Technical Report

Receipt Scanner App

By Tara Adusumilli and Niki Surapaneni

Table of Contents

Our Receipt Scanner App

By Tara Adusumilli and Niki Surapaneni

INTRODUCTION:

The receipt scanner app is an IOS app that allows the user to monitor the trend of their dietary choices by analyzing their weekly grocery haul by sorting the groceries into six categories: grains, fruits, vegetables, dairy, protein, and junk food. When the receipt is scanned (using the camera on the device), the app uses text recognition software to determine the items that are listed. It then maps them to a food category and determines the percentage of each food category and uses a charting library to visually display distribution of the food in various categories. Our hope is that, this app will help people take into consideration their dietary choices and try to initiate change to achieve a healthier lifestyle

TECHNICAL SUMMARY:

Our App had 3 main parts: scanning the receipt and recognizing the text, dictionary to categorize the receipt output and charting to help visualize the output.

We developed the app using Swift 4 and XCode 9 and developed on our mac book with ios 10. Xcode has an IPhone 8 simulator that we could use to test the app. We used a text recognition software called Tesseract OCR. We used Cocoa pods to install Tesseract and Charts libraries. We tried to find a library for the dictionary but could not find anything that was suitable.

We followed the "Getting Started" guide from Apple to create a new project in XCode. Out User Interface is a basic "Single View Application". We used a "story board" to design our UI. It has a textViewUI with a scrollbar to output the receipt as text. Below the textView, we have a chartView. The chartView is a class from the Charts module. Our App has one ViewController. This ViewController has *viewDidLoad()* method that is called when the view controller's content view (the top of its view hierarchy) is created and loaded from a storyboard (Figure 1 and 2 in Appendix). Here we wrote our code to use Tesseract and get the output and show it in the testView.

```
let tesseract = G8Tesseract(language: "eng")
…
textView.text = tesseract.recognizedText
```

Then we defined a dictionary and used it to find the percentage of various food groups. (Fig 2 , 3 and 4 in Appendix)

We then used a the percentages create PieChartData and PieChartDataSet to update the PieChart with the data (Fig. 4 and 5 in Appendix). And, of course, we used some very attractive colors to display the pie chart! We used a IPhone 8 simulator (our favorite phone) and ran the app.

RESULTS:

Our text recognition app can found on Github
https://github.com/ilovecrepes/receipt_scan_app

- There were a few glitches with Text Recognition. The software found it hard to read an image when the receipt was wrinkled. The borders with the receipt containing the grocer's name and other information like store location date/time etc. also caused the text recognition software to error. However, we tested our app with the image of a Trader Joe's receipt we found on the web and it worked well.
- Also, we did not find any dictionary to sort the food items into categories. So we created our own simple dictionary. For the app to be really useful, we still need to expand or find a proper dictionary.
- Also, every grocery chain probably uses different types of receipts and uses different short names for an item (Example: YOG vs Yogurt). So we would need to expand the app's dictionary to be able to process receipts from other grocery stores.

REFERENCES:

Getting Started Guide from Apple for IOS apps
https://developer.apple.com/library/content/referencelibrary/GettingStarted/DevelopiOSAppsSwift/

Lecture 1: Introduction to iOS 11, Xcode 9 and Swift 4
https://www.youtube.com/watch?v=71pyOB4TPRE

Easy Text Recognition with Tesseract OCR
http://www.brianadvent.com/easy-text-recognition-tesseract-ocr/

Tesseract Optical Character Recognition
https://opensource.google.com/projects/tesseract

iOS Swift Tutorial: Create Beautiful Charts https://www.youtube.com/watch?v=GNf-SsDBQ20&list=PLY1P2_piiWEY3W_aGo7r2OZmsk6Ww757Y

Beautiful charts for iOS/tvOS/OSX! The Apple side of the crossplatform MPAndroidChart. https://github.com/danielgindi/Charts

APPENDIX:

