Project Requirements Document Version 1

Project Title/Name: Intelligent Offer Categorizer
Team Name: Odyssey

Team Member Names/Emails:

- Xiao Sun (xiaosun@ucsb.edu)
- Danny Millstein (millsteindanny@gmail.com)
- Winfred Huang (whuang@ucsb.edu)
- Haochen Shi (shi@ucsb.edu)
- Delin Sun (delinsun@ucsb.edu)

Team Lead: Xiao Sun
Team Scribe: Danny Millstein

Background:

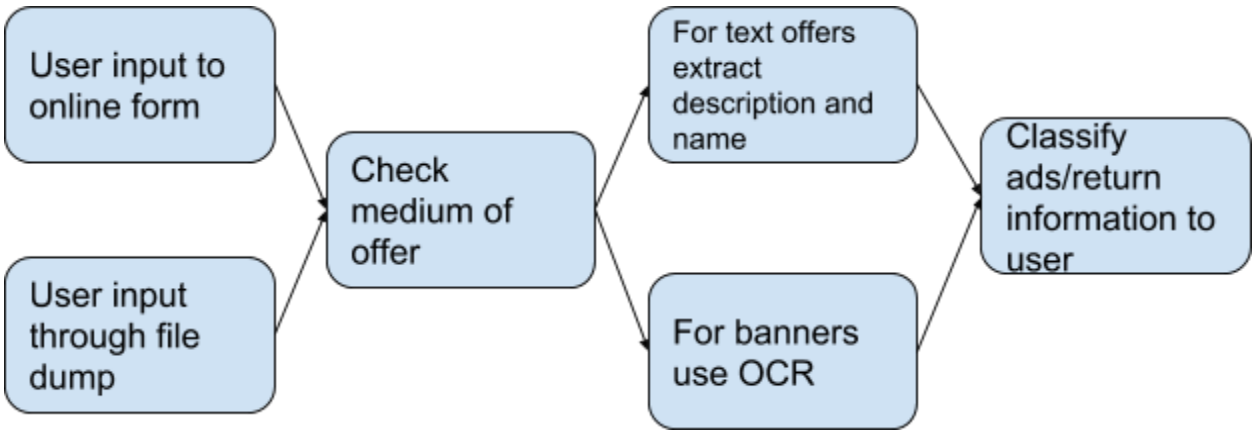
CJ affiliate as an affiliate marketing firm facilitates advertisers in finding appropriate publishers to publish their offers. In the context of affiliate marketing, an advertiser is a company with some sort of product they are selling, and a publisher is a person with access to a medium (website, instagram page, youtube channel) on which they can display ads. CJ has uncategorized offer data and no programmatic way to categorize it. CJ wants the ability to compare performance data between different categories of offers (for example: BOGO, % off, $ off, $ off when you spend 100$,....). Currently CJ Advertisers have to manually specify several properties about each offer in order to do this. CJ advertisers are concerned about offer performance, and want to be sure that the offers they are creating are actually benefiting them, or are the most optimal for them. Since CJ publishers are paid directly in proportion to the performance of a posted offer, they also want to know that the offers they are posting perform well when posted to their medium, and when directed at their audience. Having all offers categorized will help give all parties insight into what type of offers work for their specific situation. Having this paired with the recorded performance data for these offers, CJ can demonstrate to their clients, both publishers and advertisers, the value of CJ's work.

Team Goals:

The Odyssey team have multiple related goals to accomplish. The first goal is to automatically categorize an offer based on offer text data. After accomplishing this we want to use optical character recognition to parse out offer based text data from banner ads, and categorize these ads based on this data using our text offer categorizer. Next we want to parse out offer metadata of each offer, which has actual dollar amount and actual percentage, just to name a few. With this metadata we will be able to examine

the relative performance of similar offers. We want to make a web based application that can be used to categorize CJ's offer data. This web application will have an offer application form that can be filled out by a CJ advertiser. In real time our algorithm will categorize this offer, and show performance data on similar ads to help the user make the best choice for their product offer.

High level diagram



User stories/use cases

Scenario 1: Processing text data from files

As a CJ data analyst, I can upload files with offer text data into the system so that the system can automatically categorize offers, which can save a lot of time.

Given that we have a file with offer text data, we get a file containing offer data plus additional category fields for each offer.

If the program successfully processed the file, there will be a pop-up window reminding the user that the data have been processed. The result will be stored into the database/system as a JSON file data.

