильної моделі машинного навчання – це складний процес, який включає в себе кілька ключових кроків, кожен з яких має вирішальне значення для успіху моделі в здійсненні точних прогнозів на невидимих даних. Подорож

починається з обробки даних, де першим кроком є поділ наявних даних на навчальні та тестові набори. Цей крок необхідний для управління недосконалостями, такими як обмежені дані, і для забезпечення того, щоб тестовий набір представляв нові, невидимі дані.

Після поділу даних, вибір відповідного алгоритму навчання має вирішальне значення. Це рішення залежить від характеру проблеми і передбачає встановлення гіперпараметрів, які є параметрами алгоритму навчання.

Після застосування алгоритму навчання до тренувального набору продуктивність моделі оцінюється за допомогою незалежного тестового набору. Цей крок має вирішальне значення для забезпечення неупередженої оцінки продуктивності моделі на нових даних. Вона передбачає використання моделі для прогнозування міток класу на тестовому наборі та порівняння цих прогнозів з основною правдою для оцінки точності або помилки моделі.[2]

Нарешті, отримавши оцінку продуктивності моделі на невидимих даних, весь набір даних може бути використаний для навчання. Цей підхід може потенційно поліпшити продуктивність узагальнення моделі, якщо вона не досягла повної потужності. Цей останній крок передбачає використання всіх доступних даних для максимальної продуктивності моделі. [2]

1.1.5 Тренування моделі

Навчання моделі машинного навчання включає в себе ряд кроків. Процес починається з поділу наявних даних на навчальні та тестові набори. Цей поділ має вирішальне значення, оскільки гарантує, що модель може бути оцінена на нових, невидимих даних. Тестовий набір повинен представляти ці нові дані і повинен використовуватися тільки один раз, щоб уникнути упередження при оцінці продуктивності узагальнення моделі. Загальні розділи для навчання та тестових наборів варіюються від 60/40 до 90/10, залежно від розміру набору даних.[4]

Після того, як приклади тестів будуть відкладені, наступний крок передбачає вибір відповідного алгоритму навчання для проблеми. Цей вибір має

вирішальне значення, оскільки він визначає підхід, який модель буде використовувати для вивчення навчальних даних. Тут значну роль відіграють гіперпараметри, які є параметрами алгоритму навчання. Їх потрібно вказувати вручну, оскільки вони не вивчаються під час процесу встановлення моделі. Вибір цих гіперпараметрів часто передбачає поєднання експертної інтуїції та налаштувань за замовчуванням, що надаються бібліотеками машинного навчання.

Після того, як алгоритм навчання встановив модель, наступним важливим кроком є оцінка її продуктивності. Саме тут незалежний тестовий набір стає інструментальним. Оскільки алгоритм навчання не піддається впливу цього тестового набору, він забезпечує неупереджену оцінку продуктивності моделі на нових даних. Тестовий набір використовується для прогнозування міток класів, які потім порівнюються з правильними мітками для оцінки точності узагальнення моделі або помилки.

Нарешті, після отримання оцінки продуктивності моделі на невидимих даних, тестовий набір може бути реінтегрований у тренувальний процес. Цей крок заснований на припущенні, що зразки є незалежними і однаково розподіленими. Очікується, що включення більш інформативних даних покращить продуктивність узагальнення моделі, якщо вона не досягла своєї потужності.

1.1.6 Тестування та валідація

Тестування моделі є дуже важливими в процесі машинного навчання. Для тестування та оцінки результатів існують декілька критеріїв:

- точність і відкликання. Для задачі бінарної класифікації приклади можна розділити на: Істинно позитивний, Хибно позитивний, Хибно негативний, Істинно негативний.

- крива ROC (крива робочої характеристики приймача) – це графік, що показує продуктивність класифікаційної моделі при всіх порогах класифікації [5]

- AUC означає «Зона під кривою ROC» – вимірює всю двовимірну область під усією кривою ROC від (0,0) до (1,1). [5]

Для отримання необхідної інформації з документів ключовими інструментами є технології отримання тексту з зображення або документу, а також класифікації інформації, для визначення тої, яка необхідна для вирішення конкретної проблеми.

1.1.7 Оптичне розпізнавання символів (Optical Character Recognition)

Оптичне розпізнавання символів (OCR) – це технологія, яка дозволяє комп'ютерам перетворювати зображення друкованого тексту (як відскановані документи, так і фотографії) у текст, який здатний прочитати комп’ютер. Ця технологія використовується для автоматизації обробки різних документів, таких як листи, рахунки, банківські виписки та інші документи, що містять текст. [6]

Перед обробкою зображення, OCR-системи зазвичай виконують ряд операцій попередньої обробки, щоб підвищити якість зображення і відокремити текст від фону. Це може включати корекцію нахилу, видалення шуму, нормалізацію контрасту та ін. Після чого відбувається процес зчитування та перетворення символів в цифровий формат (див. рис. 1.3).

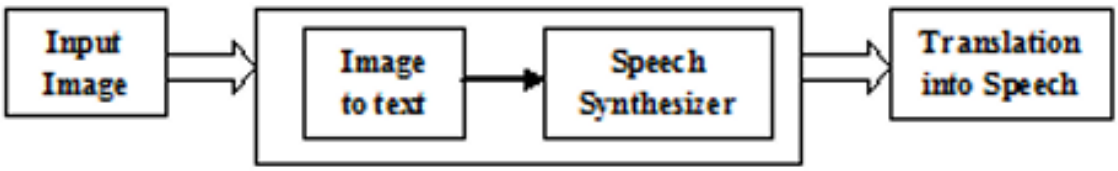

Рисунок 1.3 – Блок-діаграма OCR [7]

Основним двигуном для оптичного розпізнавання символів є Tesseract, який був вперше реалізований в 2005 році, як проєкт з відкритим кодом.

Обробка природної мови (Natural Language Processing, NLP) – це галузь комп'ютерних наук і штучного інтелекту, яка займається забезпеченням здатності комп'ютерів розуміти, інтерпретувати та реагувати на людську мову

таким чином, щоб це було значущим і корисним. Вона включає в себе ряд обчислювальних методів для аналізу та представлення людської мови з різних текстів. Ці тексти, які можуть бути усними або письмовими, зазвичай походять з реального використання, а не створені спеціально для аналізу. [8]

NLP охоплює різні рівні лінгвістичного аналізу, що відображає багато процесів, відомих в розумінні та виробленні людської мови. Кожен рівень аналізу передає різні типи значень, і системи NLP можуть використовувати різні комбінації цих рівнів. Мета – досягти обробки мови для різноманітних застосувань, включаючи пошук інформації, машинний переклад та відповіді на запитання та інші. [9]

1.1.8 Штучні нейронні мережні(Artificial Neural Network)

Дані мережі, як можна побачити з назви, мають імітувати дію біологічного нейрону, як показано на рисунку 1.4, тобто приймати багато різних сигналів x_i від сусідніх нейронів і обробляти їх заздалегідь визначеним способом. У залежності від результату цієї обробки, нейрон вирішує віддавати зворотній сигнал y_i чи ні. Вихідний сигнал може бути 0 або 1, якщо це двійковий штучних нейрон, або дійсне значення в межах від 0 до 1. [10]

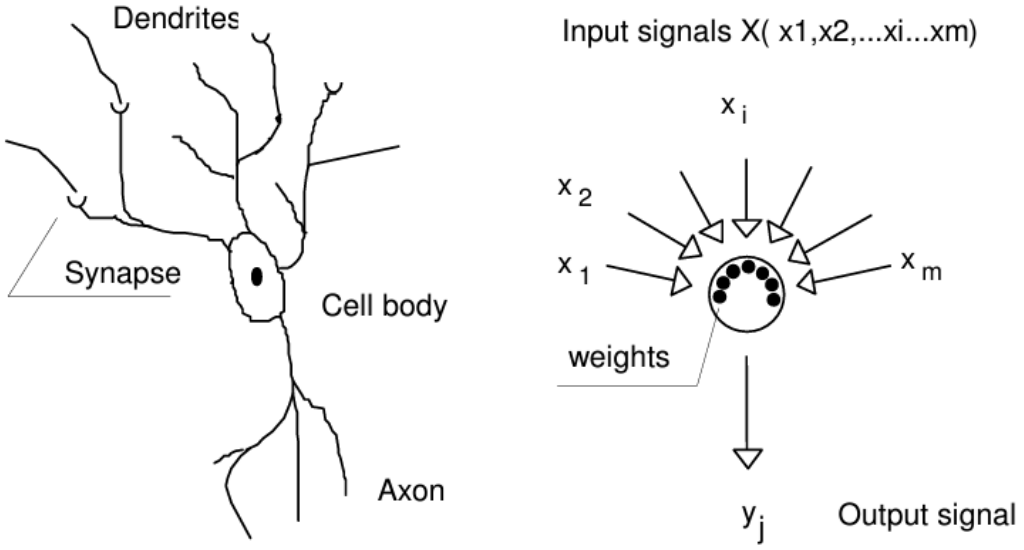

Рисунок 1.4 – Порівняння біологічного на штучного нейронів [10]

Переважно з історичної точки зору функція, яка обчислює вихід з m-вимірного вектору входу X, f(X), розглядається як складається з двох частин. Перша частина оцінює так званий нетто-вхід, Net, тоді як друга передає нетто-вхід Net нелінійним чином до вихідного значення y. Перша функція є лінійною комбінацією вхідних змінних, x1, x2, ... xi, ... xm, помножених на коефіцієнти, wji, які називаються вагами, тоді як друга функція служить як функція передачі, оскільки вона передає сигнал(и) через аксон нейрона до дендритів інших нейронів. [10]

Named Entity Recognition (NER) – це галузь обробки природної мови (NLP), яка фокусується на ідентифікації та класифікації іменованих сутностей у тексті на заздалегідь визначені категорії, такі як імена осіб, організації, місця розташування, вирази часу, величини, грошові цінності, відсотки тощо. [11] Це важливий крок у вилученні інформації з текстів і широко використовується в інтелектуальному аналізі даних, пошуку інформації та машинному навчанні.

Дана технологія, в загальному, складається з двох основних кроків: розпізнавання сутностей та їхня класифікація. Розпізнавання знаходить сутності в тексті, а класифікація додає їх до певної категорії. Варто зазначити, що традиційні моделі NER, часто використовували підходи, засновані на правилах або статистичних методах, такі як Приховані Марковські Моделі та Машини Опорних Векторів. [11] Проте такі методи вимагали великої функціональності техніки, що значно погіршувало час виконання та її ефективність. Натомість, сучасні системи переважно використовують моделі нейронних мереж, такі як CNNs, RNNs, LSTMs, CRFs та інші, які автоматично вивчають функції з набору даних, зменшуючи необхідність ручної розробки.

1.2. Огляд існуючих рішень

1.2.1 Docsumo

Переваги: Пропонує розпізнавання за допомогою OCR, розширену обробку зображень, вилучення таблиць, смарт-фільтри, інтеграцію API та

пакетне завантаження. Обіцяє до 98% точності даних і значне скорочення часу обробки і витрат.

