Self-Evolving Data Collection Through Analytics and Business Intelligence to Predict the Price of Cryptocurrency

A dissertation presented to

the faculty of

the Russ College of Engineering and Technology of Ohio University

In partial fulfillment

of the requirements for the degree

Doctor of Philosophy

Adam C. Moyer

December 2020

This dissertation titled

Self-Evolving Data Collection Through Analytics and Business Intelligence to Predict the Price of Cryptocurrency

by

ADAM C. MOYER

has been approved for

the Department of Industrial and Systems Engineering

and the Russ College of Engineering and Technology by

Gary R. Weckman

Professor of Industrial and Systems Engineering

Mei Wei

Dean, Russ College of Engineering and Technology

Abstract

MOYER, ADAM C., Ph.D., December 2020, Ph.D. in Mechanical & Systems Engineering

Self-Evolving Data Collection Through Analytics and Business Intelligence to Predict the Price of Cryptocurrency

Director of Dissertation: Gary R. Weckman

The development of the self-evolving data collection engine through analytics and business intelligence (SEDCABI) research engine along with plug-in prediction module (PPM) is demonstrated for the prediction of cryptocurrency (specifically, Bitcoin). Leveraging all data proves increase the accuracy of the prediction when compared to using only structured data, or only using unstructured data alone.

Dedication

To my loving, caring, and extremely supportive wife who encouraged and supported me through this endeavor.

Table of Contents

Page

List of Tables

Page

List of Figures

Page

Introduction

Cryptocurrency can be thought of as currency; however, it exists in digital form and is a relatively new way of representing and storing transactions using cryptography, "the enciphering and deciphering of messages in secret code or cipher" (Cryptography, 2019), in a decentralized model (ElBahrawy, 2017). As of September 2020, more than 3700 cryptocurrencies exist, "Bitcoin" is the most popular and commonly known (Investing.com, 2020). Given that we are just over 10 years from the start of actual usage of the first cryptocurrency, there is still a lot of room from improvement. The cryptocurrency landscape is complex at best, the terminology is confusing, and the cryptocurrencies have proved to be volatile making prices tough to predict.

Based on this large number of cryptocurrencies that have emerged over a short period of time, breaking them down into smaller categories gives a better idea of how cryptocurrencies which started as a mechanism for financial transactions has evolved to be much more. "There are different sectors in the stock market and it is important to understand each sector because their prices tend to move together. Likewise, you need to be aware of cryptocurrency sectors in order to understand how new innovations, laws and public interest will affect similar cryptocurrencies (Kimura, 2019)."

The decentralized aspect of cryptocurrency is what separates it from the common currency that everyone has come to know. That, and cryptocurrencies will "generally have a fixed supply and, therefore, the devaluation of cryptocurrencies through inflation is mostly nonexistent (Pauw, 2018)." No central authority that manages, controls, issues or removes cryptocurrency exists; instead, cryptocurrency is based on a decentralized

system (meaning no one controls it) referred to as “blockchain” (Clements, 2018). A cryptocurrency’s blockchain is a public list of all transactions that have ever occurred, a “distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way (Iansiti, 2017).”

While the aspect of a “decentralized system” makes cryptocurrency popular, it also helps contribute to the unpredictability of the daily price of cryptocurrency. Existing literature has been limited to using primarily structured data for price predictions. The few published articles that have branched into leveraging unstructured data in their predictive models on cryptocurrency price have focused primarily on forum related data and limited social media.

Collecting and analysis of unstructured data has been difficult due to the nature of what makes data unstructured – there is no structure for how the data may be stored and presented (e.g. Web, PDF, Word, along with many other random formats). A data collection engine will be developed as a type of automated data gathering tool to ensure not only constant collection of structured data but also the constant collection of unstructured data with self-evolving characteristics of the data that it looks for.

A formal methodology that self-evolves data collection and pre-processes both structured and unstructured data would be valuable not only for the scope of this research but extend to any research. Add to that a framework that combines the unstructured analysis output with the more traditional structured datasets leveraging plug-in predicative module, and it may be possible to predict cryptocurrency prices with greater accuracy by leveraging “all data” or, at least, comparisons can be made between

predictions made using combined structured/unstructured data and unstructured data alone for cryptocurrency price prediction which has not been attempted before based on current literature.

Deficiencies

Based on existing literature, most work has focused on utilizing traditional statistics and machine learning on structured cryptocurrency market data to be able to make predictions about cryptocurrency prices. Further review shows that very limited work has been completed on combining structured and unstructured data sources to make cryptocurrency price predictions. Of that work, the unstructured data sources themselves have been very limited in scope (e.g. a single social media source, a single source for news, a couple of forums). No current published research exists utilizing unlimited unstructured data combined with structured data for predicting cryptocurrency prices.

There appear to be issues (or at least questions) in the ranges of dates in a couple of datasets. I speculate that this is due to Twitter having increased restrictions in place (Roth, 2018), as do many social media outlets which have only grown more restrictive since the fallout of the 2016 election and the prevalence of "fake news." For example, 5 years ago, many social media platforms provided free (and surprisingly, even anonymous) access to their application programming interface (API) that would allow you to work with their databases with access to more data than they really should have been allowed. In some cases, that may have been unintentional, and in others, some took advantage of that fact like Cambridge Analytica (Granville, 2018).

In the few cases where unstructured data was obtained, there is one (Mai, 2018) where forum data was collected from Bitcointalk.org January 1, 2012, through December 31, 2014, but Twitter data only was collected September 16, 2014, through December 16, 2014, it is assumed the cryptocurrency data was collected over the longer period of time (this is not mentioned). It is unclear how that compressed timeline data was leveraged over the longer timeline data for the other datasets.

In another case (Steinert, 2018), there are described gaps in the collected social media data due to restrictions in the API (Twitter again). This means the data collected was not contiguous (for example, you could collect Tweets at 8am, 12pm, 3pm, 5pm and suggest that represents the work day, but it might have tendencies to reflect whatever people talk about at lunch depending on the time zone).

Of those using sentiment analysis (everyone who used unstructured data did this), all except one (Kim Y. B., 2016) used simplistic techniques when applying the sentiment scoring process (only counting the number of times a word was positively matched with a lexicon). It is not that this is wrong; it potentially does not correctly classify the type of opinion. For example, “opinions” vs “comparative opinions” vs “suggestive opinions” which can mislead the sentiment score used in the model (Qazi, 2017).

The issues some of the authors faced, and likely many more to come is related to data access. The dates issues identified above are related to a more restrictive mentality (which is a good thing, to an extent) across companies when to comes to data governance and how/when/where and who can access the data. Outside of healthcare and a few select

industries, many companies with public-facing data, have typically made it easy to access that data. There has been a growing trend to slow or significantly halt that process.

The proposed research utilizes a self-evolving data collection engine (SEDCABI) to vastly expand on the limited data used by all existing research. The existing research leveraged social media (Twitter only), forum data, and financial indices. This data will be expanded to include economic indicators, news, and regulatory policy (the majority of which will come from unstructured data). A very large data set will result from SEDCABI. It is hypothesized that analysis of this very large combined structured and unstructured data set will result in a better predictive ability about the patterns of cryptocurrency prices.

Hypothesis

The hypothesis of this research is that combining the traditional data sources, the unexplored aspects of incorporating unlimited unstructured data sources (including but not limited to regulatory policy, news, finance, social media, blogs, forums, etc.) along with the self-evolving data collection functionality of the proposed SEDCABI research engine can lead to better predictions of cryptocurrency prices, specifically Bitcoin. It is speculated that using structured with unlimited unstructured data sources will provide an accurate prediction of cryptocurrency prices.

Objectives

The overall objective for this research is the development of the self-evolving data collection engine through analytics and business intelligence (SEDCABI - which should be much more far-reaching than previous research and be a unique and valuable

contribution to the research community) along with plug-in prediction module (PPM) which could lead to better price prediction of cryptocurrency (specifically, Bitcoin). The PPM will be built with anticipation of it having the potential to be interchangeable with other techniques and other predictive goals in mind depending on the research it is applied to. SEDCABI should help researchers more easily incorporate automated unstructured data collection and analysis into their own research.

Significance

A formal methodology implemented into a re-useable tool that incorporates the self-evolving data collection through analytics and business intelligence (SEDCABI) research engine to work with all data (not just structured data) would be a benefit to researchers everywhere (e.g. you point it in the direction in which you would like to collect data about and the output of the tool is a preprocessed dataset primed for the analysis using whatever predictive [or other] technique the researcher wants to investigate). Additionally, this methodology will make working with unstructured data easily accessible to all researchers without a prerequisite of understanding how to numericize unstructured data so that it can be consumed by the plug-in predictive module (PPM), a custom implementation of the PPM, or their own methodology.

The goal and significance of this research is in SEDCABI, not the predictive technique used for the price prediction of Bitcoin in the PPM. As of this research, nobody has published any literature that has leveraged potentially all unstructured data sources as a predictor of cryptocurrency price. Formally combining potentially any unstructured data source with structured data for price prediction should demonstrate the usefulness of the

SEDCABI research engine. If cryptocurrency prices could be made more predictable, there can come stability which could help to reduce the “wild west” aspects of cryptocurrency and allow for a better and quicker global acceptance as a transition into a legitimate global currency.

Literature Review

Based on the relatively short time that cryptocurrency has been used, the stability and predictability of the cryptocurrency markets have proven tough to forecast. The objective of my literature review will explore existing and new techniques for using non-traditional unstructured social data sources, along with structured data sources to predict features of cryptocurrency.

The following section provides information about the articles directly related to cryptocurrency prediction on structured and unstructured data. There is incredibly limited work that has been published based on both structured and unstructured datasets. With the cryptocurrency market being relatively young compared to the traditional currency markets, it would be of great importance to understand the factors that could be leveraged in or to maximize investment opportunities. Given that there has been so little work incorporating both unstructured and structured data by means of knowledge discovery to an end of being able to successfully predict aspects of the cryptocurrency market, this review sets out to explore the literature that has attempted, both successfully and not, to make predictions about the cryptocurrency marketplace.

For the scope of this review, data will be classified as structured or unstructured. Structured will refer to data that may fit into columns and rows based on an expected structure (e.g. a list of employees, hire dates, salary, a list of Bitcoin prices, etc.), and unstructured will be data that does not have a pre-defined or expected structure (e.g. a Word document, text message, Tweet, news article, Facebook message, etc.). The articles that were reviewed could be categorized by the type of cryptocurrency (or

cryptocurrencies) analyzed, the data that was used (for example structured or unstructured), the time periods for and over which the data was collected, the type of prediction(s) being made.

Across the board, all were using some form of machine learning algorithms to help with data mining, but surprisingly only a few were using the machine algorithms in ensembles (Sun, 2018) (Alessandretti, 2018) (Mallqui, 2019). Also noting that those that did use ensemble techniques only did so on structured data sets. Working with unstructured data is challenging since, well, it lacks any predefined structure. How do you quantify the Gettysburg address? By number of words, paragraphs, psychological motivators, verbs, nouns, etc.? A prevailing technique is sentiment analysis (Qazi, 2017); however, there are many variations in the literature and variations in the implementations, but some appear to be lacking (Mai, 2018) when compared to those that have attempted to integrate machine learning into the mix (Kim Y. L., 2017).

Sentiment analysis (also known as opinion mining) was the primary tool for working with unstructured data (which is also very common for unstructured data). Techniques ranged from very basic (Mai, 2018), to chaining sentiments for multiple classifications (Kim Y. B., 2016) (Mallqui, 2019), and on the advanced side a hybrid approach which integrated topic modeling and concept building (Kim Y. L., 2017). This last one is by far the most novel idea seen in the literature for working on unstructured data. Certainly, an opportunity here for growing future work.

