Project Number: JPA 1803, MXC 0579

Financial Report Database and Analysis

A Major Qualifying Project Report

Submitted to the Faculty of

Worcester Polytechnic Institute

In Partial Fulfillment of the Requirements for the

Degree of Bachelor of Science

By

Seles Li

Brady Snowden

Date: December 14th, 2018

Approved:

Professor Jon Abraham, Advisor

Professor Michael Ciaraldi, Advisor

Professor Kevin Sweeney, Advisor

Abstract

The goals of this project were to produce a research database of financial documents for Angelo Gordon to support the firm in deciding their future investments and to simplify the comparison and analysis of these documents. Using a combination of Python and R programming, we created a central structured database of financial reports containing information regarding both public and private companies. We simplified the analysis process by automating the extraction of specific values from some PDF reports of public companies. We demonstrated the extraction by testing it on a representative set of reports from 13 companies. This project will significantly reduce the time an investment analyst needs to collect and review large amounts of information.

Acknowledgements

We would like to express our gratitude to the following individuals for their assistance in the completion of this MQP:

Scott Burton for making this opportunity possible and for supporting us in achieving our goals for this project.

Xinkai Wu for his guidance regarding the approach of this project and for clearly outlining the deliverables to allow the team to fulfill the objectives of this project.

Our sponsor Angelo Gordon for providing us the opportunity to start a new project with them.

Professor Kevin Sweeney for advising this project and for making this opportunity possible.

Professors Michael Ciaraldi and Jon Abraham, our advisors of this project, for their feedback and for their guidance in devising solutions to problems we encountered.

Professor Carolina Ruiz for her assistance in giving us a proper starting point regarding data mining.

Professors Marcel Blais and Randy Paffenroth for providing us with suggestions of analysis methods for the project.

Table of Contents

Executive Summary **v**

1 Introduction **1**

2 Background **2**

2.1 Angelo Gordon 2

2.2 Vendors 2

2.2.1 FinDox 3

2.2.2 EDGAR 3

3 Projects **8**

3.1 Project 1: Database of Annual and Quarterly Reports 8

3.1.1 Software 8

3.1.2 Process 10

3.1.3 Results 12

3.2 Project 2: Collecting Data from X-17A-5 Reports 13

3.2.1 Software 13

3.2.2 Process 20

3.2.3 Analysis 27

3.2.4 Results 29

4 Conclusion **30**

List of Figures

1 Procedure for downloading HTML reports 11

2 LSTM diagram 16

3 Procedure of a Convolutional Neural Network 18

4 Rectifier Function 19

5 Procedure for downloading PDF reports 22

6 Procedure for converting PDFs to text 23

7 Procedure for image editing and OCR process 24

8 Procedure for parsing the text 26

Executive Summary

This project was a collaboration during the Fall 2018 semester between Angelo Gordon and two undergraduate students from Worcester Polytechnic Institute. This partnership created an opportunity for the students to work as consultants of Angelo Gordon and apply their programming and analytical skills. Angelo Gordon's key objective in this project was to have an organized and automated system for downloading financial documents of companies, in order to support the firm in future investment decisions. Angelo Gordon identified two projects, titled as "Database of Annual and Quarterly Reports" (hereafter known as "Project 1") and "Collecting Data from X-17A-5 Reports" (hereafter known as "Project 2"). These two projects were a combination of database creation and financial analysis.

For Project 1, we automatically populated a database of financial reports by writing an analysis program in a Spark notebook using Python and R. We structured the database so that the reports were first separated by the company and then further organized into sub-folders based on the filing type of the report followed by the year. This system is used for files from both FinDox and the SEC's EDGAR database. The reports are stored on a cloud-based file system. We also implemented a search function to increase the ability to find specific files or information.

With Project 2, we analyzed the document database that we organized in Project 1 using a program written in R. To simplify the analysis process, we converted the documents to a consistent format. We then generated reports with the information collected from the documents and used the data visualization program Power BI to produce graphs from these reports. The final product was a functioning analysis program that can generate a report from a representative sample of documents. This project can be continued by expanding the extraction of the fields to different types of financial reports and by improving the accuracy of these extractions.

With the organized database, investment analysts will be able to find the financial information they need quickly and efficiently. Angelo Gordon can benefit from the project as they can easily visualize and analyze the information from the companies, and they can

discover new trends in the information that they may not have expected. Then, they will be able to gain new insights and perspectives and make better informed investment decisions.

1 Introduction

This report describes the work completed by our Major Qualifying Project team at Angelo Gordon. Angelo Gordon is an investment firm which specializes in alternative investment opportunities. Analysts at Angelo Gordon use many services and sources to gather information on possible investments, and the company was looking to consolidate the reports from these services into one single location. Previously any files downloaded were saved to a network drive without a well-defined folder structure. With our help, the company's goal was to create a single, cloud-based, structured database of financial documents and to automate the parsing of a specific annual summary report to save time.

This paper begins with a background of our sponsor and a brief description of their goals for this project, followed by a description of the information vendors used in this project. Next, the two projects we worked on are discussed in detail. For each of these projects, the report is broken down into sections describing the software used in the project, the process of fulfilling the project requirements, and the results. For Project 2, an additional section is added for an analysis of the process, and some recommendations on how best to use the program are added to the results section.

2 Background

This section of the report will provide background information on the sponsor company, as well as information regarding the data vendors we worked with.

2.1 Angelo Gordon

Angelo Gordon is a private investment firm located in New York, NY. The company was founded in 1988 by John Angelo and Michael Gordon [1]. Currently, Michael Gordon is the Chief Executive Officer and one of three Co-Chief Investment Officers. The other heads of the company are Josh Baumgarten, Co-Chief Investment Officer and head of the credit division, Adam Schwartz, Co-Chief Investment Officer and head of the real estate division, and Kirk Wickman, Chief Operations Officer.

Angelo Gordon operates across three main spectrums: corporate credit, structured credit, and lending. The investment process at Angelo Gordon is driven by extensive research on the equity and debt markets. Their award-winning multi-strategy platform gathers and applies market information across all three spectrums in order to make quality investments.

Recently, Angelo Gordon decided to change from their current information storage system, a multiple terabyte file system with an inconsistent structure, to a cloud storage system, unifying and structuring their information in the process. As a part of this process, they wanted to develop a way for their investment analysts to make better use of the substantial amount of information, which is where this project came in.

2.2 Vendors

Angelo Gordon uses two primary vendors for investment information: FinDox for reports from private companies and the SEC EDGAR database for reports from public companies.

2.2.1 FinDox

FinDox is a provider of financial reports, company update information and other related information of private companies. Private companies are privately owned, typically by the founder(s) or a group of private investors. Private companies are not required to disclose financial information publicly. These companies operate over a range of industries including Banking & Finance, IT Services, Retail, and more. There is no preset schedule for when the data for a company in FinDox is updated. Each company's data is instead updated as needed. A variety of file types are uploaded to FinDox, including both digital and scanned PDFs, Microsoft Word and Excel documents, and assorted text files. Filters can be used to search for company deals posted within a specific time period, industry, or category of documents [2]. There is no defined structure for reports uploaded to FinDox, so it is possible for report formatting to vary over time as well as between companies. For this project, we only looked at the reports that were generated by the analysts at Angelo Gordon.

2.2.2 EDGAR

EDGAR - Electronic Data Gathering, Analysis, and Retrieval - is a publicly accessible database system maintained by the Securities and Exchange Commission (SEC) [3]. The system is used by publicly traded companies to file reports to the SEC. Publicly traded companies are companies where all or part of their shares have been sold to the public through an initial public offering (IPO). Documents stored in the EDGAR database include annual and quarterly financial reports and other electronically submitted forms, in an assortment of file types including PDFs, HTML files, and some text files. The EDGAR database allows users to search company report filings by stock ticker symbols or Central Index Keys (CIKs), a unique identifier for each company set by the SEC. Filters can also be used to search for specific filing types and time periods. Unlike FinDox, there is a defined structure for each of the forms in the EDGAR database, established by the SEC.

During this project, we worked with a small subset of the filings available within the EDGAR database. These were the 8-K, 10-Q, 10-K, 20-F, NT 10-K, and X-17A-5 reports.