Недоліки: Не читає рукописні дані.

Найкраще підходить для компаній, які шукають прості можливості налаштування та інтеграції, особливо ті, хто має справу з великим обсягом рахунків-фактур.

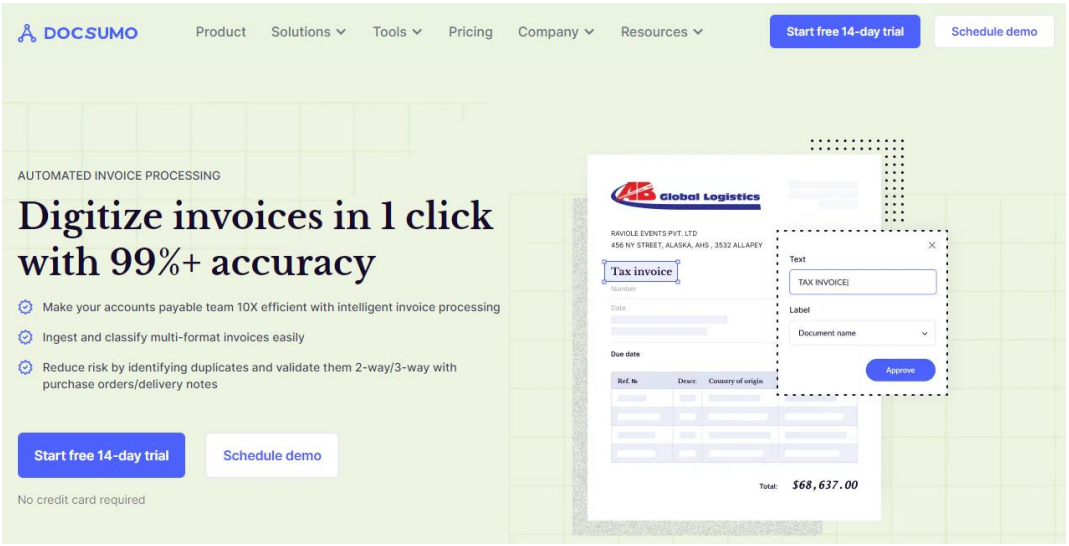

Рисунок 1.5 – Зовнішній вигляд сайту Docsumo

1.2.2 Rossum

Rossum є інноваційним рішенням для автоматизації обробки документів, яке використовує штучний інтелект для ефективної обробки великої кількості документації. Його основною перевагою є здатність до глибокого навчання, що дозволяє йому точно розпізнавати, витягувати та обробляти дані з різноманітних типів документів. Ця система особливо ефективна у витягуванні даних з рахунків-фактур, квитанцій та інших фінансових документів, значно підвищуючи продуктивність і знижуючи можливість людської помилки.

Система включає в себе можливості інтеграції з іншими бізнес-платформами та ERP системами, що робить обмін даними більш зручним і ефективним.

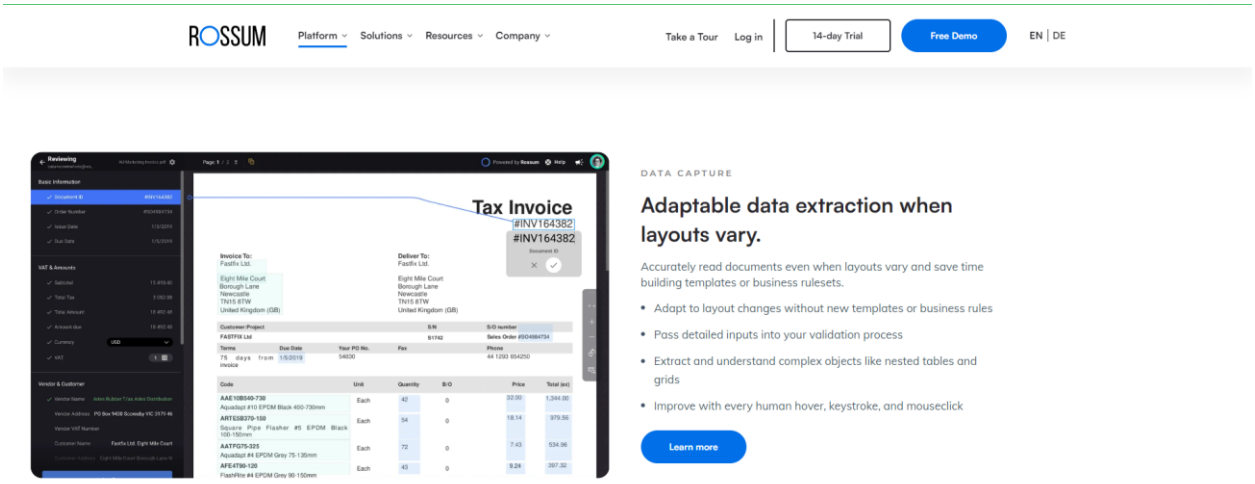

Рисунок 1.6 – Зовнішній вигляд сайту Rossum

1.2.3 Kofax

Kofax є провідною платформою для автоматизації бізнес-процесів, яка забезпечує ефективне розпізнавання документів і витягування даних. Платформа використовує передові технології оптичного розпізнавання символів (OCR) та штучного інтелекту для обробки широкого спектра документів.

Kofax підтримує інтеграцію з багатьма існуючими бізнес-системами, ERP та CRM, полегшуючи процес обміну даними. Платформа має здатність автоматично класифікувати та сортувати документи, що значно підвищує ефективність управління документацією, а також підтримує роботу з документами в різних мовах, роблячи її ідеальним рішенням для компаній з міжнародною діяльністю. Ціна відносно невелика три з половиною тисяч доларів за сто тисяч сторінок документу.

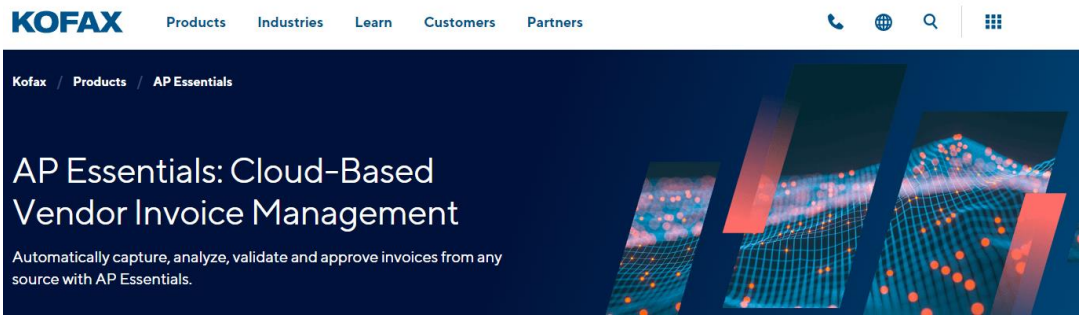

Рисунок 1.7 – Зовнішній вигляд сайту Kofax

1.2.4 Hyperscience

Hyperscience є сучасною платформою для автоматизації введення даних, яка використовує штучний інтелект для ефективної обробки документів. Платформа зосереджена на високій точності та швидкості обробки даних, використовуючи передові алгоритми машинного навчання. Hyperscience здатна автоматично обробляти велику кількість документів різних типів, включаючи складні форми та неструктуровані документи.

Система пропонує інтуїтивно зрозумілий інтерфейс, що робить її доступною для широкого кола користувачів без спеціальних технічних знань. Платформа, як і попередні, інтегрується з існуючими системами та процесами компанії, полегшуючи впровадження та використання. Ще, дана система дуже гарно розпізнає рукописні символи.

Рисунок 1.8 – Зовнішній вигляд сайту Hyperscience

1.3. Постановка задачі

Проаналізувавши аналогічні рішення та визначивши необхідні вимоги, можемо перейти до постановки задачі, яка потребує реалізації.

Кінцевий результат нашої системи повинен мати:

- Можливість розпізнавати ключову інформацію на рахунках-фактурах;
- Fine-tuning на основі специфічного набору даних;
- Місце для зберігання та взаємодії зі збереженими документами;
- Можливість інтеграції в систему Business Central 365;
- Виділення необхідних полів з документу.

2. ОПИС ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ОБРОБКИ РАХУНКІВ-ФАКТУР

2.1. Моделі та методи обробки рахунків-фактур

З появою нових компаній та швидким розвитком вже існуючих, потреба в автоматизації обробки рахунків-фактур зростає з кожним днем, через збільшення кількості даних рахунків в компанії.

Проблему обробки рахунків-фактур, як і інших схожих документів, можна розділити на декілька частин: вилучення тексту з зображень або документів, класифікації вилученої інформації та подальше застосування отриманих даних.

Convolutional Neural Networks (CNNs). Згорткові нейронні мережі є аналогами традиційних штучних нейронних мереж, оскільки вони складаються з нейронів, покращення яких відбувається за рахунок навчання. Кожен нейрон отримує вхідний сигнал і виконує певну операцію. Ключовою особливість згорткових нейронних мереж є те, що вони, в більшості випадків, використовуються для розпізнавання образів всередині зображень. Це дозволяє закодувати специфічні для зображення ознаки в архітектуру мережі, роблячи мережу більш підходящою для завдань, зосереджених на зображеннях, при цьому зменшуючи кількість параметрів, необхідних для налаштування моделі.[12] Згорткові нейронні мережі використовують набагато більше зав’язків, ніж просто ваги. Більше того, ці мережі мають деяку ступінь перекладу інваріантності. [13]

Великою перевагою є представлення зображень у вигляді тензорів. На початку розвитку комп’ютерного бачення та розпізнавання за паттернами, кольорові зображення, які являються тензорами, перетворювали в чорно-білі, які є матрицями, бо оброблювати матриці було простіше ніж тензори. Тензори є край необхідними в CNN. Вхідні данні, параметри та проміжні представлення в CNN являються тензорами.

