Semi-Automated Initialization of Simulations: An Application to Healthcare

Jose J. Padilla[1*], Saikou Y. Diallo[1], Hamdi Kavak[2,3], Olcay Sahin[2], John A. Sokolowski[1], and Ross J. Gore[1]

[1] Virginia Modeling, Analysis, and Simulation Center, Old Dominion University

[2] Department of Modeling, Simulation & Visualization Engineering, Old Dominion University

[3] Department of Systems Engineering, Turkish Military Academy

Abstract

As the number of variables and entities included in a simulation model increase, it becomes more difficult to initialize due to (a) the increasing number of input variables that are required and (b) the difficulty in finding, retrieving, and assigning the initial values of the input variables, especially in Human Social Cultural Behavior Modeling. As a result, the initialization process is generally more time consuming and error prone which motivates the need for semi-automated approaches wherever possible.

In this paper, we propose a semi-automated approach for initializing input variables that are challenging to quantify and require additional processing to be assigned their initial values. We apply this approach to initialize a healthcare simulation using data from sources such as the U.S. Census Bureau, County Health Rankings, and Twitter. Results show that the approach works well when we can find variables with existing values even for large input data sets (over fifty variables). However, additional work is required to determine whether the values assigned using this approach yield more accurate simulation results.

Keywords: Simulation initialization, data-mining, information extraction, social media

Submitted 11 July 2014, Revised 11 November 2014, Accepted 25 November 2014

[*] Corresponding author
Virginia Modeling, Analysis and Simulation Center, Old Dominion University
1030 University Blvd. Suffolk, VA 23435
Email: jpadilla@odu.edu – Phone number: 757-686-6213

1. INTRODUCTION

Simulation initialization has been explored in the literature under different applications. Wittman [1] highlights three levels of initialization. *Model-based initialization* "focuses on the types of data necessary to driving specific models". *Scenario-based initialization* considers elements such as organizations, actors, and organization-actor relationships played within the simulation and are the focus of the Military Scenario Definition Language (MSDL). *Federation-based initialization* considers consistent initial conditions across all data types in a federation.

In all cases, we can divide the simulation initialization problem into sub-problems:

- a *data description* problem that focuses on the characterization of inputs, including all necessary attributes for a simulation to accept each input as complete (for instance, data type and unit of measure);
- a *data discovery* problem that consists of finding all of the described data instances; and
- a *data injection* problem that consists of inputting data into a simulation such that it will run.

One approach to the *data description* problem is to use a standardized language to describe input variables. For example, the military uses MSDL as a common mechanism to initialize simulations with different scenarios. According to SISO [2], MSDL's advantage is that it improves consistency and reuse across federations. As such, MSDL provides the means of putting the right values at the right places allowing automated use of simulations. The creation and use of MSDL has shown that the *data description* problem, while important, can be solved by consensus and agreements through professional organizations.

On the *data discovery* side, many model-based frameworks have been developed for initializing specific simulations in transportation [3], manufacturing [4], and business [5]. These frameworks focus on initializing a single simulation with considerations on data sources, data collection, and data categorization while heavily relying on the presence of structured data in local databases. However, in some cases, we have to contend with instances where data is not available locally, data is incomplete, or data must be inferred from other structured and unstructured data sources.

Data injection is a trivial process when there are only a few variables to initialize. However, it may become problematic when dealing with a large number of input variables with varied scenarios to run. As a result, we need standardized data injection processes in order to cope with these simulations.

In this paper, we propose a semi-automated simulation initialization approach to deal with the data discovery and injection problems. The remainder of the paper is structured as follows. Section 2 proposes an initialization approach that can be used across domains. Section 3 describes an instantiation of the proposed approach and provides two use cases within a healthcare simulation. Section 4 provides a discussion on the proposed methods, potential issues, opportunities, and implications. Section 5 summarizes the paper and identifies areas for further research.

2. PROPOSED SEMI-AUTOMATED INITIALIZATION APPROACH

The approach we propose in this paper illustrates a generic way of initializing simulation variables through the application of a value assignment algorithm. The approach applies to all input variables and includes three major steps, which are (1) specifying a value assignment algorithm that determines the function to use to assign a value to a variable, (2) obtaining values for variables through local or online sources, and (3) applying the value assignment algorithm to initialize the variable. Figure 1 depicts the overall steps of the simulation initialization approach.

The approach starts with selecting an unassigned input variable. For each variable, we specify a value assignment algorithm that indicates how to assign a value to that variable. In the most straightforward case, the algorithm is a simple assignment operator (for example, assigning the age of a person their corresponding age value from a database). In other cases, the algorithm may require more complex operations such as creating a cumulative distribution function (cdf) using the prices of house in a given radius and randomly selecting the initial price of a home from that cdf. For every assignment algorithm, we identify key variables and look to assign them their initial values. The complexity of simulation initialization is directly related to the complexity involved in the assignment algorithm.