Going back to findings in the literature, when looking at the predictions, the price of the cryptocurrency was most common; actually, all except one (Demir, 2018). Some

branched out into variations including the direction of the price (Sun, 2018), the min/max/close of the price (Mallqui, 2019), and other numeric properties of the cryptocurrency. One used economic uncertainty index (ICU) as a mechanism to correlate with Bitcoin and found a positive correlation which seemed to be the most out of the box think observed in the review.

Some other findings of interest:

- Using Light Gradient Boosting Machine (LightGBM), SVM (Support Vector Machines), and Random Forests (RF), they found that using LightGBM was the best, and that "for a given cryptocurrency, the higher its comprehensive strength, the better the forecasting performance obtained (Sun, 2018)." The key finding is that they found the "forecasting performance is correlated to the forecasting period and also to the competitiveness of the cryptocurrency (Sun, 2018)."
- Models consisting of simple moving averages (SMA), XGBoost (for regression trees), gradient boosting decision trees (GBDT), and long short-term memory (LSTM) recurrent neural networks were all tested. Although the details were limited, they found that all models perform better than SMA, and "that the method based on long short-term memory recurrent neural networks systematically yields the best return on investment (Alessandretti, 2018)."
- Using deep learning neural networks (DLNN) from implementing LSTMs and generalized regression neural networks (GRNN), and found that the deep learning LSTM neural networks performed better than GRNN in predicting the future price values of Bitcoin, Digital Cash and Ripple. (Lahmiri, 2019)

- An interesting observation made that could play into future work is that “a forum is designed to be an archive of all messages; by design, Twitter focuses more on timeliness. It is not uncommon for forum users to engage in a discussion that was started days or months ago, whereas the average life cycle of a tweet is much shorter, and it is difficult for users to trace earlier tweets from an active account. (Mai, 2018)”

Overall, I found only four cases where unstructured and structured data were both used for prediction. So, that is a big limitation on its own. The benefits were in the algorithms or techniques used. The work (Kim Y. B., 2016) where Valence Aware Dictionary and sEntiment Reasoner (VADER) was used in conjunction with the Granger causality test (GCT) to determine whether the time series of a community of opinions contained predictive information regarding the fluctuations in cryptocurrency prices would be beneficial. Other items of benefit include techniques used for feature selection (Mallqui, 2019), success with Long Short-Term Memory (LSTM) (Alessandretti, 2018), and the sentiment analysis hybrid approach integrating topic modeling and concept building (Kim Y. L., 2017).

Methodology

System Components

To help identify acronyms used, refer to Table 1.

Table 1

Acronyms used in the Methodology

Acronym	Meaning
ABI	Analytics and Business Intelligence
API	Application Programming Interface
ATST	Automated Transposition Storage Technique
CAPTCHA	Completely Automated Public Turing Test To Tell Computers and Humans Apart
DDoS	Distributed Denial of Service
ETL	Extract Transform Load
GoF	Goodness of Fit
MLR	Multiple Linear Regression
PPM	Plug-in Prediction Module
RMM	Rolling Model Method
SA	Sentiment Analysis
SEDC	Self-evolving Data Collection
SEDCABI	Self-evolving Data Collection through Analytics and Business Intelligence (The Research Engine)

Table 1: continued

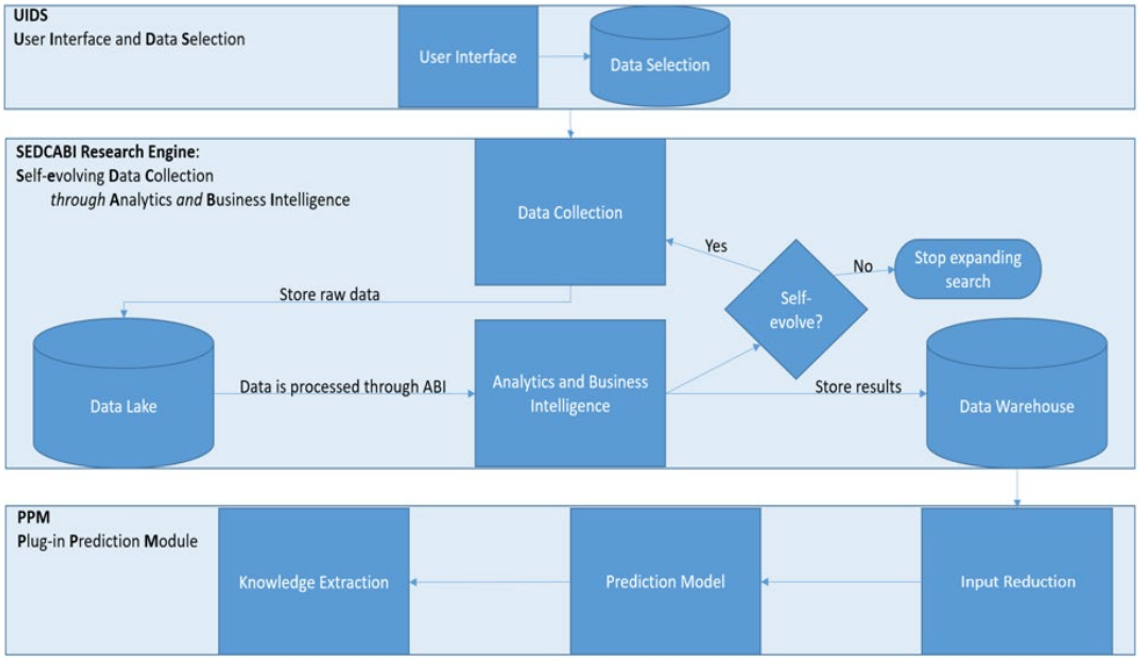
SQL	Structured Query Language
SSMS	SQL Server Management Studio
TDM	Term-Document Matrix
UIDS	User Interface and Data Selection
VADER	Valence Aware Dictionary and sEntiment Reasoner

A self-evolving data collection through analytics and business intelligence research engine (SEDCABI) was constructed to collect and store all types of data. There are no restrictions on where the data can come from based on how SEDCABI evolves it searches other than it somehow includes “Bitcoin” in the search criteria. All this aggregated data is fed into the data lake where it is stored in its raw form. From there the data will pass through the analytics and business intelligence (ABI) component. Cleaning, sentiment analysis, and topic extraction occur here for preprocessing and producing a data set which will be numericized making it suitable for statistical techniques. This process results in the construction of the data warehouse. From there, data is passed to the plug-in prediction module (PPM). At the start of the PPM, multiple linear regression (MLR) using the stepwise method is used to reduce the number of attributes from the data warehouse into inputs that are used to predict the cryptocurrency price. Finally, R^2 along with goodness of fit prediction interval (GoF) was used to evaluate the predictive model of the PPM, and simulation is presented to summarize the

usefulness of the PPM. The overall methodology, seen in Figure 1, is made up of three main components.

Figure 1

Overall Methodology



A basic user interface (UIDS: the User Interface and Data Selection component) to allow for the input of what data will be collected by the SEDCABI component. This allowed for the input of targeted URLs, social media, forums, search terms, a list of datasets with API access for which the SEDCABI engine has bots available for unsupervised collection.

Once the target data is identified through the UIDS component, an automated process (SEDCABI: the Self-evolving Data Collection through Analytics and Business Intelligence component) controlled by the data collection component initiates the

appropriate collection bots. Collection bots are designed with either general or specific data collection in mind depending on if they are targeting structured or unstructured data. For example, data can be classified as either structured or unstructured. Structured will refer to data that may fit into columns and rows based on an expected structure and unstructured will be data that does not have a pre-defined or expected structure.

Collection bots for structured data are categorized as ones that pull data through an application programming interface (API). This is very common and an example would be pulling financial exchange data where you already know what the columns of data look like and simply need to query the company's API in order to extract data that gets neatly stored in columns and rows.

Known structure bots that are targeting structured data require some initial knowledge of what the structure looks like, so the attributes do not lose any of their relational integrity about what the data represents. For example, when capturing financial data that represents the close of day stock prices that has 10 columns (attributes) and 100 rows of data and is associated with a specific capture time, that structure needs to be identified and a basic parsing schema is built and utilized by the collection bot. Once that schema is in place, it can be re-used for future collections from that specific target dataset.

Using Twitter's API, for example, where a structure may exist (e.g. date of post, user that posted, what was posted, geographic data about where the user posted from, how many likes the post received, how many followers the user has), "what was posted" would be unstructured, where the date and user would be structured data. In the cases

where unstructured data is either expected (or could be identified by performing basic analysis on the data, like “is it numeric” and if not, “have we previous categorized the attribute as structured” and if not, either treat as unstructured, or flag it for review, and eventual classification). In the following example, all columns are structured except for “user.post” (e.g. a “Tweet”) which contains data with no expected format as seen in Figure 2.

Figure 2

Example of Twitter Data as seen in SSMS

user.name	user.post	score	howManyWords	user.followers_...	user.friends
DayTradeCurre...	@dogcowuk The #Facebook #LibraCoin wi...	3	28	2	0
A. S. U. K	Facebook to Launch Testnet for Its Libra Cr...	0	16	30	51
Jimmy Neutron	I remember back in late 2017 early 2018 Fac...	0	27	63	186
Mahmudyunus	Reject Facebook And Libra: An Open Letter...	2	25	4	49
it Excellence	@LibraReserve Looooooooool @R u guys ...	0	17	45	139
Kazuhiro Sasaki	Libraね。え、なんでもっとFacebookコインでみん...	0	10	14	49
کاتب محتوی تسو...	@haythm88 @eng_mustafa1991 @Mkt_Art...	0	12	2428	8

Collection bots for unstructured data do not require any foreknowledge of the target, or even what the target is until the time instructed to collect the data. Like the “user.post” listed above, unstructured data could be a speech, research documents, news, and anything that does not have a predefined structure allowing for numerical analysis. To use the unstructured data, we apply techniques that allow us to numericize the data so that it can be used in statistical models. This is where text analysis will come into play. Once the bots were constructed, the data collection engine uses the initially provided requests for data collection to decide what it will collect via the automation framework

(think something like a task scheduler). The output of the data collection framework will be stored in the data lake as raw data.

This is where the data is preprocessed through the analytics and business intelligence (ABI) component of SEDCABI. This process encapsulates the logic that is used for the decision making required to address the identification and handling of missing values and compression of the data (described below). Additionally, ABI will address text analysis. Using sentiment analysis, along with other summarization techniques (e.g. turning date-related data into a single numeric value so it can be consumed by MLR), the goal here will be to produce numerical data from the unstructured data along with identifying possible topics to be passed back to the start of SEDCABI for additional data collection. This is the self-evolving aspect of the SEDCABI, which comes into play once ABI decides if the SEDC component needs to be re-invoked based on what it finds as topics are identified that have not been explored (*or explored enough*).

The numerical data from the text analysis will be a result of the topic extraction and classifications resulting from provided positive/negative lexicons (literal dictionaries with either positive or negative terms). Once any additional preprocessing of the data occurs (*like decisions about how time will be applied to the data*), the SEDCABI will feed into the next component, the plug-in prediction module (PPM).

The idea behind the Plug-in Prediction Module (PPM) is to allow for flexibility in accommodating for various predictive methodologies. The fact that there are are many different techniques that can be used to help understand relationships between the

attributes of data that we have collected in addition to understanding how those attributes can help predict something. The process generally sounds the same even though there are different details in techniques that are used to achieve this. So, the PPM is designed with the idea that it could be re-tooled to use other techniques and models for other predictive (or prescriptive) goals or for purposes of comparing techniques when exposed to the same dataset.

Details: Orchestration of the Overall Process

There are 20 main code components that when combined with the automated collection bots total about 40 different programs that are running 24 hours a day seven days a week to deal with all aspects of data collection including generating predictive models and applying those models to the data. These components collect structured and unstructured data from a lot of sources (details later), deal with various levels of preprocessing the data after it is collected in terms of dealing with aspects of unstructured and structured data, and perform aspects of predictive model creation and evaluation, and simulation. The entirety of the codebase took approximately six months to write, debug, and test before all components were orchestrated as necessary to be able to see the entire process from start to finish in full automation. Table 2 provides details about the software used throughout the entire orchestration of the system.