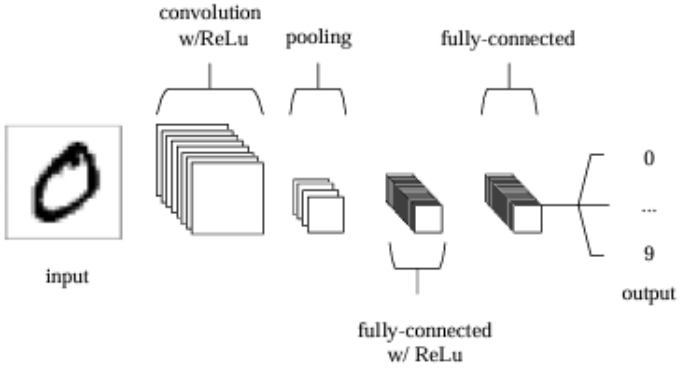

Рисунок 2.1 – Спрощена архітектура CNN [12]

Зазвичай CNN приймає тензор 3-го порядку як вхід, наприклад, зображення з H рядків, W стовпців та 3 каналами (канали кольорів R, G, B). Проте, тензори вищого порядку також можуть бути оброблені CNN подібним чином. Потім вхід послідовно проходить через серію обробок. Один етап обробки зазвичай називається шаром, який може бути шаром згортки, шаром пулінгу, нормалізуючим шаром, повністю з'єднаним шаром, шаром втрат тощо.

$$x_1 \rightarrow w_1 \rightarrow x_2 \rightarrow \ldots \rightarrow xL_{-1} \rightarrow wL_{-1} \rightarrow xL \rightarrow wL \rightarrow z \qquad (2.1)$$

Зазначене рівняння 2.1 ілюструє, як згорткова нейронна мережа працює шар за шаром у прямому проході. Вхідні дані – це x^1, зазвичай зображення (тензор 3-го порядку). Воно проходить обробку на першому шарі, який є першим блоком. Ми позначаємо параметри, задіяні в обробці першого шару, як тензор w^1. Вихід першого шару – це x^2, який також служить входом для обробки другого шару. Ця обробка триває, доки не будуть завершені всі шари у CNN, які виводять x^L. Однак, додається ще один шар для зворотнього розповсюдження помилок, методу, що вчиться визначати гарні значення параметрів у CNN. [14]

Шар розгортки є одним з найбільш задіяних шарів в даній технології, тому про нього варто згадати більш детальніше. Давайте розглянемо процес згортки матриці з одним згортковим ядром. Уявімо, що вхідне зображення має розмір 3×4, а розмір згорткового ядра становить 2×2. Якщо накласти згорткове ядро на

вхідне зображення, можна обчислити добуток чисел у відповідних позиціях ядра та зображення, а потім отримати одне число, сумувавши ці добутки. Наприклад, накладаючи ядро на верхню ліву частину вхідного зображення, отримаємо результат згортки: 1×1+1×4+1×2+1×5=12. Потім зміщуємо ядро вниз на один піксель і отримуємо наступний результат згортки. Процес продовжується до охоплення всієї матриці, як показано на рисунку 2.2.

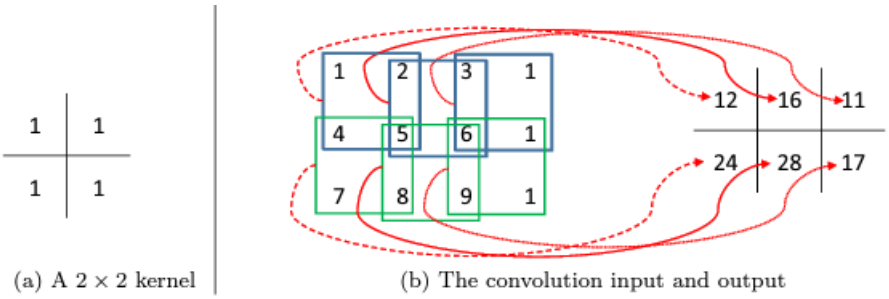

Рисунок 2.2 – Процес операції розгортання[13]

Region-Based Convolutional Neural Network (R-CNN). Архітектура R-CNN була розроблена у 2014 році і має наступний алгоритм роботи:

Спочатку, зображення розбивають на 2000 регіонів за допомогою методу вибіркового пошуку, далі з кожного регіону отримують признаки за допомогою згорткових нейронних мереж. Далі відбувається класифікація признаків за допомогою методу опорних векторів та уточнення кордонів регіонів на основі лінійної регресії. [15]

Метод вибіркового пошуку спочатку генерує багато потенційних регіонів, потім за допомогою жадібного алгоритму схожі об’єкти поєднуються у більші, які в свою чергу використовуються для створення остаточних пропозицій регіонів.

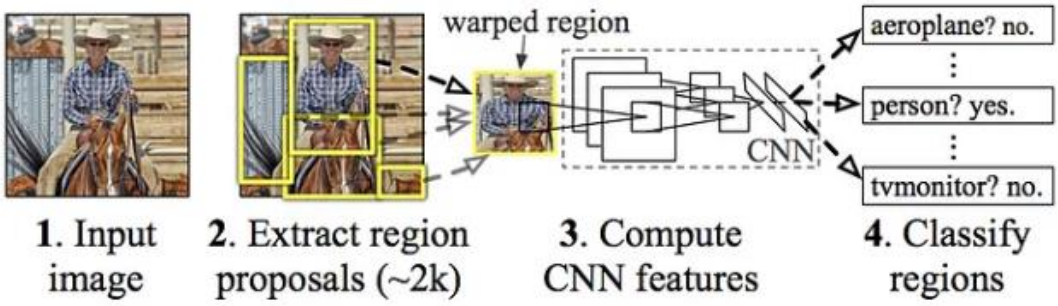

Рисунок 2.3 – Архітектура R-CNN [16]

Далі, ці 2000 пропозицій кандидатських регіонів трансформуються в квадратну форму та подаються у згорткову нейронну мережу, яка виробляє 4096-вимірний вектор ознак як вихід. CNN діє як екстрактор ознак, і вихідний щільний шар складається з ознак, видобутих із зображення. Видобуті ознаки подаються в метод опорних векторів для класифікації наявності об'єкта в цій пропозиції регіону. Крім прогнозування наявності об'єкта в пропозиціях регіонів, алгоритм також прогнозує чотири значення, які є значеннями коригування для підвищення точності обмежувального поля. [16]

Тож, використовуючи CNN для проблеми класифікації регіонів зображень, нам потрібно було б брати різні регіони, застосовувати до них CNN для класифікації наявності об’єкту в рамках регіону. Проблема такого методу в тому, що об’єкт, який нас цікавить, може мати різні положення всередині зображення та різні співвідношення сторін. Відповідно, потрібно буде обирати багато регіонів, що значним чином вплине на обчислювальні показники. [15]

Проте R-CNN також має ряд недоліків:

- Вимагає багато часу для тренування мережі, оскільки потрібно класифікувати 2000 регіонів для кожного зображення;
- Не може бути застосоване в реальному часі, оскільки потребує приблизно 47 секунд для кожного тестового зображення;
- Алгоритм вибіркового пошуку є фіксованим набором правил, тому ніяке навчання на цьому етапі не відбувається, що потенційно може призвести до генерації хибних регіонів.

Fast R-CNN. Наступним покращенням даної мережі є Fast R-CNN. Алгоритм роботи схожий до регіональної згорткової нейронної мережі, але на вхід CNN подається зображення, замість регіонів, для подальшої побудови згорткової мапи ознак. На основі цієї мапи обираються регіони, які трансформуються та нормалізуються до єдиного розміру за допомогою шару об'єднання RoI перед подачею в повністю об'єднаний шар. Далі, визначається клас регіону та коректуються обмежувальні кордони. Fast R-CNN працює швидше за R-CNN, оскільки замість подачі тисяч регіонів, згортка виконується лише один раз на зображення. [17]

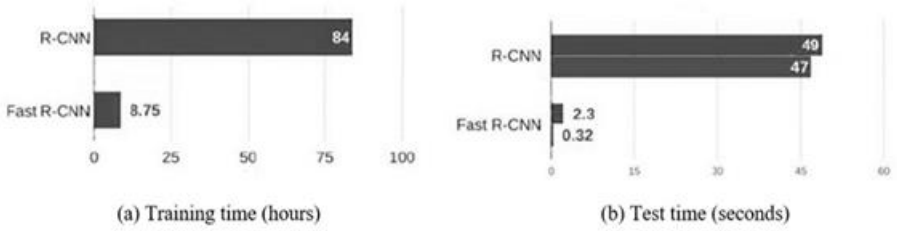

Рисунок 2.4 – Порівняння ефективності R-CNN та Fast R-CNN [18]

Така велика різниця в часі досягається за допомогою шару об'єднання RoI та softmax класифікатора, який використовується замість методу опорних векторів (див. рис. 2.5).

Також на рисунку можна помітити, що з включенням пропозицій регіонів(нижня шкала) час при тестування майже у вісім разів, що є слабким місцем даної мережі.

Обидві мережі Fast R-CNN та R-CNN використовують вибірковий пошук для пропозицій регіонів. Даний метод є повільним, що сильно впливає на ефективність мережі в цілому.

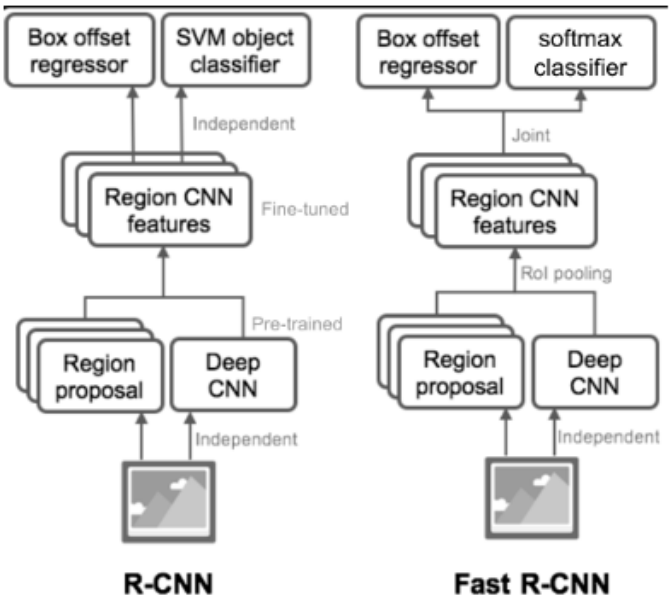

Рисунок 2.5 – Порівняння архітектури R-CNN та Fast R-CNN [18]

Faster R-CNN. У даній мережі, вибірковий пошук замінили на можливість мережі навчатися для пропозицій регіонів. Подібно до Fast R-CNN, зображення подається на вхід до згорткової мережі, яка забезпечує згорткову мапу ознак. Замість того, щоб використовувати вибірковий алгоритм пошуку на карті ознак для ідентифікації пропозицій регіону, для прогнозування пропозицій регіону використовується окрема мережа. Потім прогнозовані пропозиції регіону змінюються за допомогою шару об'єднання RoI, який потім використовується для класифікації зображення в межах запропонованої області та прогнозування значень зсуву для кордонів. [19] На рисунку 2.6 можемо побачити значно кращі результати Faster R-CNN, у порівнянні з її попередниками.

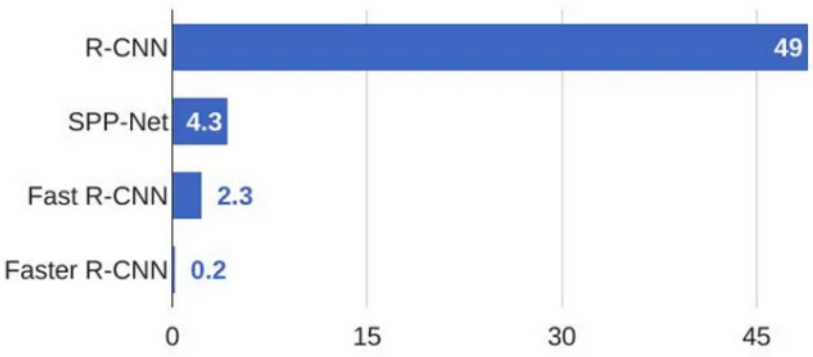

Рисунок 2.6 – Порівняння ефективності мереж

Дані результати дають змогу використовувати дану модель в реальному часі, що дає нам змогу застосувати її для імплементації в нашу систему обробки рахунків-фактур.

