Tesseract OCR vs Microsoft Azure

Research Document

Alpay Demirci & Alec Sola[1]

[1]Fontys University of Applied Sciences

June 3, 2025

Abstract

This document outlines the technical research for the text extraction process and results of Tesseract OCR vs Microsoft azure's Form Recognizer.

1 Introduction

The main objective is to extract key information from various images of identity documents, including national IDs, driver's licenses, and license plates.

This can be achieved using pre-trained models like Tesseract OCR or Microsoft Azure's Form Recognizer.

The way this document will be structured is as following. Each Identity document will have its own section and a comparison in results will be made between the two models.

2 Tesseract OCR & Microsoft Azure Form Recognizer

Before heading to the results of each identity document, it is important to explain what these models are and where they are used.

Sources: Github repository Tesseract OCR, **Azure Form Recognizer**

Tesseract OCR was originally developed at Hewlett-Packard Laboratories in Bristol (UK) and Greeley, Colorado (USA) between 1985 and 1994. Additional work was done in 1996 to port it to Windows, and in 1998 parts of the code were converted to C++. In 2005, Tesseract was open-sourced by HP. From 2006 to November 2018, it was maintained and further developed by Google.

Tesseract is an open-source optical character recognition (OCR) engine designed to run locally. Its purpose is to detect and extract text from images and scanned documents, supporting more than 100 languages. It operates by analyzing the structure of a document; identifying text blocks, lines, words, and characters, and converting them into machine-readable text.

Common use cases include document digitization (such as converting scanned books, forms, or handwritten notes into searchable and editable text), license plate recognition, and automation of data entry processes by extracting text from structured documents like tables and forms.

Azure Form Recognizer is a cloud-based AI service provided by Microsoft that automatically extracts text, key-value pairs, tables, and layout structures from documents. Unlike traditional OCR tools, it not only reads the text but also understands the layout and semantic relationships within the document.

The process involves uploading the document via an API or SDK, selecting a pre-built or custom model, after which the engine analyzes the content and layout. The extracted data is returned in a structured JSON format, which can then be integrated into downstream systems or workflows.

Form Recognizer is widely used in scenarios such as identity verification, financial document processing (invoices, receipts, purchase orders, and bank statements), logistics (shipping labels, bills of lading), human resources (digitization of employee forms), and healthcare (processing medical and insurance documents).

3 Dutch Identity Document

Figure 1: Identity Document

The input data will be the ID of Alec Sola, the reason this was chosen is due to the privacy issues it causes to have a dataset with other people's Identity Documents. The fields to extract are:

- Full name
- First name
- Nationality

3.1 Tesseract OCR

This credential presents a challenge for the model due to noise, particularly on the right side, where random letters appear. This unwanted text is often included in the output, making it harder to accurately extract the relevant fields.

To address this, the image was divided into three sections: left, middle, and right. For the purpose of this assignment, only the middle section was used, as it contained the essential information. However, even with this focused approach, the model still encountered issues in certain cases. These were resolved through hard-coded fixes.

- Month Recognition: "IUN" instead of "JUN".
- Month Structure: The desired result is "DEC"=12, however the output was being "DEC/DEC".
- Random text: "Nationltt" instead of "Nationaliteit".

Then end results though were good and the desired fields were retrieved.

3.2 Azure Form Recognizer

With Azure, the process is simple: all that's required is the API endpoint and the access key. Once set up, the input images are submitted separately, and the model handles the rest. The results are impressive, field detection is highly

```
Text Extraction > Outputs > IDAlecTextExtraction.txt
1  IDfirst_name: Juan Alejandro
2  IDname: JUAN ALEJANDRO SOLA CASTERMANS
3  sex: MM
4  nationality: Nederlandse
5
```

Figure 2: OCR ID output

accurate. Each field is treated as an object and returned in the output, which can be reviewed either through the API response or directly in the output data.

