***Business Card Name Entity Recognition***

**Abstract**

This paper is about building a project that extracts texts from business cards using a combination of the two biggest fields in Artificial Intelligence in Computer vision (CV) and Natural Language Processing (NLP) for Named-Entity Recognition tasks. The paper shows every stage from input to the retrieval of final output that runs on a local web-application written is Python bundled with other advanced front-end frameworks and Javascript library like Bootstrap, JQuery, HTML, and CSS. The project was run in python for most parts as there are libraries publicly available on the internet which help to facilitate in every task. The paper starts off by giving introduction on the subject, to allow the reader to know the trend in the space of NER (Named Entity Recognition). Background is also written in order to see where the inspiration of the paper came from. Methods show how the author built the product/project using the techniques of Image Processing. The results would show how the final output would look like in a working web application.

**Introduction**

As mentioned in the abstract, this paper is an implementation project to extract and retrieve words and texts from business cards with NLP and CV tasks for NER tasks. Named Entity recognition involves processing a text and identifying certain occurrences of words or expressions as belonging to particular categories of Named Entities (NE) (Mikheev et al., 1999). NE recognition software serves as an important preprocessing tool for tasks such as information extraction, information retrieval and other text processing applications (Mikheev et al., 1999). This project is divided into subsequent major sections: Background, Methods, Results and Conclusion.

**Background**

This paper was inspired by a developer called Mike Polinowski (Tesseract OCR on Arch Linux | Mike Polinowski, 2021). Although the code that I have written is similar to Mike’s code in the article, I have made some modifications and the libraries that I thought would be beneficial to the project. I have followed most of the steps from Mike Polinowski up to a certain point only. It is similar until the end of ‘Methods’ section. There are limitations to Mike’s project as he did not show how the project works in the real world. Additionally, he did not apply Image Processing techniques to the final output. I have run the project in the local server with other advanced front-end web technologies like Flask, etc.

**Methods**

As mentioned in the introduction section, the final output of the project is a web-app written in python that was written in Python. Figure 1 shows the last stage before extracting words and categories from the respective business card.

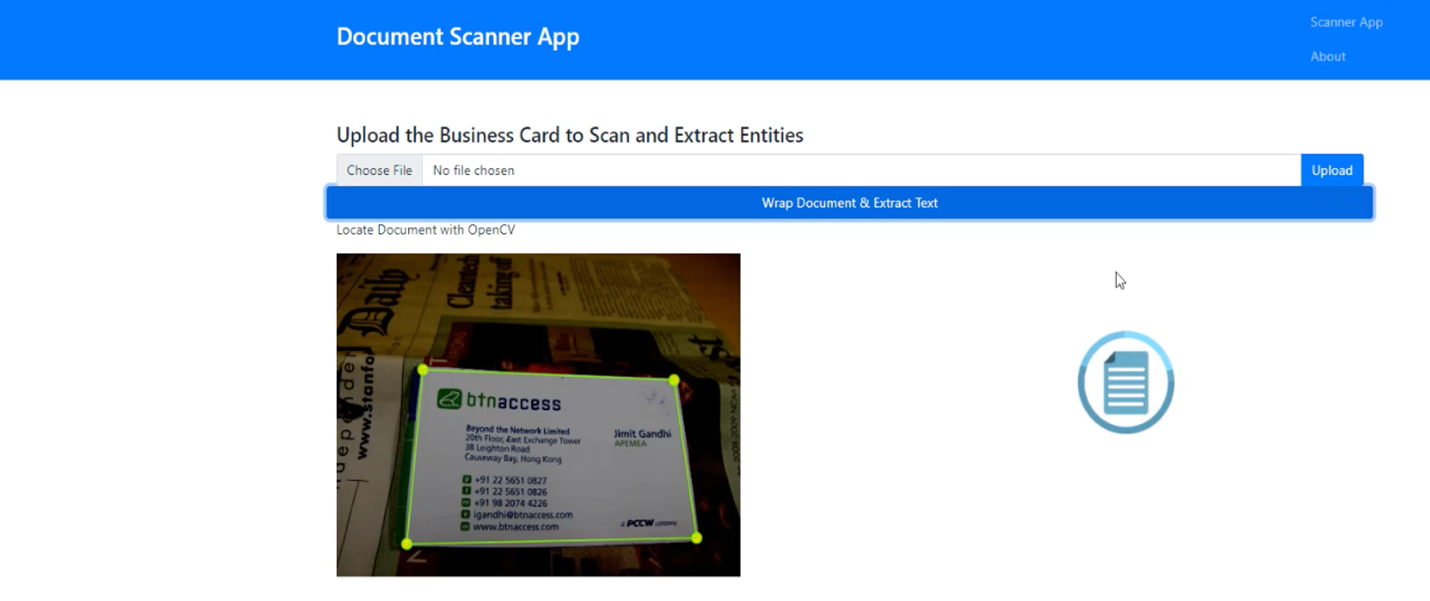


Figure 1 – Uploading Image

Figure 2 shows the final output of the project. From Figure 2, we can see that the image does not look exactly like the image the user intends to extract information from. The reason is because image processing techniques were applied to the image on the right. After Image Processing techniques were applied, extraction then starts happening by matching words found with the pre-defined categories like Name, Organization, Designation, Phone, Email, and Website. Before getting into Image Processing steps, the steps prior to the output will be explained thoroughly.