Table 2

All Software Libraries and Packages Used in the Methodology

Libraries/Packages (and Platform)

Table 2: continued

Base64 (Python)

Bitfinex (Python)

Bs4 (Python)

Datetime (Python)

Errno (Python)

Importlib (Python)

Microsoft.Office.Interop.Excel (C#)

Nltk (Python)

Numpy (Python)

Os (Python)

Pandas (Python)

Pickle (Python)

Pyodbc (Python)

Pyodbc (Python)

Random (Python)

Re (Python)

Requests (Python)

Shutil (Python)

Table 2: continued

Sklearn (Python)
SQL Server (Database)
Statsmodels (Python)
String (Python)
System (C#)
System.Data.SqlClient (C#)
Time (Python)
Tweepy (Python)
Unidecode (Python)
Urllib.parse (Python)

Technically speaking, if there were no limits on resources the number of collection bots could be scaled up significantly; however, operating as an individual, and doing this with limited network and general terms of use restrictions based on the way data can be pulled from various locations, a predetermined limit was set on the collection bots ability to do work on their own. One of the big issues in terms of data collection that had to be overcome dealt with the idea of rate limits. For example, if you were to make a request for data from a website and you do it once an hour you likely are not going to get immediately blocked; however, if you were to make several calls for that data many times per second, either the target where you are collecting data from or the network that

you might be passing through anywhere between you and the target may see that as a denial of services (DOS) attack which is a common method for hackers to try to bring down a network resource.

With that in mind and referring to rate limiting, the collection bots had random aspects of time injected into how they would delay recording and capturing data to allow them to successfully be able to do their job without unintentionally or intentionally being blocked throughout the collection process.

What the process really looks like was a little more complicated as the system evolved from where originally intended. For example, as I got into the programming of the initial capture engine and started looking at how the data could be stored, all data can always be stored in an unstructured format; however, it does not make sense to do that if you know that there is already structure within the data that you are going to target to be able to use for predictive means as there was in this case since the goal of the plug and play predictive module was set out to predict the price of Bitcoin. As a result, I scaled back the resources that were used for the entirety of the collection so I could focus on demonstrating how this theoretical process could work from start to finish. While there were no restrictions placed on where data would come from overall, the structured collection bots specifically targeted 2 locations, 1 popular social media outlet, and one major Bitcoin exchange.

Originally, a process was put together that would allow for an automated transposition storage technique of the data (ATST) which would allow any targeted structured data to be recorded over time and if the data set expanded or contracted in

terms of the number of attributes or potential inputs that might be used in the predictive model, it would still be able to record the data without any modifications to the database. Here is an example of what ATST looked like. In Figure 3, on the left, you have the initial structured dataset; and on the right is the transposed dataset with the original relational integrity of the dataset intact.

Figure 3

Example of Static Structured Data (left) Stored in a Table Using ATST (right)

Target somedataset.fake
Collection Time d/m/y h:m (1)

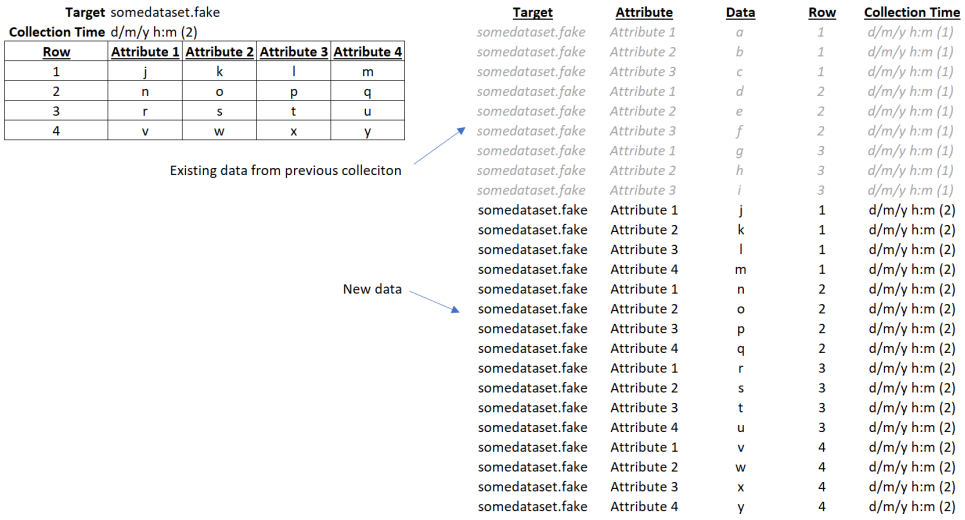
Row	Attribute 1	Attribute 2	Attribute 3
1	a	b	c
2	d	e	f
3	g	h	i

Target	Attribute	Data	Row	Collection Time
somedataset.fake	Attribute 1	a	1	d/m/y h:m (1)
somedataset.fake	Attribute 2	b	1	d/m/y h:m (1)
somedataset.fake	Attribute 3	c	1	d/m/y h:m (1)
somedataset.fake	Attribute 1	d	2	d/m/y h:m (1)
somedataset.fake	Attribute 2	e	2	d/m/y h:m (1)
somedataset.fake	Attribute 3	f	2	d/m/y h:m (1)
somedataset.fake	Attribute 1	g	3	d/m/y h:m (1)
somedataset.fake	Attribute 2	h	3	d/m/y h:m (1)
somedataset.fake	Attribute 3	i	3	d/m/y h:m (1)

Next, after collect data from the same target; however, the target dataset has evolved in some way. For example, an additional attribute has been added to the dataset (attribute 4) and we want to be able to store that larger dataset using ASTS would look like the example provided in Figure 4.

Figure 4

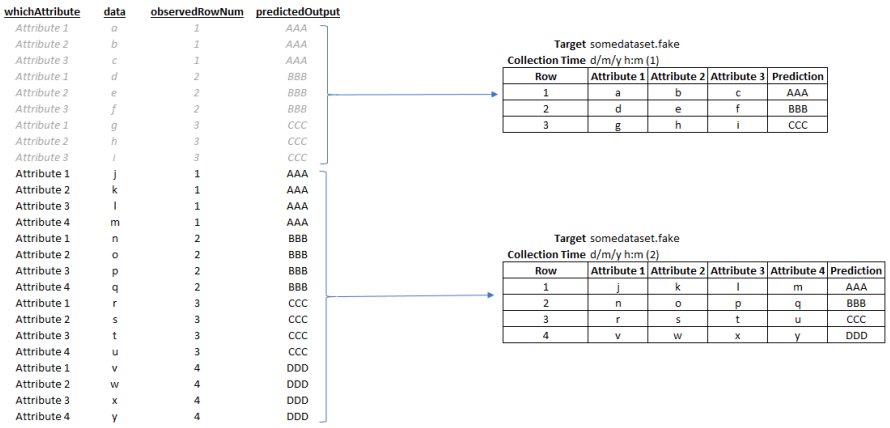
Example of Structured Data with Additional Attribute Beyond Original Data



However, the extract/transform/load (ETL) process used to manipulate that data proved to be more time-consuming from a computer resource perspective to make sense back out of the data than it did to simply create a structured database to capture the data in the first place. Figure 5 demonstrates this.

Figure 5

Reversing ATST to Create the Original Datasets



Ultimately, I backtracked on the ATST code that I put in place and replaced it with bots that were designed specifically to capture the data from the locations that they were targeting. In the end, this made it much easier to be able to relate the structured data to the unstructured data and automate the process of being able to dynamically create predictive models.

The bots designed to do the unstructured data collection, while taking longer to program, initially are simplistic since they do not do any preprocessing of the data. They are directed by a collective component based on search criteria that were initially seeded through a basic UI component and automatically evolved through the SEDCABI engine. Their general role was to do nothing more than to go out to any web-based data location, retrieve and store all data that they found.

User Interface

The UI is very minimalistic, although, for future work, I can see it evolving extensively based on the understanding of what data is collected throughout the entire process and ultimately how the data interrelates within the various tables. At this stage of the system, the only data that had to be entered for everything to come to life revolved around the word “Bitcoin.” From the social media collection perspective, the only search criteria that had to be met in the data is that it in any way included the word “Bitcoin.” In terms of the unstructured search criteria, there were four seeds initially planted which the unstructured bot used to start the initial collection process terms: “Bitcoin,” “Bitcoin news,” “Bitcoin went to buy,” and “Bitcoin went to sell.”

This is the only time the user of the system absolutely must do any work which is at the very beginning of describing what the data collection is going to be (what feeds into the start of SEDCABI). This basic interface simply stores those search criteria and from there the business intelligence and analytics component of the SEDCABI engine takes over.

A significant amount of data gets generated and recorded around what search criteria to let into the system. As a result, for future work one possible consideration would be identifying search criteria that may have a larger impact in terms of the effectiveness of the predictive model. This has not been implemented as a part of the process and all data is used in the automated creation of the predictive models.

Collection Bots

Data is constantly being captured by the SEDCABI engine from multiple locations. This includes social media and any web-based data location. The bots, whether they are structured or unstructured collection bots, are constantly running with various delays in mind that help prevents the mechanisms from being blocked from collecting content. With Twitter data, for example, maximum collection times are dictated by Twitter's API and you can only make so many calls for data in any given 15-minute window. For unstructured data, it is a bit different. There are typically 20 bots that oversee an unstructured collection of data and at any given time one bot might oversee collecting data anywhere from 25 to over 100 different sites (~per hour). They do not know where the data is going to be collected from until after leveraging multiple major search engines that yield results that are both specific to the search criteria and are limited

to the past 24 hours to help make the data collection timely and more relevant to our ability to predict following the compression that occurs on the data described later.

In addition to the existing collection bots, there are also very specifically targeted collection bots designed to pull Bitcoin data from one major public exchange. This helps ensure that we have minute by minute open, close, high, low, and volume Bitcoin data that can be used in the predictive model for the Bitcoin price.

Structured Collection Bots

The structured collection bots require knowledge of the data that they will be collecting in advance of the collection. For example, to collect from a company's API, you need to know the structure of the dataset so that the bot knows what to expect, how to access the data, and how the data will be stored in the SEDCABI database. There is no set standard for what the API will look like since it is up to the individual company to decide who can access the data, what data they will expose, and how it will be accessed. Typically, the company provides some type of API reference that a programmer uses to understand how to communicate with the API. A partial example of how the Twitter API (https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/api-reference/get-search-tweets) request and response can be seen in Figure 6 and Figure 7.

Figure 6

Example of an API Request

```
$ curl --request GET
--url 'https://api.twitter.com/1.1/search/tweets.json?q=nasa&result_type=popular'
--header 'authorization: OAuth oauth_consumer_key="consumer-key-for-app",
oauth_nonce="generated-nonce", oauth_signature="generated-signature",
oauth_signature_method="HMAC-SHA1", oauth_timestamp="generated-timestamp",
oauth_token="access-token-for-authed-user", oauth_version="1.0"'
$ twurl /1.1/search/tweets.json?q=nasa&result_type=popular
```

Figure 7

Example of an API Response


```
{
    "statuses": [
        {
            "created_at": "Sun Feb 25 18:11:01 +0000 2018",
            "id": 967824267948773377,
            "id_str": "967824267948773377",
            "text": "From pilot to astronaut, Robert H. Lawrence was the first African-
American to be selected as an astronaut by any na… https://t.co/FjPEWnh804",
            "truncated": true,
```


The API requires a very specific request from the structured bot and assuming that the request meets the company's API guidelines, then a response is issued with a specified structure that the bot was programmed to understand and use that pre-existing understanding to properly store the data in the structured database. After that response is stored in the database, it is now available as structured columns and rows. For example, social media data looks like the example provided in Figure 8, and a request/response to a Bitcoin exchange would result in something like seen in Figure 9.

