Timothy Mitchell

Professor David O’Gwynn

CSC 441: Topics in Computer Science

25 November 2020

**liGROCERYst**

**The Application of Handwritten OCR using Machine Learning**

The heavy integration of technology has allowed us to do all things digitally, so much so that most people could live their lives comfortably without needing to leave their homes often, if at all. Things that were once ordinary, such as making a list and going to the grocery store, seem to harbor feelings of redundancy as these tasks have the convenience of being done online remotely from one’s home. However, some still see value in doing things in person, physically with or without some minor assistance from our binary speaking companions. That very assistance allows for those tasks to become more efficient, in turn making such menial things somewhat more enjoyable. I have tasked myself with being one of those individuals who can bring about such enjoyment through the software I program. liGROCERYst is a project of mine that involves a program that can take a handwritten grocery list, identify characters/words, and create a checklist that allows you to strikethrough all of the items that you have obtained. This undertaking would be implemented using OCR through machine learning. Though, through the practical application, I admit that I was posed with many difficulties that made this project extremely frustrating.

To get started, I should give a brief introduction to what OCR and machine learning are. As many already know, machine learning is a form of artificial intelligence that allows our devices to essentially learn and improve on their own without needing to be explicitly programmed to do so. Machine learning uses neural network models, which are essentially models with a web of connected nodes that are responsible for simple computations, and algorithms, which are a set of rules that are followed in calculations, to aid in computer performance. Commonly, the user gives the machine learning model a set of data, and the model constructs a mathematical model that the computer uses to make decisions. There are three types of machine learning supervised, unsupervised, and reinforcement. OCR, or optical character recognition, is the conversion of handwritten, typed, or printed text into editable or searchable data. This method is used for digitizing text so that it can be edited, searched, stored compactly, or displayed inline electronically. It allows for easy access when completing machine processes such as text mining, key data, text-to-speech, machine translation, etc. The first instances of these can be traced back to as early as 1914 for OCR when Emanuel Goldberg created a machine that could read and convert characters into standard telegraph code[1]; and the mid-1950s for machine learning. OCR got its foundation from an Australian engineer Gustav Tauschek, who got a patent on OCR or Reading Machine in Germany in 1929[2]. From then on, we have had substantial growth and improvement in OCR. Convenience is the paramount factor here, and it is what those who use OCR strive to create. It has been used to scan and file documents for companies with thousands of clients, to self-check-in hotel guests, to copy-paste documents, assist the blind (text to speech), etc. Older firms like ABBYY and IRIS started tackling OCR in the ‘80s, and now everyone from IBM to Adobe, along with smaller startups, are piling on for what will be a $13 billion market by 2025[3]. The trouble is, even after decades of research, most handwriting recognition software lacks sophistication and can only produce low accuracy rates. Especially when it comes to interpreting complex mashups of information [4].

The core idea of liGROCERYst is to bring the valuable commodity of convenience and an improvement to customer experience to the stay-at-home moms and dads, the nine-to-fivers, the broke college students, the housekeepers, and caretakers, etc. It is just something for the sake of convenience. It is for the modern-day shopper. It has importance to a person whose life may be a bit hectic. When everything requires depth and is time-consuming, it is nice to have something quick and mindless for a change. People are all about wanting things done quickly and efficiently, and this helps perpetuate that need for expediency. Even though the world has gone digital, there still plenty of instances of handwritten material that are around. People in rural areas may have a harder time contesting these technological advances as there is still a substantial digital divide throughout the United States. Handwriting is king in these specific areas, and that seems to be something that is not going to change relatively quickly. Luckily, mobile technology usage in rural areas has gone up, and that allows for projects, like the kind I am creating, to get some needed data on shopper habits, user tendencies, tech usage, etc., in less technologically advanced areas. Regardless, technology such as this could really benefit local grocery stores. If liGROCERYst partnered with chain or small grocery stores, it could provide them with key data that allows them to create special customer specific deals, recommend them new products similar to ones they have previously purchased, allow them setup automated grocery delivery services, etc. The is also a chance for me to take a deep dive into the complicated and messy world of optical character recognition. Text recognition, for the most part, is a two-step process that involves detecting text appearances in images and then choosing a set of solutions that come from three main approaches. These are classic computer vision techniques, specialized deep learning, or the standard deep learning approach. This could serve as a guide for those individuals just starting in optical character recognition. Grocery may receive extra benefits in the form of a more organized work environment. They can now, more accurately, determine the amount of and time in which an item needs to be restoked.

