TUTORIAL: TEXT ANALYTICS FOR SIMULATION WITH PYTHON

Dr. Roger McHaney

Kansas State University
Management Information Systems
1301 Lover's Lane
Manhattan, KS USA 66506
mchaney@ksu.edu

ABSTRACT

Text-based data analytics offers many opportunities related to simulation practice. This tutorial describes these with three primary examples: (1) using sentiment analysis to drive a simulation; (2) using social network analytics to structure a simulation; and, (3) developing model timelines using Twitter analysis. The tutorial includes all aspects of data analytics including data collection, cleaning, analysis and use---all specifically adapted to simulation practitioners. The tutorial uses a hands-on approach with Python in Jupyter notebook and Spyder environments and data sources as JSON, CSV, Twitter/social media, and web scraping/crawling. Other techniques such as setting up a cloud-based data collection platform are also described. The three examples use VADERsentiment, Pandas, word clouds, text analytics, topic modeling, social network analytics and other tools. Example code is provided during the tutorial. Those attending the tutorial will learn ways to enhance the human element in their models based on real-world data sources.

Keywords: Data Analytics, Text Analytics, Simulation Applications, Python, Gephi

1 INTRODUCTION

Text-based data analytics offer a wide range of opportunities for understanding system operation and enhancing analysis techniques. This is true in many fields, including simulation. For instance, simulation practitioners may need to develop mechanisms that represent the frequency, categories, types, and nuances of data that enter a system. This is particularly applicable to situations where human factors come into play. Text analytics offer insights into data regarding its polarity, emotional content, topical aspect and overall intent. Data such as these may be used to drive human factor modeling and allow better representation of the unpredictable nature people bring into systems.

Many traditional data collection and analysis techniques can be supplemented or replaced with automated processes influenced by the widespread proliferation of text-based data sources such as web sites, social media, Twitter, emails, closed captioning, business documents, contracts, resumes, proposals, PDF files and many others. The current tutorial leverages this phenomenon and focuses on data collection, cleaning, analysis, and use adapted to simulation practitioners and researchers. Many of the techniques and practices described in this tutorial have broader application but all three primary examples are tailored to model development and use.

1.1 Simulating Human Behavior

Simulating human behavior is a challenge and perhaps even limitation in many models. In fact, Baines et al. (2004) suggested that human aspects were rarely included in discrete event simulations and this represented a major shortcoming in many studies. More than a decade later, Greasley and Owen (2018) found this challenge remains and requires more work.

Several approaches related to big data can help mitigate challenges related to the human element in models. For instance when building a model of cybersecurity-related policy, Twitter can be

analyzed for sentiment to reveal polarity (e.g. negative or positive feelings), future intent, or other important topics related to the human choices and potential behaviors (Gupta, Sharma, & Chennamaneni, 2016). Findings such as these can be incorporated into a model to realistically represent the ways people may react to policy changes. Likewise, in models where feelings, motivations, and actions are based on human factors, text analytics has the potential to provide realistic data for driving the model. For example, a model of consumer behavior related to purchasing airline tickets could be improved by incorporating consumer sentiments derived from real world datasets (Khan & Urolagin, 2018).

1.2 Text-based Analytics in the Modeling Lifecycle

Text-based analytics techniques fit well into several phases of the simulation modeling lifecycle. Prior research into DES modeling has used a 4-stage model to facilitate understanding in other areas (Tako & Kotiadis, 2015). We will use the same framework in this tutorial to communicate where text analytics might be applied in the modeling process. Table 1 provides examples.

Table 1 *Text Analytics Applications Mapped to DES Stage (Tako & Kotiadis, 2015, p. 557)*

DES Stage	Text Analytics Applications
Problem structuring/ Conceptualisation	Problem formulation/discovery, data acquisition, cleaning, and processing; Visualizations of input data for better understanding of system and conceptual model
Simulation model coding	Transforming data into model constructs
Obtaining solutions	Visualizations based on output data from model
Implementation	Transference of knowledge gained; Storytelling

In general, text analytics techniques can be used to discover problems, collect data for model constructs, and provide understanding in early model stages. Later, input data and acquired knowledge can be used in model construction. Model outputs can be analyzed using text analytics tools, and finally the outcomes of the model can be communicated using storytelling techniques common to data analytics.