2.2. Технологія оптичного розпізнавання символів

Технологія оптичного розпізнавання символів є дуже важливою в нашому випадку, оскільки за допомогою неї ми будемо отримувати текст із регіонів, які отримаємо після застосування Faster R-CNN.

Дана технологія, як і більшість нейронних мереж, працює за аналогією з тим, як ми розпізнаємо символи при читанні, як зображено на рисунку 1.3. За допомогою нього можна розпізнавати як рукописний текст, так і надрукований, але те наскільки точним та ефективним буде розпізнавання, напряму залежить від якості вхідних зображень.

Tesseract – це відкритий OCR двигун, спочатку розроблений компанією HP і згодом випущений для загального користування, відомий своїми можливостями у сфері оптичного розпізнавання символів.

Архітектура Tesseract включає кілька етапів: сегментацію, попередню обробку, вилучення ознак і розпізнавання, як показано на рисунку 2.7. Він розроблений для обробки як традиційного чорно-на-білому тексту, так і інверсного біло-на-чорному. Процес включає аналіз текстових рядків, їх поділ на слова та розпізнавання цих слів за допомогою двоетапної системи з адаптивним класифікатором.

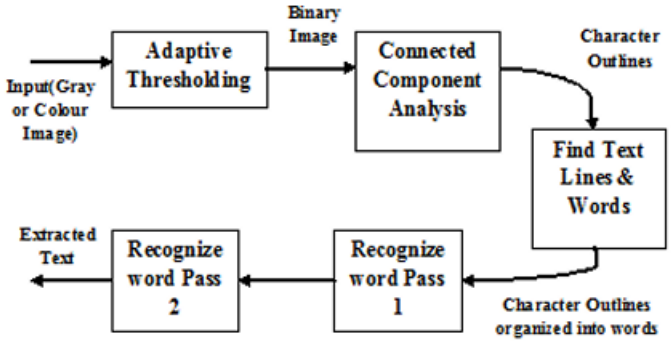

Рисунок 2.7 – Архітектура Tesseract [7]

Дана технологія здатна працювати з зображеннями, зробленими цифровими камерами, які часто містять спотворення або нерівномірне освітлення, що ускладнює розпізнавання тексту. Гнучкість Tesseract, його розширюваність, активна спільнота розробників та ефективність з коробки є ключовими перевагами серед інших рішень.

2.3. Методи та критерії валідації алгоритмів машинного навчання

Валідація алгоритмів машинного навчання є критично важливою, оскільки вона дозволяє оцінити, наскільки добре модель працюватиме на нових, раніше невідомих даних. Це варто робити для перевірки узагальнення моделі, запобігання перенавчання та оптимізації параметрів моделі. Використання таких методів, як перехресна перевірка, дозволяє отримати більш надійну оцінку ефективності моделі, аналізуючи її продуктивність на різних наборах даних.

Основними показниками у класифікаційних задачах є TP, TN, FP, та FN.

- TP (True Positive): Кількість випадків, коли модель правильно передбачила позитивний клас.
- TN (True Negative): Кількість випадків, коли модель правильно передбачила негативний клас.
- FP (False Positive): Кількість випадків, коли модель невірно класифікувала негативний клас як позитивний.
- FN (False Negative): Кількість випадків, коли модель невірно класифікувала позитивний клас як негативний.

У таблиці 2.1. описані різноманітні метрики для оцінки моделі при навчанні та тестуванні.

Таблиця 2.1

Міра	Формула	Опис
Точність (Accuracy)	$\frac{TP + TN}{TP + TN + FP + FN}$	Відсоток правильно класифікованих випадків від усіх випадків.

Точність (Precision)	$\frac{TP}{TP + FP}$	Відсоток правильно класифікованих позитивних випадків.
Повнота (Recall)	$\frac{TP}{TP + FN}$	Частка правильно класифікованих позитивних випадків з усіх позитивних.
F1-оцінка	$2 \times \frac{Precision \times Recall}{Precision + Recall}$	Гармонічне середнє між точністю та повнотою.

Міри в стилі COCO(Common Objects in Context) – це набір оціночних мір, які зазвичай використовуються для задач виявлення об'єктів та сегментації екземплярів. Ці показники засновані на наборі даних та викликах COCO, який є популярним еталоном для цих завдань комп'ютерного зору. Метрики в стилі COCO забезпечують стандартизований спосіб оцінки продуктивності моделей виявлення об'єктів і сегментації екземплярів. [13]

Середня точність – це широко використовувана метрика для виявлення об'єктів. Вона вимірює площу під кривою точного відкликання (див. рис. 2.8), яка відображає компроміс між точністю та пригадуванням при різних довірчих порогах. AP обчислюється окремо для кожного класу або категорії в наборі даних.

Середня повнота (Average Recall, AR). Повнота вимірює наскільки добре були знайдені позитивні випадки. Наприклад, ми можемо знайти 80% можливих позитивних випадків у наших головних прогнозах K.

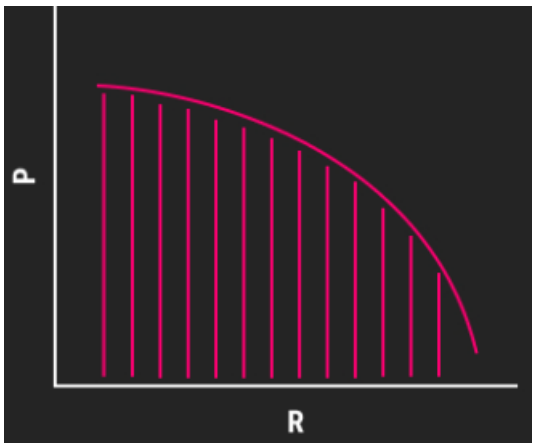

Рисунок 2.8 – Графік середньої точності[10]

Міра Жаккара (Intersection over Union, IoU). IoU, це метрика, яка базується на областях і використовується в комп'ютерному зорі для визначення того, чи можна вважати два об'єкти однаковими. Як із назви видно, метрика обчислюється шляхом ділення площі перетину двох об'єктів на площу їх об'єднання. [4]

$$IoU = \frac{\text{Площа перетину двох об'єктів}}{\text{Площа об}'\text{єднаннядвох об'єктів}} \quad (2.2)$$

Діапазон значень IoU від 0 до 1, де два ідентичні об’єкти мають IoU рівний 1, а об’єкти без перетину мають IoU рівний 0 і т. д.

Вона використовується у багатьох завданнях, деякі з яких ми вже обговорювали, такі як алгоритми виявлення об'єктів і завдання інформаційного пошуку, а також в системах сегментації, оцінці пошукових систем і в вимірюванні загальної ефективності алгоритмів пошуку.

При покращенні вашої моделі машинного навчання іноді потрібно мати одночасно число, що характеризує продуктивність. Саме це надає mAP. Ця метрика допомагає отримати середнє значення AP для всіх виявлених класів.

Валідація моделі є важливим кроком при її створенні, і використання правильних методів та технік дає змогу зрозуміти її слабкі місця та оцінити неупереджену ефективність узагальнення.

Розділення датасетів – найбільш простий метод, який може бути застосований, це коли ми розбиваємо наш датасет на два, таким чином, щоб модель навчалася на одному, як правило більшому, а інший датасет був використаний для тестування та валідації.

Притримування даних: при оптимізації гіперпараметрів моделі, може виникнути проблема перенавчання моделі, якщо ви хочете оптимізувати використання двох різних датасетів для навчання та тестування.

Для вирішення цього питання можна створити додатковий набір holdout. Найчастіше, це 10% даних, які не використовували в жодному з етапів обробки/перевірки.

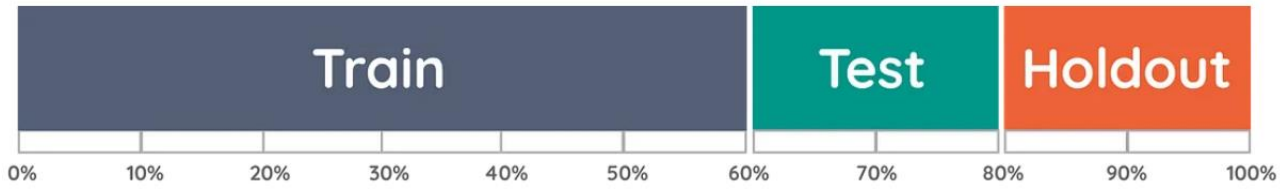

Рисунок 2.9 – Розподіл даних на три датасети [20]

Перехресне затверджування (k-Fold Cross-Validation, k-Fold CV). Щоб звести до мінімуму упередження вибірки існує метод k-Fold CV, суть якого полягає в тому, щоб дробити наш датасет не один раз, а багато і перевіряти всі комбінації. Тут відбувається k-кратна перехресна перевірка. Вона розбиває дані на k наборів, потім тренує дані на k-1 наборі і тестує на одному наборі, який залишився, як показано на рисунку. Це робиться для всіх комбінацій і потім береться середній результат на кожному екземплярі.

Рисунок 2.10 – 5- Fold CV[20]

Вкладена перехресна валідація. При оптимізації гіперпараметрів своєї моделі та використанні однієї й тієї ж стратегії k-Fold CV для налаштування моделі та оцінки продуктивності, ви ризикуєте перетренувати модель.

Замість цього використовуємо стратегію вкладеної перехресної перевірки (Nested Cross-Validation), що дозволяє відокремити етап налаштування

гіперпараметрів від етапу оцінки помилок. Для цього ми використовуємо два цикли k-fold перехресної перевірки (див. рис. 2.11):

- внутрішній цикл для налаштування гіперпараметрів;
- зовнішній цикл для оцінки точності.