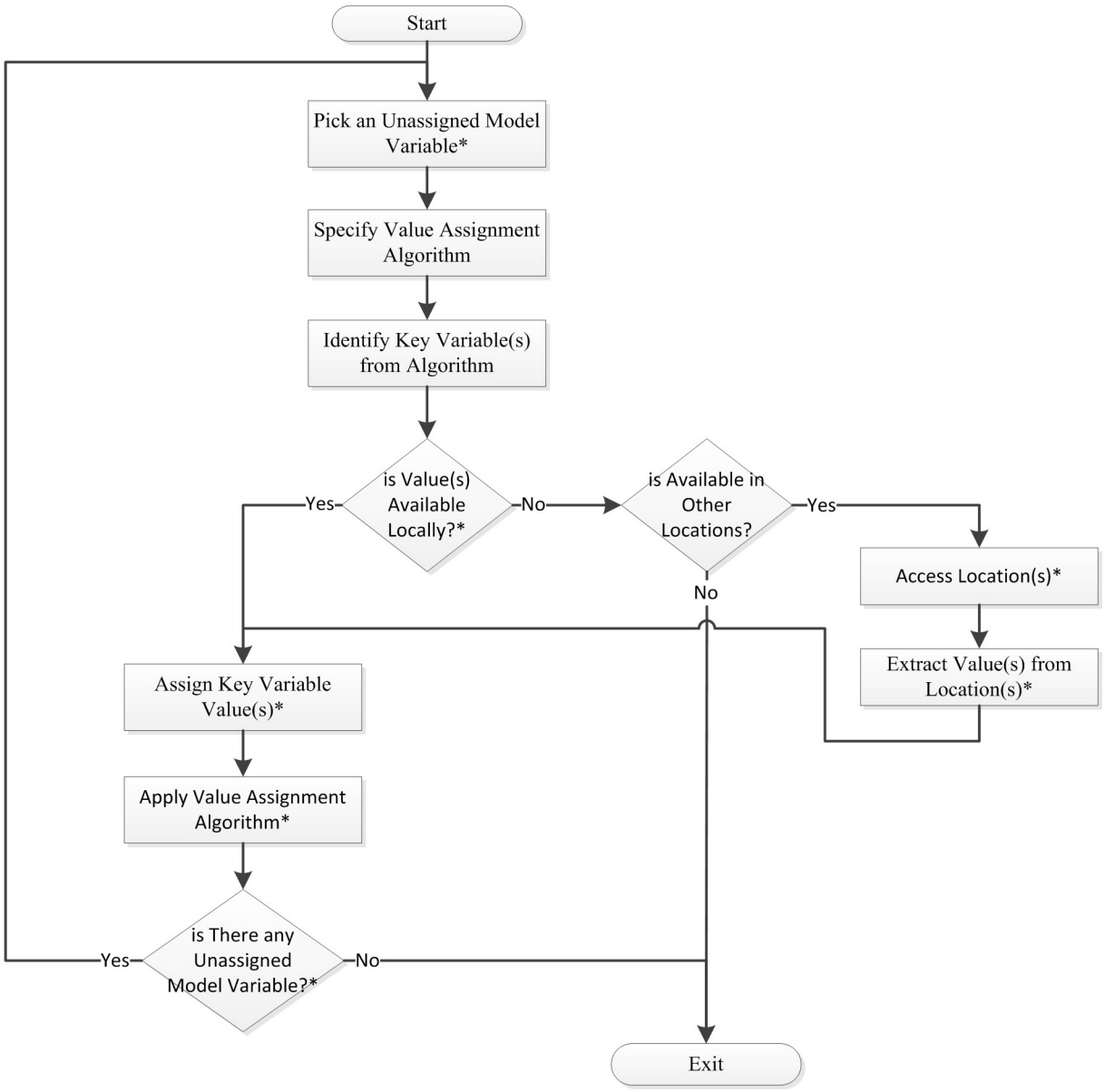

Figure 1 Overall steps of simulation initialization. The star in each box indicates a process that can be automated.

For each variable of the assignment algorithm, we need to determine whether the variable value is available locally. By locally, we mean the variable is available on a local machine or on a local network. If the value is available locally, then we assign it a value and repeat this process for every locally available value. If the value is not available locally, then we look for external locations (i.e., the World Wide Web, other networks, etc.). If we find another location, then we access that location, extract the value, and assign it to the variable. We repeat this process for every external value.

If a value is not available locally or externally, then we need to either change the assignment algorithm (i.e., use a well-known distribution function) or turn the variable into a parameter that we can explore through experimentation.

We can automate the steps that are marked with the star character (*) in Figure 1 by implementing them in a computational form (i.e., using some programming language). The potential gain in automation is twofold: (1) *increased speed* and *reduction in error* during the value assignment process, especially in the presence of a large number of variables and (2) *value extraction* when the value is only available in other locations.

3. APPLICATION TO HEALTHCARE

We apply the proposed approach to a hybrid agent-based and discrete-event model that explores the impact of obesity on the medical and healthcare system in the United States. We use AnyLogic Software (http://www.anylogic.com) to build the obesity model and run it for different cities and states under different scenarios. The goal of the obesity model is twofold: (1) determine what policies would allow obesity trends to level-off and (2) determine the tipping point at which obesity-related illnesses will exceed the medical and healthcare capacity [6, 7].

Running the simulation of the obesity model requires initializing 91 input variables divided in two groups. The first group consists of environmental variables such as population, number of restaurants, distance to those restaurants from homes and workplaces, number of exercising facilities, and the distance from homes to workplaces. The second group consists of individual variables such as food preference, weight, height, age, and gender among others. Most of these variables change based on the region being explored. In the next section, we present an instantiation of our proposed approach and two examples of initializing input variables within this model.