1.3 Tutorial Approach

The current tutorial starts with building a data analytics environment for use by simulation analysts. Real world data can be obtained for use in modelling from a variety of sources. In some instances, it may have to be derived from unstructured data sources that require careful collection, cleaning, and interpretation. This environment used in the tutorial relies on Anaconda, which is an open source implementation of Python and R programming languages for scientific computing. We use Jupyter Notebooks and/or Spyder within Anaconda's distribution to run Python scripts that assist with text analytics in this tutorial.

Within this environment, we will cover several powerful tools for text analytics. We start with data acquisition using CSV and JSON data files, Tweets from Twitter, web scraping and crawling, and other sources. Collected data will be cleaned using a variety of tools that remove stop words, short words, irrelevant tokens, and other non-meaningful symbols. Further data analysis will put the collected and cleaned data into a form ready for further analysis. This includes lists and Pandas data frames. Specific analysis takes place next. Techniques described in the tutorial will include descriptive measures, readability, sentiment analysis, text analytics (e.g. supervised and unsupervised machine learning), topic modeling, and others. Environments used in visualization and storytelling (such as

Microsoft Power BI and Tableau) will be described at this stage. Finally, the data will be transformed for use in simulation models to illustrate usefulness to modelers.

1.4 Starting Correctly

One of the most important steps in determining what tools to use and how to approach data collection, involves understanding an overall project goal. Robinson points out two important aspects of accomplishing this: first, developing a comprehensive problem description and second, formulating project goals (Robinson, 2008; Robinson, Arbez, Birta, Tolk, & Wagner, 2015). This will enable a simulation team to better understand what data to collect supported by project goals and purpose and will drive the storytelling function.

2 TOOLS FOR TEXT ANALYTICS

Text analytics requires collecting, cleaning, and working with unstructured data that comes from many sources in a variety of formats. Frequently, this data is intended for uses that do not consider collection and analysis. Therefore, a flexible tool set is required to work effectively in this area.

The current tutorial focuses on the Python programming language which has been specifically adapted to data analytics tasks with libraries, objects and scripts. We use the Anaconda distribution of Python and focus on the Jupyter Notebooks IDE (integrated development environment). An alternate approach would be to use 'R' and its scripts and toolsets. Another potential tool is LIWC (McHaney, Tako, & Robinson, 2018).

Python is an object-oriented, programming language. It originally was considered a scripting language associated with web and app development but has evolved into a high-level, widely used general purpose programming language. It has an easily understood syntax that lends it to being highly readable and, in some ways, self-documenting. Python supports user developed packages that can be installed and implemented in specialized ways. Most features used for data analytics fall into this category. For instance, Numpy, Pandas, Matplotlib, and Scikit-Learn are popular packages used in data science applications. Adding to the appeal, many Python packages, its standard library, and interpreter all are free.

Anaconda is a popular deployment of Python for data science applications. This is an open source distribution that includes both Python and R programming languages. Its primary purpose is to simplify package installation, management and deployment. Anaconda makes it possible to easily use more than 1500 data science libraries and packages. It runs on computers with Windows, Linux, or Mac operating systems. Anaconda's Python distributions include an interpreter and several Python editors (e.g. Jupyter Notebook and Spyder) as well as numerous Python tools and packages. The Anaconda distribution for data science can be downloaded from this website (Figure 1): <https://www.anaconda.com/distribution/>.

Figure 1 Anaconda Distribution for Data science Download Website

At the time this tutorial was developed, the current version of Python was 3.7. The first step in installation is to press on the appropriate installer for your computer. Figure 2 indicates saving the download and then doubleclicking on the file to initiate the local installation process. Press *run* as shown in Figure 3 then the Anaconda installer begins.

Figure 2 Download File and Double Click to Install

Figure 3 Verify Installation before Anaconda Installer Begins

As shown in Figure 4, agree to the open source license and install as *Just Me* to avoid problems later when running packages on your computer for this tutorial. Finally, choose an installation location that provides read/write permission like `C:\Users\yourname\anaconda3` in Windows environments.

Figure 4 Select Options to Ensure Usable Installation

Finish the installation by adding Anaconda to the PATH environment variable by checking the boxes under advanced options (Figure 5). After that, proceed with the defaults on the following step.