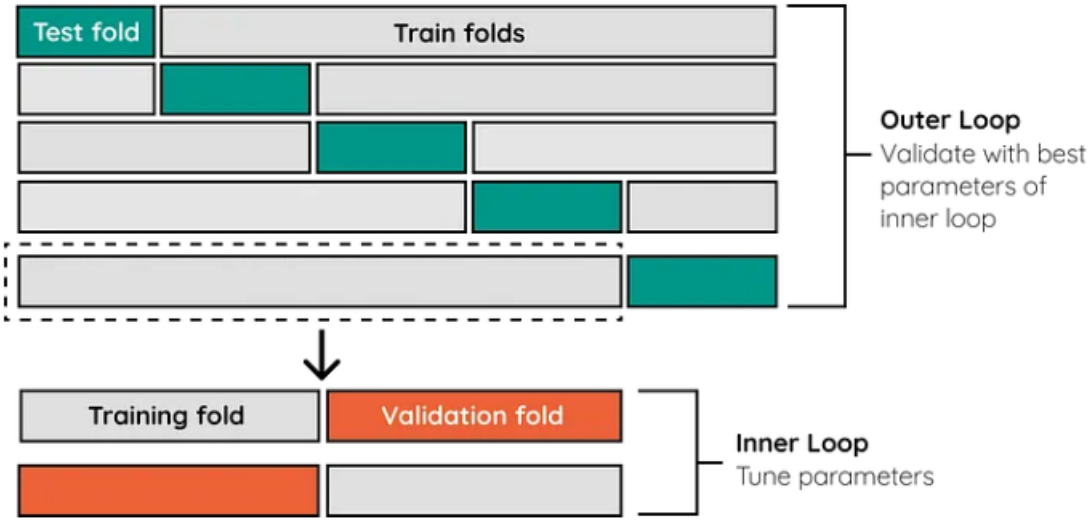

Рисунок 2.11 – Вкладена перехресна валідація[20]

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Формування навчальних та тестових датасетів

Навчання буде проводитися на попередньо натренованій моделі zoo з бібліотеки Detectron2 за допомогою Faster R-CNN на наборі даних, який складається з 78 зображень рахунків-фактур та анотацій у форматі ‘coco.json’. Анотації використовуються для позначення розташування та категорії об’єктів на зображенні.

Формат coco.json містить такі ключові елементи:

- Images: Інформація про зображення, включаючи ID, ширину, висоту тощо.
- Annotations: Дані про анотації, включаючи полігональні координати об'єктів, ID категорій, ID зображення тощо.
- Categories: Категорії об'єктів, які можуть бути виявлені.

Також два набори даних для тестування та валідації по 6 та 8 зображень з анотаціями відповідно.

Набір даних містить 13 класів, серед яких «buyer details», «invoice date», «invoice no.», «seller details», «total amount» та інші.

3.2. Короткий опис програмного забезпечення

При виборі мови програмування для задач, які пов’язані з машинним навчанням або наукою про дані, Python беззаперечно є найбільш підходящим рішенням. Python має велику кількість бібліотек, які спеціально розроблені для машинного навчання, включаючи TensorFlow, PyTorch, Scikit-learn, Keras та інші. Ці інструменти спрощують процес розробки, навчання та розгортання моделей ML. А також для візуалізації результатів навчання та тренування.

Python має велику спільноту, що значно полегшує процес розробки та вирішення помилок, пов’язаних з кодом.

Саме на цій мові програмування вже написано та натреновано дуже багато моделей, які можна просто підігнати під свої цілі та вимоги та отримати вже готове рішення, витративши невелику кількість зусиль.

Звісно, це не найшвидша мова програмування, проте вона пропонує достатню продуктивність для багатьох застосунків, особливо коли використовуються оптимізовані бібліотеки. Більше того, вона має простий синтаксис, що робить її доступною для багатьох людей.

Pytesseract – це бібліотека Python, яка працює як обгортка для двигуна Tesseract-OCR від Google. Pytesseract спрощує використання Tesseract, надаючи інтерфейс Pythonic, що дозволяє розробникам легко інтегрувати можливості OCR у свої програми Python.[6]

До переваг використання Pytesseract можна віднести:

- Простота використання: Pytesseract відомий своєю простотою і простотою використання. Ви можете встановити його через pip, і він забезпечує простий Python API для виконання OCR на зображеннях.
- Крос-платформа: Pytesseract є крос-платформенною і може використовуватися в різних операційних системах, включаючи Windows, macOS і Linux.
- Підтримка декількох мов: Tesseract підтримує більше 100 мов, а Pytesseract дозволяє легко вказати мову, яку ви хочете використовувати під час розпізнавання.
- Попередня обробка зображення: Pytesseract дозволяє застосовувати різні методи попередньої обробки для підвищення точності розпізнавання, такі як зміна розміру, пороги та зменшення шуму.
- Параметри конфігурації: Ви можете точно налаштувати поведінку OCR, надаючи параметри конфігурації Pytesseract, що дозволяє оптимізувати результати OCR для конкретних випадків використання.

3.3. Результати машинного навчання

3.3.1 Тестовий набір даних

Провівши оцінку нашої моделі на тестовому наборі даних, отримали наступні результати (див. рис. 3.1).

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.567
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.958
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.598
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.538
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.686
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.609
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.609
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.609
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.542
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.763
[12/10 16:43:52 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs  |  APm   |  APl   |
|:------:|:------:|:------:|:-----:|:------:|:------:|
| 56.665 | 95.820 | 59.832 |  nan  | 53.812 | 68.594 |
[12/10 16:43:52 d2.evaluation.coco_evaluation]: Some metrics cannot be computed and is shown as NaN.
[12/10 16:43:52 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category         | AP     | category       | AP     | category          | AP     |
|:-----------------|:-------|:---------------|:-------|:------------------|:-------|
| fields           | nan    | buyer details  | 76.535 | country of origin | 66.733 |
| invoice date     | 49.967 | invoice no.    | 64.579 | mode of transport | 61.782 |
| payment term     | 61.122 | seller details | 50.099 | shipment term     | 40.000 |
| shipping address | 62.277 | shipping date  | 50.198 | total amount      | 36.386 |
| total cartons    | 60.297 |                |        |                   |        |
```

Рисунок 3.1 – Результати оцінки моделі на тестовому наборі

Метрики оцінювання

Ефективність моделі оцінюється за допомогою середньої точності (AP) та середньої повноти (AR) на різних порогах перетину об'єднання (IoU) та розмірах об'єктів.

Загальний AP: 56,942% (AP при IoU=0.50:0.95), з високою продуктивністю при IoU=0.50 (96,947%) але нижчою при IoU=0.75 (56,465%).

AP залежно від розміру: Немає даних для малих об'єктів (APs), 51,618% для середніх об'єктів та 68,988% для великих об'єктів.

Загальний AR: Стабільний на різних максимальних виявленнях (61,3%).

Holdout набір даних (10%)

Провівши оцінку моделі на наборі даних, який не брав участі в навчанні та тестування, були отримані наступні результати (див. рис. 3.2).

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.567
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.958
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.598
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.538
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.686
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.609
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.609
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.609
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.542
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.763
[12/10 16:43:52 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs  |  APm   |  APl   |
|:------:|:------:|:------:|:-----:|:------:|:------:|
| 56.665 | 95.820 | 59.832 |  nan  | 53.812 | 68.594 |
[12/10 16:43:52 d2.evaluation.coco_evaluation]: Some metrics cannot be computed and is shown as NaN.
[12/10 16:43:52 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category         | AP     | category       | AP     | category          | AP     |
|:-----------------|:-------|:---------------|:-------|:------------------|:-------|
| fields           | nan    | buyer details  | 76.535 | country of origin | 66.733 |
| invoice date     | 49.967 | invoice no.    | 64.579 | mode of transport | 61.782 |
| payment term     | 61.122 | seller details | 50.099 | shipment term     | 40.000 |
| shipping address | 62.277 | shipping date  | 50.198 | total amount      | 36.386 |
| total cartons    | 60.297 |                |        |                   |        |
```

Рисунок 3.2 – Результати оцінки моделі на holdout наборі

Метрики оцінювання

Подібні до набору даних для валідації.

Загальний AP: 56,665% (AP при IoU=0.50:0.95), також з істотним падінням при IoU=0.75 (59,832%).

AP залежно від розміру: Схожі тенденції, як у наборі даних для тестування, без даних для малих об'єктів.

Загальний AR: Трохи нижчий, ніж у наборі даних для тестування, але стабільний на різних максимальних виявленнях (близько 60,9%).

Візуалізуємо роботу нашої моделі, надіславши на вхід одне зображення, яке не брало участь у навчання. На рисунку 3.3, можна побачити те, які регіони були виділені, та яку інформації воно містили.

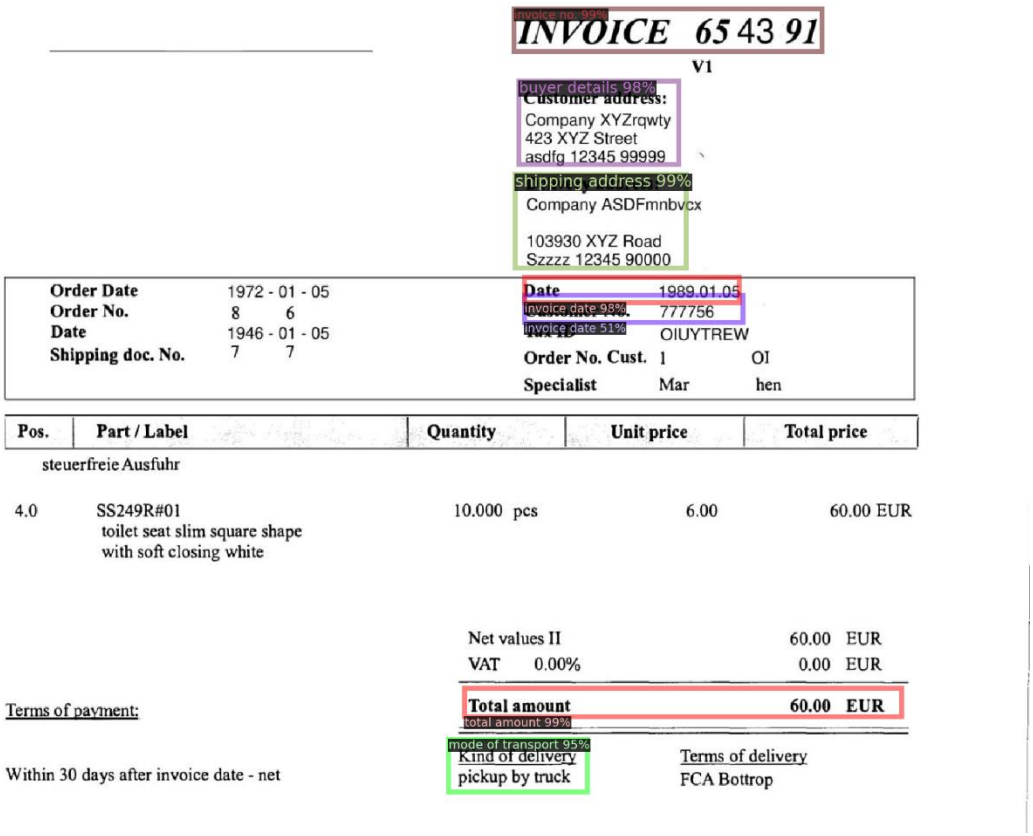

Рисунок 3.3 – Результат розпізнавання регіонів

Далі з цих регіонів можна отримати текстову інформацію, яку в подальшому використовувати для інтеграції з Business Central, а саме для автоматичного створення рахунків-фактур.

3.4. Інтеграція з Business Central

Після успішного процесу донавчання моделі на нашому наборі даних, можемо перейти до вирішення головної задачі – обробки рахунків-фактур для інтеграції з Business Central.