3.1 Instantiation of the Proposed Approach

We instantiate the proposed approach and depict its steps in Figure 2. The instantiated approach is a concrete version of the generic approach with steps specifically designed for automation where appropriate. We highlight three noticeable differences in the instantiated approach.

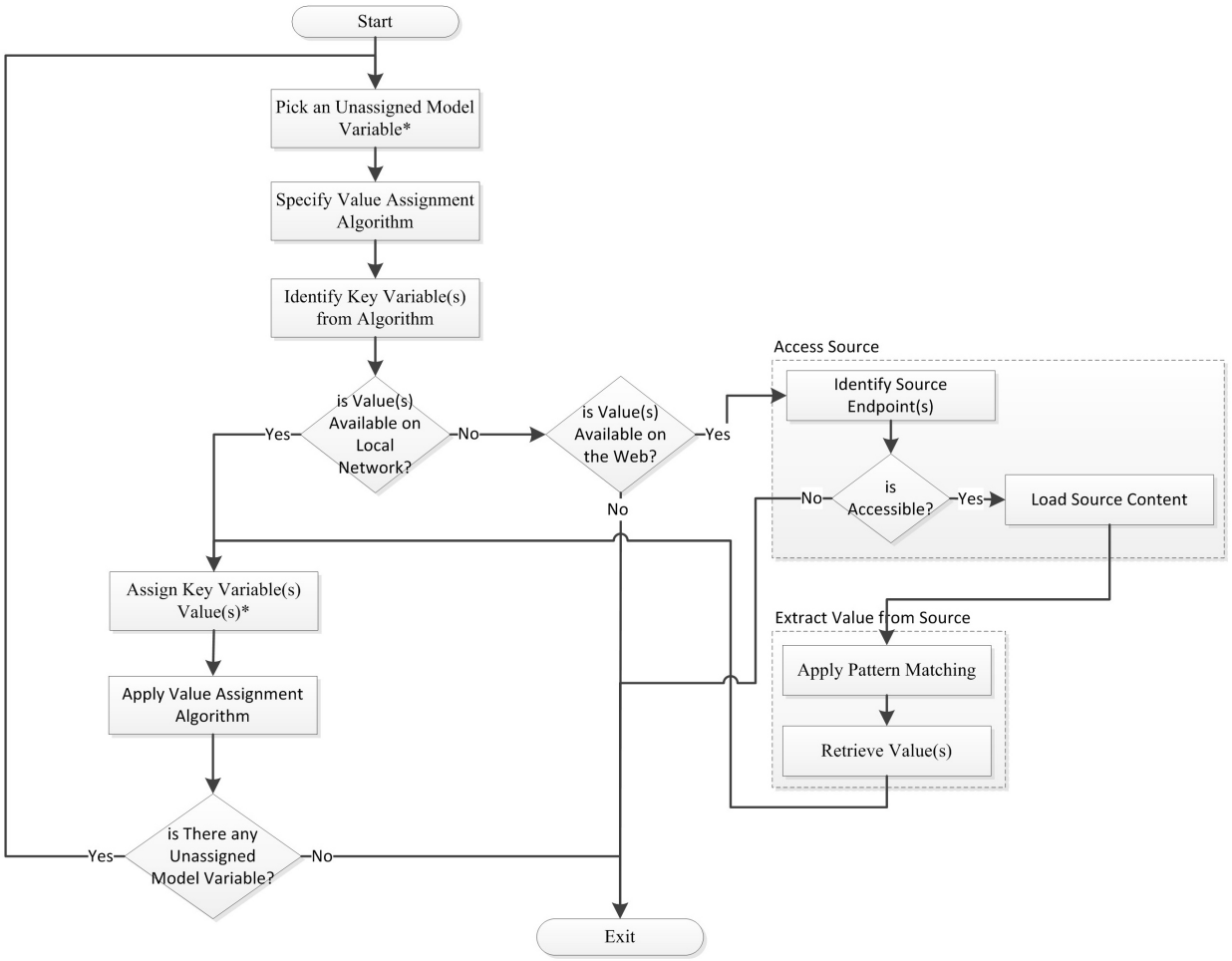

Figure 2 An instance of the proposed approach.

1. *The value availability scope:* Value availability is limited to the local network and the World Wide Web. Therefore, the approach first looks to see if key variable values are already available on the local network. When values are not available on the local network we look for values on the World Wide Web. We can automate this step if a standardized data repository (a structured database for instance) exists on the local network.
2. *Source access:* When a value is available on the Web, we first identify its endpoint, which is usually a Uniform Resource Locator (URL) referencing a web page, a web service, or an Application Programming Interface (API). If the endpoint is accessible, then we load the content and extract the value that we need. We can automate both the accessibility check and the content loading steps.

3. *Value extraction:* Typically, once loaded, content needs further processing before we can retrieve the value of interest. For that purpose, we can apply automated pattern matching to match the value that we are searching for with the retrieved content.

Based on our approach, there are two possible paths depending on the location of the values. If all key variable values are available on local networks in a well-defined structure, the approach becomes an information retrieval mechanism that loads simulation variables from a well-defined structure at a known location. We refer to this case as Simulation Initialization with Retrieval (SIR). When some values are not available on the local network and are not in a well-defined structure, we need to perform some form of data mining and analysis. We refer to this path as Simulation Initialization with Data Mining and Analysis (SIMA). SIR is straightforward and applied in everyday simulation initialization. In this example, we focus on SIMA since it is less common and more complex.