Figure 5 Add Anaconda to PATH environment variable

Once the environment is installed, Python can be accessed and used through Jupyter Notebook. It runs in several ways. It can be started from the command prompt by typing: *jupyter notebook*, or from the Windows Start Icon as shown in Figure 6, or from within Anaconda as shown in Figure 7. Once Jupyter Notebook starts, it will appear in a browser window as shown in Figure 8.

Figure 6 Starting Jupyter Notebook from Windows start icon

Figure 7 Starting Jupyter Notebook from Anaconda Navigator

Figure 8 Jupyter Notebook File Browser Window

Example files for this tutorial can be edited and run by clicking on names from the Jupyter Notebook file listing window.

3 DATA ACQUISITION

Unstructured text data can be acquired from several sources. It can be input directly into Python code. It can be read from files including CSV, Excel, and JSON. Or it can be scraped from web sites or obtained using an API such as the one provided by Twitter. The following sections provide examples of each.

3.1 Data Loaded Directly into Python

Perhaps the least efficient way to acquire data for further analysis is by pasting it directly into Python code. This method can be useful when smaller data sets are used or when cutting and pasting is the only method available. The following lines of code provide an example of how this might appear:

```
Text1 = "Line 14 produced 12 errata pieces today."
Text2 = "It was a hard day at work, we had 3 people die in the emergency room."
Text4 = "Nothing happened at all. It was very boring."
```

3.2 Data Loaded from a File

A more commonly used method is to read existing information from a file. The following lines of code demonstrate this for data contained in a csv file:

```
# Create an empty list
DataList = []

# Open a CSV file
openfile = open('data/facultysalary.csv', 'r', encoding = "ANSI")

# Read the file into an object
r = csv.reader(openfile)

# Process each line and append to a list
for i in r:

    # Append the data to the list
    DataList.append(i)

# Close the file
openfile.close()
```

Other file types can be used as well. For example, JSON is a very popular file format, particularly when web services or API interfaces are used. The following code provides example JSON formatted data (that data would be found in an external file rather than in Python code):

```
data = [

    {
        "display": "Student JavaScript Tutorial",
        "url": "http://www.w3schools.com/js/default.asp"
    },

    {
        "display": "Student HTML Tutorial",
        "url": "http://www.w3schools.com/html/default.asp"
    },

    {
        "display": "Student CSS Tutorial",
        "url": "http://www.w3schools.com/css/default.asp"
    } ]
```

3.3 Data Loaded from Twitter or other APIs

Many social media services and other information-rich sources offer APIs (application program interfaces) which include routines, protocols (e.g. rules), and tools for accessing data and other objects. For example, Twitter provides an API for gathering Tweets. To collect Tweets, it is necessary to get a Twitter Developer account. The following steps detail methods for doing this:

- 1) Obtain a regular Twitter Account (if you don't have one already);
- 2) Go to the Twitter Developer Website here:

<https://developer.twitter.com/en/apply-for-access.html>

- 3) Click on the Apply Button as shown in Figure 9;

Figure 9 *Apply for Twitter Developer Account*

- 4) Be sure to indicate your account is for academic use. More information about the application process can be found here: <https://docs.inboundnow.com/guide/create-twitter-application/>.

Once the Twitter Developer account is approved (which may take several days), you will have access to API keys which permit you to acquire data. To access your API keys, login to your account at: developer.twitter.com and then go to *App* as shown in Figure 10.

Figure 10 *Getting to your Twitter API keys. First Apps, then to Details*

Then, the Keys and Tokens can be accessed and used in Python scripts or other programmatic ways. See Figure 11

Figure 11 *Get Keys and Tokens for Use in Scripts and Software*

Figure 12 provides a screen shot of a Python script that uses Twitter API keys to aid in data collection. The script can be customized to collect data based on hashtags, keywords, users or other items of interest.

Figure 12 *Use API keys Python Script to Aid with Data Collection*

For applications that might require a long period of time for data collection, running the script in a cloud environment may be necessary. Google, Microsoft, and Amazon all provide cloud facilities that can host data analytics applications. For this tutorial, we use a free cloud-based environment called PythonAnywhere (<https://www.pythonanywhere.com>). Figure 13 shows where a limited version of this cloud-based tool can be accessed with a free account.