Для початку, розширимо функціонал Business Central, додавши на сторінку Purchase Invoice можливість автоматичного заповнення цього документу з файлів (див. рис. 3.4).

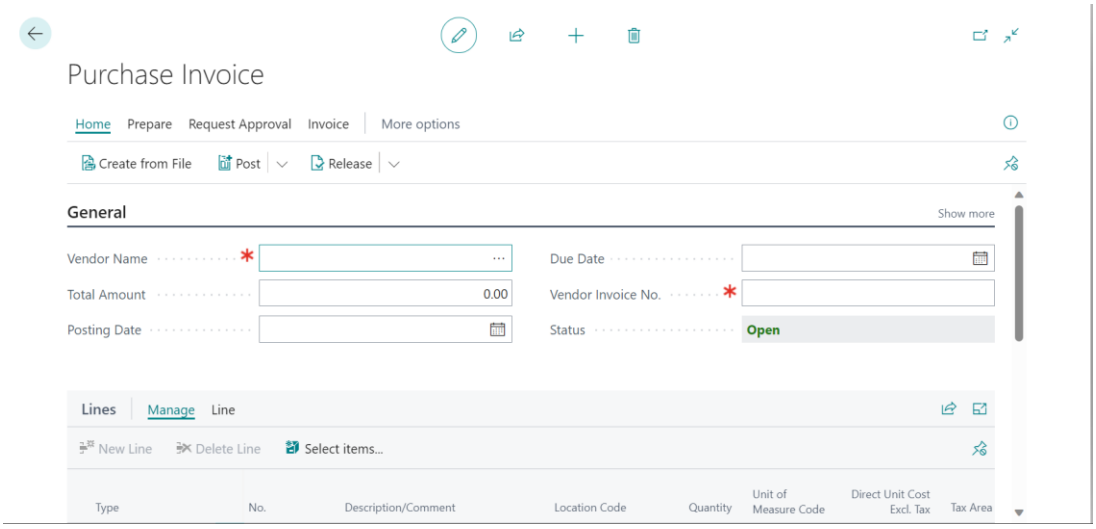

Рисунок 3.4 – Сторінка для створення рахунку-фактури

Даний функціонал має вікно, в яке користувач може завантажити зображення рахунку-фактури (див. рис. 3.5). Натиснувши ОК, дане зображення піде в сховище Google Drive за допомогою відповідного API, а ідентифікатор цього файлу в сховищі буде переданий в запиті на API, який ми розробили.

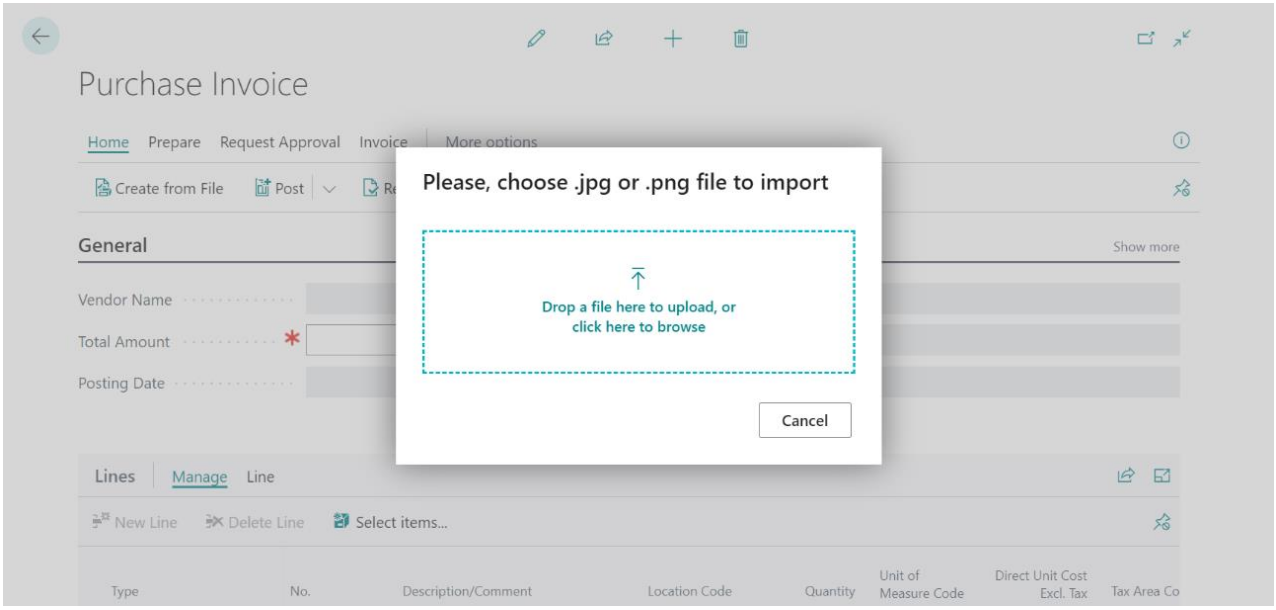

Рисунок 3.5 – Вікно для завантаження файлу рахунку-фактури

Після завантаження файлу, в разі успішної обробки, користувач отримає повідомлення про успішну обробку файлу (див. рис. 3.6), та як результат заповнені поля Total Amount, Due Date та Vendor Invoice No. (див. рис. 3.7).

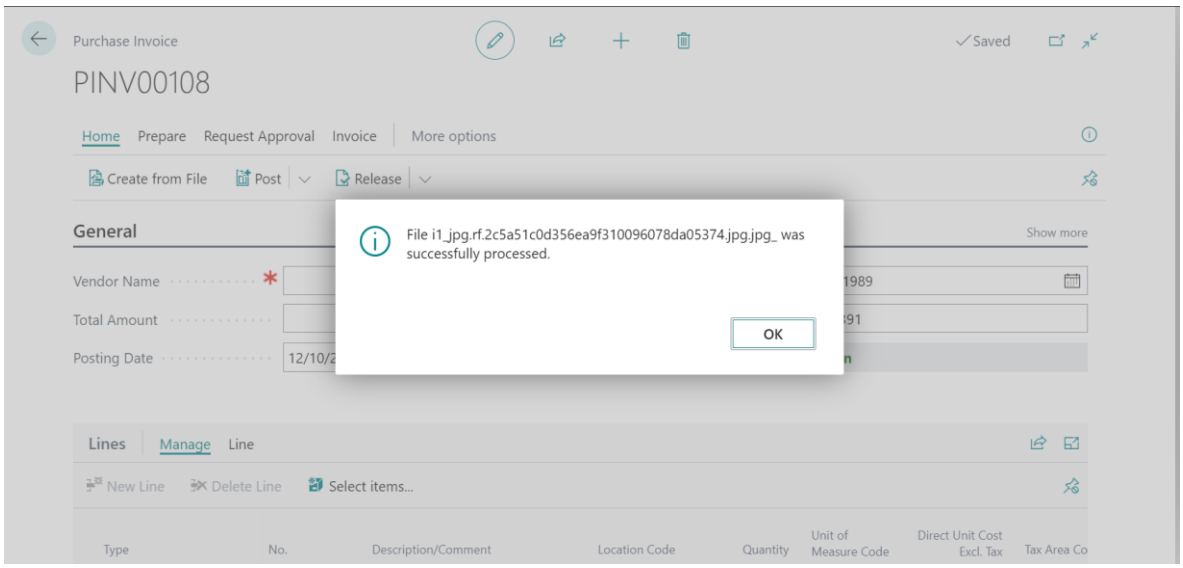

Рисунок 3.6 – Повідомлення про успішну обробку рахунку-фактури

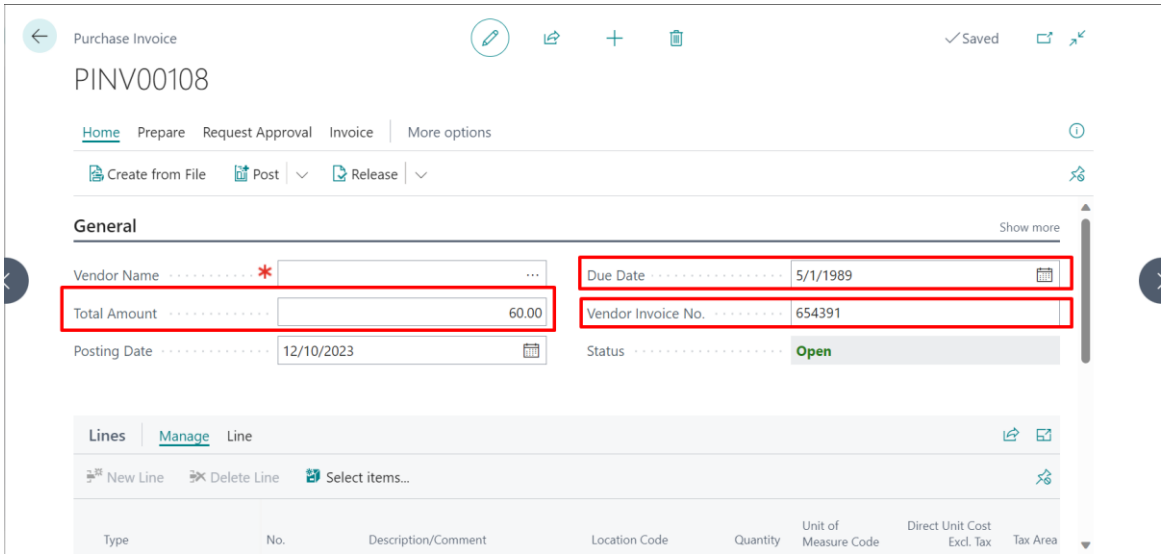

Рисунок 3.7 – Успішно заповнені поля рахунку-фактури

Для того щоб розробити API для отримання та обробки записів використаємо бібліотеку Flask для Python. Даний endpoint – /process_invoices, обробляє файл зображення рахунка-фактури, завантажений користувачем, і повертає отримані з нього дані за допомогою попередньо натренованої моделі. Це POST запит, який має містити в собі «file_id»

Отриманий з запиту ідентифікатор, використаємо для завантаження файлу в нашу систему для розпізнавання ключових полів. Після чого видаляє файл, для економії місця (див. рис. 3.8).