Fig 1: View Controller.swift

```
//
//  ViewController.swift
//  TextRecognition
//
//  Created by Tara on 3/17/18.
//  Copyright © 2018 Tara. All rights reserved.
//
import UIKit
import TesseractOCR
import Charts

class ViewController: UIViewController, G8TesseractDelegate, ChartViewDelegate {
    @IBOutlet weak var textView: UITextView!
    @IBOutlet weak var pieChart: PieChartView!

    var grainsDataEntry = PieChartDataEntry(value: 0)
    var fruitDataEntry = PieChartDataEntry(value: 0)
    var dairyDataEntry = PieChartDataEntry(value: 0)
    var vegetablesDataEntry = PieChartDataEntry(value: 0)
    var junkDataEntry = PieChartDataEntry(value: 0)
    var proteinDataEntry = PieChartDataEntry(value: 0)

    var allFoods = [PieChartDataEntry]()

    override func viewDidLoad() {
        super.viewDidLoad()

        self.textView.layer.borderWidth = 1
        self.textView.layer.borderColor = UIColor.black.cgColor

        pieChart.chartDescription?.text = "food categories"

        grainsDataEntry.label = "grain"
        fruitDataEntry.label = "fruit"
        dairyDataEntry.label = "dairy"
        vegetablesDataEntry.label = "vegetables"
        junkDataEntry.label = "junk"
        proteinDataEntry.label = "protein"
```

Fig 2: Tesseract OCR for Text Recognition

```
if let tesseract = G8Tesseract(language: "eng"){
    tesseract.delegate = self
    tesseract.image = UIImage(named: "traderjoes")?.g8_blackAndWhite()
    tesseract.recognize()

    textView.text = tesseract.recognizedText

    let myStringArr = textView.text.components(separatedBy: "\n");
    let userinfo = ["grains": ["barley" , "bread" , "cereal" , "cornmeal" , "grits" , "oat" , "oats" , "pasta" , "refined
        grains" , "rice" , "wheat", "whole grain"], "fruits": ["lemon", "mango", "orange", "papaya", "peach", "pear",
        "pineapple", "pomogranate", "pumpkin", "rasberries", "strawberry", "tomatoe", "watermelon"], "dairy":["butter",
        "cheese" , "milk" , "yogurt"], "vegetables": ["artichoke", "arugula", "asparagus", "beets","bok choy", "broccoli",
        "brussel sprouts", "cabbage", "carrots", "cauliflower", "celery", "collard", "corn", "cucumber", "edamame",
        "eggplant", "fennel", "ginger root","green bean", "kale", "leeks", "lettuce", "mushroom", "mustard greens",
        "okra","onions", "parsnip", "peas", "pepper", "potato", "raddish","shallots", "spinach", "turnip","zucchini"],
        "protein":["almond","anchovies","bacon", "beans","steak","chicken","fish", "nuts","meat", "crab", "egg","hummus",
        "lamb", "lobster", "pork","prawns", "seeds", "salmon", "sardines", "sausages", "tofu", "turkey"], "junks":["french
        fries","fried chicken","fried", "soda", "pizza", "hamburger","hot dog","taco","chicken nuggets", "nachos",
        "custard", "pudding","cake","pie", "flan", "sorbet", "chocolate", "brownie", "sundae", "tirmisue", "trifle",
        "meringue", "baked alaska", "waffle", "pancake", "baklava", "panna cotta", "gelato", "turnover","gulab jamun",
        "tooti frotti", "jalebi","pastry", "tart","ice cream"]]
    var graincount = 0
    var totalcount = 0
    var fruitcount = 0
    var vegetablescount = 0
    var dairycount = 0
    var junkcount = 0
    var proteincount = 0
```

Fig 3: Determining percentage of food categories

```
for line in myStringArr
{
    totalcount += 1
    print(line)


    for (key, values) in userinfo
    {
        for item in values
        {


            if line.lowercased().range(of:item.lowercased()) != nil {

                print("found")
                print("The key: \(key)")
                switch key
                {
                case "grains":
                    graincount += 1
                case "fruits":
                    fruitcount += 1
                case "veggies":
                    vegetablescount += 1
                case "dairy":
                    dairycount +=  1
                case "protein":
                    proteincount += 1
                case "junk":
                    junkcount += 1
                    break
                default:
                    break
                }
                break
            }
        }
    }

}
```

Fig 4: Determining percentage of food categories

```
    let grainspercent = (Double)(graincount)/(Double)(totalcount) * 100
    print("Grains Percent: \(grainspercent)")
    grainsDataEntry.value = grainspercent

    let fruitspercent = (Double)(fruitcount)/(Double)(totalcount) * 100
    print("Fruits Percent: \(fruitspercent)")
    fruitDataEntry.value = fruitspercent

    let vegetablespercent = (Double)(vegetablescount)/(Double)(totalcount)*100
    print("Vegetables Percent: \(vegetablespercent)")
    vegetablesDataEntry.value = vegetablespercent

    let proteinspercent = (Double)(proteincount)/(Double)(totalcount) * 100
    print(proteinspercent)
    proteinDataEntry.value = proteinspercent

    let junkspercent = (Double)(junkcount)/(Double)(totalcount) * 100
    print("Junks Percent: \(junkspercent)")
    junkDataEntry.value = junkspercent

    let dairypercent = (Double)(dairycount)/(Double)(totalcount) * 100
    print("Dairy Percent:\(dairypercent)")
    dairyDataEntry.value = dairypercent
}
```