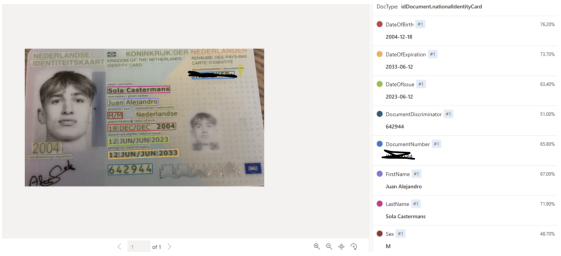

Figure 3: Form Recognizer ID screenshot

4 License Plate

OCR performs well when it comes to reading license plates, whereas Azure currently does not offer a pre-built model specifically designed for license plate text extraction.

To improve text recognition, the image was preprocessed using OpenCV (cv2). First, it was converted to grayscale to simplify the image and reduce computational complexity. Then, a bilateral filter was applied to reduce noise while preserving important edges, which are crucial for accurate character detection. Finally, adaptive thresholding was used to enhance contrast between the text and the background, especially in cases where lighting conditions vary across the image. The results were satisfactory.

Figure 4: License Plate Preprocessing

5 Drivers License

This drivers license was found in CBS, which is accepted to the public for use.

Figure 5: Drivers License

Fields to be extracted:

- First name
- Full name
- Issue & Expiry date
- License number
- License category

5.1 Tesseract OCR

The driver's license was processed in a similar way to the ID; however, this document contained significantly less noise.

The image was cropped to focus on the right side, where all the relevant information is located. Basic preprocessing was applied using OpenCV (cv2), but in this case, only grayscale conversion was needed.

Despite this, the model still struggled with:

- Name: The credential had the word "Specimen" next to the name field, this caused confusion.
- License Category: Close to the signature and it was getting the letter "D" from it.

These issues were hardcoded and the results ended up being solid, the fields desired were collected.

Text Extraction > Outputs > DriversLicenseTextExtraction.txt

```
first_name: Tamara D
name: TAMARA D VAN BOURGONDIË
date_of_birth: 19.02.1990
issue_date: 20.01.2013
expiry_date: 20,01,2023
license_number: 1234567890
categories: AM-A-BE-C
```

Figure 6: Drivers License Output

5.2 Azure Form Recognizer

As with the identity document, the process was straightforward. The results were excellent, accurate and fast on the first attempt. Additionally, the model was tested on other driver's licenses to evaluate its consistency, and it worked good.

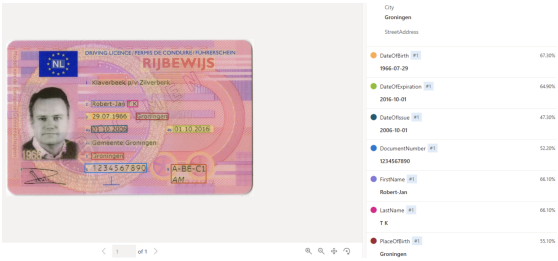

Figure 7: Azure Drivers License

6 Conclusion

After working with both models, drawing conclusions becomes straightforward. The evaluation is based on three key aspects: **efficiency, sustainability, and ethical considerations**.

For the task of automatically filling out a car accident report by extracting text from images, **Microsoft Azure Form Recognizer is clearly the better choice** for this specific step. It is user-friendly, requiring little to no code thanks to its API, and it is both fast and highly efficient. Most importantly, it accurately understands the layout and structure of documents, meaning there's no need for hardcoded rules or manual corrections which make the process not sustainable.

While there may be concerns about privacy, since Form Recognizer is a cloud-based solution, it's important to note that Microsoft provides robust data protection measures, including compliance with major security standards (like ISO/IEC 27001, GDPR, and SOC). Data can also be processed within a specific region to align with local privacy regulations.

Tesseract OCR, on the other hand, is best suited for reading license plates, as Azure currently lacks a pre-built model for that specific task. It may also be preferred in cases where cost is a major concern, as Tesseract is completely free and runs locally, while Azure Form Recognizer operates on a subscription-based pricing model.

In summary, while Tesseract may still play a role in isolated parts of the pipeline, Microsoft Azure Form Recognizer significantly outperforms Tesseract OCR in terms of efficiency, simplicity, and document understanding.