10-Q A 10-Q report is a quarterly report to be filed by publicly traded companies. These reports are summaries of a company's performance over the previous fiscal quarter. There are ten items that make up these reports, divided into two sections. The first section begins with financial statements comparing the previous two quarters. These statements are followed by an analysis of the financial conditions and results of operations. Following the analysis comes market risk disclosures, both qualitative and quantitative [4]. The final item of the first section is an evaluation of the company's disclosure controls and procedures [5].

The second section begins with a description of any legal proceedings involving the company over the past quarter, followed by a subsection disclosing any risk factor changes. The next item is a listing of any unregistered sales of equity securities. The third item in section two is a notification of any payment defaults. This is followed by a statement disclosing any violations of mine safety, if applicable. Item five is a collection of any other information that should be reported on Form 8-K during this period, which is described below. The final item is a listing of all reference files attached to this report along with a description of each [4].

10-K A 10-K report is an annual report to be filed by publicly traded companies. It is a comprehensive summary of the performance of a company over the last fiscal year. There are four parts that comprise these reports, each broken down into multiple items. The first part starts with a breakdown of the business, including information about its products and employees. This is followed by a discussion of the risk factors facing the company. After that comes a listing of the properties owned by the company and any legal proceedings that the company was involved in over the last fiscal year. The final item in the first part of a 10k is a disclosure of any mine safety violations.

The second part of these reports starts with information about the company's presence on the stock market. The next item is a five year comparison of selected financial data, followed by a management analysis of said data. This is followed by the full financial statements.

The third part of these reports is a breakdown of the corporate governance of the company, including director and officer information, information on executive compensation and

security ownership, and information regarding accounting fees and services. The fourth and final part of these reports is a listing of all other documents filed as part of the submitted report and a summary of the report as a whole.

8-K An 8-K report is a report to be filed by publicly traded companies within four business days of one of the events listed below. These reports act as updates or amendments to a previously filed 10-Q or 10-K report. An 8-K must be filed under one or more of the following conditions:

- Entry into/Termination of an agreement
- Bankruptcy
- Mine shutdown/pattern of safety violations
- Acquisition/Disposition of assets
- Creation of/Changes to a direct financial obligation
- Impairment of assets
- Delisting from the stock market
- Unregistered sale of securities
- Modification to security holders' rights
- Change of auditing accountant
- Non-reliability in past financial statements
- Changes to directors or other officers
- Changes to articles of incorporation/bylaws
- Change of fiscal year
- Changes to code of ethics
- Change in shell company status
- Change of trustee
- Change in credit enhancement

Any report submission must contain all relevant information to the conditional event(s) [6].

NT 10-K A NT 10-K report is a report that must be filed when a publicly traded company is unable to file a 10-K report within the required time frame. After a NT 10-K report is submitted within 90 days of the end of the company's fiscal year, a 10-K report must be filed within 15 days. This report should contain the reason why the 10-K report was unable to be filed before the deadline, as well as an indication of whether a significant change in earnings is expected as compared to the previous filing period. This indication must be accompanied by a explanation and estimate of the change if one is expected [7].

20-F A 20-F report is an annual report that must be filed by all publicly traded foreign private issuers [8]. Foreign private issuers are companies that have less than 50 percent of their shares traded in the United States. Once they exceed that threshold, they need to file the above reports instead [9]. A 20-F report must be filed within four months of the end of a fiscal year or upon the change of a company's fiscal year-end date. The following information should be included in a 20-F report [10]:

- Role of individuals involved in company's listing or registration
- Offering and methods and important dates related to the offering
- Key information of company's financial data and risk factors
- Business operations of company and related factors
- Financial condition and related results
- Company's directors, managers and employees
- Major shareholders and transactions
- Consolidated financial statements
- Listing of company's securities and distribution plan
- Disclosure on market risk
- Description of securities
- Dividend payments
- Material modification to rights of security holders and use of proceeds

- Management of internal control over financial reporting
- Financial statements
- Other disclosures

X-17A-5 An X-17A-5 report, also known as a Financial and Operational Combined Uniform Single (FOCUS) report, is an annual report that must be filed by all broker-dealers registered with the SEC. A broker-dealer is a firm that buys and sells securities either for their own gain or on behalf of customers. X-17A-5 reports are broken down into three parts. The first part contains general information about the registered broker-dealer [11], the second part contains the complete breakdown of an annual audit [12], and the third part contains the identification information for the broker-dealer, as well as consolidated statements of their financial condition [13].

3 Projects

The work we completed on this project is separated into two sub-projects. The first was building a central database of annual and quarterly reports, and the second was collecting data from a specific type of report from the EDGAR database.

3.1 Project 1: Database of Annual and Quarterly Reports

This sub-project was established with the goal of creating a single consolidated database of financial reports from both FinDox and EDGAR along with a way to find files within the database. A previous in-house project at the sponsor company had produced scripts for downloading some reports from both vendors, as well as for reducing the downloaded EDGAR reports to just textual information. This sub-project required us to familiarize ourselves with the provided source code, with the end goal of porting the local code to a cloud service and organizing the output files in a centralized location.

The following sections will discuss the different services used to accomplish this goal, the process for accomplishing the goal, and the final deliverables.

3.1.1 Software

To accomplish this goal, we used three services under the Microsoft Azure umbrella. Microsoft Azure is a cloud services solution developed for business use [14]. Azure provides a wide array of services, ranging from storage to data analysis and more. These services are hosted on Microsoft servers, and the user's payment tier determines the processing power and storage allocated for their usage.

The main Azure services used in this project are Azure Databricks, Azure Data Lake, and Azure Search Services.

Azure Databricks Azure Databricks is a collaborative cloud-based Apache Spark programming environment. Databricks supports code written in Python, R, Scala, Spark, and SQL. Programs on Databricks are written in Jupyter notebooks hosted on Microsoft servers.

These notebooks consist of cells of code, which can be run all together or individually. Individual cells can contain only a single coding language, but notebooks can contain cells of different languages.

Programs written using this service are run on clusters, which are individually configured run-time environments with custom settings and resources. Clusters can simultaneously process and execute multiple notebooks. Notebooks can be set up to run on a recurring schedule, in which case a dedicated cluster is created at the time of execution and released when the job finishes.

For this project, Databricks was used to host and execute the code for downloading and processing reports from both vendors. The project code was written using a combination of R and Python, with R used for the FinDox file processing and Python used for the EDGAR files. The choice of which language to use was determined from the base code provided from the previous project. The use of Databricks allowed for simple integration with the Azure Data Lake Storage system, described below.

Azure Data Lake Storage Azure Data Lake Storage is a cloud-based file storage system built around the Apache Hadoop Distributed File System (HDFS) standard [15]. Data Lake Storage is designed to support atomic file and folder operations, which are operations that cannot be interrupted or left partially completed. Data Lake Storage operations can be performed in multiple ways, including through Representational State Transfer (REST) APIs, a web portal, and standard Python file system operations.

For this project, Data Lake was used for the storage of the downloaded reports. The Data Lake system can easily be accessed by anyone at the company, meaning the reports would only need to be downloaded from the databases a single time, rather than every time an analyst needed them.

Azure Search Azure Search is an analytic service solution providing in-depth searches over stored data [16]. The search service processes structured data files or text documents and maps the data to index fields. Using these fields, massive amounts of data can easily

be combined and searched through using a simple REST API. When searching through numerical data, fields can be filtered using basic logical operations. For text data, terms can be searched within fields using either exact matches or string similarity.

For this project, Azure Search was used to demonstrate the ability to search through and filter some of the many text files downloaded from both FinDox and the EDGAR database. Using the REST API or web interface, a user can enter a search term and receive a list of all the reports that contain that term. The search output can be modified to return differing levels of information, such as the filing date or the full text of the report.

3.1.2 Process

As mentioned above, the goal of this project was to familiarize ourselves with the previous project's code and adapt it to function on the cloud.

Downloading the EDGAR files This section of the script accesses the EDGAR database and downloads the requested reports. This process can be seen in the flowchart below. The script begins by reading a user-provided CSV file into memory. Each line of this file should contain a company name and the SEC designated CIK for that company. The company name is used for naming folders and files, so the user should avoid using punctuation when entering these names. Once the CIKs are stored in program memory, the script iterates through each one.