Figure 8

Example of Social Media Data Stored by Structured Bot Following the API Response as seen in SSMS

unprocessedText	user.followers_count	user.friends_count	user.listed_count	user.favourites_count	user.statuses_count	user.created_at
@itsmeMharkie NO to both. This is BSV only. ...	1249	336	9	14193	4239	2019-12-26 22:47:26.000
@realcryptobuzzb Ppl shilling decentralizatio...	1356	482	9	2866	2457	2019-12-14 17:56:10.000
PayPal is hiring #Crypto and #blockchain exp...	487	391	3	29581	6960	2019-07-13 10:11:21.000
RT @CryptoBull: #Bitcoin is on the verge of a ...	1326	4828	3	100700	13445	2012-05-11 01:11:43.000
The time is Jun 22 2020 22:58:32 UTC and Bitc...	15	0	0	0	900	2020-05-14 12:13:28.000
RT @ElectrumSV: In order to move forward in...	1004	346	8	1536	40803	2019-05-16 15:48:46.000
LAST TRADE: SELL 0.12078494BTC@8585.3EUR...	7163	25	154	102112	249408	2014-03-17 11:30:26.000
This Technical Factor Suggests Bitcoin Will So...	733	816	2	1646	2527	2018-10-18 15:43:24.000
RT @Cointelegraph: PayPal is hiring crypto pr...	51	151	0	18660	1504	2016-11-06 03:18:55.000

Figure 9

Example of Bitcoin Price Data Stored by Structured Bot Following the API Response as seen in SSMS

timeStamp	open	close	high	low	volume	imputed
2020-06-14 04:00:00.000	9441.1000000000	9440.5838333000	9441.1000000000	9439.8000000000	0.4915954100	False
2020-06-14 04:01:00.000	9440.4712792500	9438.1000000000	9440.4712792500	9438.0281950000	0.2100000000	False
2020-06-14 04:02:00.000	9439.0000000000	9438.9000000000	9440.0000000000	9437.4000000000	0.1597720500	False

Unstructured Collection Bots

After the initial seeding of search terms in order to get the bots working, the collection engine is constantly going out and looking at major search engine results based on the search criteria and then goes out and collects the results from any web-based location. An example of search criteria including the initially seeded search terms along with other search terms generated by SEDCABI, in addition to details about how long to attempt to collect based on the search criteria, can be seen in Figure 10.

Figure 10

Examples of Search Criteria for an Unstructured Bot as seen in SSMS

searchCriteria	requestedDateTime	activeSearch	collectUntilDateTime	lastHeardFromBotDateTime
bitcoin	2020-06-27 16:55:43.167	True	2021-08-29 20:44:27.123	2020-10-01 20:10:28.537
bitcoin news	2020-06-27 16:55:43.167	True	2021-08-29 20:44:27.123	2020-10-01 20:39:08.770
bitcoin when to buy	2020-06-27 16:55:43.167	True	2021-08-29 20:44:27.123	2020-10-01 19:58:45.580
bitcoin when to sell	2020-06-27 16:55:43.167	True	2021-08-29 20:44:27.123	2020-10-01 20:35:04.287
bitcoin original	2020-09-09 23:27:56.123	False	2020-09-10 00:27:56.123	2020-09-10 00:24:12.993
bitcoin stock	2020-09-09 23:27:56.127	False	2020-09-10 00:27:56.127	2020-09-09 23:48:31.870
bitcoin currency	2020-09-09 23:27:56.130	False	2020-09-10 00:27:56.130	2020-09-10 00:00:26.170
bitcoin digital	2020-09-09 23:27:56.147	False	2020-09-10 00:27:56.147	2020-09-09 23:55:23.880
bitcoin say	2020-09-09 23:27:56.157	False	2020-09-10 00:27:56.157	2020-09-10 00:18:20.387
bitcoin company	2020-09-09 23:27:56.160	False	2020-09-10 00:27:56.160	2020-09-10 00:24:04.493

It is here that results in the largest chunks of data collected by the bots. As previously indicated, within a little over two months, collected data included more than 10,000 unique domains (e.g. “nrp.org”), 75,000 web sites (e.g. https://www.npr.org/sections/news/), and 835,000 collections of data from those websites. Once the search criteria and collection time frames are identified, an unstructured bot is tasked to start looking for data by querying major search engine provided (in all cases, they are based in the United States, which could introduce potential bias in the data or not be able to get results from countries that may limit where data can be accessed from, which is something discussed in the Limitations section of this paper). An example of what the unstructured bot might identify to collect looks like is demonstrated in Figure 11.

Figure 11

Example of Sites Identified for Data Collection by an Unstructured Bot

```
current collection attempt started at: 2020-8-01 05:09:43.027085 - searching for "bitcoin+stock"
collecting until: 2020-8-01 05:01:57.630000

found 24 URLs from SE1
https://www.forbes.com/sites/billybambrough/2020/09/30/serious-warning-issued-over-300000-bitcoin-stock-to-flow-price-mc
https://www.newsbtc.com/2020/10/01/why-post-debate-stock-market-performance-could-be-bad-for-bitcoin/
https://otcpm24.com/2020/10/01/serious-warning-issued-over-300000-bitcoin-stock-to-flow-price-model/
https://sharecaster.com/2020/10/01/serious-warning-issued-over-300000-bitcoin-stock-to-flow-price-model/
https://news.bitcoin.com/canadian-firm-3iqs-bitcoin-fund-listed-on-gibraltar-stock-exchange/
https://www.nasdaq.com/articles/3iqs-bitcoin-fund-gets-second-listing-of-2020-this-time-on-gibraltars-stock-exchange-202
https://icryptodesk.com/2020/10/01/why-post-debate-stock-market-performance-could-be-bad-for-bitcoin/
https://bitcoinslate.com/2020/09/30/canadian-firm-3iqs-bitcoin-fund-listed-on-gibraltar-stock-exchange/

new content stored for: https://www.forbes.com/sites/billybambrough/2020/09/30/serious-warning-issued-over-300000-bitcoi
new content stored for: https://www.newsbtc.com/2020/10/01/why-post-debate-stock-market-performance-could-be-bad-for-bit
new content stored for: https://otcpm24.com/2020/10/01/serious-warning-issued-over-300000-bitcoin-stock-to-flow-price-mc
new content stored for: https://sharecaster.com/2020/10/01/serious-warning-issued-over-300000-bitcoin-stock-to-flow-pric
no change for: https://news.bitcoin.com/canadian-firm-3iqs-bitcoin-fund-listed-on-gibraltar-stock-exchange/
no change for: https://www.nasdaq.com/articles/3iqs-bitcoin-fund-gets-second-listing-of-2020-this-time-on-gibraltars-stc
new content stored for: https://icryptodesk.com/2020/10/01/why-post-debate-stock-market-performance-could-be-bad-for-bit
new content stored for: https://bitcoinslate.com/2020/09/30/canadian-firm-3iqs-bitcoin-fund-listed-on-gibraltar-stock-ex
avoid URL: https://www.marketwatch.com/investing/cryptocurrency/btcusd/charts
CAPTCHA required for https://www.globenewswire.com/news-release/2020/10/01/2101855/0/en/Atari-Token-Agreement-with-Bitcc
```

Collection Avoidance

Data is only recorded for a website after passing a series of checks. For example, since it is unknown what data might be found based on the search criteria, it would be possible that some sites may have a Completely Automated Public Turing Test To Tell Computers and Humans Apart test (CAPTCHA) which is a mechanism designed to test to see if you are a bot or if you are a human looking at the web page in question. This type of result is generally found on login pages and as a result, the collection bots identify the CAPTCHA and avoid attempting to collect anything from the page since the manual exploration of all instances of this type of result had no useful data on the page with a CAPTCHA. Once a site has been identified as requiring CAPTCHA, it is marked to be avoided in subsequent searches for a week prior to attempting to collect from the site again (if the search criteria happen to result in the same site at a later time), and if the CAPTCHA is removed (not likely), data can be collected, otherwise, avoided for another week. Similar to CAPTCHA, there are sites that use a mechanism designed to test for bots which could be used for Distributed Denial of Service (DDoS) identification which is a technique that a malicious individual may use to disrupt a site from being able to let users access their site. If this is identified, a similar avoidance note is made in the database, and the site is avoided for a week prior to attempting to access it again if requested.

Another check put in place required a little bit of preprocessing of the incoming unstructured data that was like the type of preprocessing performed on the social media data previously described. As a result, a comparison of the incoming data to existing data

within the database related to the website being collected is made. If the data has not changed the instance of data currently being looked at is not re-recorded; otherwise, the collection of data was new and it was recorded in the database along with the preprocessing step that was used to perform the test to avoid having to unnecessarily regenerate that preprocessed data during the overall preprocessing that would occur on unstructured data after this collection point.

The domains are also classified by category which resulted in approximately 280 unique classifications. This was something that I did not initially anticipate that I would be able to accurately account for about the domains that were identified during the collection process. Initially, it was believed that it would be necessary to classify based on manually identifying sites. Following that, either manually classifying them or running through and generating training sets based on the data that was actually observed in the collection sets and then using that to build an automated classification component to classify the domains. However, after some research, I discovered that Open DNS offered a public service that is primarily used by parents or anyone wanting to try to better control the network traffic that their kids or other users of their network are permitted to browse. Similar to Figure 6, a request including the domain (e.g. www.forbes.com) is sent to Open DNS, and the response includes the classification (e.g. Financial Institutions). After identifying this, the service was incorporated into the system by asking their API how the domain was classified and this yields very accurate results since the way that they go about the classification process is based on a voting

mechanism from there 90 million userbase. Examples of categories might be individual or combined like:

- Financial Institutions
- Politics
- News/Media
- Forums/Message boards
- Social Networking
- Software/Technology, Business Services
- Search Engines, News/Media
- Ecommerce/Shopping, News/Media, Financial Institutions
- Research/Reference, Forums/Message boards
- News/Media, Financial Institutions
- Ecommerce/Shopping, Business Services

Some domains have a single categorization, some have multiple. As a result of this type of categorization being recorded with all observations, it would allow the preprocessing steps in theory to take multiple paths. For example, currently, all preprocessing done involves all data; however, the preprocessing steps could be re-completed for each individual category that way a prediction could be made for example only on a single category like “News/Media,” or “Ecommerce/Shopping, Weapons, Sports, Advertising, Business Services.” Or the preprocessing steps could include data from multiple categorizations like a combination of Gambling, News/Media, Forums/Message boards, Government, and Social Networking, for example.

Unstructured Data

Once the classification has been made, and the decision to record the data has been made, the bot stores the data (e.g. what site, time, classification, and data from the request response from the site) in the database for processing. To make it clear what this unstructured data might look like, a human might visit a site (e.g. https://www.coindesk.com/bitcoin-trounces-bch-bsv-fork-wannabes) and see something like Figure 12, where the unstructured collection bot sees it differently, as demonstrated in Figure 13, which includes formatting data to make it easier for a human to read.

