ReceipTrack - Receipt Image Processing

Jake Ashkenase, Nicolai Jacobsen, Jacob Kulik, David Pogrebitskiy

Northeastern University, Boston, MA, USA

Abstract

Tracking and understanding spending behavior is an aspect of personal finances that is often overlooked by the average consumer. However, this is essential in order to promote good spending habits. Our project, ReceipTrack, is an interactive dashboard that allows users to scan receipts, which will then return valuable insights on the spending habits of the user. The dashboard interface allows users to access information on past receipts, such as items purchased and store information. Additionally, the application will categorize each item in the receipt, which then gives the user insights into what grocery categories they are spending the most on. The dashboard is generated using Dash, which results in an elegant display with a simple user interface. The receipts uploaded to the dashboard are processed using Optical Character Recognition (OCR) through the PyTesseract package, which transforms the receipt image to a string. The string of information is then parsed and relevant information is extracted through a series of algorithms. This information is then further processed, for example to categorize the items in the receipts. The goal of developing ReceipTrack was to gain an understanding of image processing and text recognition in the field of data science, as well as improving our skills in language parsing and developing dashboard applications.

Introduction

ReceipTrack stemmed from the desire for a project that would not just be used following its completion but something that we could integrate as a useful tool in our everyday lives. Along with functionality, we also wanted to push ourselves to try a new niche of programming which ended up being image processing. The goal of our project was to create an interface where someone could upload pictures of their receipts and be given back valuable insights on their spending habits along with a medium to store information so it was no longer necessary to hold onto every receipt. Another goal of the project was to create our model so it could function with a number of different receipt formats including both physical and online types. We thought that if we could build out this idea we could create something that was not only useful to us, but something that a large market of people would be excited to integrate into their lives. We found this work to be significant because while the idea itself is quite simple, it offers incredible functionality to solve a real world issue. We felt that this project was useful enough that a heavily built out version of our project could be built into an app for consumers.

Methods

Object Design

The PyTesseract package is an Optical Character Recognition (OCR) tool that transforms text in images [1, 3]. With a goal to get all data directly from receipt pictures rather than through hardcoding, any uploaded or referenced file should be able to be transformed into a long string with certain configurations, such as black/white imaging and specifying the language to English [2]. Following the translation of the text in an image to a string, a receipt object is built, where the constructor helps search for certain important attributes in the text. Each receipt object is built out in accordance with the 3 parts of every receipt: the vendor, the items, and the totals. Firstly, the phone number and date taken from the top of every receipt is used due to logos and font types not being read in properly by tesseract and a difficulty to remain dynamic when working with an address. The subtotal and total is easily identified in the majority of receipt formats, because the two words are usually on separate lines with its respective value and can therefore be identified by checking if the words are a constituent element in the string followed by a float. Identifying items was more difficult due to various receipt formats displaying this information differently. The most accurate solution we found to this problem was using the receipt total to find the correct set of values (prices) in the receipt. This was done by iterating over subsets of combinations of all floats that could be found in the receipt string, then identifying the one that had a sum equal to the total. We could then find the item names for each respective value in the set. While this is a complex solution to the problem, we found it was highly accurate in identifying the items for any receipt format. Using this information, the object then makes sure to classify each item, in an attempt to produce visualizations based on purchase patterns. Finally, these attributes in the object are used directly by the dashboard script to transfer the information into a clear table format for the user's convenience.