For each CIK in the list, the database is queried for the most recent 8-K, 10-Q, 10-K, 20-F, and NT 10-K report submissions, checking for up to 100 of each per company. Because report submissions consist of multiple files, the HTTP response is an XML document containing links to the relevant report submissions. For each of these links, the script queries the database with another HTTP GET request to retrieve the link to the HTML file for each submission. For each of these links, the script retrieves the report using a final HTTP GET request and saves it under the file path `SEC-Edgar-data/`*`Company Name/Filing Type/Company Name_upload-date_Filing Type`*`.html`. For example, the 2016 10-K

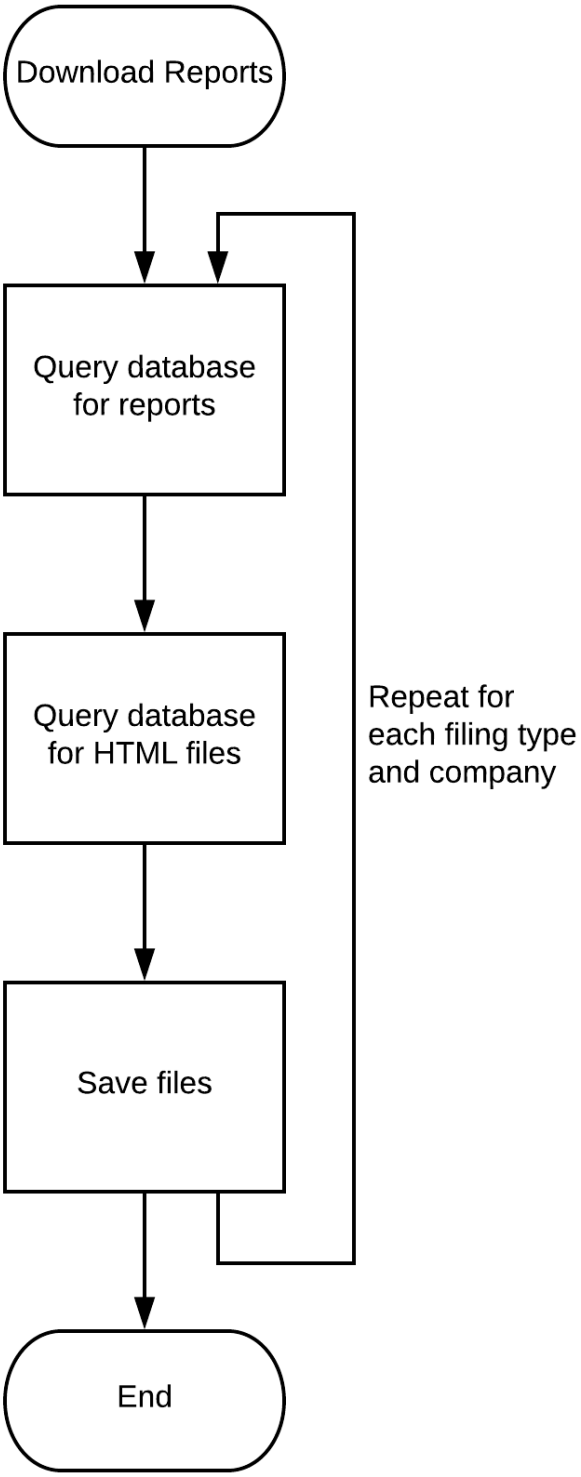

Figure 1: Procedure for downloading HTML reports

report from IBM would be located at `SEC-Edgar-data/IBM/10K/IBM_2016_10K.html`.

Downloading the FinDox files This section of the script accesses the FinDox database and downloads all available reports.

This script begins by attempting to log in to the FinDox website using an HTTP POST request. The credentials used are hard-coded into the script to allow it to run without requiring any user input, and they can be easily changed if necessary. Once the script has received a successful login message from FinDox, the database is queried for all deals available. The script then proceeds to download all files from each available deal, saving the files under the path `findox/Angelo Gordon/`*`Company Name/Deal Name/file`*`.` For example, the 2018 Quarter 2 financial report from Uber might be located at `findox/Angelo Gordon/Uber/2018 Q2 Investor Financials.pdf`.

Converting to plain text This section of the script converts the downloaded HTML files into human-readable text files. Because all of the FinDox files are uploaded in a human-readable format, this script only executes on the EDGAR files. This script works by removing all of the HTML tags and entities (, &lt, etc.) from each file. The remaining text consists of solely the body text from each file, which is then saved in a new text file.

3.1.3 Results

There were two final results for this project: the new structured report storage and the Azure Search implementation. The new storage structure provides a central location to access any relevant report from both EDGAR and FinDox, and the search implementation provides a strong proof of concept for a useful way to retrieve files containing the desired information. Both of these results provide Angelo Gordon with the opportunity to expand in the future, either by adding reports to the list of downloads or by moving to a full search over all files. With this database, Angelo Gordon can efficiently find the financial information of companies they need.

3.2 Project 2: Collecting Data from X-17A-5 Reports

The purpose of this sub-project was to take a series of lengthy reports and extract a small set of information from each. The reports in question were X-17A-5 reports from the EDGAR database, which are annual reports summarizing the current financial standing of a company. Each of these reports contains large amounts of information, which made them useful examples for extracting just a few fields. The end goal of this project was a program that could pull specific data out of a report and present it to analysts in a useful way. The three fields this project focused on were cash amounts, total assets, and total equity. Each of these three values provide an important insight into the state of a company. The cash value represents the liquidity of the company, specifically how much cash is available to the company beyond its liabilities. Together, the total assets and total equity values provide the leverage ratio, which indicates the amount of debt that company has when compared to its assets.

The following sections of the report will discuss the software used to accomplish this goal, the process our code goes through, and some analysis of the process alongside possible improvements for the future.

3.2.1 Software

To accomplish the goal of this project, we used three separate R packages: Pdftools, Tesseract, and Magick.

Pdftools Pdftools is a R package that is used to extract text data from PDF files. This package is built upon the Poppler c++ interface, which is a free library for rendering PDF files. In addition to extracting text, Pdftools also supports converting individual pages of PDF documents into JPEG, PNG and TIFF formats [17]. For our project, this package is used to extract the text from digital PDF files and to convert the pages of scanned PDF files into images for use with Tesseract.

Tesseract Tesseract is an R package used to detect and read text from images using optical character recognition (OCR). It was developed by Hewlett Packard Labs and made to be open-source in 2005 [18]. Google adopted the project since 2006. It was originally based on traditional computer vision algorithms, which includes image classification, object detection, and object tracking. As of version 4.0.0 (the latest stable release as of this report), this package has developed to be Deep Learning based and to have a primary recognition engine based on Long Short Term Memory networks (LSTM) [18].

As a basis for understanding a LSTM, a Recurrent Neural Network (RNN) is a type of neural network designed for capturing information from sequences or time series data. It is recurrent because the outputs of each state are fed back into the loop as the inputs of the next state. To train a neural network and improve the accuracy of an algorithm, backpropagation is needed. It is used to calculate the derivatives of the error of the network by using the chain rule as the weights of one layer affect the weights of the next layer [19]. Depending on the input values of the network, this algorithm repeatedly adjusts the weights and biases in the relative proportions to those changes, which will minimize the difference between the actual and desired output. Gradient descent is used for updating the weights of the neural network, and hence, minimizing the cost function J below, which represents the difference mentioned above [20]. It involves determining the changes in the weights for thousands of training examples and averaging the desired changes. It is based on the following cost function J:

$$J = \frac{1}{n}\sum_{i=1}^{n}(y\_predicted_i - y_i)^2,$$

where $y\_predicted_i = mx_i + b$

Gradients, the rates of change of the cost function over the rate of change of a certain input, are defined as the following:

$$\frac{\partial J}{\partial m} = \frac{2}{n}\sum_{i=1}^{n} x_i(mx_i + b - y_i)$$

$$\frac{\partial J}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(mx_i + b - y_i)$$

If the gradient is negative, that means the weight value should increase. If the gradient is positive, that means the weight value should decrease. A user-defined learning rate is needed to control how much the weights of the network are adjusted with respect to the gradient [21]. In order to know the direction to go to minimize the error, the above two equations are then multiplied by this learning rate, resulting in the following equations for two new values of m and b:

$$m = m - \frac{\partial J}{\partial m} * learning\_rate$$

$$b = b - \frac{\partial J}{\partial b} * learning\_rate$$

This complete procedure is essentially how neural networks learn. With a particular number of iterations, the algorithm can find the global minimum of the difference between the output and the desired output. However, the smaller the learning rate, the longer it takes for the algorithm to reach the minimum point. Tuning the learning rate will improve the efficiency of the algorithm.