Figure 13 Use PythonAnywhere to Run Data Collection for Long Periods of Time

3.4 Data Loaded from Web Scraping and Crawling

Another source of data used in unstructured data analytics is web pages. Data that is displayed on a website can be either scraped or crawled depending on the way the HTML is structured. Web scraping generally refers to obtaining data from a single page whereas crawling involves setting up a program that scraps multiple pages. Web scraping can be accomplished using a query tool called XPATH. This tool aids in drilling into the page's HTML structure and programmatically retrieving desired information. Figure 14 provides an example of XPATH. XPATH code can be tested using a sandbox such as the one found at: <https://www.freeformatter.com/xpath-tester.html>

//li[a]	Selects the li elements which enclose an a element.
---------	---

Figure 14. XPATH Statement

3.5 Data Loaded from Files

Often, data is obtained from an organization's data warehouse or other sources and may be found in JSON, CSV, or Excel files. Each can be readily imported using a set of Python commands. Figure 15 provides an example.

```
# Define list
sms = []

# Get data from csv file
openfile = open('data/smsdata_classification.csv', 'r')

r = csv.reader(openfile)

# Put into list
for i in r:
    sms.append(i)
    sms_data.append(i[0])
    sms_labels.append(i[1])

openfile.close()
```

Figure 15 Sample Code to Load CSV File

4 DATA CLEANING

A primary activity of data analytics is to clean captured data. In most cases, the data will contain a variety of special characters, white space and other formatting that must be removed. Python provides several tools for accomplishing these tasks so ‘clean’ data can be stored in lists or data frames for later analysis. Pandas is a popular Python library with tools for cleaning data, fixing missing values, organizing data, and changing data types. Other tools can be used as well. For instance, Python has functions for making letters upper or lower case. Likewise, Python includes a library of commands, called RegEx, or Regular Expressions, that are sequences of characters used to form search patterns or perform character replacement/substitutions in strings. Figure 16 provides a brief example of built-in functions and a RegEx string to clean data.

```
# Lowercase
tokens = tokens.lower()

# Remove useless numbers and alphanumerical words
tokens = re.sub("[^a-zA-Z0-9]", " ", tokens)
```

Figure 16 *Sample Code to Help Clean Data*

5 DATA ANALYSIS

After collecting data and ensuring it has been cleaned and made ready for analysis, several different things can be done with it. Descriptive statistics can be derived from unstructured data. These include frequency distributions for words, topics, sentences, and other tokens. Likewise, word clouds can be developed to communicate outputs based on frequency.

Many times, the data includes words that do not contribute to the meaning of the analysis. These are words known as stop words and can be removed from the data. Other words have different forms due to plurals, tense and other features. They can be combined into a common form for analysis. This is done through stemming or lemmatizing. Code examples provided in the tutorial provide examples of performing these operations in ways useful to simulation practitioners.

Following cleaning, the normalized data set can be used for sentiment analysis, which provides information about positive or negative feelings, future intentions, and topics related to sentiments. Other text analytics can be performed to classify unstructured data into categories using machine learning techniques, text analysis, topic modeling, and social network analysis. These types of analyses can be useful in development of simulations and models. Specific code examples will be provided in the tutorial.

6 EXAMPLES RELEVANT TO SIMULATION

Unstructured data can be useful in simulation models, particularly when dealing with human elements and decision making. For instance, real world review data can be used to derive a distribution of happy versus unhappy customers. The distribution can be incorporated into a model to determine different patterns of behavior. Likewise, intention analysis, can be used to drive model entity behavior according to real world distributions.

Other useful information relevant to simulations includes timing data. In other words, the rate with which comments are made in Twitter or other social media platforms is useful. Modeling consumer behavior can be based on product reviews or other information from a variety of text-based sources. Likewise, text analytics and topic modeling can be used as sources for data built into simulation models. For example, text analytics provides classification models. Various attribute values attached to simulation entities can be used for categorization and ultimately direct entities down different paths

within a model. Topic modeling can be used to better understand dynamics within a corpus of unstructured data. For instance, using topic modeling to analyze adverse medical event data can help modelers structure their simulation in ways that match reality more closely, and provide categories and distributions for the occurrence of particular events (Zhu et al., 2019). The current tutorial will provide code examples for each of these categories to further illustrate usefulness to simulation development. Three examples are provided in the following subsections.