Food Classification

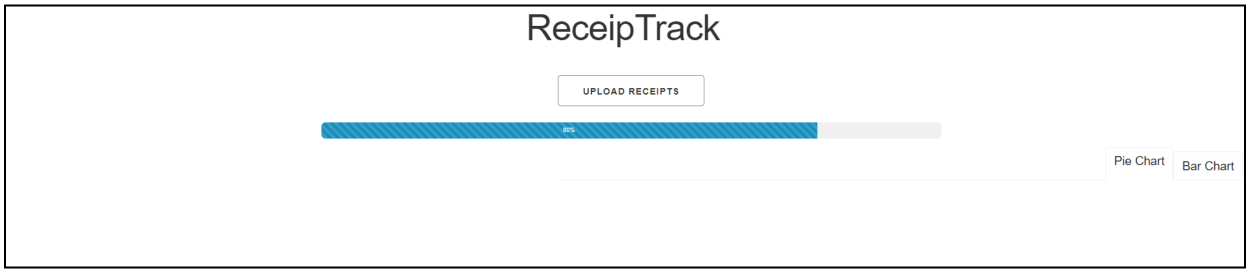
One of the main goals of the project was to be able to take the different items from the receipts and be able to identify them as different items, allowing us to then classify them in categories. We found this especially difficult because receipts come in so many different forms and there is so much excess text that we needed to ignore. Many items were abbreviated, and so while the human eye can make out what food is being bought, it is extremely hard for the computer to do so. We had two main ways to try and fight this issue and get the classification for some of these hard to read items. The first of these methods was using a database of 900+ foods to try and match the items on our receipt to ones in the dataset. However, This wasn't enough to deal with abbreviations. After looking at trends in different receipts we found that vowels were usually the letters to be taken out when an item is abbreviated so for every item we would create a list of potential abbreviations taking out different combinations of vowels from the item. This allowed us to match up a lot of the abbreviated food items with entries in our dataset, which in turn allowed for the classification of those items. The other solution we employed to deal with items that were abbreviated past the point of our system being able to understand them was trying to identify the brand name. We compiled 7000+ unique brands that could potentially show up in the receipt and then be used to classify an item. One issue with this method is that for it to work each brand must only have products in one category, otherwise we could potentially misidentify an item because we know nothing about the product and only have the brand name to go off of. This reduced our list of brands from 7000+ to 3000+ but still left us with valuable information to improve our classification system. Using these two techniques we were able to put together a classification method that is effective at picking up the category of different items and allows us to create valuable insights for consumers on their spending habits between different categories.

Dashboarding

Plotly Dash was used to effectively and elegantly display our results outputted from the above methods. The dashboard we came up with consists of three main sections: data upload, individual receipt information, and summary visualizations. For data upload, a simple Dash Core Component was used. This component encodes each file as base64 encoded string so we were required to decode it, output the image to a temporary location, then feed it back into our processing methods. Because our methods include a fair amount of iteration, sometimes the upload process can take a little while. We decided to implement a progress bar to indicate to the user that we dashboard is still loading up. This was done using the tqdm library which helps track progress through a for loop. Because it natively outputs the progress to the console, we had to write the console output to a file, then read that file on an interval to increment the progress bar on the screen. To display individual receipt information, we chose to display html-style tables. These tables were then combined into a dbc (Dash Bootstrap Components) accordion component which allows users to view all the information without taking up too much screen space. Finally, we chose to display a pie chart and a bar chart. The pie chart shows the breakdown of spending by food category while the bar chart shows the amount spent on each purchase over some period of time. These were then displayed to the dashboard on different tabs using a combination of Core Components and Bootstrap Components.

