****

**DATA SCIENCE PROFESSIONAL CERTIFICATE CAPSTONE PROJECT FINAL REPORT**

**Group:** Global Economics

**Topics:** “Creation of a recommendation tool using the International Monetary Fund’s Article IV Reports”

Team Members:

|  |
| --- |
| Jac Chan |
| Ayoub Mharzi  Cat Din Tran |
| Daria Ulybina  Meagan Thompson |
|  |

*“This report summarizes our capstone project as part of the Georgetown data science certificate. Please refer the GitHub repository for any codes or scripts. This project might still be subject to further modifications and the update will be posted in the same repository.”*

**TABLE OF CONTENTS**

**Abstract**

**Problem and Hypothesis**

**Approach**

**Ingestion**

**Wrangling**

**Computation and Analysis**

**Modeling and Application**

**Reporting and Visualization**

**Discussion of Methodology**

**Discussion of Final Product**

**Conclusion and Lessons Learned**

**Abstract**

As part of its **Surveillance** mission, the International Monetary Fund conducts an ongoing annual process with member countries. These are known as “Article IV consultations” as they fall under Article IV of the IMF’s list of agreements. Each annual exercise results in a report, one per country per year, referred to as Article IV reports. Each report contains an assessment of the country’s economic and financial policies and discusses possible developments and future policies. The recommendations and policies contained in the Article IVs are used internally at the IMF, and by country authorities, other institutions, and companies. All of the articles produced are available via the IMF external website in a PDF format. The current practice for finding information contained in the Article is to download the document and read through or search the PDF for instances of the economic indicator. Our capstone project undertook text analysis on PDF data to allow IMF economists to search for Article IV reports based on key economic indicators.

**Problem and Hypothesis**

In the case of an economist working for the IMF and looking for a specific issue, policy, or recommendation, if the recommendation is not related to a specific study country or a small set of countries, the economist would need to look through annual reports for 186 member countries to find Article IV reports for countries with similar issues. Currently, a person would reach out to the country teams or people working on the same topics, etc. This is a very tedious process and is time consuming. The external website where the reports are available provides a very basic tagging system but this is not useful in the case of an in-depth search related to the content of an Article IV.

The idea behind our project comes from the question: ***Is it possible to identify similar economic situations through space and time using the Article IVs and combine that with quantitative data to surface trends and insights?***

**Approach**

We created a tagging system that identifies sets of keywords related to different topics to detect similarities based on the Article IV content. To do so our idea was to analyze the text of each Article IV using Natural Language Processing (NLP) and Machine Learning.

**Ingestion - Downloading the Article IVs**

The first step was to create a process to download the documents and store them locally. We wrote a python script using BeautifulSoup and requests libraries to download PDFs from the IMF external website starting from a specific year (specified within the code) and saved the documents in a local directory. The script also generated two additional outputs:

* The “log”: a txt file containing a list of the newly downloaded documents compared to the previous download;
* A summary table: a csv file containing metadata of the reports (link to the PDF, author, size, publication date, subject, etc…).

**Wrangling - PDF to HTML**

Once downloaded, the next step was to convert the documents to a format other than PDF, as creating a generic process to extract data from PDFs is quite difficult. We converted documents to HTML format, which is easier to map and clean. We also chose this format because the conversion is well documented and easy to use with “pdfminer”, and some team members had experience with this process.

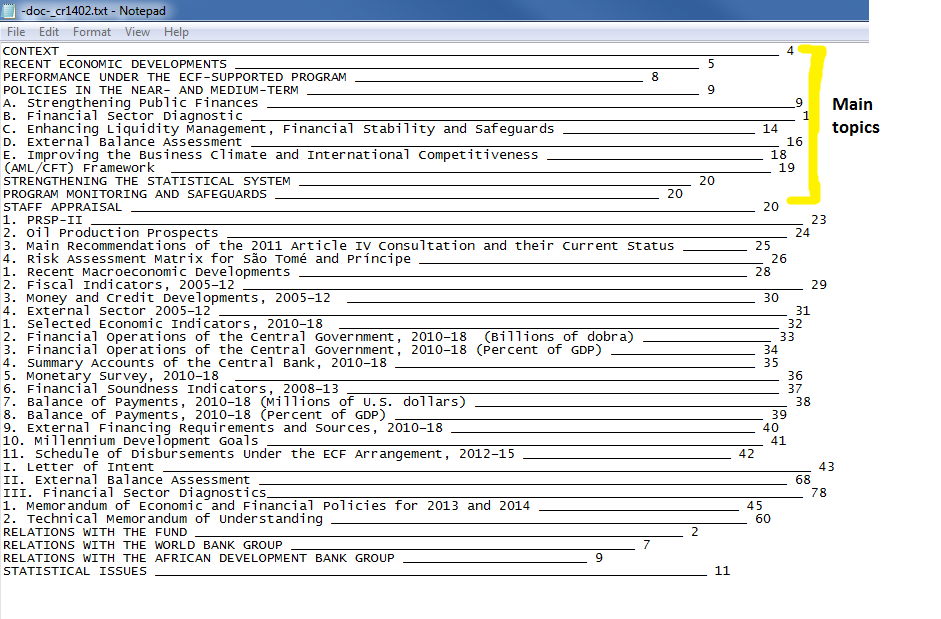
The script we created called the converter module of the pdfminer library and generated the HTML equivalent of each Article IV. However, while running this on the entire set of documents, we noticed that some of the documents were not processed at all, while some would process indefinitely and not move to the next record. After some research, we realized that this was due to the existence of horizontal tables and text which was not handled by the converter module. To avoid this, we added a timer to the script that would kick a document out and move to the next one if no output was generated after 180 seconds. The updated script generated a set of 372 HTML Article IV reports from 2013 to 2017.