3.2 SIMA Case 1: Initializing Environmental Variables

This example covers initializing the environmental variables of the obesity model. A sample of the environmental variables from the model is listed in Table 1.

Table 1 Sample of environmental variables from the obesity model.

Number of Businesses	Number of Homes
Size of Population	Number of Primary Care Physicians
Number of Specialists	Number of Hospitals
Number of Primary Care Facilities	Expected Treatment Times for Diseases

We followed the proposed approach to generate values for each environmental input variable. We pick an unassigned input variable and specify a value assignment algorithm. The algorithm, in this case, becomes a simple assignment operator. For instance, if the input variable is the size of the population, then the assignment algorithm will assign the population number based on the region under exploration. We identify the input variable itself (i.e., population of a specified area) as a key variable from the assignment algorithm.

For each of the environmental variables, we determine that their key variable value is not available on local network so we look on external locations such as the World Wide Web. When considering all environmental input variable of the simulation, we are able to identify two sources: QuickFacts service of the US Census Bureau website and County Health Rankings website. The QuickFacts service provides statistics about people, businesses, and geography for given region. The County Health Rankings website provides statistical information of states and counties of the US for wide range of health related issues such as obesity, HIV and infant mortality as well as health system related measures, healthcare costs, and insurance. We identify two URL structures for these websites respectively.

- http://quickfacts.census.gov/cgi-bin/qfd/extract?{StateCode}
- http://www.countyhealthrankings.org/app/{StateName}/{Year}/downloads

Within these URL structures, we parameterize some parts of the URL so that it becomes easy to locate different regions and different versions of data. For instance, {StateCode} indicates 5-digit state code ranging from “01000” (Alabama) to “56000” (Wyoming), {StateName} indicates lowercase state names such as “alabama” and “virginia" and, {Year} indicates 4-digit years between “2011” and “2014”.

We load the content using the URL structures identified. Source content is in text-based format when loading file from the QuickFacts service and it is in HyperText Markup Language (HTML) file format when loading from the County Health Rankings. Within the loaded text, we extract the variable value needed using Java’s standard regular expression library (java.util.regex).

Table 2 2 shows an example method that loads and extracts a parameter from the QuickFacts service. For example, when calling the example function using parameters "51000" and "Population, 2013 estimate," it returns the population of the state of Virginia from 2013, which is “8,260,405.” After extracting the value of the variable, we assign the value to the input variable, and then repeat these steps for all input variables.

Table 2 Example Java code to load a parameter from the QuickFacts service.

```
private static String QuickFactsURLStructure =
                        "http://quickfacts.census.gov/cgi-bin/qfd/extract?{StateCode}";
private static String QuickFactsURLParamStateCode = "{StateCode}";

/**
* @param stateCode - Census state code between 01000 and 56000.
* @param paramName - Name of the parameter such as Population, 2013 estimate.
* @return Corresponding value for paramName.
* @throws IOException
*/
public String RetrieveQuickFactsData (String stateCode, String paramName) throws IOException{

        String targetURL = // identify the URL
                        QuickFactsURLStructure.replace(QuickFactsURLParamStateCode,
stateCode);
        String inputLine;
        URL censusURL = new URL(targetURL);

        // use BufferedReader to read the file
        BufferedReader censusReader =
                        new BufferedReader(new InputStreamReader(censusURL.openStream()));
        StringBuffer sbCensus = new StringBuffer();

        // read the file line by line and append each one to StringBuffer
        while ((inputLine = censusReader.readLine()) != null){
                sbCensus.append(inputLine);
        }

        // This regular expression is used to match pattern.
        // The target attribute (i.e., population) is sent as paramName parameter
        Pattern pattern = Pattern.compile( paramName+ "\\s+(.+?)\\s+");
        Matcher matcher = pattern.matcher(sbCensus.toString());

        while (matcher.find()){
                return matcher.group(1); // group(0) returns the whole part. group(1) returns the
target

        }
        return "";

}
```

This approach may not work properly in case of an accessibility problem such as broken Internet connection or the source website is not reachable. In that case, one solution is to keep caches of the extracted values stored in a local database. This database can be updated in regular intervals and serve as a local data source. When the source website is moved or the content structure is

altered, we need to notify the user so that they can modify the code to match the changes in location or structure. Assuming we have a working source website and a fixed structure, we can begin the initialization process.

3.3 SIMA Case 2: Initializing Individual Preferences

We follow the proposed approach to initialize restaurant preferences of agents within an area. In the obesity model, agents choose one of three restaurant types: fast food, non-fast food, and market. This preference is important because it has a direct effect on the potential calorie intake of agents. At initialization, the model requires that we provide the percentage of agents that prefer to eat at a given restaurant type. These preferences are represented by three input variables: (1) percentage of people eating at a fast food restaurant, (2) percentage of people eating at a non-fast food restaurant, and (3) percentage of people eating at a market. Since we have no data for restaurant preference for individuals in an area, we generate these values by conducting calibration experiments to match the historical obesity-level trends of each state between 1995 and 2010. As part of the experiment, we vary the restaurant preference percentage and record the values for the simulation run that best matches the historical data with a 95% confidence level. These values are then used during initialization to predict obesity levels and trends in the next thirty years. Using the proposed approach, we introduce an alternative way to generate this preference value.