Figure 12

Example Image of Unstructured Data as seen by a Human

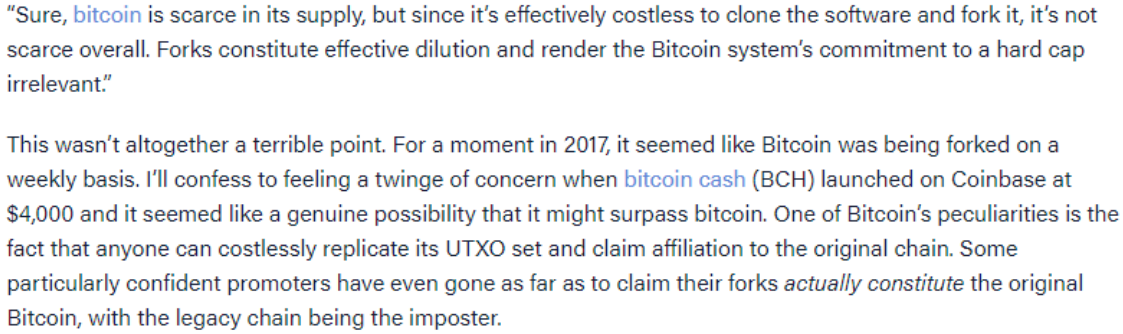
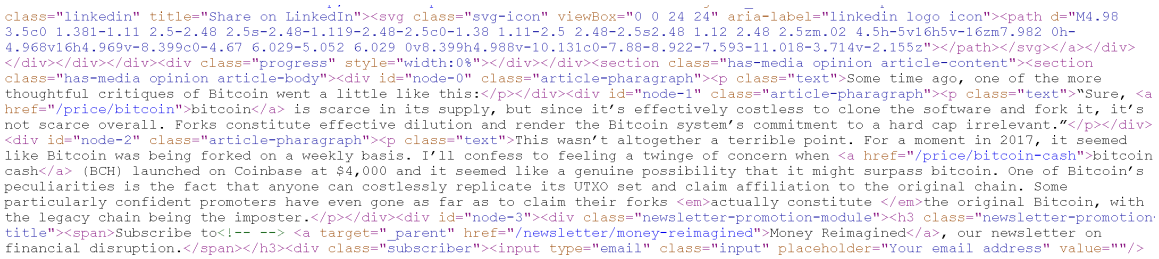


Figure 13

Example of Unstructured Data as seen by an Unstructured Bot



Social Media Preprocessed Data

Much of the data surrounding anything captured from social media is structured in nature. For example, when comments were posted and how many times it was replied to, how many followers, how many favorites, or is the user verified or not, etc. most of that data does not require any initial preprocessing. It is the payload data or the purpose of the post in terms of what the user said which requires additional processing in order to numericize the data in a way that we will be able to leverage it at the end of the process in the predictive model.

To achieve this preprocessing step, several transformations must occur on the data. In general, it would start by retrieving the unprocessed text (e.g. a Tweet), and then operate on it by making all text lowercase, removing character data that might be non-alphanumeric such as emojis, removing all white space characters (like extra spaces, tabs, and newline characters), lemmatization, and removing stopwords. The purpose of lemmatization is to take a word and reduce it to the base or dictionary form of the word (e.g. “caring” would reduce to “care”). Stopwords are words that one would consider being used frequently but do not add significant value to the sentence (like “a” “for” “the” “me” “you”). In Figure 14, you can get a sense of what unprocessed data looks like and then what the preprocessed data looks like.

Figure 14

Example of Data Collected by a Structured Bot and then Preprocessed

```
Original Data:

 Bitcoin has some catastrophic risks.
 The severity is sorted from high to low.
 1. personal risk; 2. platform risk 3. policy risk #Bitcoin

Preprocessed Data:

 bitcoin catastrophic risk
 severity sort high low
 1 personal risk 2 platform risk 3 policy risk bitcoin
```

All of this is necessary for us to create a term-document matrix (TDM) which is a way for us to count how many times a word shows up in the data and make comparisons against other similar data since we have basically made the data all similar (e.g. making it lower case, along with lemmatization would make two people sound more generally the same in terms of how they speak). For example, based on Figure 14 you would come up with the number of times a word shows up as seen in Figure 15.

Figure 15

Example of a TDM Generated from Preprocessed Data as seen in Python

bitcoin	catastrophic	high	low	personal	platform	policy	risk	severity	sort
2	1	1	1	1	1	1	4	1	1

From here we can count the number of positive terms and negative terms to introduce new features of the dataset. This count is also stored within the preprocessed data to operate as an input to the predictive model later. These counts are used to perform

a basic sentiment analysis which allows us to classify the unstructured data as either positive or negative to produce additional features for the preprocessed data set. Details are provided in the Limitations section of this paper that this basic form of sentiment analysis has some shortcomings wherein may inaccurately classify some data as positive or negative depending on the context of the data being processed.

Another feature added to the preprocessed dataset is captured at the beginning of the process that led to the TDM which is a count of all data that is non-alphanumeric within the text following. This is done in order to capture something like when a user that puts many exclamation marks or several continuation periods (…) just to provide an additional data point to be able to pass off in later stages of preprocessing data for the predictive model.

Once all of this is completed, the dataset is stored within the database with the ID of the original data that it relates to so these newly created attributes can effectively be joined to the original data which increases the amount of structured data that we will be able to pass off at a later stage when the data will be compressed and combined with other datasets that would otherwise be unrelatable with the exception of time. By storing this data, the steps do not need to be repeated which would put an unnecessary burden on the computer.

Social Media Popular Words

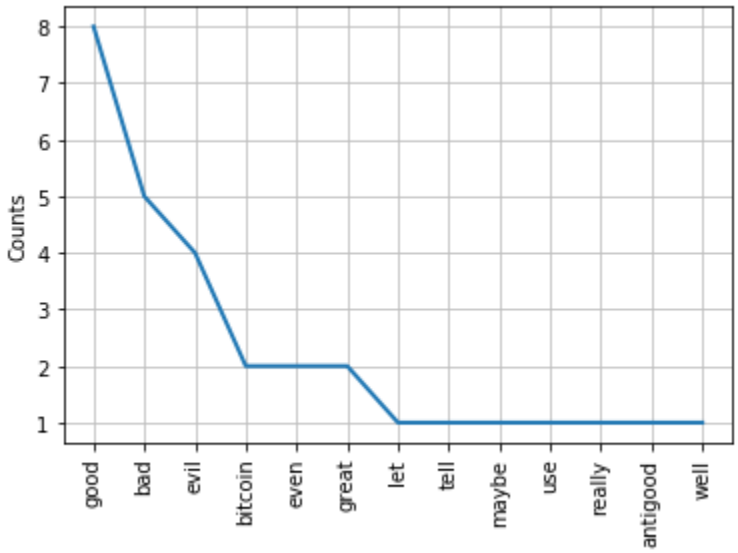
This is one of the components that help create the feedback loop within the SEDCABI engine to help evolve and propagate search criteria that other than the one common thread of Bitcoin has nothing to do with any of the UI initially seated search

criteria. Given a starting hour to look at we can grab a small slice of the data that was captured within the hour in question. From there we can take that data and tokenize the text which basically just means come up with a list of all words found within the data and apply a count to them. This is identical to the Figure 15, except, instead of looking at a single count of terms from one social media post, all social media posts for the hour in question are used to generate a much larger TDM, in which, if a term showed up 1000 times across many social media posts, the TDM count would reflect that summed count of each time the term showed up.

By doing this we can identify what is most popular within the data set by identifying which terms are used most frequently. Given the example in Figure 15, we can easily generate a frequency distribution as seen in Figure 16 that can be used to identify what is most frequently being talked about.

Figure 16

Example of a Term Frequency Distribution as seen in Python



Given what is identified to be most popular, those terms are used to create new search criteria to feedback into the SEDCABI engine to let the system go out and start looking for data associated with Bitcoin along with the appropriate popular term. To ensure that the search does not run forever, time parameters are additionally passed with the search criteria to ensure that the feedback loop does not persist indefinitely and can be easily controlled and set to terminate within an hour.

Unstructured Preprocessed Data

The initial preprocessing of the unstructured data is a very similarly laid out process as to what was done with the social media data. Similar transformations must occur on the data. Initially, all same preprocessing steps that are performed on social media occur on the unstructured data; however, some additional cleaning of the data is needed on the unstructured data. Removing any possible Hypertext Markup Language (HTML) data embedded in the text (this is formatting data, for example, that makes a bold word look like **this**. Removing JavaScript (code that may be embedded in the webpage which might do something like make a picture larger when you point the mouse at it) and Cascading Style Sheet (CSS) tags (font and information about how to display data). Based on the example provided in Figure 13, after being preprocessed would look like the example provided in Figure 17.

Figure 17

Example of Preprocessed Data as seen in SSMS

sure bitcoin scarce supply since effectively costless clone software fork scarce overall fork constitute effective dilution render bitcoin system commitment hard cap irrelevant wasnt altogether terrible point moment 2017 seem like bitcoin fork weekly basis ill confess feel twinge concern bitcoin cash bch launch coinbase 4000 seem like genuine possibility might surpass bitcoin one bitcoins peculiarity fact anyone costlessly replicate utxo set claim affiliation original chain particularly confident promoter even go far claim fork actually constitute original bitcoin legacy chain imposter

Again, these steps are necessary for us to create another TDM to count how many times a word shows up in the data. Classification of sentiment analysis along with positive and negative term counts are stored within the new dataset. Unlike the social media data preprocessing the non-alphanumeric data such as punctuation or graphical characters is not captured in the unstructured data.

Once this is completed the data is stored within the database with the ID of the original data that it relates to so that these newly created attributes can effectively be joined to the original data which increases the amount of structured data that we will be able to pass off at a later stage when the data will be compressed in combined with other datasets that would otherwise be unrelated with the exception of time. The outcome of each of these preprocessing steps, in regard to the data after it has been cleansed, is also recorded in the database so the preprocessing steps do not need to be repeated which would put an unnecessary burden on the computer.

Unstructured Popular Terms

This is another one of the components that help to create the feedback loop within the SEDCABI engine to help evolve and propagate search criteria that other than the one

common thread of Bitcoin has nothing to do with any of the UI initially seeded search criteria. Given a starting hour to look at we again grab the preprocessed data that was captured within the hour in question. From there we take that data and tokenize the text the same way as described in the Social Media Popular Words section of the paper. After doing this, the system identifies what is most popular within the data set, and given what is found, new search criteria is fed back into the SEDCABI engine (paired with the term Bitcoin to ensure the search engines are providing relevant searches, which is also discussed in the Limitations section of this paper). Again, to help ensure that the search does not run forever time parameters are passed with the search criteria to ensure that the feedback loop does not persist indefinitely and is directed to stop after one hour.

Compression of the Data

It is at this point where all the preprocess datasets will be able to come together. In order to do this, the social media preprocessed data runs through a compression algorithm that simply takes all data within an hour by hour time, and for that specific hour, all of the preprocessed values are summed into a single value for each feature. While most of the data is already numeric after it has been preprocessed there are still some aspects of date-time that will get numericized at this point. For example, data like how long the user has been using the social media tool is quantified by taking whatever that value is and counting the number of days since the inception of the tool. An example of this is shown in Figure 18.

Figure 18

Example of Compressed Social Media Data as seen in SSMS

startDateTime	stopDateTime	overallCount	user.verified	user.followers_count	user.friends_count	user.listed_count	user.favourites_count	user.statuses_count
2020-07-11 14:00:00.000	2020-07-11 15:00:00.000	2400	10	8640412	3620439	178945	28170346	85338305
2020-07-11 15:00:00.000	2020-07-11 16:00:00.000	2509	7	10028243	4038865	186669	29134670	86853574
2020-07-11 16:00:00.000	2020-07-11 17:00:00.000	2640	12	23181252	4747362	266943	33988026	80577286
2020-07-11 17:00:00.000	2020-07-11 18:00:00.000	1786	6	7995026	3089158	130994	22406237	50319062

After this compression occurs and all the data for that individual hour in question is transformed into a single row of summed values for the hour in question, that summed data gets stored into another database that only stores the preprocessed compressed data. The same strategy is used for the unstructured preprocessed data and stored within that same database. An example of this is shown in Figure 19.