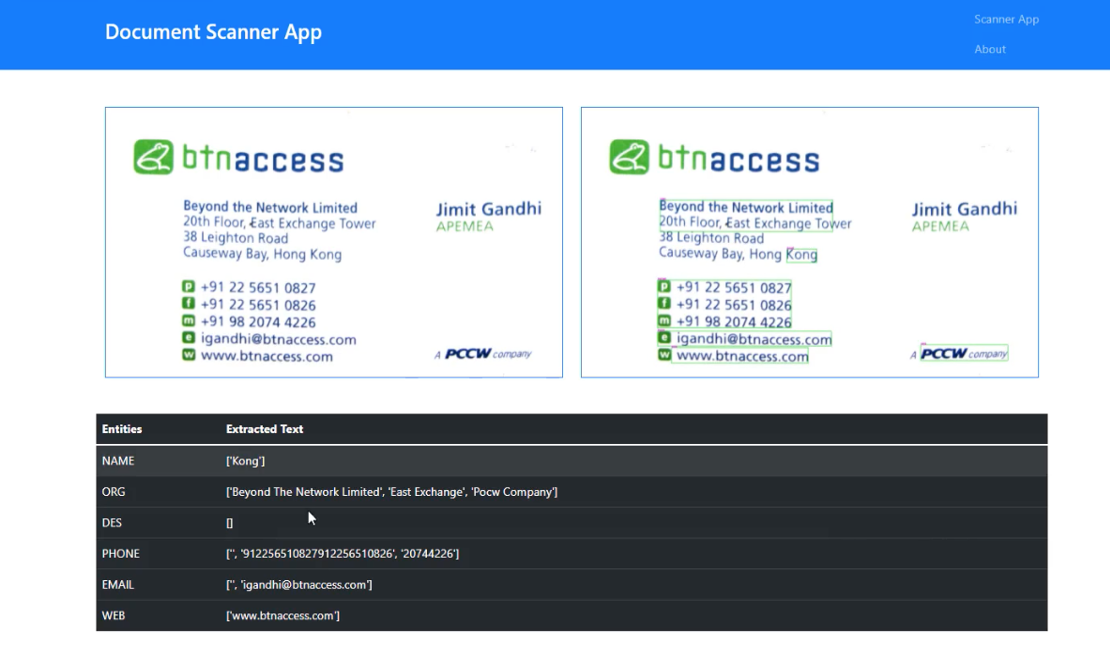


Figure 2 – Sample Final output

Figure 3 shows the overall architecture that was designed for the web-app.