First, we pick the percentage of people eating at a fast food restaurant and use function (1) as the value assignment algorithm.

$$\%\ of\ people\ eating\ at\ a\ restaurant\ type = \frac{\#\ of\ people\ choosing\ restaurant\ type * 100}{Total\ \#\ of\ people} \quad (1)$$

From this function, we need to determine over a sample population in a given area the number of people who prefer to eat at fast food restaurants. Similarly, we can apply this function to the number of people who prefer to eat at non-fast food restaurants or markets.

We can obtain this data through traditional means such as surveys and questionnaires. In this paper, we use Twitter (http://www.twitter.com) to infer eating preferences. Twitter is a micro-blogging website that lets users share their updates up to 140 characters. Twitter users share a variety of their personal information within their messages such as political affiliations, music

preferences, places they eat, and more. Data obtained from Twitter is useful to infer preferences of large groups of individuals [8]. We apply function (1) to the data obtained from Twitter to infer eating preferences for each restaurant type.

Twitter provides a designated webpage (https://dev.twitter.com) for developers that provide documentation on how to access its data using a variety of APIs. We use the Streaming API to collect public messages with no specific keywords. The API requires having a developer account and an application associated with it. Once all API requirements are satisfied, we develop a Java-based custom application that utilizes Twitter4J [9], a Java library to facilitate connection and retrieval from the Twitter APIs. We collect 6,504,229 tweets (17.8 GB uncompressed) in the Hampton Roads area in Virginia, USA between March 13, 2014 and June 19, 2014 (99 days) and pass them through a filtering process.

We create three filters with each filter consisting of keywords that reflect the eating preference. For instance, a tweet is classified as fast food restaurant preference if it mentions popular fast food restaurant names such as "McDonald's" and widely known menu items such as "Whopper." Similarly, a tweet is classified as non-fast food restaurant preference if it refers to common local food restaurant attributes such as "farm to table" and "gourmet". Table 3 shows the complete list of keywords.

Table 3 The specified keywords for each restaurant type.

	Keywords
Fast Food	mcdonald, dollar menu, big mac, burger king, big king, whopper, burgerking, wendy's, wendys, taco bell, cantina bell, pizza hut, ultimate cheese lover's, pepperoni lover's, meat lover's, chick-fil-a, chickfila, chick-fila, chickfil-a, chick-n-strips, kfc, original recipe, extra crispy, kentucky grilled chicken, hot wings, panera bread
Non-Fast Food	local food, farm to table, bistro, market, gourmet, restaurant
Market	homemade, home made, grocery, cooking, organic, fresh,

We consider only twitter messages containing at least one keyword and at the end of the filtering process, we add up the total number of people who tweeted and the total number of people who generated tweets in a given type. We assign the total number of people who tweeted to

population size and the total number of people who tweeted in each type to the total number of people and apply function (1) to for each type.

We implement keyword matching in Java using the regular expressions library. In addition to that, we use the Apache Commons Lang library (org.apache.commons.lang3) to join keyword arrays when creating regular expressions and the JSON Simple library (org.json.simple) to parse tweets and get particular properties from JSON objects. An example JAVA code that counts tweets with a particular keyword array is shown in Table 4 4.

Table 4 Example Java code that counts tweets containing given keywords.

```
/**
 * @param rootFolderPath - Root folder path where tweet folders reside.
 * @param keywords - an array of keywords to be searched in tweets.
 * @return number of tweets matching the keywords.
 * @throws IOException
 */
public int KeywordMatchCount(String rootFolderPath,
                String[] keywords) throws IOException{

        int count = 0;
        File aTweetFolder; File[] listOfTweetFiles;
        JSONObject tweetObj;
        String singleMessage = null; BufferedReader in ;

        // Following statement creates a pattern matching expression for given keywords
        String patternString = "\\b(" + StringUtils.join(keywords, "|") + ")\\b";
        Pattern pattern = Pattern.compile(patternString);

        // We loop through folders within the root folder
        for(String folderName: GetFolderNames(rootFolderPath)){
                // We load the tweet folder and list of files.
                aTweetFolder = new File(rootFolderPath  + File.separator+ folderName +
File.separator);
                listOfTweetFiles = aTweetFolder.listFiles();

                // We loop through list of files, each include up to 10000 tweets.
                for(File aTweetFile : listOfTweetFiles){
                        in = new BufferedReader( new InputStreamReader( new
FileInputStream(aTweetFile), "UTF8"));
                        // we read each file line by line.
                        while ((singleMessage = in.readLine()) != null) {
                                // we convert each line to a JSON object
                                tweetObj = (JSONObject)JSONValue.parse(singleMessage);
                                // then, retrieve text property, which is actual tweet message
                                String tweetText = ((String)tweetObj.get("text")).toLowerCase();
```

```
                                        // We check if the message contains any keywords given
                                        Matcher matcher = pattern.matcher(tweetText);
                                        if(matcher.find()){
                                                count++;// we increase the count number by one
                                        }
                                }
                                in.close();
                        }
                }
                return count;
}
```

We found that 22,114 messages from 13,209 Twitter users shared messages related with restaurant preferences. Table 5 shows the three restaurant preferences with the corresponding number of Twitter messages, number of users, and the percentages.

Table 5 Twitter message and user classification results.

