Automative filling of an Accident Report

Domain Understanding

Alpay Demirci[1] and Alec Sola[2]

[1]Fontys University of Applied Sciences

May 3, 2025

Abstract

This document outlines the technical research for the individual project. Since the domain context is already covered in the data challenge and business proposal, the focus here will be more technical.

1 Introduction

Extracted from NSC Injury Facts, originally from National Safety County: "Between 1913 and 2022, the number of motor-vehicle deaths in the United States (which include all types of motor vehicles, including passenger cars, trucks, buses, and motorcycles) increased 996%, from 4,200 deaths in 1913 to 46,027 in 2022."

This article highlights how fatal accidents have evolved over time. While data on non-fatal accidents is limited, it offers useful perspective.

Filing a car accident report is often one of the most stressful parts of an already stressful situation. Right after a collision, emotions are running high, people might be injured or in shock, and yet they're expected to calmly fill out a detailed, legally relevant document on the spot. The process is time-consuming usually taking 15 to 20 minutes, and not always straightforward. Many people struggle with:

- Figuring out what information goes where.
- Drawing the situation sketch accurately.
- Avoiding mistakes that could later cause problems with insurance claims.

To make things worse, the form is often confusing, especially for those unfamiliar with legal paperwork or not fully fluent in the language. And if someone is injured or shaken up, it's even harder to write clearly or think straight.

In short: at the very moment when drivers need clarity and simplicity, they're faced with a slow, complex, and frustrating process.

2 Objectives

Automatic filling of an accident report on the phone. There are two separate tasks to complete.

Aim 1: Automatic extraction of variables from documents and filling in the report.

- Store original Identity Documents.
- Transform original pictures to annotated ones where the important sections are highlighted.
- Autofill the digital report.

Aim 2: Generation of an image representing the accident.

- User fills a prompt describing the accident with a specific structure.
- Generation of a 2D drawing/picture representing the accident.

3 Aim 1

The objective of Aim 1 is to automatically extract key variables from identity-related documents and images provided by the users. The following steps outline the proposed process:

3.1 Document Storage and Preprocessing

All uploaded documents (scanned ID cards, driver's licenses, license plates) are securely

Figure 1: Original ID example

Figure 2: Annotated Identity Document with YOLOv8

stored in their original format. This is still not clear, however it could be that each image undergoes preprocessing to improve accuracy:

- Resolution optimizing overall.

3.2 Region Detection

To extract only relevant sections of each document (name, ID number, expiration date), object detection models are employed, these are different alternatives:

- **YOLOv8** (You Only Look Once, version 8) is a real-time object detection model known for its high speed and accuracy. It is designed to run efficiently even on mobile devices and edge hardware, making it suitable for extracting fields from documents in real time.

  Source: ultralytics

- **EfficientDet** is a scalable object detection model developed by Google. It uses a lightweight backbone (EfficientNet) and a compound scaling method to balance accuracy and efficiency, making it a strong choice for deployment in resource-constrained environments.

  Source: Cornell University

  These models are fine-tuned to detect text-containing fields on various document types.

3.3 Text Extraction via OCR

The detected regions are passed to an OCR engine to extract text:

- **Tesseract** is a widely used open-source OCR engine developed originally by HP and now maintained by Google. It's effective for printed, high-contrast text in structured documents and supports over 100 languages. While powerful, it struggles with noisy or complex layouts.

  Source: Github

- **EasyOCR** is a Python-based OCR library that uses deep learning under the hood (e.g., CRNN + CTC). It supports 80+ languages and tends to outperform Tesseract in cases involving rotated or irregular text, though it may be heavier on resources.
  Source: Github

- **TrOCR** (Transformer-based OCR) is a model developed by Microsoft that uses a Vision Transformer (ViT) encoder and a text-generating Transformer decoder. It achieves state-of-the-art results on difficult OCR tasks, including handwritten or distorted text, thanks to its end-to-end learning approach.

  Source: Hugging Face

3.4 Field Structuring and Classification

Once raw text is extracted by OCR, it must be structured, meaning, each piece of text (like a name, ID number, or date of birth) needs to be correctly classified and mapped to the appropriate field in the accident report. This is not trivial, since many documents contain complex layouts, multiple columns, and labels in varying positions.