6.1 Sentiment Analysis to Drive Model Human Factors

Sentiment analysis uses computation techniques to identify and classify opinions expressed in text messages or social media. The outcome determines the attitude of the writer toward the subject of the analyzed message. Usually the outcome will be positive, negative or neutral. The current tutorial will demonstrate how to scrape text from a set of product reviews then use that material to build a distribution for use in a model. As brief example, Figure 17 shows use of vaderSentiment to build scores based on customer comments. The resulting scores are loaded into EasyFit to develop a distribution (Figure 18) for use in a model where a decision point is located. As can be seen, the customers either loved or hated the product.

```

vaderSentiment
• http://www.nltk.org/howto/sentiment.html

In [1]: # Import Needed Libraries from Natural Language Tool Kit
import nltk
from nltk.corpus import stopwords

# Sentiment analysis items required (e.g Vader)
from nltk.sentiment.vader import SentimentIntensityAnalyzer

In [2]: comments = ["I hate this product and its taste.", # positive sentence example
                    "I absolutely love wheatabix.", # positive sentence example
                    "Wheatabix is my top choice and I like it alot", # punctuation emphasis handled correctly (sentiment intens
                    "Buy me more Wheatabix! Healthy and delicious.", # booster words
                    "Nasty stuff that Wheatabix. DAMN nasty.", # emphasis for ALLCAPS handled
                    "Wheatabix is my fav! Hate it!!!!", # combination of signals - VADER appropriately adjusts int
                    "Wheatabix sucks and I hate it. FRIGGIN Nasty stuff!!!", # booster words & punctuation
                    "Wheatabix is good.", # positive sentence
                    "Wheatabix is kind of good.", # qualified positive sentence is handled correctly
                    "Wheatabix was good, but the consistency is unappealing and the flavor is not great.", # mixed negation sentence
                    "At least it isn't a horrible cereal.", # negated negative sentence with contraction
                    "Make sure you :) or :D today when you eat wheatabix!", # emoticons handled
                    "Wheatabix SUK!", # negative slang with capitalization emphasis
                    "Wheatabix only kinda sux! But I'll get by on it, lol" # mixed sentiment example with slang and constrastive c
                    ]

analyzer = SentimentIntensityAnalyzer()
# Get Sentiment Scores for Each Data Item
for comment in comments:
    vs = analyzer.polarity_scores(comment)
    print(vs["compound"])

```

```

-0.5719
0.6697
0.5106
0.7888
-0.8918
-0.4344
-0.9136
0.4404
0.3832
-0.5409
0.431
0.8633
-0.5461
0.2228

```

Figure 17 Python VaderSentiment Application

Figure 18 Data from Sentiment Analysis Fit to Beta Distribution

6.2 SNA Approach to Structure Model

Social network analysis investigates social structure and formation with network development based on graph theory. The result of a network analysis is a set of nodes (e.g. actors or entities) and edges (e.g. relationship or links). SNA is useful in model structure development and can help inform developers of an agent-based network or communication within a network of entities. Figure 19 provides a glimpse of a network of station communications developed using the open source SNA tool, Gephi. Communication patterns and frequencies can be derived and used to structure a simulation model.

