HTML Table data extraction

Ideation challenge

Author: Anatolijs Gorbunovs
Handle: gorbunov

Introduction

This challenge requires to convert a HTML that contains some data in a form of tables into a JSON that could be easily parsable by a computer. The main problem that we are faced is that modern HTML is a mix of representation and a data. The question is to extract the data without loosing any useful information.

Data mining

After investigating the HTML source of the referenced example documents, it is clear that they all use Workiva Wdesk1 software for generating these tables. However, this seems to be a proprietary software with semi-open documentation. Thus, a decision was made to make a more or less universal table processor that can handle the provided demo documents and documents generated by the same tool, but not more. This decision was necessary in order to not over-complicate the solution.

In order to make more or less valid assumptions about the data in HTML, it was decided to prototype (code) the actual algorithms in Java.

Brainstorming

The key to success in development of this application is, in my opinion, to exclude loosing any useful meta-information about the extracted data. If we fail to do so, the numbers and/or text in the returned JSON might be useless for the end-user.

It is more or less clear how to extract the numbers and textual data out of the table, and it will be covered later in the document.

So what is exactly this meta-information and why it is so important? The meta-information includes but is not limited to:

- the units of information for the number: currency, amount, percents, share, date, …;
- is the number pro forma;
- row header;
- column header;
- table header;
- name of the company associated with the table;
- date from the column header, if any;
- date of the document;
- type of the document;
- any other auxiliary information.

Extracting table headers

Table headers are put in capital letters in the provided HTML documents. Use this additional info to extract them.

The name of the company goes directly before the table heading. Use this into to extract it.

Any auxiliary info about the table goes directly after the heading in a **div** element.

Extracting numbers

In order to extract numerical data, let us go with the following approach.

1. Choose a table to process with.
2. Find **tr** and **th** tags in it.
3. Iterate over child **td** tags of the current row, with storing index of the current column. Keep in mind the **colspan** attribute.
4. For each such a tag:
    - Stripe all the HTML tags inside of it.
    - Assume that the content of a cell is a number. Split it into whole and fractional part (separated by dot). Fractional part might be absent.
    - Split the whole part into tokens, by using whitespace characters and commas as separators. Ignore empty tokens.
    - Try to join the numbers separated by commas and/or spaces, into a single token and mark it as a number.
    - If not succeeded, this cell contains no numerical information. In case it is the first cell of a row, store its data as a row heading. Otherwise, if there were no numerical data is any row above the current, store the column header for the column number for the current column index, into an array/list. Otherwise, add the text to a list of unknown data cells.
    - Add meta-information from the row, column, table and document heading, as well as cell itself.

The most interesting part of this is joining the numbers separated by commas and/or spaces, into a single number. Here is one possible way of doing it:

1. Iterate over all the tokens.
2. If the current token is a number, and we are not inside a number right now, set the **inside** flag. Parse the number and set the **current** number to the parsing result.
3. Otherwise, if the current token is a number, and we are **inside** already, count the number of digits in the token **n**. Multiply the **current** number by **pow(10, n)**. Parse the number and add it to the **current** number.
4. If the current token is comma or whitespace, ignore it.
5. Otherwise, if **inside** flag is set, save the **current** number by adding it to all the outputs. Unset the **inside** flag.
6. If there is a unit symbol or name nearby the number (left or right of it), then store it.

After this procedure is done, multiply the number by a unit of measurement from the table aux info, like millions, thousands etc.

Detecting the units

Detecting the units of measurement is not trivial in general. However, the given demo documents seems to conform to a standard on how the auxiliary info about the table is structured. This info can be used to detect the units of measurement. Need to handle the common multiple nouns (million, thousand, etc.) as well as the word “except”.

Dates

The tokens inside **td** tags should be converted to dates as follows:

1. Iterate over all the tokens.
2. If the current token is a number between 1900 and 2100, save it as a year component and set **inside** flag to true.
3. If the current token is a string and it matches one of 12 month English names, or its abbreviation with 3+ characters, store it as a month component. Use hashset with pre-stored names of months and abbreviations to improve performance.
4. If the current token is a number between 1 and 31, and we do not have a day component yet, store it as day component.
5. If the current token is a number between 1 and 31, and we already have a day component, swap day component with month component.
6. Validate the date. Check if month is in range 1..12, and it contains at least the same amount of days as day component.

Excluding useless data

Some tables present in the HTML do not have any value associated with them. Usually their title starts with “FORM”, “__” (underscores), or “TABLE OF CONTENTS”. Just ignore them.

Also, ignore any malformed data. Log warning about the issue.

Pro forma or not pro forma

Detecting if a document conforms to a pro forma is non trivial. I suppose that the purpose of this challenge is to extract the data from document while maintaining all the relevant information. Detecting if a document is in pro forma can be done as a separate Data Science/Machine learning challenge. They can receive the output of this challenge, as well as the full HTML as an input, and return the **confirms/does not confirm** answer as an output.

Details

The output JSON file should contain only data from cells with numerical data. The metadata of these cells should describe information about their row and column etc.

Putting it all together

A simple deterministic solution for a given task was presented here.

In case the current simple solution is not enough, it might be worth to launch a short Data Science/Machine learning challenge with aim to fulfill the missing parts.

Additionally, a prototype of this approach is included in the final deliverable. To get the results, compile it and run the “java -jar html-table.jar **filename**”, where filename is a name of a file on a local drive, or it can be omitted, in the latter case a default demo file will be used.