CHƯƠNG 4: PHÂN TÍCH YÊU CẦU VÀ TRIỂN KHAI HỆ THỐNG

Chương này tập trung khảo sát hiện trạng, tổng quan chức năng, đặc tả chi tiết use case quan trọng, và đưa ra các yêu cầu phi chức năng. Toàn bộ nội dung nhằm làm rõ phần "cần xây dựng" trước khi bước sang giai đoạn thiết kế.

0.1 Khảo sát hiện trạng

Hiện nay, việc quản lý giấy tờ trong nhiều tổ chức, doanh nghiệp vẫn chủ yếu dựa trên nhập liệu thủ công hoặc các hệ thống OCR cơ bản. Ví dụ:

- **Giải pháp thủ công:** Nhân viên đọc thông tin trên giấy tờ và nhập lại vào hệ thống. Ưu điểm là linh hoạt, xử lý được nhiều loại giấy tờ, nhưng nhược điểm là chậm, dễ sai sót, tốn nhân lực.
- **Các hệ thống OCR phổ biến:** Một số tổ chức đã sử dụng Tesseract OCR hoặc ABBYY để trích xuất văn bản. Tuy nhiên, các giải pháp này thường khó xử lý bố cục phức tạp, ít hỗ trợ tiếng Việt, và không tự động phân loại loại tài liệu.
- **Ứng dụng tương tự:** Các ngân hàng, công ty bảo hiểm có hệ thống nhận dạng CCCD/giấy tờ. Tuy nhiên, hầu hết hệ thống là giải pháp thương mại đắt đỏ, khó tuỳ chỉnh theo yêu cầu riêng.

Bảng 1 dưới đây tổng hợp so sánh:

Giải pháp	Ưu điểm	Hạn chế
Thủ công	Linh hoạt, không cần phần mềm	Tốn thời gian, dễ sai sót, không mở rộng
Tesseract OCR	Miễn phí, dễ tích hợp	Hỗ trợ tiếng Việt chưa tốt, không phân loại
ABBYY OCR	Chính xác cao, thương mại	Chi phí lớn, ít tùy biến
Giải pháp đề tài	Kết hợp OCR + PhoBERT, phân loại tự động	Cần tập dữ liệu huấn luyện, hiệu năng phụ thuộc GPU

Bảng 1: So sánh một số giải pháp hiện tại

Từ khảo sát trên, có thể thấy nhu cầu cấp thiết về một hệ thống vừa OCR, vừa phân loại tự động, chi phí thấp, dễ mở rộng.

0.2 Tổng quan chức năng

Hệ thống "Nhận dạng và phân loại giấy tờ" có các chức năng chính:

- Cho phép người dùng tải ảnh giấy tờ (JPG, PNG).
- Thực hiện OCR (PaddleOCR + Tesseract tuỳ chọn) để trích xuất văn bản.

- Xử lý văn bản, đưa qua PhoBERT để tạo vector đặc trưng.
- Phân loại tài liệu bằng Logistic Regression, trả về loại giấy tờ (CCCD, Hóa đơn, Hợp đồng…).
- Hiển thị văn bản OCR, ảnh gốc, ảnh đã xử lý, và kết quả phân loại.
- Cho phép quản trị viên huấn luyện lại mô hình với dữ liệu mới.

0.3 Môi trường và phát triển công cụ

0.3.1 Môi trường lập trình

1. Cài đặt các thư viện

Sử dụng các lệnh sau để cài đặt toàn bộ thư viện cần thiết cho dự án:

```
pip install torch torchvision torchaudio
pip install transformers
pip install scikit-learn
pip install joblib
pip install paddleocr
pip install paddlepaddle -U
pip install opencv-python
pip install pillow
pip install pytesseract
pip install streamlit
```

2. Cài đặt Tesseract OCR Engine

Hệ thống sử dụng Tesseract OCR Engine, do đó cần tải xuống và cài đặt thêm công cụ này.