In the example of a Recurrent Neural Network (RNN), all the nodes in each layer contribute to the output of the network. Thus, when updating the weight back through time, the gradient gets multiplied multiple times. Such multiplying with a weight close to 0 will lead the value of the gradient decrease quickly even closer to 0 [22]. This is called the Vanishing Gradient Problem. This will be bad for the machine since it is not improving over time. To resolve this issue, LSTM is introduced.

Essentially a special type of RNN with a chain like structure, a LSTM has additional interactions within the loops, including a cell state and three gates (a forget gate, an input gate and output gate), in the network.

As shown in Figure 2, the cell state is the horizontal line on the top that runs through the entire chain. To protect and control the cell state, the network requires gates that consist of sigmoid neural network layers, denoted as yellow rectangular boxes, and pointwise

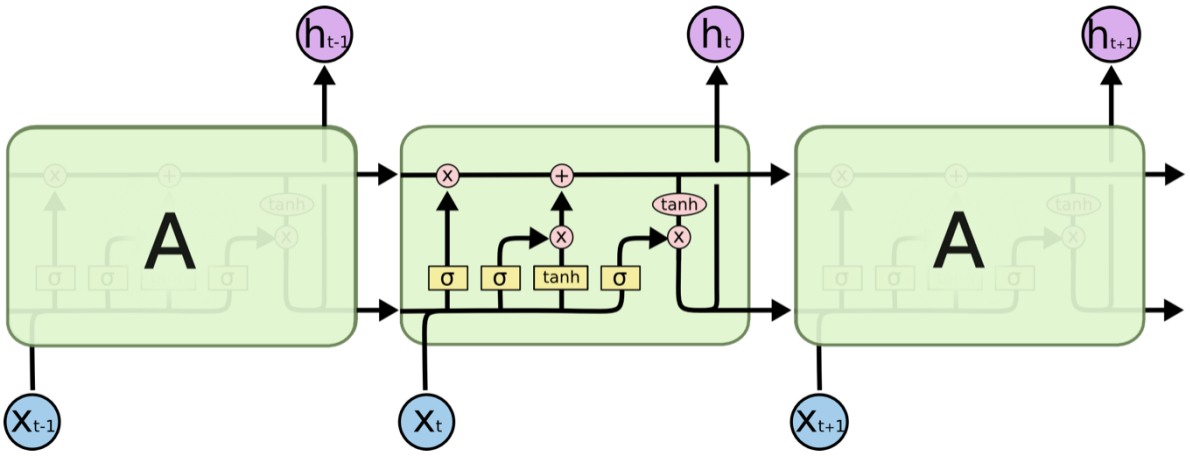

Figure 2: LSTM diagram [23]

multiplication operations, denoted as pink circles. A sigmoid neural network contains a sigmoid function, which introduces non-linearity to the network. Such function is a type of activation function, which defines the output of the neural network. The sigmoid function is defined as $A = \frac{1}{1+e^{-x}}$ and outputs values ranging from 0 to 1 [24]. The network layer outputs a number between 0 and 1 inclusively, representing a probability of the prediction of the layer's input.

With the interactions in a loop shown in Figure 2, a LSTM repeats the following general procedure on each loop iteration, along with the required computations [23]:

1. The forget gate layer, the first sigmoid layer, identifies the information that should be removed. When the forget gate is on, i.e. the output of the activation function is close to 1, the gradient does not vanish. The output value of this layer can be between 0 and 1, corresponding to the amount of useful information passed through [25]. When a value of 0 is the output, the information inputted into the gate will be removed. When a value of 1 is the output, all information in a component will be passed through. In reality, it is unlikely that the output value from this layer will be exactly 0 or 1.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

2. The input gate layer i_t, the second sigmoid layer, decides which values to be updated.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

3. The tanh layer creates a vector of new candidate values, $\tilde{C}_t$, to be added to the cell state.

$$\tilde{C}_t = tanh(W_C[h_{t-1}, x_t] + b_C)$$

This layer uses the tanh function, an activation function, to allow some memory to be removed. The function is defined as $tanh(x) = \frac{2}{1+e^{-2x}} - 1$ and has a range between -1 and 1 [26].

4. Step 2 and 3 together update the old cell state, C_{t-1}, into a new cell state, C_t.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

5. The output gate layer o_t, the last sigmoid layer, determines which parts of the cell state will be the output. The cell state will be put through the tanh layer, which pushes the candidate values to be between -1 and 1, and multiplied by the output of the sigmoid layer.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * tanh(C_t)$$

The concept of convolution should also be understood before the functions of convolutional neural networks (CNNs) are introduced. As shown in the definition below, a convolution is a combined integration of two functions, with one function modifying the other [28].

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

A CNN has two functions, feature classification and feature extraction, and is built by

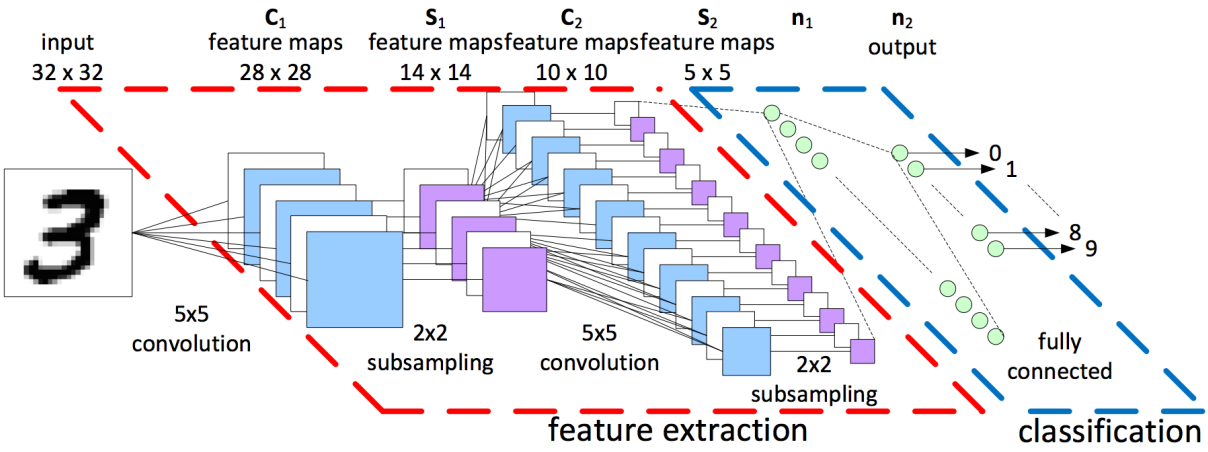

Figure 3: Procedure of a Convolutional Neural Network [27]

following the stages below [28]:

1. Convolution is first done to extract features from an input image by taking smaller squares of the input data and interpreting the image as a matrix of pixels.

2. A rectifier function, another type of activation function, is applied to increase non-linearity in the CNN. It is defined as $Y = max(0, x)$, where x is the input of a node in a layer of the network. The function is rectified at the bottom for negative input values and linear for all positive input values, as shown in Figure 4 below. Since it outputs 0 for all negative input values, it is sparsely activated, meaning it can give the model a better prediction and reduce the issue of overfitting [29]. Overfitting occurs when the neural network cannot provide a good generalization and predictions for new incoming data [30].

3. Pooling is applied to detect features in the images regardless of the difference of lighting in the images and the angles of the images. One type of pooling, max pooling, can be done by having a 2x2 matrix on the feature map and taking the largest values as the entire map is passed through. These values will then form a pooled feature map in a matrix format. This will keep the main features of the image while reducing the image size. Subsampling, also known as average pooling, can be done to reduce the distortion of the image by taking the average of the elements in the feature map [31].

