HTML Parser Report

Contents

Working on meat_0 JSON File

We start with the meat_0 file which contain 2000 records from which we have to extract maximum data from the rawHtml fiel.

Starting with Parsing of the JSON file

Before we get started with the parsing, we have to load in our data in our local environment and process it.

Initialising and Processing data

We install the library BeautifulSoup with which we would parse the rawHtml present and extract relevant information. After importing the required libraries and loading in our data, we first make a Pandas DF of the JSON data to properly access and HTML field inside.

Parsing on a sample HTML

We take a single html record from the above DataFrame and create a soup object. Now, we can use the soup API to extract and find information according to patterns of tags present.

Formatting the HTML record

First, we pretty print our sample html record to observe the structure of the document and relevant patterns present for relevant information. This is done through an online formatter and stored offline, for future reference.

Making a list of relevant information and tags present

Through our observation above, we make note of all the text we can extract and also the tags present alongside them. This would come useful for our reference later when we are extracting that data with Soup.

Building the Parser and Testing it

We see that there's 11 potential fields that we can extract our data in. Some of them are grouped out of necessity due to the functionality of the Soup object. For the order details, we find the text by searching for its relevant class and html tag, followed by pre-processing of the text to remove unnecessary bits. We follow the same process for the rest of the fields. For data which can't be separated with the Soup object, they are taken in a List of lists format in the Parsing stage which we can split into different columns later in the cleaning stage. All functions for each data are printed after addition of more functions, to test if all above code still work.

Building the Parsed Dataset

We leverage all the extract statements from above and create a function to append a tuple with each field when a rawHtml is passed to it. The list is then converted to a Pandas DF for easy manipulation.

Handling N/A Exceptions

Once we have our parsed dataset, we check to see if there are entries with missing values through the order_num (ID) column. There's 59 such records and so we remove them from the set.

Report after Parsing meat data

We have now completed the Parsing of meat json. Out of a total of 2000 mails, 1941 of them were parsed successfully. These 59 mails were from the Customer Care to individuals containing instructions to just complete their payment and had no real information. Thus, **97%** of the data was parsed successfully.

Cleaning the Parsed meat data

We now have to clean out unwanted symbols, irrelevant words and make anymore transformations needed to the parsed data.

Cleaning every column

Columns have symbols or words with them which don't allow us to do further analysis on the same values. Hence, we are removing them through Regex here. For instance, the order_num (ID) column should retain only the ID string and nothing else. The same is done for the rest of the columns such as datetime transformation, extraction, currency conversion and string concatenation,

Transformation and Feature Splitting

The second transformation we have to do is feature splitting of the columns which have grouped data after Soup extraction, such as order_list column. We do it directly and also make a new DF for the split features for the relevant columns. All the new columns are then appended to our cleaned DF with properly rename columns.

Report after Cleaning meat data

We were able to retrieve 1941 rows with 18 columns of data, and hence, we retained 100% of our data from the Parsed DF. Out of them, half the columns don't have any null values while the rest have about 60 (not relatively significant). The paid_online column has about 25% null values, assuming that payment for those orders did not happen online and were COD.

Working on flip_oc_0 JSON File

We start with the flip_oc_0 file now which contain 2000 records in which we have to extract maximum data from the rawHtml field given.

Starting with Parsing and Processing of our file

Before we get started with the parsing, we have to load in our data in our local environment and process it. We first load in our data to make a Pandas DF of the JSON data to properly access and HTML field inside.

Doing EDA

We explore this DF to get an idea of the data inside and inspect any differences in the structure based on the sender and receivers. After noting the format of the rawHtml, we can now start with building the Parser to extract data.

Parsing on one sample HTML

We sample one record from our DF for testing purposes of our extraction module. Since this structure doesn't have a lot of complexity and data, we can directly use the get_text function of Soup to get a corpus of all the text in the file at once.

Building the Parser for flip DF with Regex

Now we build the Parser. The extracted corpus is filtered and stripped of any unnecessary characters so we're left with a clean paragraph of words in the end. Regex is then utilized to write match expressions for each column. If-else statements are included to return None when no match is found to make the function fail-proof. 15 patterns are written for a total of 13 columns in our DF.

Report after parsing flip data

Out of a total of 2000 mails, we were able to extract values in all our columns. However, we do see almost all columns missing 22 records which we will investigate in the Cleaning process. Thus, 100% of the data was parsed with Regex with the information in the corpus. The parsed data is exported to a CSV.

Cleaning the parsed flip data

Not much cleaning needs to be done here due to the simple nature of the data and our efficient expressions. The columns are renamed first with representative names. After checking the CSV manually, we can see that there are certain errors in detecting the amount paid for Gift Vouchers and it doesn't hold correct data. Also, most columns for gift vouchers are blank since their email structure is different and don't contain much information so we can remove these rows. There are 22 such rows so it shouldn't affect our future insights by a huge margin. Our final DF is now ready and it's exported to a CSV.

Report after cleaning flip data

We were able to retain 1978 rows out of 2000 rows, with 12 columns of data, coming to about 99% retrieval of the entire dataset. 75% of the columns have only non-null values. The column with maximum null values is item_bought, from which we retrieved 86% of the values completely. This can be due to more variations in the formatting of the text present in the rawHtml.

End of Report.