- Đối với Windows: tải bộ cài đặt từ trang GitHub chính thức của Tesseract
- Sau khi cài đặt, thiết lập đường dẫn đến file `tesseract.exe` trong mã nguồn. Cụ thể, thêm dòng lệnh sau vào đầu file:

```
import pytesseract
pytesseract.pytesseract.tesseract_cmd =
    r"C:\Program Files\Tesseract-OCR\tesseract.exe"
```

0.3.2 Cấu trúc và luồng xử lý hệ thống

Giao diện được xây dựng bằng Streamlit, một thư viện giúp tạo các ứng dụng web tương tác một cách nhanh chóng. Giao diện người dùng được thiết kế đơn giản và trực quan, gồm các thành phần chính sau:

Gồm trang chính:

Tải ảnh: Cho phép người dùng tải lên một hoặc nhiều file hình ảnh hóa đơn từ máy tính.

Vùng hiển thị kết quả: Một cột hiển thị ảnh gốc và ảnh sau khi đã được tiền xử lý, giúp người dùng dễ dàng so sánh chất lượng. Cột còn lại hiển thị kết quả OCR và các trường dữ liệu đã được trích xuất.

Vùng phân loại kết quả dự đoán: Hiển thị kết quả dự đoán phân loại giấy tờ sau khi phân tích hình ảnh.

0.3.3 Cấu trúc và luồng xử lý hệ thống

- **Nhận đầu vào:**
  - Hệ thống tiếp nhận hình ảnh từ người dùng.
- **Tiền xử lý ảnh:**
  - Hình ảnh được đưa vào hàm `preprocess()`.
  - Tự động điều chỉnh để tối ưu hóa cho OCR:
    - Chuyển ảnh sang định dạng xám.
    - Khử nhiễu.
    - Áp dụng CLAHE + Otsu hoặc Adaptive Thresholding để chuyển thành ảnh nhị phân.
- **Thực hiện OCR:**
  - Kết quả tiền xử lý được đưa vào hàm `run_ocr()`.
  - Thử nghiệm nhiều chế độ **PSM (Page Segmentation Mode)** của Tesseract để chọn chế độ phù hợp nhất với bố cục hóa đơn.
  - Trả về toàn bộ văn bản thô được nhận dạng.
- **Phân loại tài liệu:**
  - Các đặc trưng được đưa vào mô hình phân loại.
  - Hệ thống tự động xác định loại giấy tờ dựa trên **nội dung, cấu trúc và định dạng**.
  - Trả về kết quả phân loại.

0.4 Phân tích mã nguồn

0.4.1 Ý tưởng bài toán

Trong thực tế, các cơ quan, doanh nghiệp hay cá nhân thường phải xử lý một lượng lớn giấy tờ, văn bản hành chính như: hóa đơn, quyết định, đơn từ, thông

báo,… Việc phân loại và trích xuất thông tin từ các loại giấy tờ này nếu làm thủ công sẽ mất thời gian, dễ nhầm lẫn và tốn nhiều nhân lực.

- Nhận diện chữ trong ảnh giấy tờ (OCR – Optical Character Recognition).
  - Sử dụng PaddleOCR để quét và trích xuất văn bản tiếng Việt từ ảnh chụp/tệp scan.
- Biểu diễn nội dung văn bản dưới dạng vector số.
  - Dùng PhoBERT – mô hình ngôn ngữ tiếng Việt hiện đại – để mã hóa văn bản, tạo ra vector đặc trưng phản ánh ngữ nghĩa của tài liệu.
- Phân loại tài liệu theo từng nhóm.
  - Ứng dụng mô hình học máy (Logistic Regression) để dự đoán loại giấy tờ dựa trên vector đặc trưng.
- Triển khai giao diện trực quan để người dùng có thể tải ảnh giấy tờ, xem kết quả OCR, loại tài liệu và xuất ra file (CSV/Excel).

0.4.2 Huấn luyện mô hình phân loại giấy tờ

- Khởi tạo PhoBERT:

```
MODEL_NAME = "vinai/phobert-base"
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
model = AutoModel.from_pretrained(MODEL_NAME)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)
```

- Giải thích:

- PhoBERT là mô hình ngôn ngữ tiếng Việt dựa trên BERT.
- tokenizer: Chuyển văn bản thành các token số để mô hình hiểu.
- model: PhoBERT để sinh vector đặc trưng.
- device: tự động chọn GPU nếu có, tăng tốc huấn luyện.