Figure 19 Using Gephi to Develop Model Structure

6.3 Twitter Data to Derive Model Timing Sequence

Another example of deriving value from text mining can be illustrated with extracting timing information from Twitter's JSON-formatted, Tweet data. In this example, Python was used with a cloud service to collect Twitter data using the hashtag, simulation. The tweets were mined for their creation times and this was used to develop a distribution for interarrival time. Figure 20 shows the Twitter collection code and Figure 21 shows the collected times. Finally, Figure 22 provides the output for fitting the distribution using EasyFit software.

```
In [ ]: # Action 3 Required: pip if needed
!pip install algorithmia

In [1]: # Use Twitter Developer API Info to Collect Tweets!
import Algorithmia

input = {
    "query": "simulation",
    "numTweets": "50",
    "auth": {
        "app_key": "f0dVn[REDACTED]",
        "app_secret": "3q[REDACTED]",
        "oauth_token": "2[REDACTED]",
        "oauth_token_secret": "pKHiz0oIctsbmbC[REDACTED]"
    }
}

client = Algorithmia.client('simBq26eRxHp15lxcKLDLL3d06Q1')
algo = client.algo('twitter/RetrieveTweetsWithKeyword/0.1.5')
print(algo.pipe(input).result)

ended_entities': {'media': [{'additional_media_info': {'monetizable': False}, 'display_url': 'pic.twitter.com/WXtLAg173D',
    'expanded_url': 'https://twitter.com/Mr_DrinksOnMe/status/1194458362504732672/video/1', 'id': '1194325008241299457', 'id_str': '1194325008241299457', 'indices': [62, 85], 'media_url': 'http://pbs.twimg.com/ext_tw_video_thumb/1194325008241299457/pu/img/9bXIOUQsDF0LLGFP.jpg', 'media_url_https': 'https://pbs.twimg.com/ext_tw_video_thumb/1194325008241299457/pu/img/9bXIOUQsDF0LLGFP.jpg', 'sizes': {'large': {'h': 720, 'resize': 'fit', 'w': 408}, 'medium': {'h': 720, 'resize': 'fit', 'w': 408}, 'small': {'h': 680, 'resize': 'fit', 'w': 385}, 'thumb': {'h': 150, 'resize': 'crop', 'w': 150}}, 'type': 'video', 'url': 'https://t.co/WXtLAg173D', 'video_info': {'aspect_ratio': [17, 30], 'duration_millis': 7834, 'variants': [{'content_type': 'application/x-mpegURL', 'url': 'https://video.twimg.com/ext_tw_video/1194325008241299457/pu/pl/Te-0CKLQqAOcxXS.m3u8?tag=10'}, {'bitrate': 632000, 'content_type': 'video/mp4', 'url': 'https://video.twimg.com/ext_tw_video/1194325008241299457/pu/vid/320x564/FR6mlk4n0LY20onm.mp4?tag=10'}], 'bitrate': 832000, 'content_type': 'video/mp4', 'url': 'https://video.twimg.com/ext_tw_video/1194325008241299457/pu/vid/408x720/iZhtFlwE2WknRbev.mp4?tag=10'}}]}, 'favorite_count': 162457, 'favorited': False, 'geo': None, 'id': '1194458362504732672', 'id_str': '1194458362504732672', 'in_reply_to_screen_name': None, 'in_reply_to_status_id': None, 'in_reply_to_status_id_str': None, 'in_reply_to_user_id': None, 'in_reply_to_user_id_str': None, 'is_quote_status': False, 'lang': 'en', 'metadata': {'iso_language_code': 'en', 'result_type': 'recent'}, 'place': None, 'possibly_sensitive': False, 'retweet_count': 68839, 'retweeted': False, 'source': '<a href="https://about.twitter.com/products/tweetdeck" rel="nofollow">TweetDeck</a>', 'text': 'Learning to fly a helicopter. Lucky it was just a simulation. https://t.co/WXtLAg173D', 'truncated': False, 'user': {'contributors_enabled': False, 'created_at': 'Sun May 09 13:43:13 +0000 2010', 'default_profile': False, 'default_profile_image': False, 'description': 'This is our place, we make the rules. Web Designer at @MalavKarkar. Instagram: mrdrinksomme. For business or endorsements: hello@mrdrinksomme.com', 'entities': {'description': {'urls': []}, 'url': {'urls': [{'display_url': 'malavkarkar.com', 'expanded_
```

Figure 20 Python Code to Collect Twitter Data

```
In [5]: # View the Tweet Times

for i in data[:100]:
    print(i['created_at'])

Thu Nov 14 16:27:43 +0000 2019
Thu Nov 14 16:27:43 +0000 2019
Thu Nov 14 16:27:43 +0000 2019
Thu Nov 14 16:27:42 +0000 2019
Thu Nov 14 16:27:41 +0000 2019
Thu Nov 14 16:27:40 +0000 2019
Thu Nov 14 16:27:37 +0000 2019
Thu Nov 14 16:27:37 +0000 2019
Thu Nov 14 16:27:35 +0000 2019
Thu Nov 14 16:27:35 +0000 2019
Thu Nov 14 16:27:34 +0000 2019
Thu Nov 14 16:27:31 +0000 2019
Thu Nov 14 16:27:31 +0000 2019
Thu Nov 14 16:27:28 +0000 2019
Thu Nov 14 16:27:28 +0000 2019
Thu Nov 14 16:27:27 +0000 2019
Thu Nov 14 16:27:27 +0000 2019
Thu Nov 14 16:27:27 +0000 2019
Thu Nov 14 16:27:27 +0000 2019
Thu Nov 14 16:27:27 +0000 2019
Thu Nov 14 16:27:27 +0000 2019
```