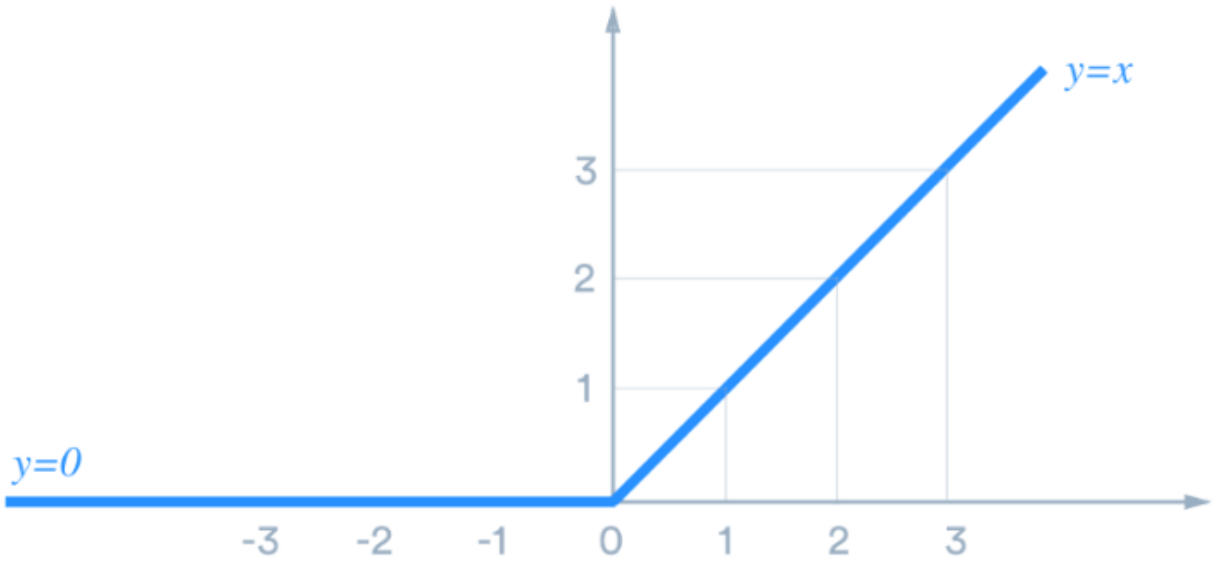

Figure 4: Rectifier Function [29]

4. Flattening is done by transforming the pooled feature map matrix obtained into a single column.

5. The flattened feature map obtained from the previous step is then passed through a neural network for processing by going through an input layer, a fully connected layer, and an output layer. A fully connected layer is defined to be a layer with every node connected to every node of another layer [32]. The predicted classes of the image are computed in this fully connected layer and are the outputs of this step. The error of the predictions can be reduced as the information is passed through the network. Using the softmax function below, the values output from the neural network will be between 0 and 1, which is the probability of each class.

$$\sigma : \mathbb{R}^K \to (0,1)^K, \text{ where } \mathbb{R} \text{ is a set of real numbers}$$

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \text{ for } j = 1, ..., K$$

In the matter of image processing techniques, a CNN takes in the image being detected as the input and applies a feature detector, which functions as the filter as shown in Figure 3. It

can produce a map of unique features while retaining the features that are crucial for image detection. This will reduce the image size and enhance the image processing efficiency [28]. Multiple feature maps of this type can be applied to prevent information loss of the image as each feature map can be used to detect different features of the image. After extracting the features of the image, the rectifier function is applied. Then, pooling or subsampling is done to reduce overfitting. Convolution and subsampling are done hand-in-hand multiple times, depending on the complexity of the input image, to complete the steps of extracting features in an image as shown in Figure3. At the final step of classification, values between 0 and 1, representing the probabilities of the predictions of the input image, will be the final output. The output with the highest probability shows what the CNN detects the input image as.

Tesseract was used in our project for extracting text from scanned image PDFs that were unable to be processed with Pdftools.

Magick Magick is a R package used for processing images. Magick is developed by rOpenSci and built upon the ImageMagick library, a comprehensive open source image processing library [33]. It supports such functions as image conversion, rescaling, and contrast adjustment. This package is used in our program to manipulate images to increase the accuracy of Tesseract, which included converting the images to black and white, increasing the pixel height of each character, and making adjustments on the level of contrast in the images.

3.2.2 Process

The main deliverable for this project was an R script to download and parse annual X-17A-5 reports from the SEC EDGAR database. The goal was to extract the cash, total assets, and total equity values from the body of the reports.

This script is divided into three main sections: downloading the reports, converting the reports into text, and parsing the text for data.

Downloading the Reports This section of the script accesses the EDGAR database and downloads the requested reports using the same process as in Project 1.

For each CIK in the list, the database is queried for the last 100 X-17A-5 reports for that company. The difference between the downloads for this script and for project 1 is that X-17A-5 reports are PDF documents. For each link to a PDF report, the script retrieves the report using a final HTTP GET request and saves it to the disk under the file path `SEC-Edgar-data/`*`Company Name`*`/x17a5/`*`Company Name_upload-date`*`_x17a5.pdf`. For example, the 2016 X-17A-5 report from IBM might be located at `SEC-Edgar-data/IBM/x17a5/IBM_2016-12-31_x17a5.pdf`.

Converting to Text This next section of the script converts all of the PDFs downloaded in the previous section into plain-text files. The script begins this process by iterating through each company that reports have been downloaded for. For each of these companies, the script iterates through each PDF report.

For each report, the script first attempts to extract the full text using the pdf_text function from the Pdftools library. This method works with all of the digitally created PDFs, but not with scanned image PDFs, which comprise approximately half of the X-17A-5 reports stored in the EDGAR database. For these files, the script uses the OCR function from the Tesseract library to extract the text. To improve the quality of the OCR output, the script performs some pre-processing on the image PDFs before they are passed to Tesseract. This pre-processing includes scaling the images up so each character has a greater pixel height, reducing the noise on the image, and increasing the contrast of the image. These image manipulations made a marked improvement on the overall quality of the text extraction. The extracted text is saved to a new file under the file path `SEC-Edgar-data/`*`Company Name`*`/x17a5/txt/`*`Company Name_upload-date`*`_x17a5.txt`. For example, the 2016 X-17A-5 report from South Street Securities might be located at `SEC-Edgar-data/SouthStreet/x17a5/txt/SouthStreet_2016-12-31_x17a5.txt`.