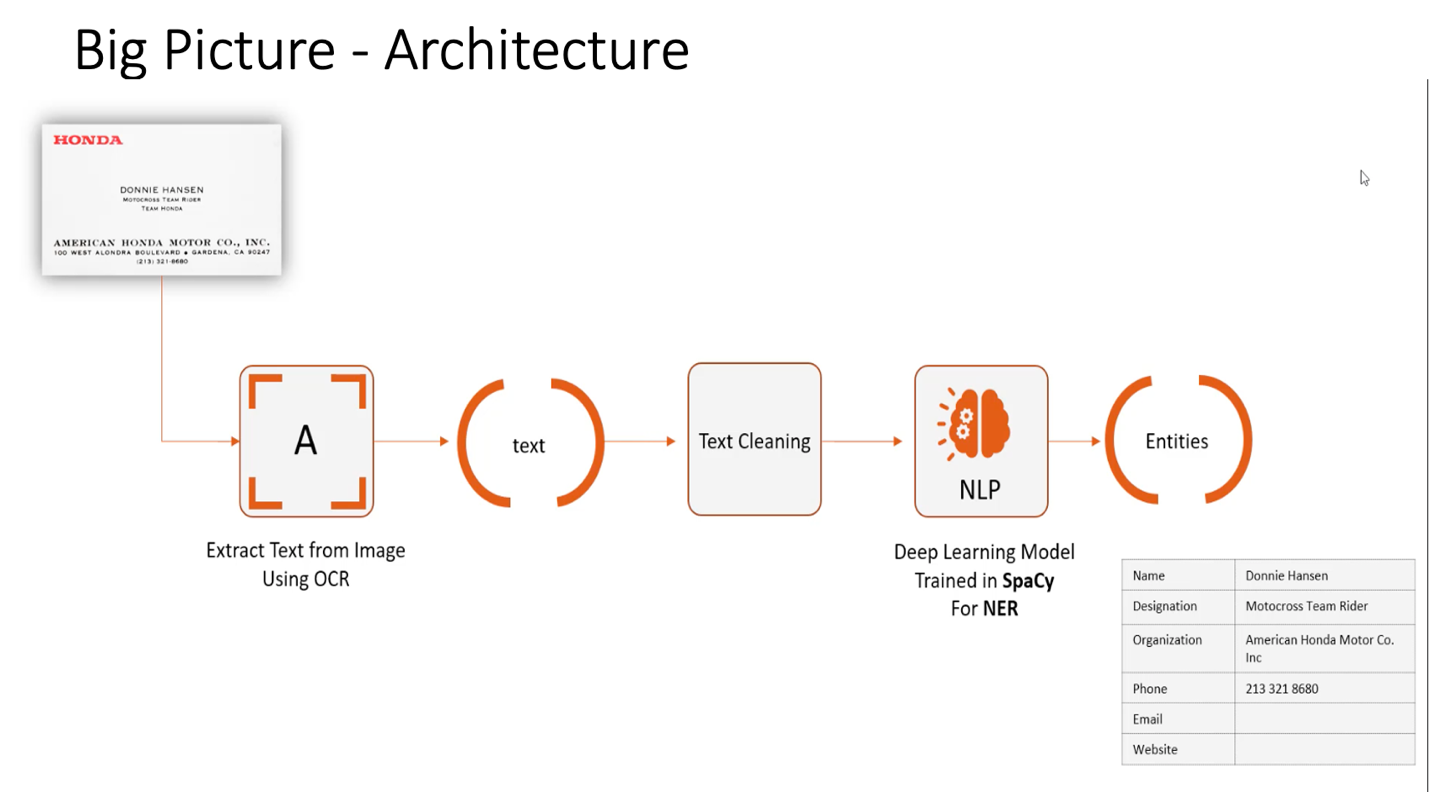


Figure 3 - Architecture

There are 6 important stages of development. Figure 4 shows an overall depiction of the stages of development.

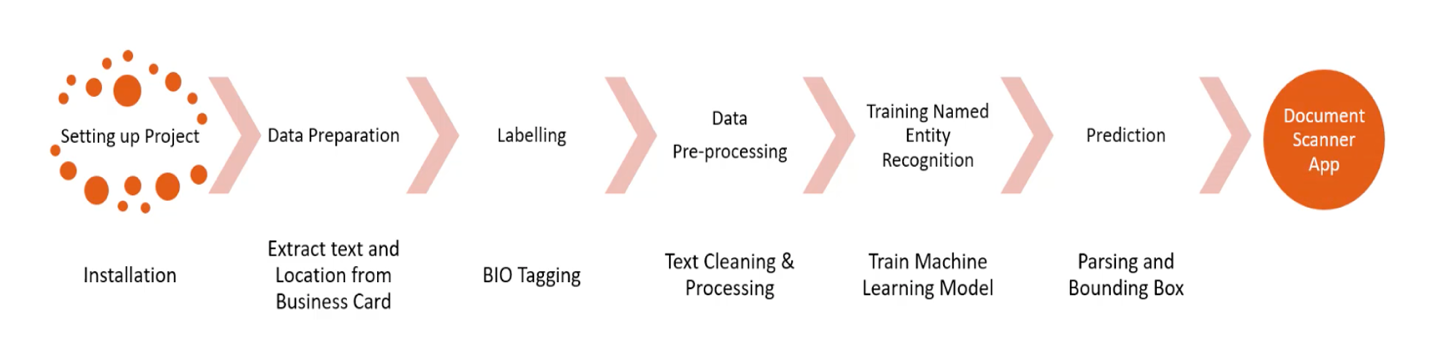


Figure 4 – Stages of Development

At figure 4, the most important step is to install all the necessary libraries. This is the first step of the project. This project was greatly supported by libraries in CV and NLP to help with all kinds of tasks. Looking at the Computer Vision aspects, the main libraries that were used are OpenCV and Tesseract OCR or Pytesseract. OpenCV is an open-source library that includes several computer vision algorithms. It can be used in Image Processing, Video Analysis, Object Detection and many more. Tesseract is an open-source text recognition (OCR) Engine, available under the Apache 2.0 license (Zelic, 2022). It can be used with the existing layout analysis to recognize text within a large document, or it can be used in conjunction with an external text detector to recognize text from an image of a single text line (Zelic, 2022). In this project, the OpenCV helped with three main tasks: Scanning Documents, Identifying Location of Texts, and Extracting Text from Image. Looking at the Natural Language Processing aspects, Pandas, SpaCy, and RegEx were used. Pandas was essential in cleaning data. Spacy library was used to train machine learning models. SpaCy allows developers to merge existing data with a JSON-like code format to train the machine learning models that go in accordance with the standard format of SpaCy-like codes. By making sure that the format aligns, developers are able to use configuration files on the SpaCy-like code for the process of Machine Learning. Moreover, Regex library was also used to make sure that junks that are found in the code and other unnecessary information are removed.

Second step is data preparation. This involves extracting texts and location of words in the business card. As mentioned, Pytesseract is the most important library in the project as it helps in extracting texts from the business card. There are 5 steps that Pytesseract take in order to extract texts from words. First, it scans the card. Second, it puts a box on all the boxes it finds in the business card. Third, it puts new boxes on paragraphs as a consequence of finding boxes in the second step. Fourth, it puts a box on every line horizontally. Fifth, it extracts every single word it finds. Figure 5 – 9 shows all the steps Pytesseract took to get to the final step of extraction of words.