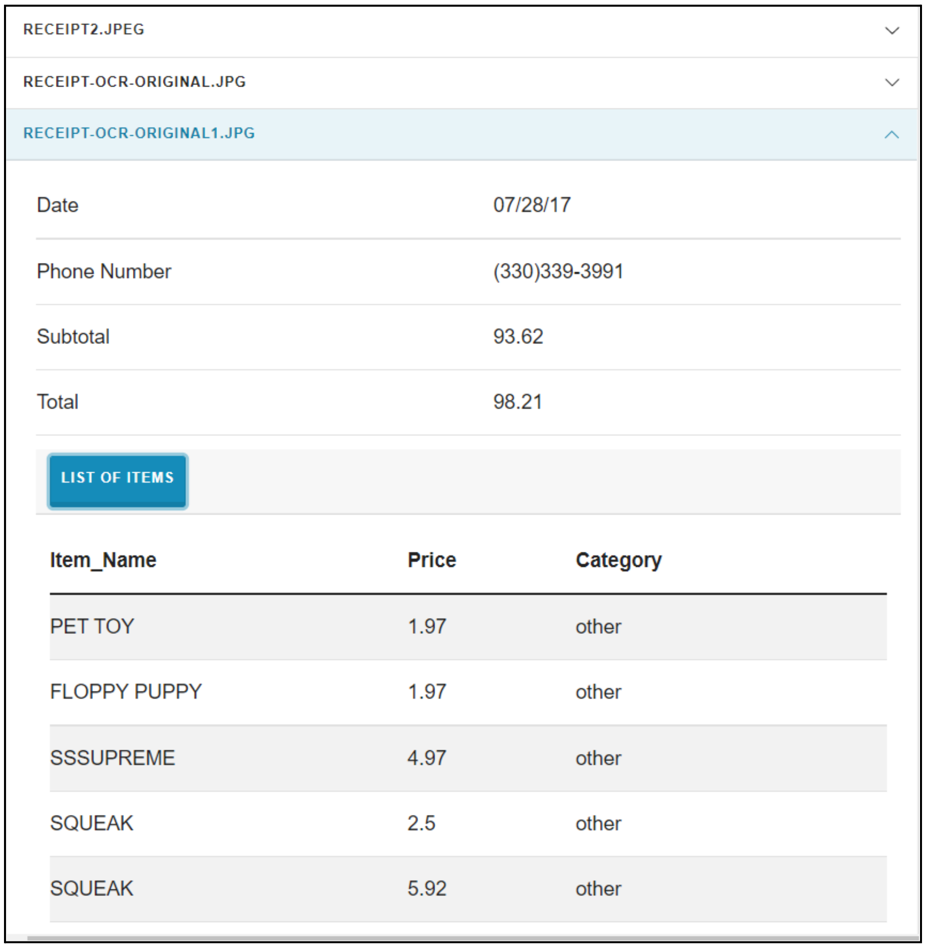
Analysis

The main outcome of our project was the dashboard we created to encompass all of our insights on the data. the first part of this dashboard can be seen in our first figure where we created a loading bar for the dashboard in order to show progress on receipts loading onto the dashboard, something that can take some time to work. Instead of leaving users clueless whether or not the dashboard is working, we used this to keep the user involved at all points of the process. The second part of our dashboard can be seen in the second figure and serves the purpose of displaying all of the information we were able to process in one place. We designed it so that multiple receipts could be seen side by side and different ones can be opened and closed in order to allow for easy comparison between them. This part of the dashboard is a result of our program's ability to sift through different types of results and still effectively pull out the same information. Our third figure really shows the power of our item categorization as it is able to show how spending is split into the different categories we have created and also continues to add the data of receipts as they are added. Our bar chart also allows for users to see their spending over time and analyze how their habits are changing in that regard.



Default dashboard screen with upload button and progress bar

Above is part of the default screen, which prompts the user to upload receipts. The progress bar then reflects the percentage of receipts that are done being processed, as it may take a minute per uploaded picture.



RECEIPT2.JPEG

RECEIPT-OCR-ORIGINAL.JPG

RECEIPT-OCR-ORIGINAL1.JPG

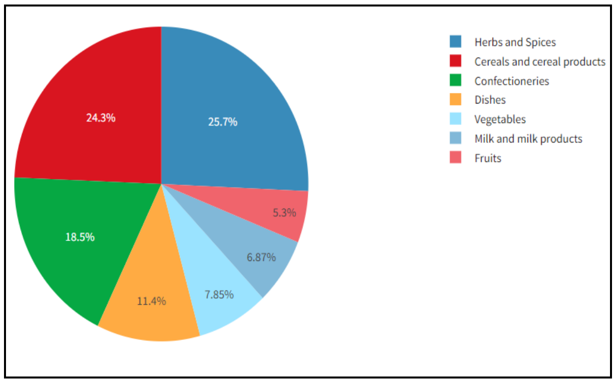
Date	07/28/17
Phone Number	(330)339-3991
Subtotal	93.62
Total	98.21