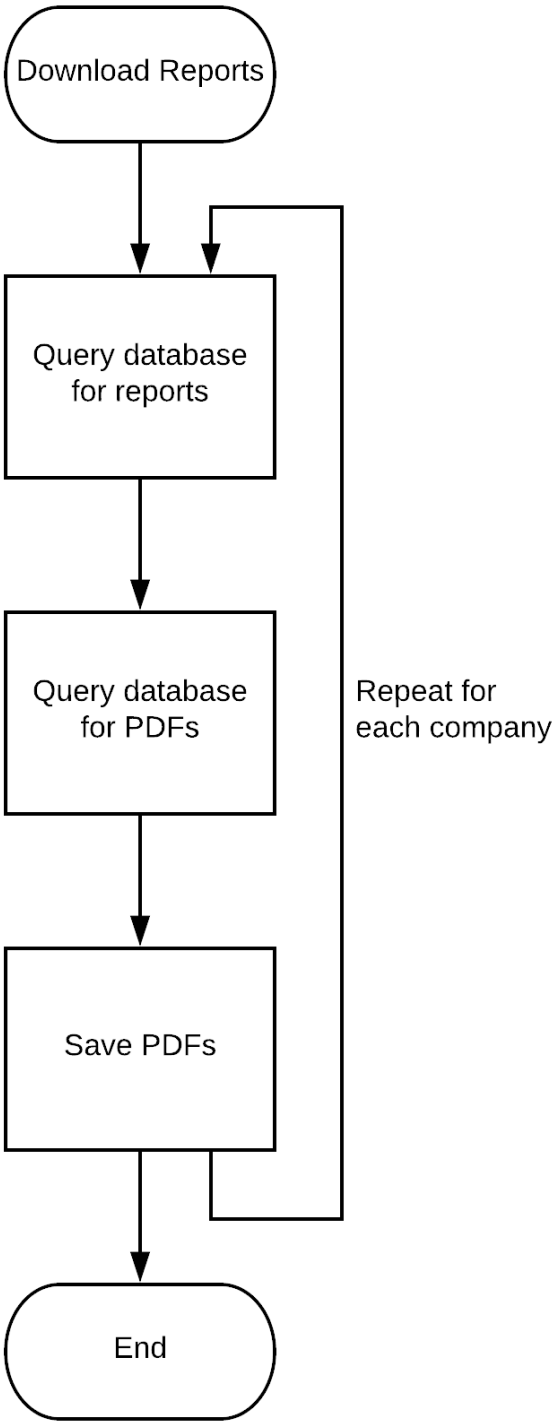

Figure 5: Procedure for downloading PDF reports

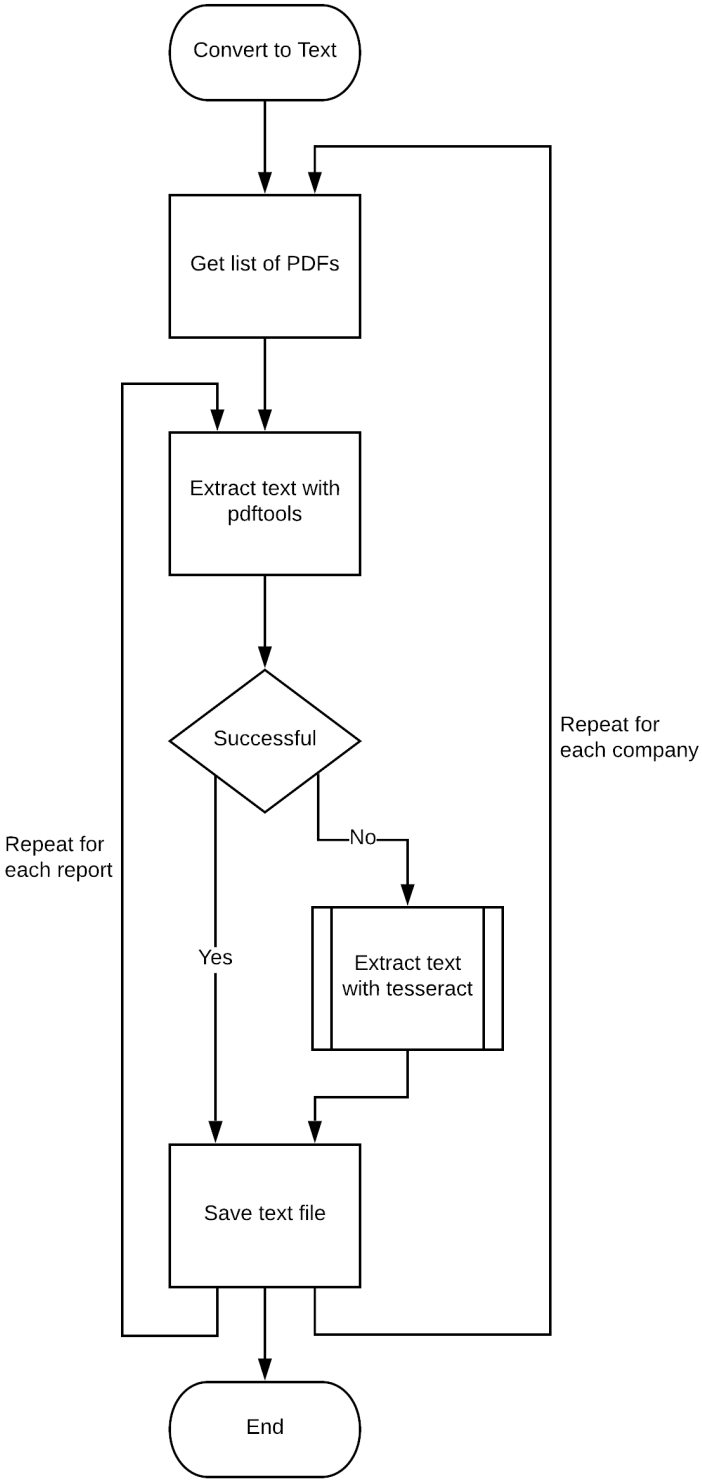

Figure 6: Procedure for converting PDFs to text

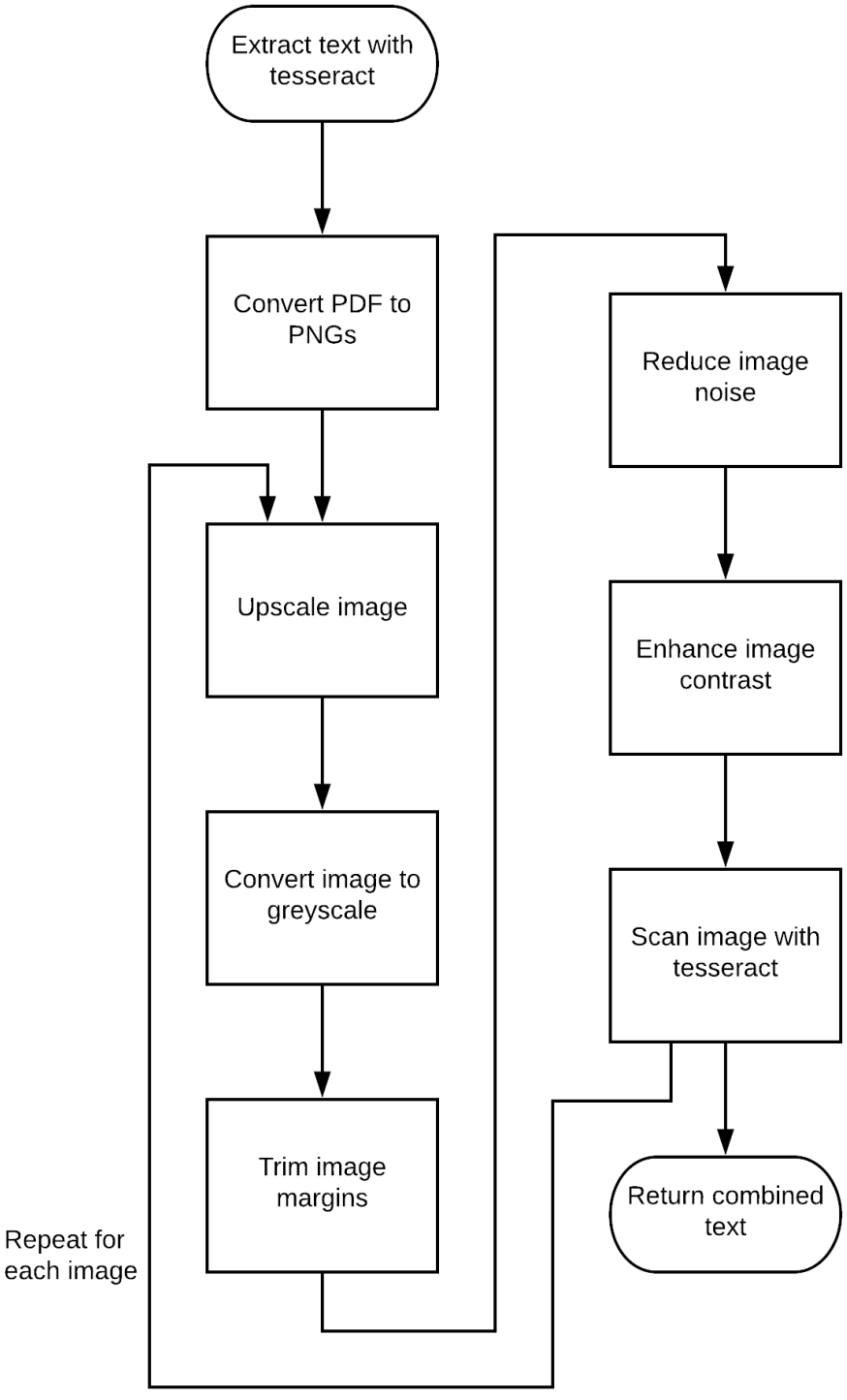

Figure 7: Procedure for image editing and OCR process

Parsing the Text The third section of the script contains the main functionality: parsing the data from the text into a CSV file. This process begins by reading a user-provided CSV file that contains computed regular expressions (regex) for each company. Each line of this file should consist of the company name, three regex patterns for selecting the line of text regarding cash, total assets, and total equity respectively, and a boolean value indicating whether to read the next or previous line of text first if the desired value is not on the same line as the text. This process starts by iterating through each company. For each company, the script then iterates through each text file from the previous section. For each text file, the script first uses some simple regex patterns to remove some extraneous characters. Some examples of this are replacing consecutive whitespace characters with a single space, removing commas and periods, and removing dashes and underscores.