Then the system will ask the user two options, "Do you want to stay at this page?" or "Go back to the main menu".

Test: Upload several data files into the system to see if the result matches user's requirement.

Scenario 2: Error Processing text data from files

As a CJ data analyst, I can upload files with offer text data into the system.

Given I have uploaded a file that is not containing the necessary information or is of the wrong format.

Then application will notify me that my file is of the wrong format.

Test: If any non JSON/BJSON file is entered into the application, a pop up window will appear notifying the user.

Scenario 3: Creating single new offer from a single file

As an individual advertiser user, I can create a new offer into CJ's system.

Given that the advertiser enters data into the system and they are creating a new offer in our system, the system will tag the offer based on input.

When they are trying to enter description and name, the system automatically tags the offer with the correct category.

Test: User clicks "create offer" button, the system gets into the "creating offer" page. The user can input offer information here.

Scenario 4: Creating several new offers from a single file

As an advertiser or CJ employee, I can input a single file into the application and receive that file with a category of offer applied to it in the form of an additional json variable.

Given the file inputted was in the desired BJSON format and contains enough information to generate a category.

If the file contains several offers in Json format, the system read these offers.

Test: Input a file with a known categorization, and compare the systems output to what was expected.

Scenario 5: Creating several new offers from multiple files

As an advertiser or CJ employee, I can input multiple files into the application and receive similar files with a category of offer applied to each in the form of an additional json variable.

Given the files inputted were in the desired BJSON format and contained enough information to generate categories.

Test: Input multiple files with known categorizations, and compare the systems output to what was expected.

Scenario 6: Uploading images

As a advertiser can upload image offers to the system and system will automatically categorize them.

Given that the advertiser have image offers, they want to categorize these offers by our system. When they are uploading the image offers, our system will extract texts from images. Then our system will categorize offers by text like scenario 1.

Test: use image offers as test input, manually extract text from the offers, compare to the output of the system-extracted texts.

Test: use image offers as test input, upload several images into the system, manually extract text from the offers, compare the output text with the text on the banners.

Scenario 7: Mixed medium input

As a CJ employee I can upload a file containing text offers and banner offers to the web application, so that I do not need to specify the type of offer, and I am still able to get my offers categorized correctly.

Test: Populate a json file with offer objects with known categories of medium 1. Input the file into the webapp. The app should output a new JSON of the input that includes categories.

Test: Populate a json file with offer objects with known categories of medium 3. Input the file into the webapp. The app should output a new JSON of the input that includes categories.

Test: Populate a json file with offer objects with known categories of medium 3 and 1. Input the file into the webapp. The app should output a new JSON of the input that includes categories.

Scenario 8: Automatically Categorize Offers after Uploading and Processing Data

As an advertiser I can enter the name and description of an offer, and in real time see the application label my offer with one to multiple categories.

Given the information inputted is sufficient enough to generate a category for this offer.

Test: Input a description and name that we know the corresponding appropriate category for, and in real time compare the output of the application with the expected output.

Scenario 9: Show overall performance of category

In this scenario, as a consumer of this product, I am able to pick out a category so I can see the overall performance of that specific category like a BOGO offer as an example. Assuming that the consumer has data stored in the database full of sorted and categorized offers, the consumer should be able to view some, if not all, of the offers. When the consumer chooses a specific offer, whether its BOGO or some percent off, the expected output should be a summarized performance of that offer. That summary output can have elements like monthly average earnings, which month had the most profit, or which month had the least profit just to name a few. If the output can not produced like there is not enough data to produce yearly earnings, then the program will

produce a pop-up error dialog with specific notes to alert the consumer about how the earnings were calculated. Otherwise for other errors like the computation took too long, then the pop-up dialog will show why and return back to the “home” page.

Scenario 10: Create Online Offers

In this scenario, as the advertiser or the analyst of CJ, I am able to access to CJ’s website. Then I can access the “Create Offer” page. The form with some blanks will show up. I can type the “name”, “description”, “medium”, and “advertiserID”. After I click the “Create” button, the website will gather the information and create the offer automatically for me. The categorizer will also categorize the offer and show the performance of similar past offers, as it works in scenario 9.

Github Commit link:

https://github.com/cjdev/capstone2018/commits/master

Github Issues link:

https://github.com/cjdev/capstone2018/issues

Technologies Employed:

- BJSON
- Optical Character Recognition
- Python Plotly
- Javascript