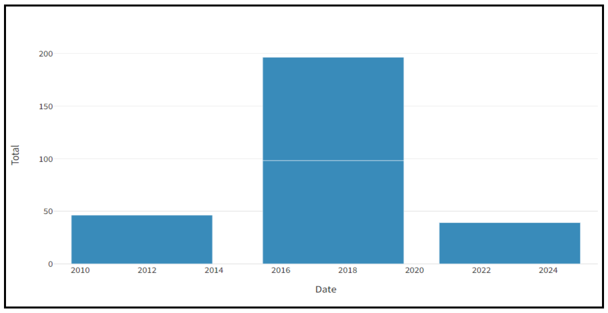
LIST OF ITEMS

Item_Name	Price	Category
PET TOY	1.97	other
FLOPPY PUPPY	1.97	other
SSSUPREME	4.97	other
SQUEAK	2.5	other
SQUEAK	5.92	other

List of items, prices, and categories under each receipt

Once done processing, the items and associated file names are displayed in a list format on the left of the screen, with a collapsible table providing more information on the date, phone number, and totals. This also leads to another collapsable table with the item name, price, and classified category for each item.





Receipt visualizations, item classification and time barplot

The dashboard and visualizations should give the end user an understanding of their spending habits over time by uploading photos of their receipts. The categorization of items should also allow users to identify categories where they might be overspending or underspending. For example, if the user is living with an unhealthy diet, they should be able to identify this easily from the pie chart and hopefully make the necessary changes to improve their diet. The historical bar plot should also help the user identify changes in their lifestyle which may have caused spending to increase or decrease, such as a change in diet. For example, a user might see an increase in their spending habits which doesn't align with their current income, helping them determine that they need to start budgeting and reducing their spending.

Conclusions

ReceipTrack is a dashboard tool that allows for users to upload receipts and see insights on their shopping habits. Each receipt is labeled by different common features such as location, phone number, and totals. The dashboard is able to track multiple receipts to highlight trends over time, while also classifying items to track the spending of different categories. Throughout the code, we've made sure to be as dynamic as possible, testing values where needed to build out receipt information without running into errors. For example, the list of items and prices associated with a receipt and checked after basic string processing to add up to the total value. This helps provide an additional layer of security and confidence in the accuracy of the displayed values.

There is room for improvement when it comes to processing receipts and displaying them, however. Computationally slow iterations, specifically when working with the pandas dataframes, were utilized in certain parts of the code that hurt run time. With a progress tracker for receipt processing, we tried to make this as seamless for the user. In the future, a more functional programming style can help with increasing speed. Further, building on top of current listification and classification capabilities will only help with visualizations. Potential future use cases of the receipt processing application would be to create a mobile application that would allow users to simply take an image of a receipt on the go and track spending like our dashboard. This could also be a potentially useful application for businesses, where employees have company-issued credit cards that should only be used for certain situations, such as work trips. Our application could help companies track the spending of employees in those situations, instead of manually checking receipts.

Author Contributions

ReceipTrack is made up of multiple functional parts. With class functionality built on top of the PyTesseract package's optical character recognition, item characterization and Plotly dashboarding come together to give a great user experience. Contributions were split across these different aspects. David worked heavily on the dashboarding, turning the objects into different charts and incorporating the visualizations. Nicolai assisted with item listification on the dashboard as well. Jacob and Nicolai's main responsibilities were developing the receipt class. Jake's work on item classification was also worked into the receipt class, helping create a more personalized analysis on a user's spending. The report was contributed to by all members. Overall, work was split up equally based on each member's interests and skills.

References

1. Babu, Sudharshan Chandra. “Digitize Receipts with Receipt OCR: Automated Receipt OCR.” *Nanonets AI & Machine Learning Blog*, Nanonets AI & Machine Learning Blog, 26 Sept. 2022, https://nanonets.com/blog/receipt-ocr/.

2. Yue, Alex. “Automated Receipt Image Identification, Cropping, and Parsing.” *Princeton Computer Science*, 2018, https://www.cs.princeton.edu/courses/archive/spring18/cos598B/public/projects/System/COS598B_spr2018_ReceiptParsing.pdf.

3. Zelic, Filip. “Tesseract OCR in Python with Pytesseract &amp; OpenCV.” Nanonets AI &amp; Machine Learning Blog, Nanonets AI &amp; Machine Learning Blog, 17 Oct. 2022, https://nanonets.com/blog/ocr-with-tesseract/amp/.