Figure 19

Example of Compressed Unstructured Data as seen in SSMS

startDateTime	stopDateTime	overallCount	overallScore...	positiveScore...	negativeScore...	howManyWords	imputed
2020-07-11 14:00:00.000	2020-07-11 15:00:00.000	321	17797	42385	24588	1381967	False
2020-07-11 15:00:00.000	2020-07-11 16:00:00.000	407	28184	66859	38675	1667946	False
2020-07-11 16:00:00.000	2020-07-11 17:00:00.000	505	34658	77993	43335	2161971	False
2020-07-11 17:00:00.000	2020-07-11 18:00:00.000	357	23765	59012	35247	1462027	False

Once compression has been completed on both the preprocessed, structured social media data and the preprocessed, unstructured data, an additional view of the data is created. To do this the aspect of the hour in question from both compressed data sets is now tied directly into the Bitcoin exchange data regarding the starting and ending time of the compressed data sets of Figure 18 and Figure 19 which get joined to the example of Bitcoin price data as seen in Figure 20. It is at this point where the complete data set that is presented to the plug-in predictive module is first observed.

Figure 20

Compressed Content Joined by Date into a Single Dataset as seen in SSMS

btcStartDateTime	btcStopDateTime	openAmount	closeAmount	high	low	volume
2020-07-11 14:00:00.000	2020-07-11 15:00:00.000	9237.5000000000	9232.3018687700	9237.5000000000	9232.3018687700	2.0284658500
2020-07-11 15:00:00.000	2020-07-11 16:00:00.000	9241.9000000000	9241.9000000000	9241.9000000000	9241.8000000000	0.1960000000
2020-07-11 16:00:00.000	2020-07-11 17:00:00.000	9238.0000000000	9237.8364156300	9238.9000000000	9237.8364156300	0.8684399200
2020-07-11 17:00:00.000	2020-07-11 18:00:00.000	9227.8968212600	9227.8968212600	9227.8968212600	9227.8058734800	0.0300000000

One other consideration that had to be addressed during the compression of the data was held deal with any missing values. Generally, this was a non-issue due to the amount of data collected; however, there were a small number of outages likely related to either the network where the data was being collected from or related to the target being unavailable at the time the collection attempt was made. For example, if no unstructured data was able to be collected for two hours, it would mean that we would be unable to compress data for that time period and ultimately it would mean that we would not be able to use other data sets like the social media data collected at that time because we would have no time reference to be able to relate the social media compressed data to the unstructured compressed data for that time period. As a result, data imputation was employed by two means. First, whatever the last previous observation within the compressed data set was those values were simply copied forward it used to impute the missing time. Secondly, a simple average of the last observation and the next observation was used to inject another imputed feature to represent the missing data. What this meant was that within the period that the data set represents, all-time could be accounted for in the predictive model. This is a potential limitation discussed later in the paper.

Predictive Model

Initially, I had envisioned that it would be necessary given the sheer amount of data that from a dimensionality reduction standpoint it would force principle component analysis in order to make sense of the inputs; however, it turned out that that was unnecessary and only made it more difficult to explain the inputs as they related to the output. As previously described, the plugin prediction module (PPM) was designed with the idea that whatever the task was in terms of the data collection the user of the system might have a specific predictive methodology in mind depending on the type of data that was collected. As a result, more than 200,000 models were implemented based on multiple linear regression with the stepwise method to perform various predictions at 6, 12, 24, 48, 72, and 168-hour intervals as seen in Table 3.

Table 3

Models Generated for Future Hours Predicted

Model	Future hours predicted
Base Line (MLR with Stepwise)	6
Base Line + Weighted Indices	6
Base Line + Weighted Indices + Shift	6
Base Line + Weighted Indices + Shift + Simple Moving Average	6
Base Line + Weighted Indices + Shift + Exponential Moving Average	6
Base Line (MLR with Stepwise)	12

Table 3: continued

Base Line + Weighted Indices	12
Base Line + Weighted Indices + Shift	12
Base Line + Weighted Indices + Shift + Simple Moving Average	12
Base Line + Weighted Indices + Shift + Exponential Moving Average	12
Base Line (MLR with Stepwise)	24
Base Line + Weighted Indices	24
Base Line + Weighted Indices + Shift	24
Base Line + Weighted Indices + Shift + Simple Moving Average	24
Base Line + Weighted Indices + Shift + Exponential Moving Average	24
Base Line (MLR with Stepwise)	48
Base Line + Weighted Indices	48
Base Line + Weighted Indices + Shift	48
Base Line + Weighted Indices + Shift + Simple Moving Average	48
Base Line + Weighted Indices + Shift + Exponential Moving Average	48
Base Line (MLR with Stepwise)	72
Base Line + Weighted Indices	72
Base Line + Weighted Indices + Shift	72

Table 3: continued

Base Line + Weighted Indices + Shift + Simple Moving Average	72
Base Line + Weighted Indices + Shift + Exponential Moving Average	72
Base Line (MLR with Stepwise)	168
Base Line + Weighted Indices	168
Base Line + Weighted Indices + Shift	168
Base Line + Weighted Indices + Shift + Simple Moving Average	168
Base Line + Weighted Indices + Shift + Exponential Moving Average	168

Predictive models are constantly built by the software and as each model is built, the aspects of the model are recorded in the database including things like what techniques went into building the model how many coefficients did the model result in, what was the R^2 value of that model, along with the calculated interval from the goodness of fit metric. Additionally, the coefficients and the model itself physically get saved and stored so that it can be recalled using on any state of the data, past, present, or future-looking. This data is used in model selection for each hour of the combined, compressed data as described once the simulation software is introduced in the Results section of the paper.

As seen in Table 3, there are 5 models built for each hour prediction. First, a baseline model is built using multiple linear regression with the stepwise method. As the

regression is performed, the coefficient with the highest p-value over 5% is removed, and regression is performed again (and repeated until all p-values are <= 5%).

Expanding on the baseline, weighted indices that represents a numerical value for time is added to the data. While discussed in the Limitations section, this is what occurred: Five indices were added into the time-ordered compressed dataset (the dataset was in order by time representing the compressed hour, so 7/1/2020 1:00 am was listed before 7/1/2020 2:00 am, etc.), and that dataset was provided to MLR (the same as described by the baseline model). An example of what they look like can be seen in Table 4.

Table 4

Example of Weighted Indices as seen in Python

Index	IndexWeighted2	IndexWeighted3	IndexWeighted4	IndexWeighted5
0	0	0	0	0
1	1	1	1	1
2	4	8	16	32
3	9	27	81	243
4	16	64	256	1024

Next, building on the same idea, adding a single shifted, lag value for each feature is introduced (for example, if there was an “OverallCount,” which is the programmatic name of an input, of 2647, it would be listed in a new column and shifted down in the

next row of data, resulting in a new input named “Lag_1OverallCount”). An example of this can be seen in Table 5.

Table 5

Example of a Shifted, Lag Variable Introduced to the Compressed Dataset

OverallCount	Lag_1OverallCount
2647	0
2556	2647
2266	2556
2010	2266
2182	2010

Next, building on the baseline, weighted indices, and a simple moving average (with a “window size of 3” as seen in the example) is added to the data. An example of what this looks like can be seen in Table 6.

Table 6

Example of Simple Moving Average Introduced to the Compressed Dataset

OverallCount	Lag_SMA_OverallCount
2647	0
2556	0
2266	2489.666667
2010	2277.333333

Table 6: continued

2182	2152.666667

Finally, building on the baseline, weighted indices, an exponential moving average is added. An example of this can be seen in Table 7. Since the software is constantly creating so many predictive models and storing the results within a database, it makes it easy to identify which models can be used for each of the time predicted scenarios.

Table 7

Example of Exponential Moving Average Introduced to the Compressed Dataset

OverallCount	Lag_EMA_OverallCount
2647	2647
2556	2630.454545
2266	2564.190083
2010	2463.428249
2182	2412.259477

Results

There are four primary databases that comprise all data for the SEDCABI engine. Altogether they total almost 1.5 terabytes of data. The main database houses the general orchestration of the entire process and includes many of the aspects that allow for the self-evolving data capture and overall intelligence of the process to occur. Specifically, it houses the domains, individual sites collected, the unprocessed and preprocessed unstructured content, details about the predictive models generated, lists of popular terms extracted from the structured and unstructured content, along with avoidance data as described in the Collection Avoidance section of the paper. Two of the databases have rules for only dealing with targeted structured data. For example, one database stores targeted structured data, like when collecting data from Twitter's API, which would store details about the user's posting (the "Tweet"), when it was posted, how many followers the user has, etc. as seen in Figure 8. Another database stores Bitcoin exchange data like seen in the example of Figure 20. Minute by minute structured Bitcoin data including things like open, close, high, low, and volume is also constantly being captured and entails approximately 137,000 observations. This ensures that there is always momentum moving forward so as data is captured and preprocessed, it can be used to generate new and up-to-date predictions.

The other structured database has the role of dealing with the final compressed data that is ultimately used to drive the prediction component. While the data collection spans several months, it does not do so in all aspects of data. This is related to the development process of the code and as a result, not all aspects of data were represented

completely, and that data was not used once the entire process was debugged and tested. Once that occurred, the initial incomplete data was removed from the database to ensure that the process had consistent data in all aspects to be considered in the predictive model (for example, if there was unstructured data only from 7/1 – 7/31, but the other databases had data from 5/1 – 7/31, all data from 5/1 – 6/30 were removed). The result of the final completed data set is 1.5 terabytes of data and represents a period of a little more than two months.

To give some examples of the quantities of data spanning the final utilized dataset, includes approximately 16,000 self-evolved searches, 5.2 million social media posts, 10,000 unique domains, 75,000 web sites, and 835,000 collections of data from those websites. Examples of domains can be seen in Table 8 where data was collected from and the number of collections from that domain (rather than displaying all 10,000 unique domains, the list is shortened to include domains where the data was collected at least 7000 times).

Table 8

Domain Examples and Number of Collections for the Domain

Domain	Collections
finance.yahoo.com	51441
cointelegraph.com	39401
www.forbes.com	23263
news.bitcoin.com	23062

Table 8: continued

otcpm24.com	22997
www.coindesk.com	18345
en.wikipedia.org	17833
www.bit-cointalk.com	16198
cryptomoneyteam.co	13494
www.newsbtc.com	12789
www.americancryptoassociation.com	12680
elevenews.com	12604
www.investing.com	10107
bitcoinslate.com	9601
coinmarketcap.com	9578
www.reddit.com	8271
www.tradingview.com	8008
www.justcryptocurrencies.com	7730
medium.com	7645
www.coinbase.com	7592
dailyhodl.com	7296

As described in the methodology, the domains are classified by category which resulted in approximately 280 unique classifications. For example, www.npr.org is classified as “Radio, News/Media, Non-Profits, Podcasts,” and www.facebook.com is classified as “Social Networking.” Below, in Table 9, it includes the breakdown of the

classifications by sites collected within the classification (the list is shortened to only include classifications that contained more than 5000 collections).

Table 9

Site Classification Examples and Number of Collections by Category

Type of Classification	Collections
Unclassified	608360
News/Media	90628
Search Engines, News/Media	53333
Financial Institutions	44384
Research/Reference, Non-Profits, Research/Reference	17964
News/Media, Financial Institutions	13187
News/Media, Forums/Message boards	8271
News/Media, Business Services	7215
News/Media, Software/Technology	6648
Social Networking	6319
Research/Reference	5467

Once all data was preprocessed, it was compressed as described in the Compression of the Data section of the paper, which resulted in a dataset used to generate various predictive models listed in Table 3. These data sets are automatically grown depending on how they get fed into the model. For example, there might be aspects of time that get indexed and possibly squared or cubed, or there might be aspects of lag that

get introduced. For example, possible copying and shifting of columns, or the introduction of different types of moving averages like simple moving averages or exponential moving averages. Examples of this can be seen under the Predictive Model section of the paper.

Simulation

Simulation software was constructed to get an idea of how effective the predictions could potentially be. Based on the compressed data which represents an hour by hour time frame predictive models were constructed to include all data up until the hour that was used for prediction. At no point in time was a prediction model used that had any future-looking data. This allowed me to simulate 6, 24, 48, 72, and 168-hour predictions based on data given at any specific hour and to compare it relatively to the actual price of Bitcoin for the current hour (e.g. 8/2/2020 at 1 am) to what would actually be the price 6, 24, 48, 72, and 168-hours from the hour in question (e.g. 168 hours after 8/2/2020 at 1 am is 8/9/2020 at 1 am).