	Number of Twitter messages	Number of Twitter users	Percentage
Fast Food Restaurants	10901	5848	44.27%
Non-Fast Food Restaurants	6644	4106	31.08%
Market	4569	3255	24.65 %

According to the data we collected, 44.27% of the users tend to go to fast food restaurants, 31.08% of the users tend to go to non-fast food restaurants, and 24.65 % of the users tend to choose markets. We use this data to run a Monte Carlo simulation of the obesity model to 1) evaluate the accuracy of the model in predicting the obesity level in the Hampton Roads and 2) compare the model's predictions with the values against the predictions obtained from using the calibration values. Table 6 shows a comparison between the initial values obtained from Twitter and those obtained experimentally.

Table 6 Comparison of initial values assigned to the variables.

	Twitter	Calibration
Fast Food Restaurants	44.27	63.58
Non-Fast Food Restaurants	31.08	36.42
Market	24.65	0

Although the Twitter data shows a lot more nuance and credibility, Figure 3 shows that its ability to calibrate against existing data is not as good as the experimental data.

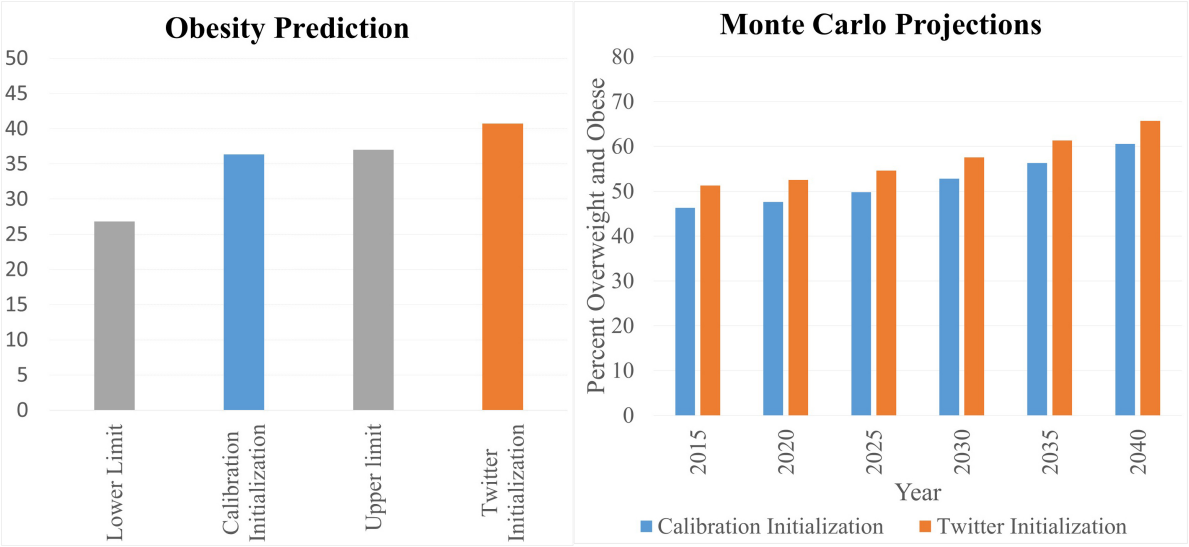

Figure 3 (Left) Prediction of the obese population percentage for the year 2013. (Right) Obese population percentage projection of the simulation up until the year 2040. In each year, left and right bars represent calibration initialization and Twitter initialization respectively.

The image on the left shows prediction results for the Hampton Roads area when we run the simulation for the year 2013 based on 2010 data. The image on the right shows model projections until the year 2040. In the prediction case, The Twitter-based initialization predicted 40.7%, about 4% off from historical limits that fall between 26.8% and 37% (obtained from County Health Rankings). The calibration-based initialization approach predicted 36.3% for 2013. In the projection case, the Twitter-based initialization projected more overweight and obese population than the calibration-based initialization. We discuss the impact of these findings in the next section.

4. DISCUSSION

Social media sites, such as Twitter and Facebook, are potential data source alternatives and complement to traditional approaches such as conducting surveys or consulting subject matter expert. They have the advantage of providing access, with some restrictions, to large and longitudinal data pools. In our model, we use messages collected from Twitter to capture preferences of individuals residing in an area. According to the prediction results given in Figure 3 (left), the simulation initialized with Twitter-based data performs 3.7% above the given upper bound whereas the experimental data obtained from calibration is within the confidence interval. According to projection results given in Figure 3 (right), the simulation initialized with the

Twitter-based data projects that over time, there is a higher incidence of obese and overweight compared to the calibration-based projection which is consistent with existing projections. Therefore, if we believe existing projections, the use of Twitter data results in over projections in the short term but might be more accurate in the long term.

The discrepancy between the Twitter results and the experimental results might be attributed to several factors. The Twitter data we collected is relatively small for the area compared to the approximate 1.7 million population residing within the area. We are able to get a daily average of approximately 65,000 messages but only 220 of these messages on average are related to restaurant preferences. These numbers might be an indication that only a portion of the population uses social media to share eating preferences and therefore the data we ultimately use for initialization is not representative of the preferences of the general population. In order to increase accuracy, we have collected Twitter messages from the whole US population for a total of 3.5 million data points (9.5 GB uncompressed) daily. Considering four months of data collection for the whole U.S. population, we can easily approach the Terabyte range. We are currently processing this data but wanted to highlight the size of the data involves in this approach.