To handle this, **LayoutLMv3** is a transformer-based model developed by Microsoft that processes documents not just as plain text, but as a combination of:

- Textual content.
- Visual features (pixel-level embeddings).
- Spatial layout (bounding box coordinates of each token)

This multimodal approach allows it to "read" forms and documents more like a human would understanding that a number next to the label "Date of Birth" likely belongs in that field, even if the document layout is unusual.

It is trained on large-scale document datasets and performs especially well on form understanding, invoice parsing, and identity documents.

3.5 Autofill Report

The structured data is automatically inserted into the digital accident report.

4 Aim 2

4.1 Overview

1. User fills in a structured prompt describing the accident (vehicle directions, location, impact point, time of day, weather...).
2. An LLM parses the description and converts it into a formal scene specification.
3. A rendering module uses that specification to generate a 2D image of the accident.

4.2 Prompt Parsing with a Language Model

A central component is the use of a **Large Language Model (LLM)** to convert the user prompt into a machine-understandable scene. Two implementation options are proposed:

Option 1: Customized GPT (via OpenAI's GPTs platform)

A custom GPT can be trained or configured using OpenAI's GPTs interface to:

- Guide the user in writing the prompt in a structured format ("Car A was heading north on Street X; Car B was turning left from Street Y").
- Parse this text into a JSON object describing scene elements such as:
  - Vehicle positions and headings
  - Type of collision (rear-end, side-impact)
  - Road layout (crossroads, roundabout)
  - Environmental conditions (night, rain)

Option 2: Fine-tuned open-source LLM

Alternatively, a local or open-source model like **LLaMA**, **Mistral**, or **Phi-2** can be fine-tuned on synthetic or historical accident descriptions to produce these representations. These models

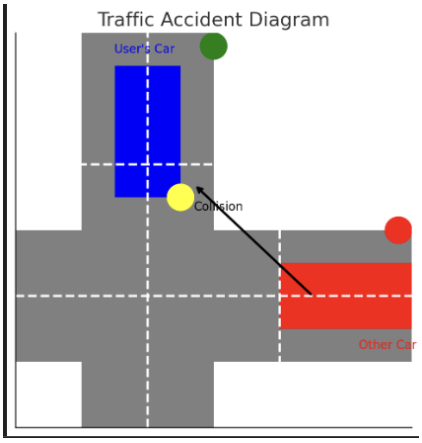

Figure 3: GPT diagram example

are transformer-based language models, meaning, similar in architecture to GPT because they are capable of understanding and generating natural language. They are trained on large corpora and can be adapted to specific domains like accident reporting via supervised fine-tuning.

Source LlaMA: Meta

Source Phi-2: Hugging Face

5 Validation Metrics

As a disclaimer, just because specific metrics are presented, does not mean that they will be implemented. This section covers research on some that could eventually fit.

5.1 Aim 1: Document Understanding and Autofill

This task involves detecting and extracting fields correctly (name, license number, expiration date) from structured documents.

- **Precision, Recall, F1 Score** for field-level classification (is "John Doe" correctly identified as the name?).
- **Exact Match Accuracy** measures how often the extracted field exactly matches the ground truth.
- **Word Error Rate (WER)** for OCR outputs, compares how many insertions, deletions, or substitutions occurred in text recognition.
- **IoU (Intersection over Union)** if bounding boxes are predicted (via YOLO), this measures how well the predicted region matches the ground truth box.

5.2 Aim 2: Scene Interpretation and Image Generation

This task is more complex because it is converting a text prompt into a structured scene representation (JSON) and then into a visual output. Two steps can be evaluated:

1. Text-to-JSON Accuracy (LLM evaluation)

- **Exact Match Accuracy**. Does the model output the correct structure (vehicle positions, impact type)?
- **Slot-Filling Accuracy** measures whether each required field ("direction", "vehicle ID") is correctly filled.

2. Image Accuracy

Since 2D drawings are deterministic renderings of the structured scene:

- **Human evaluation**. Does the image reflect the correct number of cars, directions, collision point?
- **Scene Similarity Score (custom)** comparing elements like objects present, spatial layout, or arrows between generated and reference image.

5.3 What counts as accurate:

This is yet to be determined with the teachers. For now it is important to be ambitious, but at the same time, realistic.

- 60% slot-filling accuracy in structured output.
- Human-rated match with ground truth in 45-50% of generated drawings.
- No missing core elements (Missing one of the vehicles).

6 Conclusion

Throughout this document, the various technical approaches for completing the two main tasks have been presented. From this point forward, conversations will be held with the teachers to evaluate what is best.