Once the text has been cleaned up, the text is searched using the regex for that company from the CSV file, as well as the phrases "In Thousands"/"In Millions" to account for reports where values are written omitting the last three or six figures respectively. For each of these three lines of text, the script removes all text before a dollar sign, then extracts the first numerical value from that line. In the case where no numbers are found, the script proceeds to check each adjacent line for the number. Finally, the script calculates the leverage ratio (assets/equity) and the cash/asset ratio for that report. At this point, if any values have not been found or if the equity and asset values are equal, the report is re-scanned using the same OCR process as before. The equity and asset values being equal means the "total liabilities and equity" line was read instead of the "total equity" line. The parsing process is then repeated for this new text. After the values have been extracted from each text file for a company, they are put into a data frame and written to disk under the file path `CSV Output/`*`Company Name`*`.csv`.

After all of the companies have been processed, the script reads each of the data frames back into memory and appends them onto a single large data frame, which is then saved as `CSV Output/all.csv`.

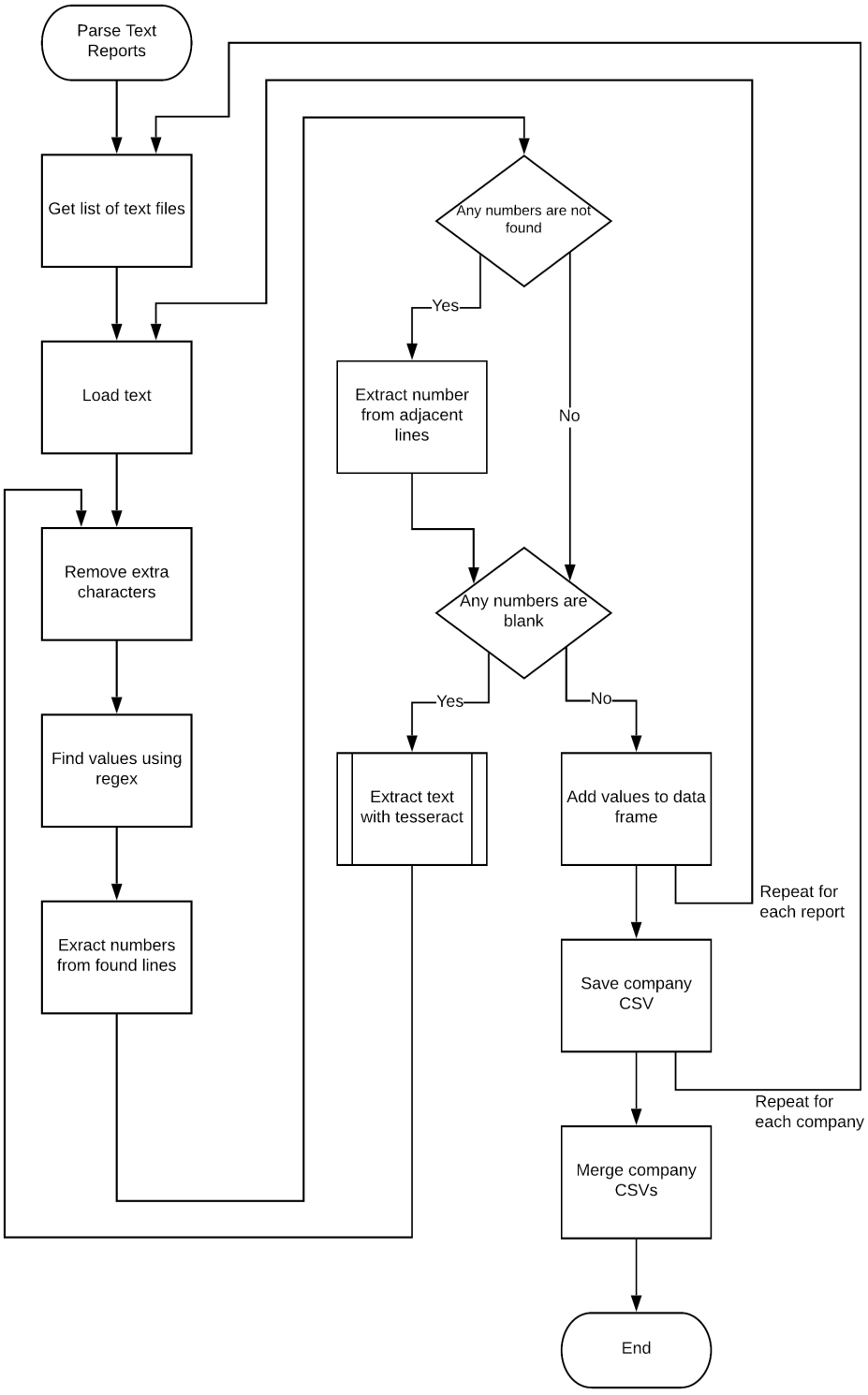

Figure 8: Procedure for parsing the text

3.2.3 Analysis

The following section will discuss the reasoning behind some project choices, some shortcomings of specific sections of this project, and how some improvements could be made.

Regex The regex patterns were derived through examining the extracted report text to determine unique strings that could identify the desired lines and nothing else. Each company uses slightly different phrasing, so a single pattern cannot apply to all reports. For new companies where there isn't already a regex pattern established, a default set of patterns was created using common patterns across the currently processed companies.

For the cash pattern, the default pattern is "(Cash and) | (Cash\s*\$? \s*\d+) | ^Cash$". The first element of this pattern selects lines of text that contain the words "Cash and". This pattern is used to match the phrase "Cash and cash equivalents" without being over-fitted. The second element of this pattern is "Cash\s*\$?\s*\d+". This pattern is used to match the word "Cash" followed by any number of spaces and a number. The question mark following the dollar sign selects lines either with or without a dollar sign preceding the number. The third element of the pattern is "Ĉash$", which matches lines which consist of the word "Cash" and nothing else. This is for text files where the number appears on the next or previous line of text.

For the asset pattern, the default pattern is "(Total assets) | (Total\s+ [Ll]iabilities\s+ and) | (Total\s*\d+)". The first element of this pattern is used to select lines of text that feature the label "Total assets". The second element of the pattern is used to account for reports where the asset value can't be found, so the equivalent total liabilities and equity value is substituted. The third element of the pattern is used for reports where the total assets line consists of only the word "Total" followed by some whitespace and the value.

Deriving the default equity pattern was not as clear-cut as the cash and asset patterns. Both of the previous patterns were created using common patterns between companies, but there is a lot of variation in the equity pattern between companies. The default equity pattern chosen is "([Mm]ember.? | Owner.? | [Ss]tockholder.?) [s*'*]*\s+ (([Ee]quity\s*\d*)

| (equity . \d+))". Unlike the cash and asset patterns, which were comprised of multiple smaller patterns, the equity pattern is a single expression. The first part of the pattern, "([Mm]ember.? | Owner.? | [Ss]tockholder.?) [s*'*]*\s+", will select lines that start with either "Members", "Owners", or "Stockholders". These are the three different labels used to identify the equity value in a report. The second part of the pattern, "(([Ee]quity\s*\d*) | (equity . \d+))", select the three common possible endings to the desired line of text. Respectively, these patterns account for lines with no numbers and lines with a single character between equity and the number.