Figure 5 – Pytesseract Page/Card

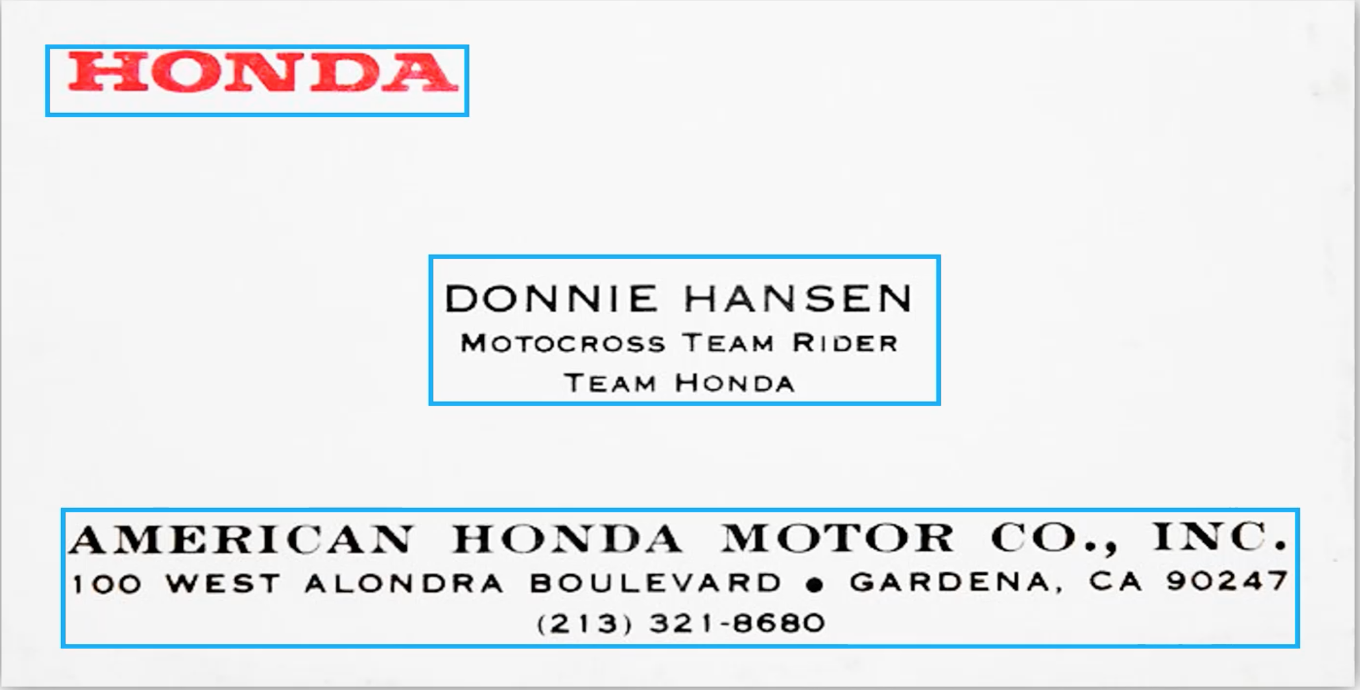


Figure 6 – Pytesseract Block



Figure 7 – Pytesseract Paragraph

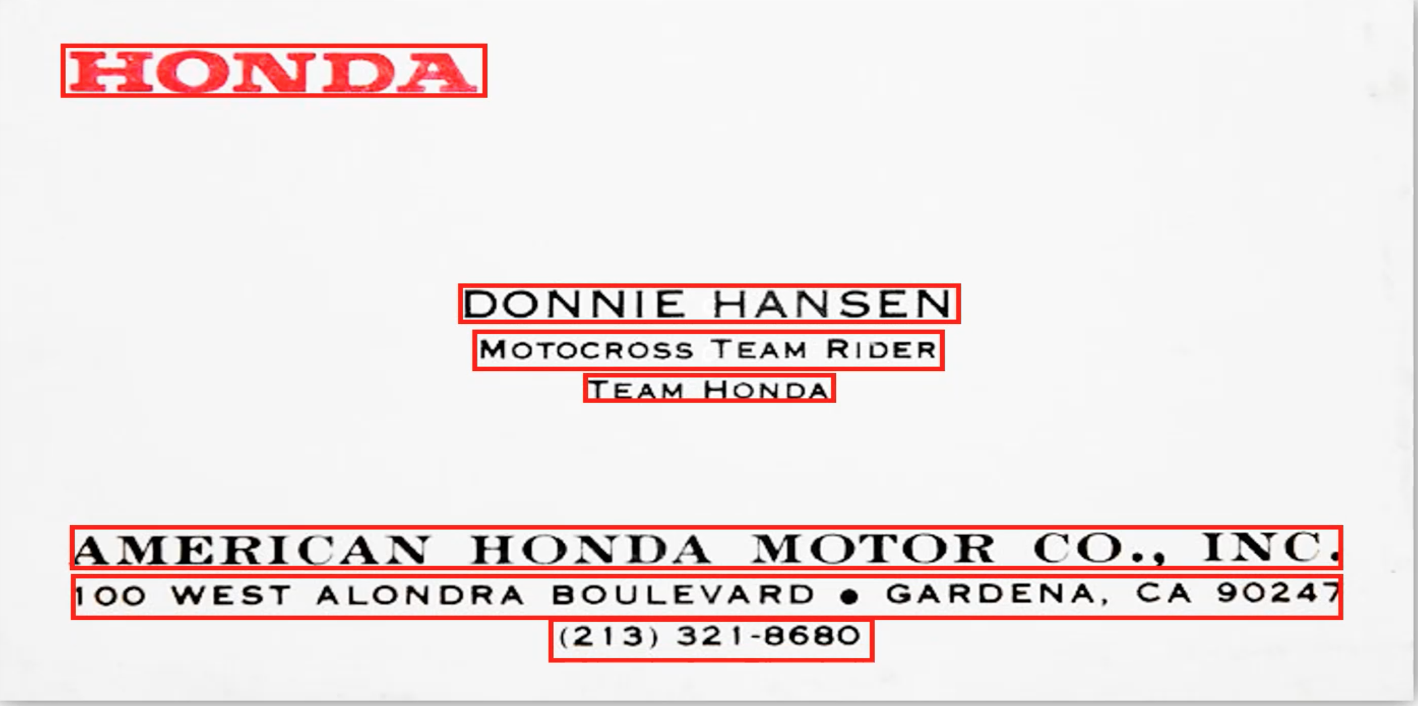
****

Figure 8 – Pytesseract Line

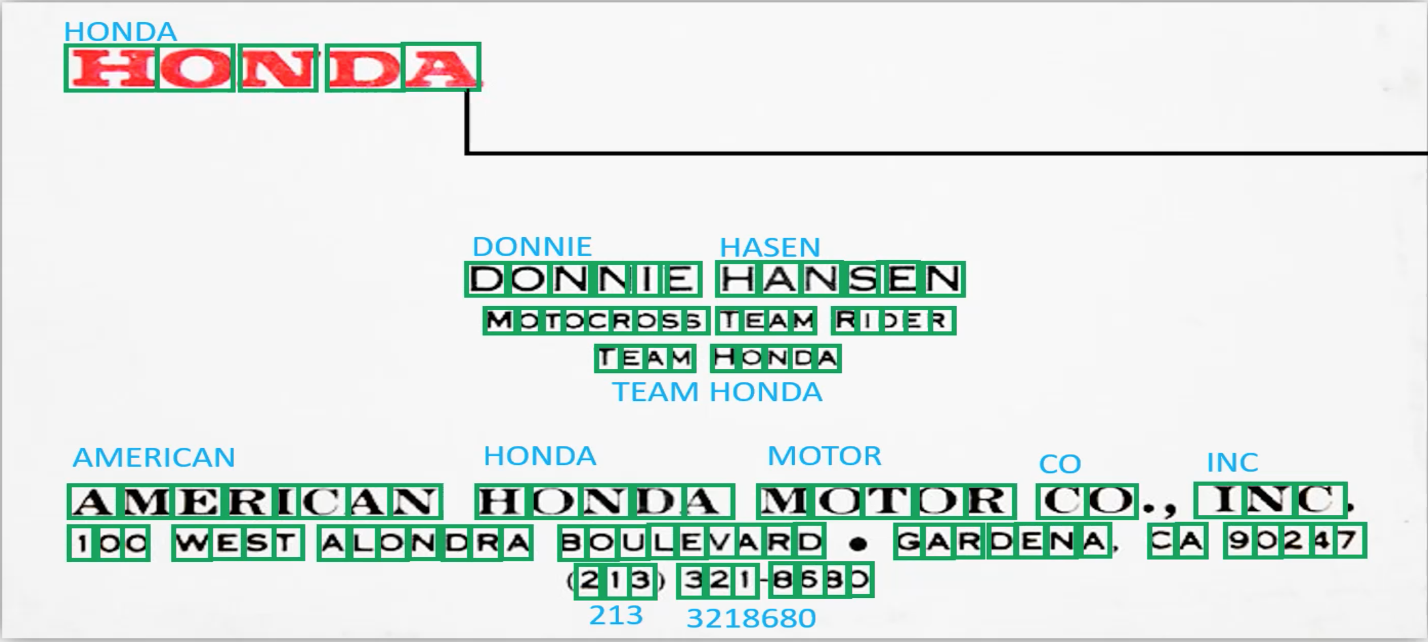
****

Figure 9 – Pytesseract Words

Third, after the data preparation step is done and there is enough data, labeling is done in order to match words with categories. This is a crucial step of Machine learning pipeline as the model will learn what categories are for what words. 273 Images were used for labelling. Labelling is done manually with a technique of labeling called BIO (Beginning, Inside, and Outside). Figure 10 shows how labeling is done in the process. Looking at the excel sheet in row 4, B-PHONE means the beginning of the Phone number box it finds. On row 5, I-PHONE would be what is inside of the B-PHONE on row 4. O is other incomplete information or junk that the algorithm cannot make sense out of that it finds.