Another potential reason for this result is the keyword-based classification of the tweets. We could improve it in two ways. First, we could add more keywords related to each of three classes, which would result in more tweets being used. Second, we could introduce a supervised learning algorithm to classify tweets as either one of the three categories or as unrelated. This process includes identifying a training set, building a model, training the model, testing the accuracy of the model, and classifying the rest of the messages. It is important to note that we are comparing results based on a simulation. Ideally, we would compare against empirical data, but data for this type of variables is not available hence the proposed approach.

These limited experiments show that web-based data provides an alternative way of inferring data. However, we need to use caution when it comes to the representativeness of the target population in the collected data. The approach we propose in this paper provides a means of formalizing the initialization process and retrieving initialization related data from local and online data sources. Further, it semi-automates the steps to initialize simulations. One of the crucial points here is ability to use web-based sources to initialize simulation variables.

In the first case, we utilize the US Census Bureau and Country Health Rankings as data source among many other online sources. For instance, The World Health Organization (http://apps.who.int/gho/data/node.main) provides numerous data repositories about broad statistics such as child mortality rate as well as specific statistics such as number of mumps cases per country. CIA World Factbook (https://www.cia.gov/library/publications/the-world-factbook) serves a wealth of country statistics such as age distribution, population growth rate, birth rate, death rate, infant maternal mortality rate, and life expectancy. When choosing a data source, we need to use caution about the credibility of the source and the method that the source followed when collecting the data.

In the second case, we initialize input variables that are challenging to quantify using messages collected from Twitter. Data obtained from social media websites like blogs or micro-blogging services are appearing in the literature for analysis of meme diffusion [10] and validation of models [11]. Yet, a limited number of studies [12] use them for simulation initialization as input data.

Studies also show that it is possible to gather healthcare related measures using a variety of web-based data. Ginsberg et al. [13] demonstrate how search engine queries can help gathering influenza epidemic trends. Corley et al. [14] identify influenza-like illness trends by mining online blogs and Signorini, Segre and Polgreen [15] do so using Twitter data. Other than epidemics, it is possible to get different health related measures from the Web; for example, Paul and Dredze [16] use Twitter data to generate measurements about ailments such as allergies, depression, cancer, and obesity. If these measures are used as simulation inputs, we can benefit from them because they provide 1) near real-time measurements and 2) the ability to scale the area that is under exploration.

While the proposed approach provides opportunities, it also generates challenges like establishing whether the sampled data is representative of the initialization variable and having the technical capabilities to process large datasets. Here, large dataset refers to data “that may not fit the main memory of the computer” [17]. Mislove et al. [18] show the challenge of sampling bias in data obtained from Twitter. The sampling bias requires dedicated efforts like the one made by Culotta [19], which aims at reducing the bias. The challenge of processing large

datasets may require using non-conventional data processing and storage tools such as Apache Hadoop and MongoDB as well as more computing power.

Hadoop is a scalable and distributed storage and computing system, which provides two main subsystems called the Hadoop Distributed File System (HDFS) and MapReduce. The HDFS is "designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user application" [20]. "MapReduce is a programming model and an associated implementation for processing and generating large datasets that is amenable to a broad variety of real-world tasks" [21]. Hadoop is suitable to provide analysis capabilities in petabytes scale [20]. Important advantages of Hadoop are (1) the ability to process data in "hundreds of megabytes, gigabytes, or terabytes in size", (2) the ability to run on inexpensive hardware, and (3) linear scalability [22]. On the other hand, Hadoop is not a well-fit solution there is a need for low-latency access to data and when there are many small sized files [22]. Our preliminary experiments show that Hadoop provides high scalability with twitter data analysis when tweets are combined into single files rather than having a separate file per tweet.

MongoDB is "a database management system designed for web applications and internet infrastructure" [23]. What makes it attractive is the ability to store documents in JSON-like structure and the "ability to scale easily with automatic failover" [23]. Nowadays, many social media website APIs provides data in JSON format, which can be stored in MongoDB without modification. MongoDB also provides ability to query this JSON data as well as standard database features like indexing and aggregation. In our case, we can use MongoDB to store JSON-based data acquired from the Web and provide fast filtering capabilities without explicitly creating a database structure. Tools like MongoDB Connector for Hadoop provide a close connection between the two technologies where MongoDB acts as an input data source or output destination for Hadoop.

5. CONCLUSIONS AND FUTURE WORK

Semi-automating the simulation initialization process is crucial because it can reduce time and error. It is important for simulation models that support a large number of scenarios and have a large number of variables. In some cases, we might need to initialize qualitative variables such as

population sentiments and preferences and in those cases researchers often rely on assumed probability distributions functions.

This paper proposes an approach for semi-automated simulation initialization. The approach facilitates the retrieval of content from structured and unstructured data sources and generates the input data available to simulations. The approach assists the user with the process of dealing with a large number of variables and different initial conditions while having the characteristic of relying on large data sets analysis (terabyte, petabyte) using existing technologies like Hadoop and MongoDB in order to make sense of such data.

We show two use case applications in a simulation model of the obesity epidemic. The first use case is to obtain data from reliable sources for initializing demographic and health statistics for specified areas. The second case is to capture the preferences of people on particular types of eating locations. Future work focuses on evaluating the reliability of social media data at different geographical scales (city, state, and country) for initialization and calibration as well as validation.

ACKNOWLEDGEMENT

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors. We thank the anonymous reviewers for evaluating the paper and for their valuable comments towards improving the quality of the paper.