Pdftools The package Pdftools is useful for extracting text from PDFs, but it only works with digital PDFs. For scanned PDFs or other document types, other packages such as Tesseract must be used. The reason Pdftools does not work with images is because the core library Pdftools is built on extracts the text data directly from the file. With images, the text data is not encoded into the file and therefore cannot be automatically extracted.

Tesseract The Tesseract package, which we used for this project, can be a very powerful tool for extracting text from scanned PDFs. However, it still has some limitations. It can perform poorly when there is a significant amount of noise in the image quality, or when the image is not prepossessed well, e.g. when a scanned PDF is shown at a certain angle. Therefore, parameter changes have to be made manually. Moreover, not much documentation currently exists to explain the values of the underlying parameters, so it can be hard to understand the factors of the processing rate of the package and to adjust some of these significant parameters when needed.

A possible solution to fixing some of the inaccuracy of Tesseract could be to train a new language model specifically for these PDF reports. While not feasible to accomplish during this project, a future project could focus on solely this aspect of Tesseract. It would involve creating and manually labeling a full set of training and testing data from past X-17A-5 scanned reports. This project could also include in-depth testing and documentation of the different parameters available within Tesseract.

3.2.4 Results

On average, the script takes about 15-25 minutes to download and process the reports of a single company. Given the small group of 13 companies we tested with, it would take about 5.5 hours to run the process for all of those companies. Since the goal is to continue adding companies to this process, we recommend that the code be run overnight or over the weekends once each fiscal quarter, in order to have more efficient time usage and to account for different companies ending their fiscal year at different times. The code could also be adjusted so that companies that have already been processed would not be reprocessed, which would reduce execution time.

By completing this goal and extracting subsets of information from a report, we have streamlined a part of the process an investment analyst must go through when considering a future investment. Instead of manually searching through years of reports to put together a timeline of the overall value of a company, analysts can use this program to do the same task in minutes. This is a huge time-saver, especially if the program is run during off hours. This would mean that the analysts can immediately begin the analysis on the data, rather than having to format it themselves.

4 Conclusion

At the conclusion of this project, all of our desired goals were met. Through the first sub-project, we created a central, cloud-based database of research documents, with the ability to easily add more companies. We also provided the basis for a search feature to be built upon and implemented in the future. Through the second sub-project, the other goal of extracting specific values from detailed reports was accomplished. The script for extracting values can be adapted in the future to handle different fields or filing types of documents with ease. Going forward with this project will be beneficial to the firm in providing guidance on the decisions of their future investments.

References

[1] Angelo Gordon. History - Angelo Gordon. https://www.angelogordon.com/about/history/, 2018. Accessed: 2018-11-06.

[2] FinDox. FinDox. https://www.findox.com/, 2018. Accessed: 2018-12-10.

[3] SEC. SEC.gov | EDGAR | Search Tools. https://www.sec.gov/edgar/searchedgar/webusers.htm, 2018. Accessed: 2018-11-07.

[4] SEC. Form 10-Q. https://www.sec.gov/files/form10-q.pdf, 2018. Accessed: 2018-12-08.

[5] CFR. Disclosure controls and procedures. https://www.law.cornell.edu/cfr/text/17/229.307, 2003. Accessed: 2018-12-07.

[6] SEC. Form 8-K. https://www.sec.gov/files/form8-k.pdf, 2018. Accessed: 2018-12-08.

[7] SEC. Form 12b-25. https://www.sec.gov/files/form12b-25.pdf, 2018. Accessed: 2018-12-08.

[8] SEC. SEC.gov | Using EDGAR to Research Investments. https://www.sec.gov/oiea/Article/edgarguide.html, 2018. Accessed: 2018-12-08.

[9] SEC. Accessing the U.S. Capital Markets — A Brief Overview for Foreign Private Issuers. https://www.sec.gov/divisions/corpfin/internatl/foreign-private-issuers-overview.shtml, 2017. Accessed: 2018-12-08.

[10] SEC. FORM 20-F. https://www.sec.gov/files/form20-f.pdf, 2018. Accessed: 2018-12-08.

[11] SEC. Form X-17A-5 Part I. https://www.sec.gov/files/formx-17a-5_1.pdf, 2018. Accessed: 2018-12-08.

[12] SEC. Form X-17A-5 Part II. https://www.sec.gov/files/formx-17a-5_2.pdf, 2018. Accessed: 2018-12-08.

[13] SEC. Form X-17A-5 Part III. https://www.sec.gov/files/formx-17a-5_3.pdf, 2018. Accessed: 2018-12-08.

[14] Microsoft. What is Azure - Microsoft Cloud Services - Microsoft Azure. https://azure.microsoft.com/en-us/overview/what-is-azure/, 2018. Accessed: 2018-11-07.

[15] Microsoft. Azure Databricks - Microsoft Azure. https://azure.microsoft.com/en-us/services/databricks/, 2018. Accessed: 2018-11-07.

[16] Microsoft. What is Azure Search - Microsoft Docs. https://docs.microsoft.com/en-us/azure/search/search-what-is-azure-search, 2018. Accessed: 2018-11-07.

[17] Jeroen Ooms. Text Extraction, Rendering and Converting of PDF Documents. https://cran.r-project.org/web/packages/pdftools/pdftools.pdf, 2018. Accessed: 2018-11-13.

[18] Vaibhaw Singh Chandel. Deep Learning based Text Recognition (OCR) using Tesseract and OpenCV. https://www.learnopencv.com/deep-learning-based-text-recognition-ocr-using-tesseract-and-opencv/, 2018. Accessed: 2018-11-19.

[19] Sefik Ilkin Serengil. The Math Behind Neural Networks Learning with Backpropagation. https://serengil.wordpress.com/2017/01/21/the-math-behind-backpropagation/, 2017. Accessed: 2018-12-12.

[20] Suryansh S. Gradient Descent: All You Need to Know. https://hackernoon.com/gradient-descent-aynk-7cbe95a778da. Accessed: 2018-12-12.

[21] Hafidz Zulkifli. Understanding Learning Rates and How It Improves Performance in Deep Learning. https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10, 2018. Accessed: 2018-11-28.

[22] Sebastian Moncada. THE VANISHING GRADIENT PROBLEM. https://www.superdatascience.com/recurrent-neural-networks-rnn-the-vanishing-gradient-problem/, 2018. Accessed: 2018-12-03.

[23] Christopher Olah. Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015. Accessed: 2018-11-19.

[24] Avinash Sharma V. Understanding Activation Functions in Neural Networks. https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0, 2017. Accessed: 2018-12-03.

[25] Rohan Kapur. Rohan & Lenny #3: Recurrent Neural Networks & LSTMs. https://ayearofai.com/rohan-lenny-3-recurrent-neural-networks-10300100899b, 2017. Accessed: 2018-12-07.

[26] Eugine Kang. Long Short-Term Memory (LSTM): Concept. https://medium.com/@kangeugine/long-short-term-memory-lstm-concept-cb3283934359, 2017. Accessed: 2018-12-07.

[27] Rahul Vernwal. How should I start with CNN? https://becominghuman.ai/how-should-i-start-with-cnn-c62a3a89493b, 2017. Accessed: 2018-12-10.

[28] Derrick Mwiti. A Beginner's Guide to Convolutional Neural Networks (CNN). https://heartbeat.fritz.ai/a-beginners-guide-to-convolutional-neural-networks-cnn-cf26c5ee17ed, 2018. Accessed: 2018-11-19.

[29] Dan-Ching Liu. A Practical Guide to ReLU. https://medium.com/tinymind/a-practical-guide-to-relu-b83ca804f1f7, 2017. Accessed: 2018-12-03.

[30] Piotr Skalski. Preventing Deep Neural Network from Overfitting. https://towardsdatascience.com/preventing-deep-neural-network-from-overfitting-953458db800a, 2018. Accessed: 2018-12-03.

[31] David Stutz. Understanding Convolutional Neural Networks. https://davidstutz.de/wordpress/wp-content/uploads/2014/07/seminar.pdf, 2018. Accessed: 2018-12-11.

[32] the data science blog. An Intuitive Explanation of Convolutional Neural Networks. https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/, 2016. Accessed: 2018-12-11.

[33] ImageMagick. Magick++ API. https://www.imagemagick.org/Magick++/, 2017. Accessed: 2018-12-01.