The simulation for each hour prediction interval was performed three times, once using all compressed data, once with the unstructured data removed, and once with the structured data removed (except for the Bitcoin price data). This occurred using compressed data generated from 6/27/2020 through 8/31/2020 in all simulations. For each of the simulations, two selection methods were used (one based only on R^2, and the second using the GoF prediction interval) to demonstrate the difference that the selection method had on the simulation.

Model Selection for Simulation

Since so many models were generated, only one model could be selected during the simulation based on the time interval predicted. For example, not only considering R^2 but looking at the overall goodness of fit interval metric (see Table 10 for an example), for each hour, a model was selected and used for the prediction. This was achieved for a specific hour by looking at the databases which kept track of the R^2 and GoF prediction interval for the model. The database was queried to return a single model for the hour in question by looking at the lowest interval for the hour, and if there was a tie (for example two models had the same interval, the one with the highest R^2 value was selected). For comparison of selection methods in the simulation, an additional selection method of only using the R^2 value was simulated to show the difference a change in the selection method could have on the results of the simulation.

The simulation used this data to select a model for the hour in question as is rolls forward, hour by hour, through the compress data making predictions for the various hour prediction intervals (6, 24, 48, 72, 168). This rolling model method (RMM) picks the best model for the hour being predicted, makes the prediction based on the model section, and moves on to the next hour, and repeats the process. Noted in the Limitations section of the paper, the RMM does not account for imputation in the data when selecting a model which can lead to a larger difference in the system's prediction for a Bitcoin price vs the actual Bitcoin price for a given hour.

Table 10 provides an example of the goodness of fit metrics that the system would generate (this example was based on a time frame that included 6/27/2020 through

7/31/2020 10:00 am), along with Figure 21 which shows a visualization of the same prediction interval.

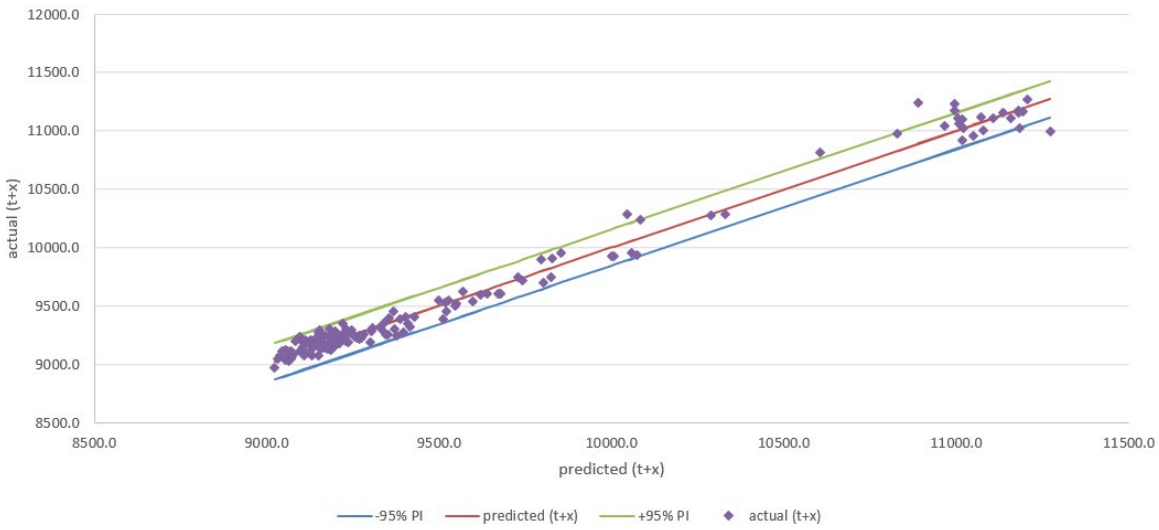
Table 10

Example of Goodness of Fit Metrics for 6 Hour Prediction Interval

Metric	Values
n	160
dof	159
CE	1369.503271
Avg E	8.559395441
MAE	57.51646328
SSE	994771.6944
MSE	6217.32309
RMSE	78.85000374
MAPE	0.005883383
R^2	0.986218991
alpha	0.05
STEYX	78.86697103
T.INV.2T	1.974996213
Interval (+/-)	155.7619691

Figure 21

Goodness of Fit 1:1 Plot with +/- 95% Prediction Interval for 6 hours



The 0 base was removed from the scale to give a better visual of the goodness of fit since the price of Bitcoin fell within $8,500 to $12,000 during the observed time. Overall, about 200,000 models were generated to account for the overall period that the data was collected, and this data was recorded with each model. Eventually, the component that generates models had logic added to it that only allows it to record a model for a given hour if the model is better (in terms of R^2 and the prediction interval) than the other models that exist for the given hour. This helps to slow the creation of models that would not be picked by the simulation software.

Simulation Using All Data

As of 8/31/2020, considering a prediction interval of 168 hours for an example (and using the GoF prediction interval as the model selection method), when I compared the actual price (e.g. what is the actual price of Bitcoin right now) to the predicted price the simulation demonstrated that methodology was able to generate a direction accuracy

for the price as seen in Table 11 with 92% accuracy a week out. What this means is if a week from now the price of actual Bitcoin would be up or down compared to right now, the simulation predicted and correctly agreed (this is what is shown in the "Agreement" column) that the price would be up or down accurately 92% of the time overall. The difference of the actual price at the hour interval minus the predicted price for the simulation was averaged to produce the "Mean Difference." The same holds true for the "Median Difference" except for using a median function on the difference. Using the GoF prediction interval was slightly more than 6% more accurate at the 168-hour interval than the R^2 selection method.

Table 11

Simulation Accuracy using All Data

Hour Predicted	Selection Method	Agreement	$ Mean Difference	$ Median Difference
6	R-squared	61.22%	(3.39)	(6.36)
12	R-squared	66.74%	5.67	7.35
24	R-squared	74.03%	7.80	8.81
48	R-squared	73.30%	37.85	(8.98)
72	R-squared	83.11%	5.63	0.25
168	R-squared	85.99%	(8.03)	(25.73)
6	GoF Interval	88.14%	0.33	(0.06)
12	GoF Interval	87.46%	4.45	(0.37)
24	GoF Interval	86.08%	17.91	0.69

Table 11: continued

48	GoF Interval	90.42%	12.26	(0.13)
72	GoF Interval	91.19%	14.84	0.62
168	GoF Interval	91.99%	(79.14)	(6.44)

Simulation with Unstructured Data Removed

Doing the same simulation as described using all data, but removing the unstructured data from the compressed data, for the same time period, and 168-hour prediction interval, and using the GoF prediction interval as the model selection method, the simulation predicted the price would be up or down 83% of the time overall. Using the GoF prediction interval was slightly more than 2% more accurate at the 168-hour interval than the R^2 selection method. This can be seen in Table 12.

Table 12

Simulation Accuracy with Unstructured Data Removed

Hour Predicted	Selection Method	Agreement	$ Mean Difference	$ Median Difference
6	R-squared	61.40%	8.66	5.85
12	R-squared	66.48%	13.86	14.47
24	R-squared	76.11%	11.41	6.08
48	R-squared	77.37%	6.65	(17.39)
72	R-squared	83.46%	(15.68)	(16.09)
168	R-squared	81.66%	(116.15)	(94.71)

Table 12: continued

6	GoF Interval	75.17%	7.13	0.49
12	GoF Interval	75.90%	10.91	1.93
24	GoF Interval	80.20%	11.25	1.77
48	GoF Interval	82.44%	10.47	(3.57)
72	GoF Interval	86.77%	(13.31)	(2.56)
168	GoF Interval	83.78%	(114.21)	(72.94)

Simulation with Structured Data Removed

Doing the same simulation as described using all data, but removing the structured data from the compressed data (with the exception of Bitcoin price data, otherwise the purpose is immediately defeated since the Bitcoin data is considered structured data), for the same time period, and 168-hour prediction interval, and using the GoF prediction interval as the model selection method, the simulation predicted the price would be up or down about 88% of the time overall. Using the GoF prediction interval was slightly more than 2.5% more accurate at the 168-hour interval than the R^2 selection method. This can be seen in Table 13

Table 13

Simulation Accuracy with Structured Data Removed

Hour Predicted	Selection Method	Agreement	$ Mean Difference	$ Median Difference
6	R-squared	63.09%	4.36	(1.21)

Table 13: continued

12	R-squared	67.83%	10.51	(0.10)
24	R-squared	74.00%	0.07	28.16
48	R-squared	76.40%	20.56	(4.39)
72	R-squared	82.04%	(1.35)	(3.03)
168	R-squared	85.22%	(134.00)	(89.17)
6	GoF Interval	77.54%	5.23	0.12
12	GoF Interval	76.68%	12.93	1.11
24	GoF Interval	79.65%	23.89	3.31
48	GoF Interval	81.25%	17.05	0.72
72	GoF Interval	84.56%	(0.26)	(0.35)
168	GoF Interval	87.88%	(120.57)	(59.82)

Comparison of the Simulations

Comparing Table 11 through Table 13, and considering the three simulations, using all data, using only structured data, and using only unstructured data, in all cases using the GoF prediction interval selection method always performed better than using R^2 alone. Additionally, in all cases, using all data always outperformed using either structured data alone or unstructured data alone. See Table 14 for a breakdown of how each hour interval predicted, along with which type of data was used in the simulation to see the percentage improvement by using all data to the other restricted datasets.

Table 14

Comparison of Using All Data to Structured or Unstructured Data Alone

% Improvement	Data Used	Hour Predicted	Selection Method
12.97%	Structured	6	GoF Interval
11.56%	Structured	12	GoF Interval
5.88%	Structured	24	GoF Interval
7.98%	Structured	48	GoF Interval
4.42%	Structured	72	GoF Interval
8.21%	Structured	168	GoF Interval
10.60%	Unstructured	6	GoF Interval
10.78%	Unstructured	12	GoF Interval
6.43%	Unstructured	24	GoF Interval
9.17%	Unstructured	48	GoF Interval
6.63%	Unstructured	72	GoF Interval
4.11%	Unstructured	168	GoF Interval

Conclusion

Based on the results of the simulations, it has been demonstrated that leveraging all data produces better results for all prediction intervals than using structured data alone or unstructured data alone. Summarizing the finding from the simulations (Simulation Using All Data through Comparison of the Simulations) based on those three data sources is shown in Table 15.

Table 15

Summary of Simulation with Different All/Structured/Unstructured Data

Hour Predicted	Data Used	Selection Method	Agreement
6	All	GoF Interval	88.14%
12	All	GoF Interval	87.46%
24	All	GoF Interval	86.08%
48	All	GoF Interval	90.42%
72	All	GoF Interval	91.19%
168	All	GoF Interval	91.99%
6	Structured	GoF Interval	77.54%
12	Structured	GoF Interval	76.68%
24	Structured	GoF Interval	79.65%
48	Structured	GoF Interval	81.25%
72	Structured	GoF Interval	84.56%
168	Structured	GoF Interval	87.88%

Table 15: continued

6	Unstructured	GoF Interval	77.54%
12	Unstructured	GoF Interval	76.68%
24	Unstructured	GoF Interval	79.65%
48	Unstructured	GoF Interval	81.25%
72	Unstructured	GoF Interval	84.56%
168	Unstructured	GoF Interval	87.88%

With the agreement of the prediction of the Bitcoin price ranging from 86% to almost 92%, it appears that the entire orchestration of the described methodology has been successful and should prove to be useful for anybody conducting research that needs to be able to leverage the aspects of all data, not only structured data, or unstructured data.