Figure 21 Collected Time Data from Twitter

Figure 22 Fitting Distribution for Interarrival Time using EasyFit Software

7 VISUALIZATION AND STORY TELLING

Big data visualization and storytelling tools such as Power BI or Tableau are useful in both simulation model input data analysis and to provide output data reporting and analysis. Both tools contain a wide range of visualizations that help ensure data is more easily understood. Simulations are excellent tools for generating big data visuals to illustrate how outputs can add another dimension to a model's usefulness. This tutorial will provide examples.

8 CONCLUSIONS

The purpose of this tutorial is to provide a working set of tools specific for unstructured data analysis for use by simulation practitioners. The examples use the Python programming language with Jupyter Notebook from Anaconda. Specific examples derived from descriptive analytics, text analytics, topic modeling and sentiment analysis were demonstrated with data from sources that included web scraping, Twitter API, and CSV files.

ACKNOWLEDGMENTS

The author would like to acknowledge Dr. Bongsug Chae from Kansas State University for developing some of techniques described in this tutorial.

REFERENCES

- Baines, T., Mason, S., Siebers, P.-O., & Ladbrook, J. (2004). Humans: the missing link in manufacturing simulation? *Simulation Modelling Practice and Theory*, **12**(7–8), 515–526.
- Greasley, A., & Owen, C. (2018). Modelling people's behaviour using discrete-event simulation: a review. *International Journal of Operations & Production Management*, **38**(5), 1228–1244.

- Gupta, B., Sharma, S., & Chennamaneni, A. (2016). Twitter Sentiment Analysis: An Examination of Cybersecurity Attitudes and Behavior. In *Proceedings of the 2016 Pre-ICIS SIGDSA/IFIP WG8.3 Symposium Innovations in Data Analytics*. **17** (pp. 1–10). Dublin, Ireland: AIS.
- Khan, R., & Urolagin, S. (2018). Airline Sentiment Visualization, Consumer Loyalty Measurement and Prediction using Twitter Data. *International Journal of Advanced Computer Science and Applications*, **9**(6), 380–388.
- McHaney, R., Tako, A., & Robinson, S. (2018). Using LIWC to choose simulation approaches: A feasibility study. *Decision Support Systems*. <https://doi.org/10.1016/j.dss.2018.04.002>
- Robinson, S. (2008). Conceptual modelling for simulation Part I: definition and requirements. *The Journal of the Operational Research Society*, **59**(3), 278–290.
- Robinson, S., Arbez, G., Birta, L. G., Tolk, A., & Wagner, G. (2015). Conceptual modeling: definition, purpose and benefits. In *Proceedings of the 2015 winter simulation conference* (pp. 2812–2826).
- Tako, A. A., & Kotiadis, K. (2015). PartiSim: A multi-methodology framework to support facilitated simulation modelling in healthcare. *European Journal of Operational Research*, **244**(2), 555–564.
- Zhu, L., Reychav, I., McHaney, R., Broda, A., Tal, Y., & Manor, O. (2019). Combined SNA and LDA methods to understand adverse medical events. *International Journal of Risk & Safety in Medicine*, (Preprint), 1–25.

AUTHOR BIOGRAPHY

ROGER MCHANNEY is a Professor of Management Information Systems and University Distinguished Teaching Scholar at Kansas State University, College of Business. He holds the Daniel D. Burke Chair for Exceptional Faculty. McHaney is the author of books on simulation, education, and health technology. Prior to becoming an academic, he developed simulations primarily in manufacturing and materials handling. His current research includes understanding how simulation and modelling techniques can best be applied in business organizations, and data-driven simulations. His email address is mchaney@ksu.edu.