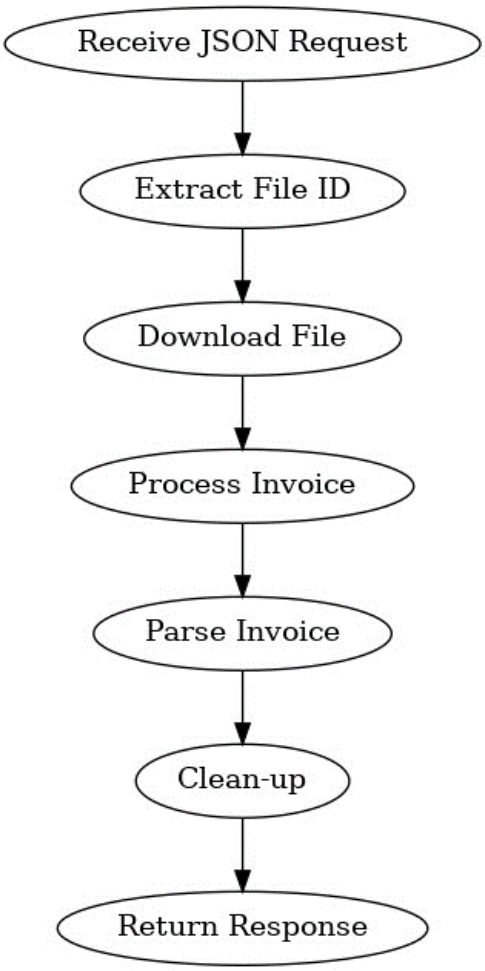

Рисунок 3.8 – Flow chart діаграми API

За допомогою flask та ngork створимо та розгорнемо наш API. На рисунку 3.9 бачимо, що він успішно функціонує та може оброблювати вхідні запити.

```
 * Serving Flask app '__main__'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
 * Running on http://9927-35-193-221-70.ngrok.io
 * Traffic stats available on http://127.0.0.1:4040
```

Рисунок 3.9 – Успішно розгорнутий API

Створимо тестовий запит на даний API, в якому передаймо ідентифікатор файлу з Google Drive, який попередньо туди завантажили. Запит має містити file_id, як показано на рисунку 3.10.

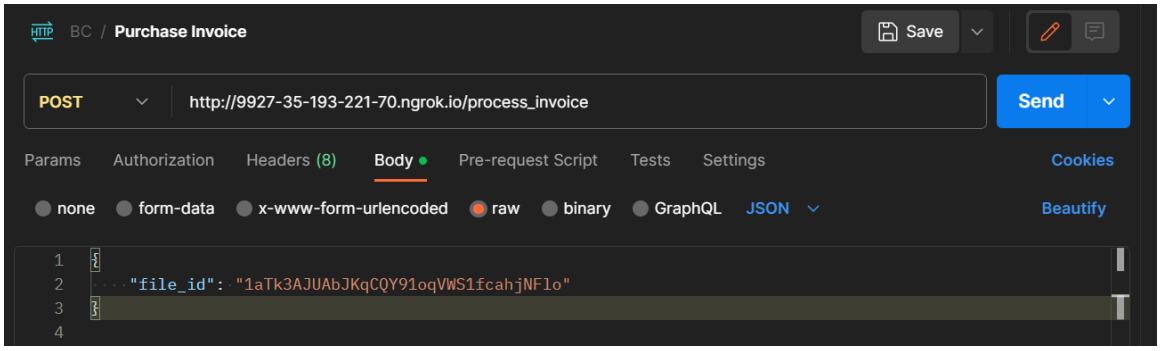

Рисунок 3.10 – Тестовий запит на API

У результаті отримає відповідь, у форматі JSON, яка містить поля, які ми попередньо визначили для отримання (див. рис. 3.11).

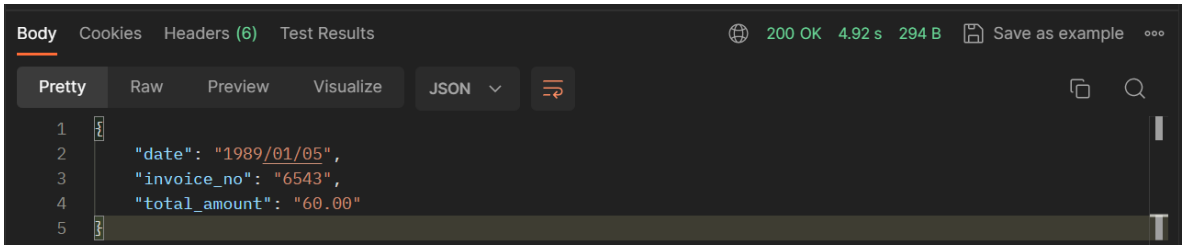

Рисунок 3.11 – Відповідь від API

На цьому етап розробки є закінченим та успішним.

ВИСНОВКИ

У рамках даної роботи було досліджено та описано основні теоретичних аспектів машинного навчання та вибору оптимальних інструментів для вирішення завдань у цій галузі. Також було проведено огляд існуючих рішень у сфері машинного навчання, огляд та порівняння моделей для машинного навчання.

Була розроблена інформаційна технологія обробки рахунків-фактур, яка базується на оптичному розпізнаванні символів та попередньо натренованій моделі, а також інтегрована в систему ведення бізнесу та обліку Microsoft Dynamics 365 Business Central.

Був розроблений API для збереження та зчитування файлів зі сховища, що допомогло інтегрувати нашу інформаційну технологію в Business Central.

Враховуючи описані вище аргументи, можемо зробити висновок, що завдання, поставлене на початку виконання роботи, було успішно виконано.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ebert J., Hauptmann C. Microsoft Dynamics 365 Business Central // Microsoft Dynamics 365 Business Central. 2023.
2. Janiesch C., Zschech P., Heinrich K. Machine learning and deep learning // Electronic Markets. 2021. Vol. 31, № 3.
3. Linardatos P., Papastefanopoulos V., Kotsiantis S. Explainable ai: A review of machine learning interpretability methods // Entropy. 2021. Vol. 23, № 1.
4. Kubat M. An Introduction to Machine Learning // An Introduction to Machine Learning. 2021.
5. Kevin P. Murphy. Probabilistic Machine Learning // Gaussian processes for machine learning. 2022.
6. Lestari I.N.T., Mulyana D.I. IMPLEMENTATION OF OCR (OPTICAL CHARACTER RECOGNITION) USING TESSERACT IN DETECTING CHARACTER IN QUOTES TEXT IMAGES // Journal of Applied Engineering and Technological Science. 2022. Vol. 4, № 1.
7. Ahmed M., Awel M.A., Abidi A.I. Review on Optical Character Recognition // International Research Journal of Engineering and Technology(IRJET). 2019. № July.
8. Zhou M. et al. Progress in Neural NLP: Modeling, Learning, and Reasoning // Engineering. 2020. Vol. 6, № 3.
9. Singh S., Mahmood A. The NLP Cookbook: Modern Recipes for Transformer Based Deep Learning Architectures // IEEE Access. 2021. Vol. 9.
10. Dongare A.D., Kharde R.R., Kachare A.D. Introduction to Artificial Neural Network (ANN) Methods // International Journal of Engineering and Innovative Technology (IJEIT). 2012. Vol. 2, № 1.
11. Lample G. et al. Neural architectures for named entity recognition // 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference. 2016.

12. O’Shea K. and N.R. An Introduction to Convolutional Neural Networks // Int J Res Appl Sci Eng Technol. 2015. Vol. 10, № 12.
13. Bouvrie J. Notes on convolutional neural networks // In Pract. 2006.
14. Wu J. Introduction to Convolutional Neural Networks // Introduction to Convolutional Neural Networks. 2017.
15. He K. et al. Mask R-CNN // Proceedings of the IEEE International Conference on Computer Vision. 2017. Vol. 2017-October.
16. Wang X., Shrivastava A., Gupta A. A-Fast-RCNN: Hard positive generation via adversary for object detection // Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017. 2017. Vol. 2017-January.
17. Huang H.N. et al. Image segmentation using transfer learning and Fast R-CNN for diabetic foot wound treatments // Front Public Health. 2022. Vol. 10.
18. Chaudhuri A. Hierarchical Modified Fast R-CNN for Object Detection // Informatica (Slovenia). 2021. Vol. 45, № 7.
19. Cao C. et al. An Improved Faster R-CNN for Small Object Detection // IEEE Access. 2019. Vol. 7.
20. Kaya M., Bilge H.Ş. Deep metric learning: A survey // Symmetry. 2019. Vol. 11, № 9.