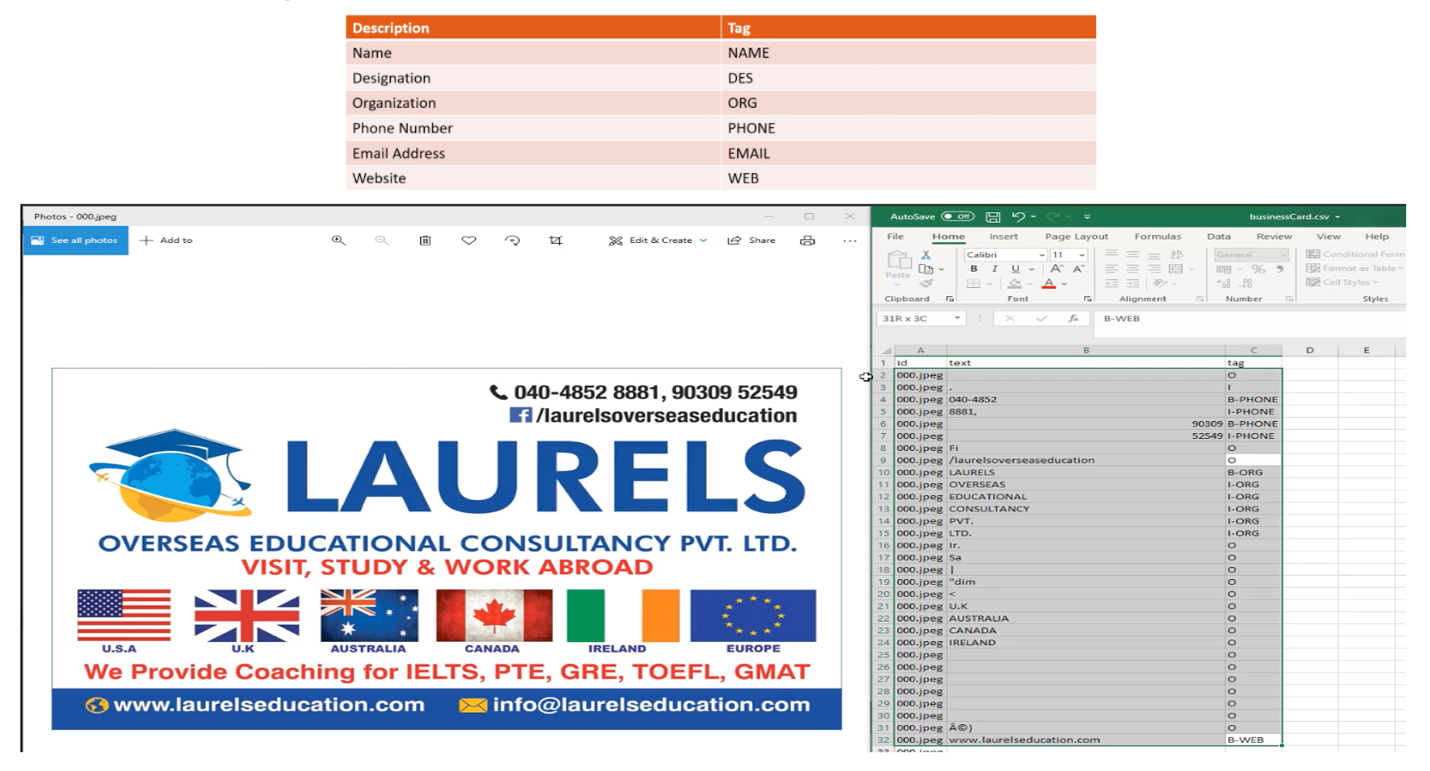


Figure 10 – BIO Labelling Based on Pre-defined Categories

Fourth, the data in the excel/csv files are cleaned as a step for Natural Language Processing was used with libraries like Pandas and Regex. Texts are cleaned before the data enter the actual training step in the Machine learning pipeline. This is very important as the SpaCy library used in the project has its own format. Cleaning is done to make sure that all the data is in accordance with the SpaCy format called entities as seen in Figure 11. The numbers in top of the first line in figure 11 is the exact location in numerical terms for each item in the business card.



Figure 11 – Entities(Goes into SpaCy)

Fifth, Training and Testing with Machine Learning starts with a distribution of 90% for Training data and 10% for Testing data. Figure 12 shows the completion of the process of labelling. Each element gets paired to its category. What is needed in the final output is only the main categories. Any elements found with the technique of BIO will be added up to the main categories. This step goes in sync with the prediction step as it puts a box on elements to describe the big headings that are found.

