COA Document Understanding Write-up

Clayton Cohn
February 8, 2021

Abstract

This paper is written in support of a Jupyter Notebook that was created for the COA Document Understanding mini project assigned to me by Specright. The assignment was to extract a subset of fields from various COA PDF files and document the process. A detailed list of instructions for use of the code can be found in the Jupyter Notebook. My approach, limitations, and reflections are listed in the sections that follow.

1 Approach

In this section, I describe my approach to extracting the data from the COAs. While I did not have time to attempt Task B with code, I have nevertheless included my intended approach as per the guidelines specified in the mini project description.

1.1 Task A

The first thing I did was examine the various files. Although the COAs for Task A were all from the same supplier, I noticed that there was still a considerable lack of uniformity regarding terminology used across documents, and knew that this would need to be addressed. I decided to extract the data via a Python PDF parser and experimented with three different packages: PDFMiner, PyPDF2, and pdftotext. I decided on pdftotext, as this package most accurately parsed the COA files.

There were several issues that arose during the parsing process. The pdftotext package parses documents page by page, so I had to identify the COAs with multiple pages and handle them individually. pdftotext also occasionally mixed up the line order (once or twice in only a couple of documents), so this also had to be corrected manually. The "product name" and "lot number" were not particularly difficult to identify, as all seventy of the COA instances contained both strings. The tests, conversely, were much more difficult. The test categories varied considerably, so parsing based off of test category was not the best option. Instead, I noticed that all of the COAs (except one) included the test table immediately after the "Expiration Date," so I knew that I could identify the starting point for most COA test tables with minimal additional engineering. To identify the end of the test table, I found that nearly all test tables directly preceded the "the above data" disclosure or the disclaimer, so I knew that was where I could mark the end of the test table in each COA. Once I had the starting and ending lines of the test tables, parsing was easier. The first row in each test table was always the table's labels. Rows consisting of only parentheticals were ignored.

I then had to decide in which format to store the extracted data. The assignment itself did not list a downstream objective, so I tried to use an implementation that would be as universally applicable as possible. I decided on storing the extracted data in a Pandas DataFrame due to its ease of use and ubiquity in Artificial Intelligence (AI) and Machine Learning (ML). The more difficult problem was deciding how to store each COA's tests within the DataFrame. Each COA had multiple types of tests that were accompanied by varying descriptive categories. Furthermore, the number of test-related categories was not uniform across all instances. In the end, I decided to store the tests as nested lists, where the first item in each list consisted of the test labels for that particular COA. While straightforward, this approach necessitates that an additional step be taken before the training process (i.e., extracting the individual tests from the DataFrame). As such, it is recommended that the tests be extracted all at once and prior to training, as this will minimize the overhead that would have otherwise been encountered during training. To aid in the extraction of the tests, I provided a helper function at the bottom of the Jupyter Notebook that returns a given row's (COA) tests as a two-dimensional NumPy Array, which can then be utilized easily and efficiently.

1.2 Task B

The purpose of Task B was to extract the same data required for Task A, but from a completely disparate body of COAs taken from multiple suppliers. Because of the wide range of vocabulary used across the COAs, often to

describe the same feature, I would not spend much time trying to find words in common. Instead, I would create a hash set for each category. For instance, the "product_name" hash set could consist of "Product," "Product Name," "Common Name," etc. Once the hash sets were created for each field that needed to be extracted, I would try and parse the documents and check for edge cases. While this strategy would suffice for most of the fields, it would not suffice for identifying the tests. The complete lack of uniformity with regard to formatting would make it very difficult to approach parsing the tests the same way that I did in Task A. Instead, I would first try a PDF table parser such as the tabula package. I am already familiar with tabula, and have used it in the past to parse PDF tables when importing data.

Another issue that arises with the COAs from multiple suppliers is that many of them appear to have hand-written notes. I imagine that this would not bode well for a PDF parser. If these notes were pertinent, and needed to be included in the list of extracted fields, then I would likely have to eschew the PDF parser completely in favor of an Optical Character Recognition (OCR) approach. Some of the multi-supplier COAs also have multiple tables, so even with a PDF table parser, I would need to determine which table was the one specifying the results. At first glance, it appears that the test tables are nearly always the largest tables in those COAs consisting of multiple tables, so that would be a logical place to start. From there, it is just a matter of experimenting with parsing the various COAs to determine which methods would be optimal from the standpoints of accuracy and efficiency.

2 Limitations

Although the result was acceptable (i.e., the necessary data was accurately extracted), my method was not without its limitations. For one thing, my code will not be resilient to new COA instances. Even if they are from the same company as the others, the code will likely need to be tweaked to account for new edge conditions as they present themselves. Multi-page COAs, in particular, will have to be handled on a case-by-case basis as it currently stands. Additionally, there are a couple of areas in my code that can be better-optimized, but that I did not have time to do (these areas are identified in the comments of the Jupyter Notebook).

3 Reflections

Upon reflection, there are a few things I would do differently (or at the very least, experiment with) if I were to redo the assignment. I mentioned using hash sets in Task B, but in hindsight I should have opted for hash sets from the start. Any instance where an array is traversed in order to determine if it contains an item could be more efficiently implemented with a hash set. I also mentioned using a PDF table parser in Task B, which I should have also opted for in Task A. While the pdftotext package worked relatively well for the product and lot number fields, it proved very inefficient to parse the test tables. The majority of my code involves manually discerning the test tables and the resulting edge conditions. I imagine that the code would be much more concise had I used a package like tabula to parse the tables instead of doing so myself. I mentioned using OCR as well in Task B, which I should have also experimented with in Task A. Because the overall format of the single-supplier COAs wad relatively uniform, it may have been easier to program an OCR reader to read the files as opposed to parsing the underlying text myself. I also imagine that this would have made the code considerably shorter as well.

Lastly, I should have asked more questions. In an actual research project, I *always* make sure to have a complete understanding (at least as much as possible) of what the team is trying to accomplish, and this is done by asking questions. This ensures that I extract the data in a way that will be conducive to our research goals. It also ensures that tasks are completed correctly the first time.

I very much enjoyed working on this, and I look forward to hearing from you. Thank you for your time and careful consideration!