REFERENCES

1. Wittman RL. Standards-based combat simulation initialization using the military scenario definition language (MSDL). In TOLK A (ed.) *Engineering Principles of Combat Modeling and Distributed Simulation*. John Wiley & Sons, Inc., 2012, pp. 579-605.
2. SISO. Standard for: military scenario definition language SISO-STD-007-2008. 2008.
3. Palagummi V, Fujimoto R and Hunter M. Techniques for rapid initialization in in-vehicle traffic simulators. In *Proceedings of the Winter Simulation Conference*, 2009, pp. 2446-56.
4. Bergmann S, Stelzer S and Straßburger S. Initialization of simulation models using CMSD. In *Proceedings of the Winter Simulation Conference*, 2011, pp. 2228-39.
5. Robertson N and Perera T. Automated data collection for simulation? *Simulation Practice and Theory*; 2002, 9: 349-64.
6. Sokolowski JA, Banks CM, Diallo SY, Padilla JJ and Lynch CJ. A methodology for engaging modeling and simulation to assess a corollary problem to the obesity epidemic.

In *Proceedings of the International Workshop on Applied Modeling & Simulation*, 2012, pp. 30-37.
7. Sokolowski JA, Banks CM, Diallo SY, Padilla JJ and Lynch CJ. A simulation analysis to weigh the impact of obesity: corresponding patient need with medical capacity. In *Proceedings of the Summer Computer Simulation Conference*, 2013, pp. 1-8.
8. Golbeck J and Hansen DL. Computing political preference among twitter followers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2011, pp. 1105-1108.
9. Yamamoto Y. *Twitter4J - A java library for the twitter API*, http://www.twitter4j.org.
10. Colbaugh R and Glass K. Early warning analysis for social diffusion events. *Security Informatics*; 2012, 1(18).
11. Weng L, Flammini A, Vespignani A and Menczer F. Competition among memes in a world with limited attention. *Scientific Reports*; 2012, 2(335).
12. Padilla JJ, Diallo SY, Kavak H, Sahin O and Nicholson B. Leveraging social media data in agent-based simulations. In *47th Annual Simulation Symposium*, 2014, pp. 71-80.
13. Ginsberg J, Mohebbi MH, Patel RS, Brammer L, Smolinski MS and Brilliant L. Detecting influenza epidemics using search engine query data. *Nature*; 2008, 457(7232): 1012-1014.
14. Corley CD, Cook DJ, Mikler AR and Singh KP. Text and structural data mining of influenza mentions in web and social media. *International journal of environmental research and public health*; 2010, 7(2): 596-615.
15. Signorini A, Segre AM and Polgreen PM. The use of twitter to track levels of disease activity and public concern in the US during the influenza A H1N1 pandemic. *PloS one*; 2011, 6(5), e19467.
16. Paul MJ and Dredze M. You are what you tweet: analyzing twitter for public health. In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, 2011, pp. 265-272.
17. Babu TR, Murty MN and Subrahmanya SV. *Compression schemes for mining large datasets*. London: Springer-Verlag, 2013.
18. Mislove A, Lehmann S, Ahn YY, Onnela JP and Rosenquist JN. Understanding the demographics of twitter users. In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, 2011, pp. 554-557.
19. Culotta A. Reducing sampling bias in social media data for county health inference. In *Joint Statistical Meetings Proceedings*, 2014.
20. Shvachko K, Kuang H, Radia S and Chansler R. The hadoop distributed file system. In *IEEE 26th Symposium on Mass Storage Systems and Technologies*, 2010, pp. 1-10.
21. Dean J and Ghemawat S. MapReduce: simplified data processing on large clusters. *Communications of the ACM*; 2008, 51(1): 107-113.
22. White T. *Hadoop: the definitive guide*. Yahoo Press, 2012.
23. Banker K. *MongoDB in action*. Shelter Island, New York: Manning Publications Co., 2011.

AUTHOR BIOGRAPHIES

Jose J. Padilla is a research assistant professor at the Virginia Modeling, Analysis, and Simulation Center at Old Dominion University. He received his Ph.D. in Engineering Management from Old Dominion University.

Saikou Y. Diallo is a research assistant professor at the Virginia Modeling, Analysis, and Simulation Center and adjunct Professor of Modeling, Simulation, and Visualization Engineering at Old Dominion University. He received his M.S and Ph.D. in Modeling and Simulation from Old Dominion University.

Hamdi Kavak is a Ph.D. student at the department of Modeling, Simulation, Visualization, and Engineering at Old Dominion University.

Olcay Sahin is a Ph.D. student at the department of Modeling, Simulation, Visualization, and Engineering at Old Dominion University.

John A. Sokolowski, Ph.D. (Modeling and Simulation Engineering) is the Executive Director for the Virginia Modeling, Analysis, and Simulation Center (VMASC) of Old Dominion University, supervising 50 researchers and staff with an annual funded research budget of $10 million. He administers research and development in Transportation, Homeland Security, Defense, Medical M&S, Decisions Support, Business & Supply Chain, and Social Science M&S applications. He is contributor and co-editor of Principles of Modeling and Simulation (Wiley, 2011) and Modeling and Simulation: A Multidisciplinary Approach (Wiley, 2009).

Ross J. Gore is a research assistant professor at the Virginia, Modeling Analysis, and Simulation Center at Old Dominion University. He holds a B.S. in Computer Science from the University of Richmond, and a M.S. and Ph.D. in Computer Science from the University of Virginia.