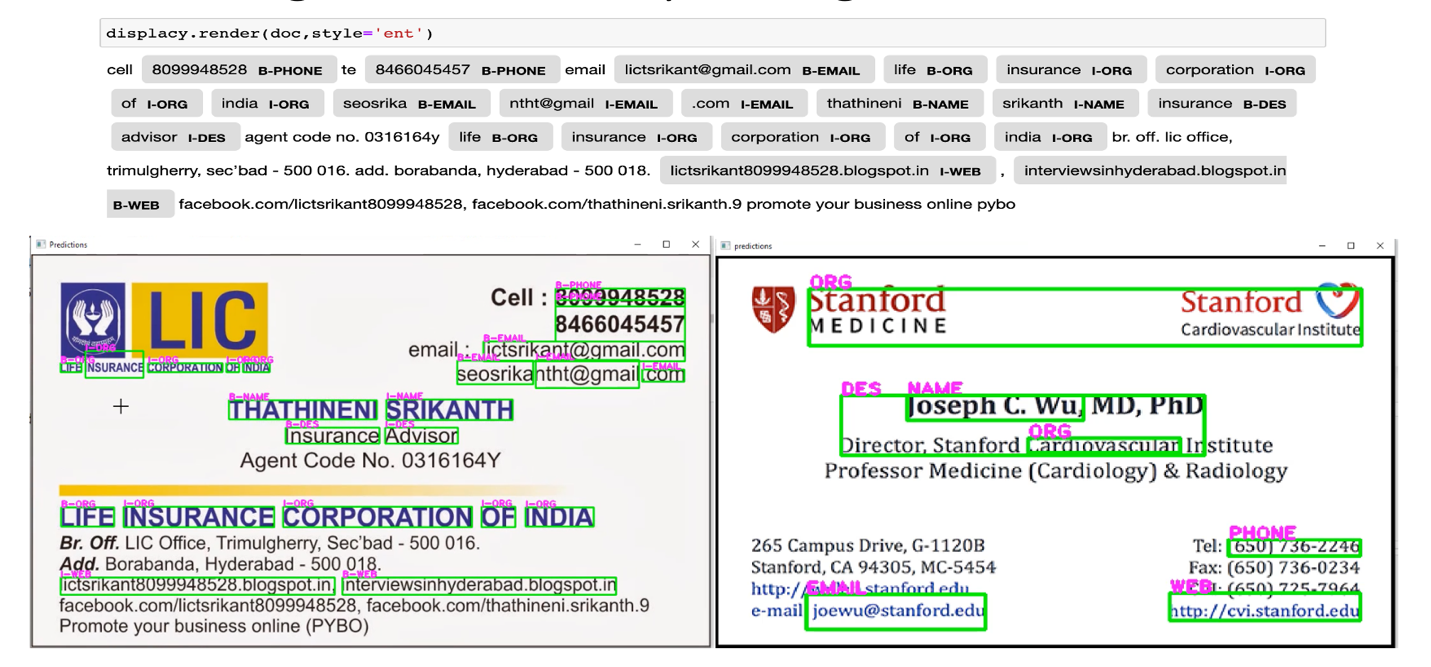


Figure 12 – SpaCy Words Matching

Figure 13 shows the prediction results that were found, although not formally displayed in the web-app, but that will be shown later after the Image Processing stage. Here, only the values from ‘token’ and ‘label’ columns will be matched with each other at the output stage.

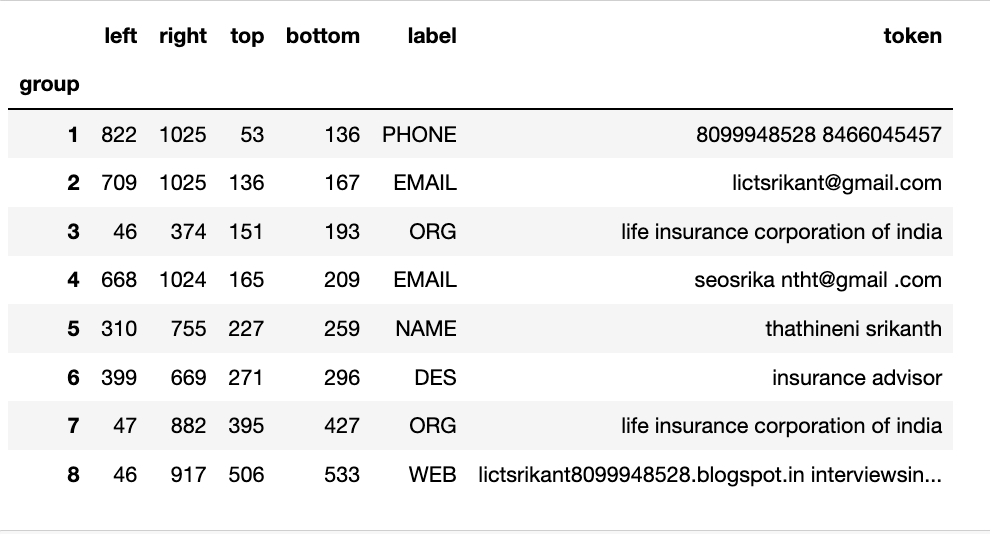


Figure 13 – Output prior to being Displayed on Web App

**Results**

This section will cover the Image Processing step that make modifications on the final Image. The steps are Resizing, Enhancing, Canny filters, Morphological Transformation, Contours finding, and Four-corners finding. Figure 14 shows the code for the Image Processing step. As we see with enhancement, more sigma means more detail enhancement. Gaussian Blur was also used at 5x5 that is passed on to edge detection stage for canny filters. After that the image goes into the stage of dilation and then closing. Later on, it finds contours from the image that has gone through dilation and closing and the four corners of the business card. Figure



Figure 13 – Code for Image Processing.



Figure 14 - Resizing

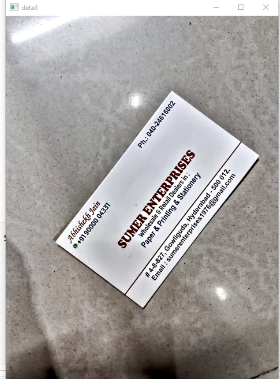


Figure 15 – Detail Enhancement



Figure 16 – Canny Edge Detection.

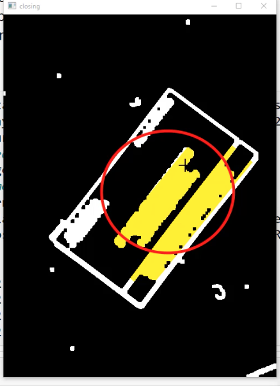


Figure 17 - Closing



Figure 18 - Dilation



Figure 19 – Closing



Figure 20 – Contour and four points.