- Khởi tạo OCR:

```
ocr = PaddleOCR(use_angle_cls=True, lang="vi")
```

- Giải thích:

- Sử dụng PaddleOCR để đọc chữ trong ảnh tiếng Việt.
- use_angle_cls=True: hỗ trợ xoay chữ.
- Kết quả OCR sẽ được dùng làm đầu vào cho PhoBERT.

- Hàm OCR ảnh:

```
def extract_text(image_path: str) -> str:
    result = ocr.ocr(image_path)
    lines = []
    if result:
        for res in result:
            for line in res:
                lines.append(line[1][0])
    return " ".join(lines)
```

- Giải thích:

- Đọc ảnh và trích xuất các dòng văn bản.
- Kết quả trả về là một chuỗi text.

- Hàm sinh vector từ văn bản:

```
def embed_text(text: str) -> np.ndarray:
    if not text.strip():
        return np.zeros(768)
    inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=
    with torch.no_grad():
        outputs = model(**inputs)
    return outputs.last_hidden_state.mean(dim=1).cpu().numpy().flatten()
```

- Giải thích:

- Nếu text rỗng → trả về vector 768 số 0.
- Với text có nội dung:
  - tokenizer: mã hóa văn bản.
  - model: PhoBERT tạo ra biểu diễn vector.
  - mean(dim=1): lấy trung bình embedding của các token → vector duy nhất (768 chiều).

- Hàm xây dựng dataset:

```
def build_dataset(dataset_dir: str):
    X, y = [], []
    labels = sorted(os.listdir(dataset_dir))
    for idx, label in enumerate(labels):
        folder = os.path.join(dataset_dir, label)
```

```
        ...
                text = extract_text(img_path)
                vec = embed_text(text)
                X.append(vec)
                y.append(idx)
    return np.array(X), np.array(y), labels
```

- Giải thích:

- Duyệt qua thư mục dữ liệu (theo từng nhãn).
- Với mỗi ảnh:
  - OCR → văn bản.
  - Văn bản → vector PhoBERT.
  - Lưu vector vào X, nhãn vào y.

- Huấn luyện và lưu mô hình:

```
clf = LogisticRegression(max_iter=2000)
clf.fit(X, y)

joblib.dump(clf, "doc_classifier.pkl")
with open("labels.json", "w", encoding="utf-8") as f:
    json.dump(labels, f, ensure_ascii=False, indent=2)
```

- Giải thích:

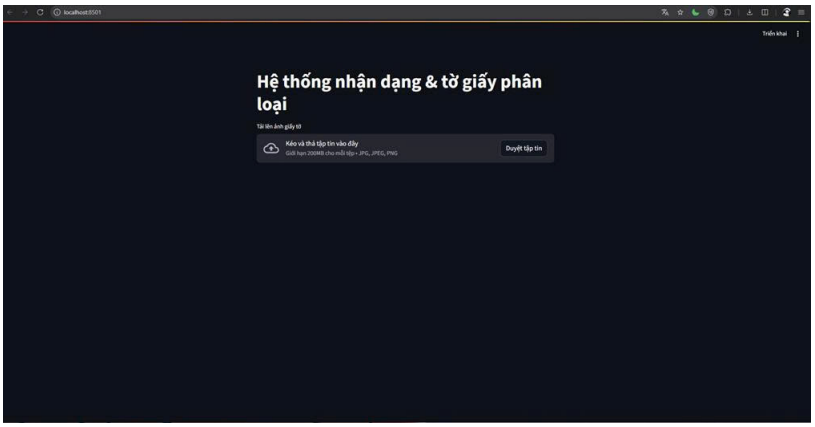
- Logistic Regression được chọn làm mô hình phân loại.
- Sau khi train:
  - Lưu model (doc_classifier.pkl).
  - Lưu danh sách nhãn (labels.json).

Sử sụng lệnh để chạy: streamlit run app.py

Khi chạy lệnh sẽ hiển thị ra trang giao diện như sau:

Sau đó, người dùng tải ảnh cần phân loại lên, ví dụ:

Kết quả sau khi quét, nhận dạng được đây là giấy tờ hóa đơn



Hình 0.1: Dao diện chương trình.

Hình 0.2: Hình ảnh cần phân loại.

Hình 0.3: Kết quả sau khi nhận dạng.