ДОДАТОК А

```
!pip install -q git+https://github.com/huggingface/transformers.git
!pip install -q datasets seqeval
!python -m pip install -q
'git+https://github.com/facebookresearch/detectron2.git'
# Import necessary libraries
import detectron2
from detectron2.utils.logger import setup_logger
setup_logger()
# Import some common libraries
import numpy as np
import os, json, cv2, random
from google.colab.patches import cv2_imshow

# Import Detectron2 utilities
from detectron2 import model_zoo
from detectron2.engine import DefaultTrainer
from detectron2.config import get_cfg
from detectron2.utils.visualizer import Visualizer
from detectron2.data import MetadataCatalog, DatasetCatalog
# Mount Google Drive (to access dataset)
from google.colab import drive
drive.mount('/content/drive')
# Set the path for the dataset and annotations
dataset_path = "/content/drive/MyDrive/InvoiceDataset/train_split"
coco_json_path =
"/content/drive/MyDrive/InvoiceDataset/train_split/annotations/_annotations.
coco.json"
import json
import os
from PIL import Image

# Paths to your dataset and annotations
dataset_path = "/content/drive/MyDrive/InvoiceDataset/train_split"
annotations_path =
"/content/drive/MyDrive/InvoiceDataset/train_split/annotations/_annotations.
coco.json"

valid_dataset_path = "/content/drive/MyDrive/InvoiceDataset/valid"
valid_annotations_path =
"/content/drive/MyDrive/InvoiceDataset/valid/annotations/_annotations.coco.j
son"
test_dataset_path = "/content/drive/MyDrive/InvoiceDataset/test"
test_annotations_path =
"/content/drive/MyDrive/InvoiceDataset/test/annotations/_annotations.coco.js
on"

# Load the COCO annotations
with open(test_annotations_path, 'r') as file:
```

```
    data = json.load(file)

# Check and correct the image sizes
for image in data['images']:
    image_file = os.path.join(test_dataset_path, image['file_name'])
    actual_size = Image.open(image_file).size  # (width, height)

    if (image['width'], image['height']) != actual_size:
        print(f"Correcting {image['file_name']}: from {(image['width'],
image['height'])} to {actual_size}")
        image['width'], image['height'] = actual_size

# Save the corrected annotations
with open(test_annotations_path, 'w') as file:
    json.dump(data, file)
# Register the dataset in COCO format
from detectron2.data.datasets import register_coco_instances
register_coco_instances("invoice_dataset", {}, coco_json_path, dataset_path)

valid_dataset_path = "/content/drive/MyDrive/InvoiceDataset/valid"
valid_annotations_path =
"/content/drive/MyDrive/InvoiceDataset/valid/annotations/_annotations.coco.j
son"

test_dataset_path = "/content/drive/MyDrive/InvoiceDataset/test"
test_annotations_path =
"/content/drive/MyDrive/InvoiceDataset/test/annotations/_annotations.coco.js
on"

register_coco_instances("test_dataset", {}, test_annotations_path,
test_dataset_path)
register_coco_instances("valid_dataset", {}, valid_annotations_path,
valid_dataset_path)
cfg = get_cfg()
cfg.MODEL.DEVICE = 'cuda'
cfg.merge_from_file(model_zoo.get_config_file("COCO-
Detection/faster_rcnn_R_50_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("invoice_dataset",)
cfg.DATASETS.TEST = ()
cfg.DATALOADER.NUM_WORKERS = 2
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-
Detection/faster_rcnn_R_50_FPN_3x.yaml")
cfg.SOLVER.IMS_PER_BATCH = 4
cfg.SOLVER.BASE_LR = 0.001

# cfg.SOLVER.WARMUP_ITERS = 1000
# cfg.SOLVER.MAX_ITER = 3000
# cfg.SOLVER.STEPS = (1000, 1500)
# cfg.SOLVER.GAMMA = 0.05
```

```
cfg.SOLVER.MAX_ITER = 1000
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 13
cfg.OUTPUT_DIR = "/content/drive/My Drive/final_model"
# Train the model
os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = DefaultTrainer(cfg)
trainer.resume_or_load(resume=False)
trainer.train()
from detectron2.data import build_detection_test_loader
from detectron2.evaluation import COCOEvaluator, inference_on_dataset



# Set up the validation dataset
cfg.DATASETS.TEST = ("valid_dataset", )
evaluator = COCOEvaluator("valid_dataset", cfg, False,
output_dir="./output/")
val_loader = build_detection_test_loader(cfg, "valid_dataset")

# Evaluate on the validation dataset
print("Evaluating on validation dataset...")
inference_on_dataset(trainer.model, val_loader, evaluator)
# Set up the test dataset (if different from validation)
cfg.DATASETS.TEST = ("test_dataset", )
evaluator = COCOEvaluator("test_dataset", cfg, False,
output_dir="./output/")
test_loader = build_detection_test_loader(cfg, "test_dataset")

# Evaluate on the test dataset
print("Evaluating on test dataset...")
inference_on_dataset(trainer.model, test_loader, evaluator)
import locale
locale.getpreferredencoding = lambda: "UTF-8"

!pip install pytesseract
!sudo apt update
!sudo apt install tesseract-ocr
import cv2
import torch
from detectron2.utils.visualizer import Visualizer
from detectron2.data import MetadataCatalog, DatasetCatalog
from detectron2.engine import DefaultPredictor
from detectron2.config import get_cfg
from detectron2 import model_zoo
from google.colab.patches import cv2_imshow
from pytesseract import image_to_string

# Load the trained model for inference
cfg = get_cfg()
```

```
cfg.merge_from_file(model_zoo.get_config_file("COCO-
Detection/faster_rcnn_R_50_FPN_3x.yaml"))
cfg.MODEL.WEIGHTS = "/content/drive/MyDrive/final_model/model_final.pth"  #
Path to the trained model
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 13  # Set the same number of classes as in
training
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5  # Detection threshold

# Create predictor
predictor = DefaultPredictor(cfg)

# Load an image
image_path =
"/content/drive/MyDrive/InvoiceDataset/test/i1_jpg.rf.2c5a51c0d356ea9f310096
078da05374.jpg"
image = cv2.imread(image_path)

# Ensure the image is loaded correctly
if image is None:
    raise ValueError(f"Image at path {image_path} could not be loaded. Check
the file path.")

# Make predictions
outputs = predictor(image)
for prediction in outputs["instances"].to("cpu").pred_boxes:
    # Extract the coordinates of the bounding box
    x1, y1, x2, y2 = prediction.numpy().astype(int)

    # Crop the region of interest from the original image
    roi = image[y1:y2, x1:x2]

    # Use Tesseract to do OCR on the cropped image
    text = image_to_string(roi, config='--psm 6')
    print(text)

# Visualize predictions
from detectron2.data import DatasetCatalog, MetadataCatalog

# Suppose you registered your dataset for training as "my_dataset_train"
dataset_dicts = DatasetCatalog.get("invoice_dataset")
my_metadata = MetadataCatalog.get("invoice_dataset")

v = Visualizer(image[:, :, ::-1], metadata=my_metadata, scale=1.2)
# v = Visualizer(image[:, :, ::-1],
MetadataCatalog.get(cfg.DATASETS.TRAIN[0]), scale=1.2)
out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
# Display the image using OpenCV
cv2_imshow(out.get_image()[:, :, ::-1])
!pip install flask
```

```
!pip install flask-ngrok
import os
import io
import requests
import datetime
from googleapiclient.http import MediaIoBaseDownload
from googleapiclient.discovery import build
from google.oauth2 import service_account
from flask import Flask, request, jsonify
from flask_ngrok import run_with_ngrok
import cv2
import torch
from detectron2.utils.visualizer import Visualizer
from detectron2.data import MetadataCatalog, DatasetCatalog
from detectron2.engine import DefaultPredictor
from detectron2.config import get_cfg
from detectron2 import model_zoo
from google.colab.patches import cv2_imshow
from pytesseract import image_to_string
import re
import json


SCOPES = ['https://www.googleapis.com/auth/drive']
SERVICE_ACCOUNT_FILE =
'/content/drive/MyDrive/InvoiceStorage/service_account.json'
PARENT_FOLDER_ID = '1Gf1cVdVLa5mzGUsAOGRWjoMxf7hfy0ms'

app = Flask(__name__)
run_with_ngrok(app)

def authenticate():
    creds =
service_account.Credentials.from_service_account_file(SERVICE_ACCOUNT_FILE,
scopes=SCOPES)
    return creds

def download_file(file_id):
    creds = authenticate()
    service = build('drive', 'v3', credentials=creds)

    request = service.files().get_media(fileId=file_id)

    fh = io.BytesIO()
    downloader = MediaIoBaseDownload(fd=fh, request=request)

    try:
        done = False
        while not done:
            status, done = downloader.next_chunk()
```

```
            print("Download %d%%." % int(status.progress() * 100))

        fh.seek(0)

        target_dir = 'downloaded_invoices/'
        file_name = 'invoice' + '.jpg'
        file_path = os.path.join(target_dir, file_name)

        with open(file_path, 'wb') as f:
            f.write(fh.read())

        print(f'File downloaded successfully: {file_path}')
        return file_path

    except Exception as e:
        print(f'An error occurred: {e}')
        return None


#download_file('1j0B7V71FVWB4lfkD82fCWZCdjJ5APKCJ')

def process_invoice(file_path):
    """Process the invoice and extract information"""

    # Load the trained model for inference
    cfg = get_cfg()
    cfg.merge_from_file(model_zoo.get_config_file("COCO-
Detection/faster_rcnn_R_50_FPN_3x.yaml"))
    cfg.MODEL.WEIGHTS =
"/content/drive/MyDrive/final_model/model_final.pth"  # Path to the trained
model
    cfg.MODEL.ROI_HEADS.NUM_CLASSES = 13  # Set the same number of classes
as in training
    cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5  # Detection threshold

    # Create predictor
    predictor = DefaultPredictor(cfg)

    # Load an image
    image_path = file_path
    print(image_path)
    image = cv2.imread(image_path)
```

```
    # Ensure the image is loaded correctly
    if image is None:
        raise ValueError(f"Image at path {image_path} could not be loaded.
Check the file path.")

    # Make predictions
    # outputs = predictor(image)
    # for prediction in outputs["instances"].to("cpu").pred_boxes:
    #     # Extract the coordinates of the bounding box
    #     x1, y1, x2, y2 = prediction.numpy().astype(int)

    #     # Crop the region of interest from the original image
    #     roi = image[y1:y2, x1:x2]

    #     # Use Tesseract to do OCR on the cropped image
    #     text = image_to_string(roi, config='--psm 6')
    #     print(text)
    #     return(text)

    outputs = predictor(image)
    extracted_texts = []


    for prediction in outputs["instances"].to("cpu").pred_boxes:
        # Extract the coordinates of the bounding box
        x1, y1, x2, y2 = prediction.numpy().astype(int)

        # Crop the region of interest from the original image
        roi = image[y1:y2, x1:x2]

        # Extract text and add to the list
        extracted_text = image_to_string(roi, config='--psm 6')
        extracted_texts.append(extracted_text)

    # Combine all extracted text into a single string
    combined_text = "\n".join(extracted_texts)
    return combined_text


def parse_invoice(text):
    # Define patterns
    invoice_no_pattern = r"INVOICE (\d+)"
    date_pattern = r"Date (\d{4}\.\d{2}\.\d{2})"
    total_pattern = r"Total amount (\d+\.\d+) EUR"


    # Search for patterns in the text
    invoice_no_match = re.search(invoice_no_pattern, text)
    date_match = re.search(date_pattern, text)
```

```
    total_match = re.search(total_pattern, text)

    # Extracting the information
    invoice_no = invoice_no_match.group(1) if invoice_no_match else None

    date = date_match.group(1) if date_match else None
    date_obj = datetime.datetime.strptime(date, "%Y.%m.%d")
    formatted_date = date_obj.strftime("%Y/%m/%d")

    total = total_match.group(1) if total_match else None

    # Constructing the result as a dictionary
    result = {
        "invoice_no": invoice_no,
        "date": formatted_date,
        "total_amount": total
    }
    return result


@app.route('/process_invoice', methods=['POST'])
def api_process_invoice():
    """API endpoint to process invoice"""
    data = request.json
    file_id = data['file_id']
    file_path = download_file(file_id)
    invoice_text = process_invoice(file_path)
    invoice_data = parse_invoice(invoice_text)

    # Clean up: remove the downloaded file
    os.remove(file_path)

    return jsonify(invoice_data)

if __name__ == '__main__':
    app.run()
```

```
pageextension 50100 "Purchase Invoice Ext" extends "Purchase Invoice"
{
    actions
    {
        action(CreateFromFile)
        {
            ApplicationArea = All;
            Image = DownloadFile;
            Caption = 'Create from File';
            Promoted = true;
            Enabled = true;
            0 references
            RunObject = Page "Download File Page";
        }
    }

    procedure UploadFileToGoogleDrive(FileContentStream: InStream): Text
    var
        HttpClient: HttpClient;
        Response: HttpResponseMessage;
        AccessToken: Text;
        FileId: Text;
    begin
        AccessToken := GetAccessTokenFromOAuth2();

        HttpClient.Get('https://www.googleapis.com/upload/drive/v3/files?uploadType=media')
                .Headers.Add('Authorization', 'Bearer ' + AccessToken)
                .Headers.Add('Content-Type', 'application/json')
                .Entity(InStreamToBytes(FileContentStream));

        Response := HttpClient.Send();

        if Response.IsSuccessStatusCode() then
        begin
            FileId := JsonAPIParser.GetResponseValue(Response, 'id');
        end;
```

```
        exit(FileId);
    end;

    procedure SendFileIdToExternalAPI(FileId: Text)
    var
        HttpClient: HttpClient;
        Response: HttpResponseMessage;
        JsonRequestText: Text;
    begin
        JsonRequestText := JsonAPIBuilder.BuildObject('{"fileId": "' + FileId + '"}');

        HttpClient.Post('http://9927-35-193-221-70/ngrok.io/process_invoice')
                .Headers.Add('Content-Type', 'application/json')
                .Entity(JsonRequestText);

        Response := HttpClient.Send();
    end;
}
```