**Wrangling - Extracting content**

Running a clustering and tagging algorithm on the entire text contained in an Article IV would generate a lot of noise since we were only targeting some of the main topics such as “fiscal” or “monetary” and also due to the existence of a lot of tables, charts and numeric analysis that was not of a very big importance to us in the scope of this project. To avoid this, the idea was to extract specific chunks of text related to the main topics discussed in the reports and run the algorithm on those.

Once we started analyzing the previously generated HTML documents we noticed that for many cased the HTML tags were not very well structured and there is no generic path that would allow to easily and generically extract text related to the main topics of each Article IV.

After trying several approaches, we observed that the thing in common to all the HTML files was how the tables of contents were identified (**Topic\_\_(a number certain of underscores)\_\_ page number**). Having the table of contents of each document would ease the text extraction since we can identify the main topics and extract the text related to each one. The extraction of the tables of contents was done by a python script looping through the HTML documents and using regular expressions for the extraction. The output of the script was a list of txt files named after the source document as shown below.



After a quick look at the created tables of content, we observed that all the main topics are discussed before that “Staff appraisal” of each Article IV.

**Creating the JSON dictionary:**

The next step was to extract the text chunks related to each topic discussed in the reports using the table of contents and the HTML files then save the output in a format easy to use for the rest of our project. The output format we agreed on was JSON, since it’s a lightweight format and would be easy to use for the rest of the project.

Before beginning the extraction we agreed to create a JSON dictionary of the table of contents. This would allow us to simplify and have a better structure from the table of contents. Additionally, this method would allow us to target only the main topics (everything above the “Staff appraisal”) and get rid of everything else. To do so we created a script that loops through the txt files and creates one big JSON dictionary with the following structure: {**“Document1 name”: [“Topic 1”,” Topic 2”,…], “Document2 name”: [“Topic 1”,” Topic 2”,..], …}**

The “Document Name” is the id given to the document on the IMF website (Unique id per document) and that is saved as the document’s name at the download.

After doing this we started creating our python script to generate a second JSON dictionary containing the text related to each topic. This was using both the tables of contents (in JSON format) and the Article IVs (in HTML format). In this process, we used the following libraries: **BeautifulSoup**, **json** and **collections.** The idea was to create a script that uses regular expressions (regex) to extract the chunk of text between two topics and that uses “STAFF APPRAISAL” as a stop word. The output had a bit of noise (minor tables and some numeric data was extracted as well) but this was negligible and did not alter our final results.

The python script generated one JSON dictionary containing all documents with the following structure: **[ {“content”: [ {“head”: “topic1 title”, “tail”: “topic1 text body”}, {“head”: “topic2 title”, “tail”: “topic2 text body”}…..], “doc”: “ Document 1 name”}, {“content”: [ {“head”: “topic1 title”, “tail”: “topic1 text body”}, {“head”: “topic2 title”, “tail”: “topic2 text body”}…..], “doc”: “ Document 2 name”} ….]**

**Adding the year and country**

The documents in the created JSON dictionary were identified by their id at the download (EX: -doc-cr1753), but in order to identify the similar economic situations in space and time, we needed to add the country name and publication year to each object inside the dictionary. This was easy since this information was stored in the summary table created at the download of the PDF documents. We created a JSON object structured as follow: **{“document1 id”: [“Country name”, “publication year”], “document2 id”: [“Country name”, “publication year”] ...}.** After that, we merged this dictionary with the one containing the topics and texts using the document id as a reference for the merge.

**Computation and Analysis**

I think this is where we need to talk about the clustering and tagging. Basically, I need the steps that we took and what the outputs look like, how we fine-tuned hyperparameters, etc… Daria – can you start filling this in?

**Reporting and Visualization**

Need some details

**Results**

**Lessons learned**

1. Don’t mess with PDFs. This isn’t entirely true, but we did learn that there is not a standard way of extracting text from PDFs and without some subject matter expertise to guide you, it is difficult to decide what is important
2. Never underestimate the time it takes to process and wrangle the data so it is useful for your application.
3. Version control is very important. Group needed to spend more time understanding and adhering to GitHub version control standards.

**Location of code:** <https://github.com/georgetown-analytics/global-economics>

**References**

<https://www.imf.org/external/about/econsurv.htm>

<http://docs.python-requests.org/en/master/#>

<https://github.com/euske/pdfminer>

<https://www.crummy.com/software/BeautifulSoup/>

<http://bdewilde.github.io/blog/2014/09/23/intro-to-automatic-keyphrase-extraction/>

<http://brandonrose.org/clustering>

<https://docs.python.org/3/library/collections.html>