In comparison to previously published work, and given the limited research that has leveraged unstructured data with structured data, the closest comparison that I could make based on the previous work, and given the prediction intervals of 6, 12, 24, 48, 72, 168 was an accuracy rate of 74.51% at 168 hours (Kim Y. B., 2016). As seen in the Simulation Using All Data section of this paper, based on the compressed output of SEDCABI sent to the PPM, using the RMM was able to achieve 91.99% at 168 hours which is a 17.48% improvement.

The compression technique described that is inherent to the output of the SEDCABI engine demonstrates a way to compress data that would have otherwise unrelatable and somewhat meaningless to time-series data that lends itself well to many

popular predictive algorithms. Even if the PPM component were not considered, the output of the SEDCABI engine can produce datasets based on the described techniques which could be leveraged by any predictive methodology. Given the sheer volume of predictive models created by the PPM, the RMM, even with the described limitations, is demonstrated as being effective in the simulations at model selection over the compressed data generated by SEDCABI. All of this occurs with only four initial seed terms (“Bitcoin” “Bitcoin news” “Bitcoin when to buy” “Bitcoin when to sell”), the rest of the work is completed by SEDCABI.

Limitations

Possible Bias in the Data

There are potential inherent shortcomings in the way the data is collected by the system which is outside of the control of the system. There is a potential bias in the data since there is no distinction on where the data comes from considering that some may come from syndicated sources and some might be freelance for example. Since the system generates the search is on its own the data that comes back from the searches can come from anywhere and that includes potentially legitimate sources but also potentially illegitimate sources or sources that may be propaganda or potentially fake news and so on. In terms of the way the compression on the data works it uses all of the data so it does not consider the source as a reliability factor. That is where there might be biased one direction or another in terms of the way the data might have an effect once it makes its way to the PPM.

Additional limitations in terms of the way the system collects data may be directly related to where the searches originate from. For example, all data collected by the system currently is done within the United States and that's not to say that the bots might find sources of data that exist outside of the United States however, due to limitations that may exist or be imposed on what sites you're allowed to access from the United States may be a restriction which could also potentially bias the data. The system for example may not be allowed to collect data right in certain countries due to the restrictions imposed by those countries on their network borders.

From a social science perspective, the data even though not restricted by the collection mechanism itself but rather restricted based on controls outside of the system that may exist potentially limit the data that can be collected. At this time, the system leverage is all data that the bots are able to collect and there is no mechanism in the system to weed out a source is based on the perceived reliability of the source.

It would be possible to add additional controls to the system if you wanted to be able to identify certain domains as being reliable or not and that might be the easiest approach to being able to potentially weed out the unreliable sources. The problem is given the sheer amount of locations that exist that the system potentially could collect data from it would not be reasonable to manually identify what is reliable and not so it's possible that restrictive controls could be put in place to literally limit the collected data from a list of pre-known generally excepted reliable sources.

The selection process for resources is also limited to major search engine providers within the United States and it's possible that the way that they prioritize a response to search criteria may also be a source of bias depending on where the search engine stands on certain topics even though the presumption is based off of which sites are being clicked on the most in so it would not necessarily be a bias from the search engine provider but rather the users of the search engine and what they happen to be searching for the most which may promote certain data or sites more so than other sites. Again, a potential issue of the liability of the data. Who is writing the data found on the websites could be propaganda or fake news even from reliable sources so not necessarily categorizing a domain as reliable may eliminate the bias.

Another limitation of the system is related to how the sentiment analysis is performed it is literally a count of positive and negative terms that results in an overall score but does not consider the context in which the data is used. This has the possible result of a joke for example that when taken out of context may be considered very positive when in fact it may be very negative or vice versa. Here is an example of a sentiment analysis using that basic technique on a sentence compared to a different technique that VADAR uses. In the basic sentiment analysis: “At least the book isn’t horrible.” is negative; whereas using VADER: “At least the book isn’t horrible.” is positive.

There is also a limitation related to the way the bots present themself so they expect in the English language if the site that they collect data from is incapable of offering an English version of the data it is not included in the search data. Potentially offering the bots the ability to run from multiple locations could possibly help to expand the amount of data collected along with expanding beyond the English language.

Data Collection

One limitation with the SEDCABI engine it must constantly be collecting data to produce compressed datasets to be used by the PPM. If a network connection drop for a day, that timeframe must either be ignored or imputed. If collection bots stop running, there is no feedback loop, in which case, the system will only be collecting data based on the initial seed searches (e.g. “Bitcoin” “Bitcoin news” “Bitcoin when to buy” “Bitcoin when to sell”). Where other literature has used known structured or unstructured data sources (Bitcoin prices or forums) which provide a timeline which can be accessed at a

later time, SEDCABI relies on unknown unstructured data sources, so you could not reliably search for older data and make use of it in the simulation unless you were sure that the data could be reliably timestamped. For example, today I search for "Bitcoin news" and find an article that was published three months ago, and unless the article was timestamped, and that timestamp could be extracted from the data, you could not reliably consider that historic data to generate the compressed dataset as the output of the SEDCABI engine. For this reason, it must constantly be running to ensure the timeliness of the data captured so that it can be related via time to the structured data.

Another limitation is not knowing how related the unstructured data is to the prediction of Bitcoin price. It is possible some sources are very high, and some are very low. The feedback component of SEDCABI that identifies popular terms and feeds them back to the collection bot is limited to a single term rather than popular phrases which may restrict the potential data collected.

Model Selection

Based on issues described in Data Collection above, the RMM does not account for imputation in the data when selecting a model that can lead to a larger difference in the system's prediction for a Bitcoin price vs the actual Bitcoin price for a given hour. For example, if structured data for a specific hour was imputed for any reason (like a software issue that causes the structured bot to stop collecting) it may be combined during the compression phase of SEDCABI along with compressed unstructured data that was not imputed; or, vice versa (unstructured was imputed for some reason, but structured was not). Another scenario may involve some type of longer outage (like a

network outage, or the system being rebooted and interrupting the SEDCABI process for a period of hours, etc.) in which case both structured and unstructured may need to be imputed. Technically, the system can recover the structured data missing for a given time period since it likely can be accessed via the original source of the API; however, the component of SEDCABI that compresses the data may not know that the data could still be obtained and goes ahead and imputes the missing data. Depending on these types of scenarios, it may result in a greater magnitude of difference between the system's prediction and the actual Bitcoin price for a given hour. In these scenarios, it is possible that adjusting the model selection process beyond what was previously described to be a little bit more flexible in terms of the timeframe considered (maybe using a model generated prior to the issue) may help to improve the volatility in terms of the magnitude of the difference of the predicted versus actual price of Bitcoin. This may also be a situation where different imputation methods could be considered, and possibly introducing other techniques to handle creating the predictive models in the PPM.

Avoided Data Collection

As described in Collection Avoidance , not all data can be collected for various reasons. Common examples that the system encountered included sites with embedded CAPTCHA (e.g. you must solve a puzzle prior to being allowed to access the site), or other forms of protection designed to ensure that it is a human visiting the site rather than some type of bot, like the collection bots of SEDCABI. This places additional restrictions on where data can be collected from which could act as additional bias in the data.

Volume of Data

Finally, the sheer volume of data collected and generated by the overall system is massive. Not only that the computational requirements are also very high, so for one computer to encompass all aspects of the system is tough. It would be much better to leverage cloud-based services where you can translate more money to more machine and storage power, but the “more money” part is restrictive.

Future Research

There are several potential avenues for future research that I believe can come out of what has been presented in this paper. From a preprocessing through predictive model creation perspective I believe that all of the aspects that were captured from the SEDCABI engine could be considered as a way to further evolve the usefulness of the compressed data that would be presented to the PPM or other external predictive methodologies. As previously mentioned, an example using something like the category of the unstructured data could be used as a filtering mechanism to better target data to be provided to the PPM, or introducing a technique to help reduces bias in the data as described as a limitation of the collected data. Considering other imputation techniques could also improve the way that the compressed data could be consumed by the PPM (or external predictive techniques).

Another area to further explore would be changing up the PPM to including additional predictive methodologies beyond MLR in order to see how they would be able to leverage the compression technique used to feed data into the PPM, as well as improving the model selection process when evaluating the type of data that it has been presented and understanding there may be aspects of imputation that one model might be better suited at handling than another.

Additionally, based on the compression technique introduced to be able to summarize otherwise unrelated data sets, it may be possible to consider rolling windows of data and then using that as a way to not only predict a future value but from a

simulation perspective, introduce a strategy component to the system and have it actually control trades based on the accuracy of the simulation to automate buy/sell strategies.

References

Alessandretti, L. E. (2018). Anticipating Cryptocurrency Prices Using Machine Learning. *Complexity*, 1-16.

Clements, R. (2018). Assessing the Evolution of Cryptocurrency: Demand Factors, Latent Value and Regulatory Developments. *Michigan Business and Entrepreneurial Law Review*, 73–100.

Cryptography. (2019, 5 17). *Definition of cryptography*. Retrieved from Merriam-Webster.com: https://www.merriam-webster.com/dictionary/cryptography

Demir, E. G. (2018). Does economic policy uncertainty predict the Bitcoin returns? An empirical investigation. *Finance Research Letters*, 145–149.

ElBahrawy, A. A.-S. (2017). Evolutionary dynamics of the cryptocurrency market. *Royal Society Open Science, 4*(11), 170623. doi:https://doi.org/10.1098/rsos.170623

Granville, K. (2018, 3 18). *Facebook and Cambridge Analytica: What You Need to Know as Fallout Widens*. Retrieved from TheNewYorkTimes.com: https://www.nytimes.com/2018/03/19/technology/facebook-cambridge-analytica-explained.html

Iansiti, M. L. (2017). The Truth About Blockchain. *Harvard Business Review*. Retrieved from Harvard Business Review: https://hbr.org/2017/01/the-truth-about-blockchain

Investing.com. (2020, 9 1). *All Cryptocurrencies*. Retrieved from Investing.com: https://www.investing.com/crypto/currencies

Kim, Y. B. (2016). Predicting Fluctuations in Cryptocurrency Transactions Based on User Comments and Replies. *PLoS ONE*, 1–17.

Kim, Y. L. (2017). When Bitcoin encounters information in an online forum: Using text mining to analyse user opinions and predict value fluctuation. *PLoS ONE*. doi:https://doi-org.proxy.library.ohio.edu/10.1371/journal.pone.0177630

Kimura, H. (2019, 1 17). *The Cryptocurrency Sectors Cheatsheet*. Retrieved from TradingHereos.com: https://www.tradingheroes.com/cryptocurrency-sectors-cheatsheet/

Mai, F. S. (2018). How Does Social Media Impact Bitcoin Value? A Test of the Silent Majority Hypothesis. *Journal of Management Information Systems*, 19–52.

Mallqui, D. C. (2019). Predicting the direction, maximum, minimum and closing prices of daily Bitcoin exchange rate using machine learning techniques. *Applied Soft Computing Journal*, 596–606.

Pauw, C. (2018, 7 24). *How Cryptocurrency Prices Work, Explained*. Retrieved from cointelegraph.com: https://cointelegraph.com/explained/how-cryptocurrency-prices-work-explained

Qazi, A. R. (2017). A systematic literature review on opinion types and sentiment analysis techniques tasks and challenges. *INTERNET RESEARCH*, 608–630.

Roth, Y. J. (2018, 7 24). *New developer requirements to protect our platform*. Retrieved from Twitter.com: https://blog.twitter.com/developer/en_us/topics/tools/2018/new-developer-requirements-to-protect-our-platform.html

Steinert, L. a. (2018). Predicting altcoin returns using social media. *PLoS ONE*, 1–12.

Sun, X. L. (2018). A novel cryptocurrency price trend forecasting model based on LightGBM. *Finance Research Letters*. doi:https://doi-org.proxy.library.ohio.edu/10.1016/j.frl.2018.12.032

Thesis and Dissertation Services