What does an implementable example of liGROCERYst look like? Given an image of a grocery list, the software should be able to identify, if the list is ordered as one item on each row, each item on that list separately and provide them each with its corresponding bounding box. Once all the bounding boxes are set, it should allow you, for the time being, to mark picked items at your leisure and to rearrange boxes in whatever order you choose. I also want the software to correspond with grocery stores to categorize products by proximity to one another to decrease shopping time, to get sales for certain items that are currently happening for that particular store, etc. For this example, I used two different tools Pytesseract and TensorFlow, which gave very different results. First, I will begin with TensorFlow, and TensorFlow is an open-source machine learning library that is maintained and supported by Google. It allows someone with a complex math background to create machine learning models and neural networks. It has two integral components graphs and sessions. The graph essentially functions as a unit for defined computations that have been written in code. Graphs are not for storage, and they do not do the computing. The sessions are what causes the parts of the graphs to be executed. When creating the python script for this application, there are four essential libraries needed OpenCV, a library designed to solve computer vision problems; NumPy, a library of arrays, matrices, and high-level math functions; Matplotlib, a library that allows for the visualization of mathematical data; and of course, TensorFlow. Next, locate a dataset of handwritten digits or characters. I recommend Kaggle A-Z since it covers sixty-two ASCII hexadecimal characters corresponding to digits 0-9, capital letters A-Z, and lower-case letters a-z. Kaggle is all packaged in an easy to use CSV file. This new data should then be split into two, one is the training data, and the other is the testing data. Once separated, the data needs to be normalized. With this, we are just gray scaling, changing the values from 0-255 to 0-1. Now it is time for us to build the model. The code will look something like this:

**model = tf.keras.models.Sequential()**

**model.add(tf.keras.layers.Flatten()**

**model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))**

….

As follows, we set the parameters for training the model, train the model, calculate validation loss and accuracy, save the model, make a prediction, and finally, check prediction accuracy [5]. The model made with TensorFlow had very few issues, and while the accuracy was not the best, it was a lot better at actually recognizing characters than Pytesseract. Pytesseract was the first tool I tried with this project. It is an optical character recognition tool for python that recognizes, reads, and prints recognized text in text embedded images. With this model, I used the libraries ArgParse, Cv2, OS, PIL, Pytesseract, and Time. First, I constructed the argument parse and then parsed the arguments. This means that when you run the script in the terminal, for example, python3 tesseract.py --image someimage.png, you need to add the “--image” argument before the file name. Once the image is loaded, the program converts it to a grayscale image. It could also have been given some preprocessing arguments like thresh or blur that aid in the removal of noise. Now it is time to write the grayscale image to the disk as a temporary file. That file is then loaded as a Pillow image, read through the OCR, and then deleted. Once all possible words are found, we are free to set up the bounding boxes around all identified words in the grayscale image. Finally, that image is given a new filename and saved as a new image. Many complications accompanied the Pytesseract implementation. While the accuracy with digital characters was halfway decent, it was stumped when it came to most handwritten images. The biggest challenge is the accuracy of the OCR. People's handwriting tends to differ vastly, and the smallest detail could throw the system off. Cursive is especially hard because it is difficult to interpret handwriting with no distinct separation of characters. In my case, however, this is not that big of a deal since it must recognize whole words instead of individual characters. Individual character recognition is still important. Without it, how will the system know where the words begin and end and whether the thing it is looking at is a word? Another issue could arise from a lack of examples for the system to learn from in the database. The training of a system requires an immense number of examples to account for unique cases. There needs to be a consideration for the lighting, folds and grooves in the paper, unusual or super stylistically written characters, image quality, the darkness of the ink or lead, the thickness of the strokes, the distance between letters, etc. There are so many determining factors that it seems near impossible to achieve consistent accuracy. Some images that had words on lined paper would not identify anything at all. Other times, there was an unusual phenomenon where a lighting issue would cause most of the paper, when gray scaling, to become black. What is even more interesting is that it would still create one giant bounding box around the entirety of the list. None of the actual individual words would be identified as words though.

liGROCERYst is meant to be a means of heightening the effectiveness of practical tasks that involve characters created through penmanship. Essentially, I am creating a piece of software that allows an individual to manage uploaded handwritten grocery lists. This should allow them to mark out selected items, categorize items, etc. I have been able to identify key issues with the program that have hindered my progression. The challenge from here on out is finding a solution to the pressing issues. I also want to work on a secondary voice software that works in conjunction with this one that listens for a specific phrase. Once that phrase is heard, the software will give the user an indicator, and the user will begin to list grocery items. The software will note all items, repeat a list back to the user, and place the order itself. Depending on the store, the user would have the option for delivery or in-store pickup. Hopefully, this process will create an individual who somewhat proficient at OCR and machine learning. I want to be able to bring that skill set to a future employer.

[1] Multiple Authors. 2020. Optical character recognition. (Last Update: November 2020). Retrieved November 15, 2020 from <https://en.wikipedia.org/wiki/Optical_character_recognition>

[2] Harry Thornburg. Unknown. Optical Character Recognition. (unknown). Retrieved November 15, 2020 from <https://history-computer.com/ModernComputer/Basis/OCR.html>

[3] Jess McCuan. 2020. Why handwriting recognition is still so tricky. (March 2020). Retrieved November 25, 2020 from <https://automationhero.ai/blog/handwriting-recognition-tools-translate-unstructured-data/>

[4] Jess McCuan. 2020. Why handwriting recognition is still so tricky. (March 2020). Retrieved November 25, 2020 from <https://automationhero.ai/blog/handwriting-recognition-tools-translate-unstructured-data/>

[5] NeuralNine. 2020. Python Machine Learning Tutorial #8 - Handwritten Digit Recognition with Tensorflow. Video. (12 February 2020). Retrieved November 05, 2020 from <https://www.youtube.com/watch?v=Zi4i7Q0zrBs&t=116>s