Fig 5: Charting

```
        allFoods = [grainsDataEntry, fruitDataEntry, dairyDataEntry, vegetablesDataEntry, junkDataEntry, proteinDataEn

        updatePieChartData()
        // Do any additional setup after loading the view, typically from a nib.
    }
    func progressImageRecognition(for tesseract: G8Tesseract!) {
        print("Recognition Progress \(tesseract.progress) %")
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    func updatePieChartData(){
        print("===allFoods")
        print(allFoods)

        let chartDataSet = PieChartDataSet(values: allFoods, label: "foods")
        let chartData = PieChartData(dataSet: chartDataSet)
        let colors = [UIColor.blue, UIColor.red, UIColor.green, UIColor.black, UIColor.purple, UIColor.yellow]
        chartDataSet.colors = colors
        pieChart.data = chartData
    }
}
```

Fig 6: Simulator showing App output for a sample Trader Joe's receipt

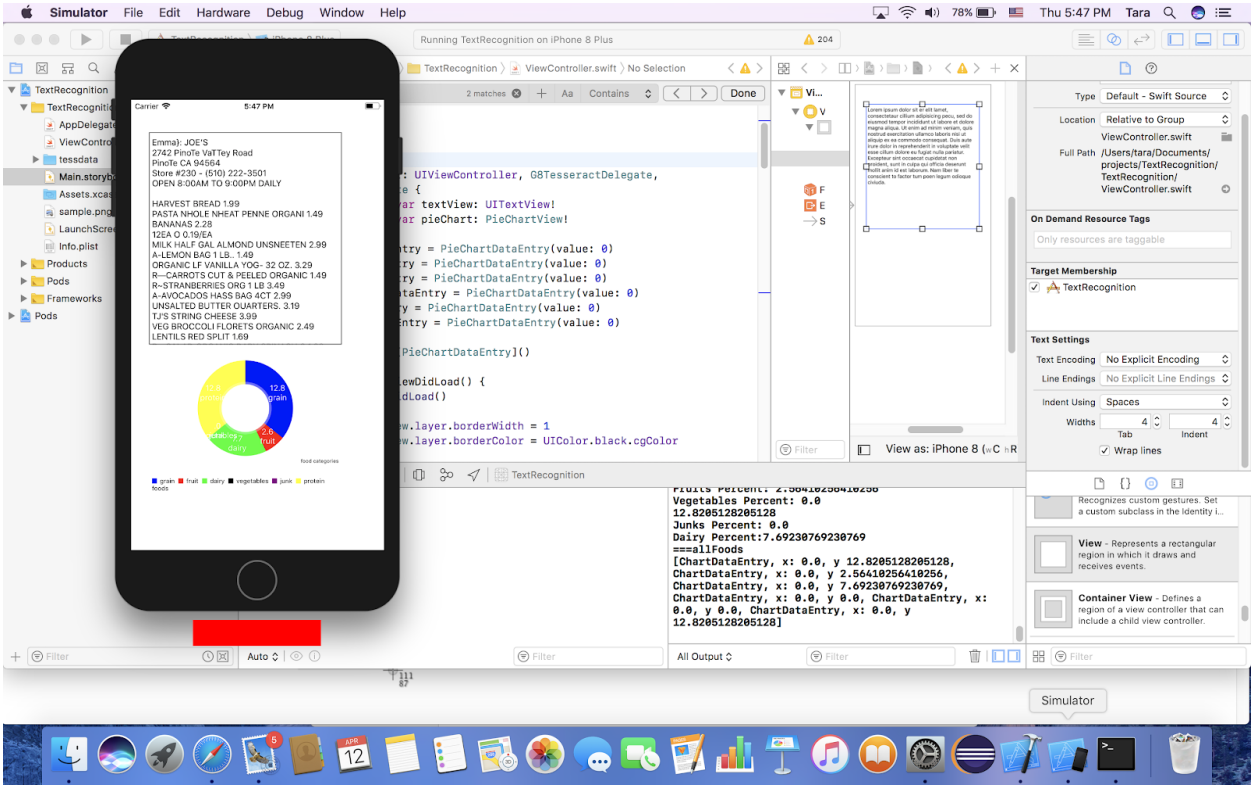

Fig 7: Simulator showing App output for a sample Trader Joe's receipt