After Image Processing techniques were used, the brightness and contrasting were applied. Figure 21 shows that a brightness of 50 and a contrast value of 40 give optimal results. High brightness value without any use of contrast makes the image less visible.



Figure 21 – Brightness and Contrast

Finally, the output is shown in Figure 22 in a web-app.

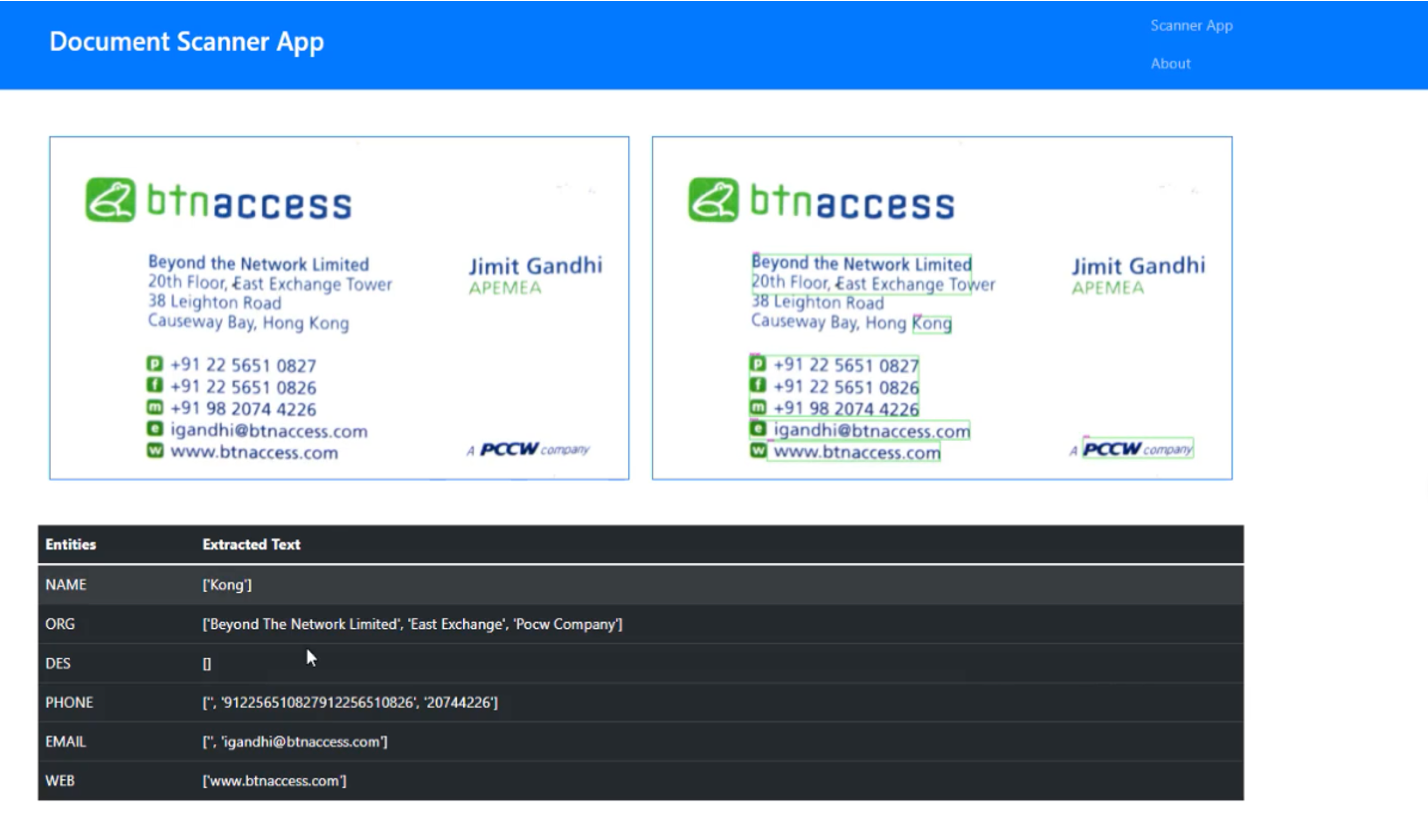


Figure 22 – Final Result of Output

**Conclusion**

The paper shows how CV and NLP coupled with Image Processing techniques are used to extract words from business cards. The article looked deeply into Background, Methods, and Results. The article started off with Abstract giving on overview of the paper and the introduction to explain what NER is to the reader.

**References**

*Mikheev, A., Moens, M., & Grover, C. (1999). Named Entity recognition without gazetteers.* *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics  -*. https://doi.org/10.3115/977035.977037

*Tesseract OCR on Arch Linux | Mike Polinowski*. (2021, October 31). https://mpolinowski.github.io/docs/IoT-and-Machine-Learning/ML/2021-10-31--tesseract\_ocr\_arch\_linux/2021-10-31/

*Zelic, F. (2022, October 17).* *Tesseract OCR in Python with Pytesseract & OpenCV*. Nanonets AI & Machine Learning Blog. https://nanonets.com/blog/ocr-with-tesseract/