

Large-Scale Multi-agent-Based Modeling and Simulation of Microblogging-Based Online Social Network

Maíra Gatti, Paulo Cavalin^(✉), Samuel Barbosa Neto, Claudio Pinhanez, Cícero dos Santos, Daniel Gribel, and Ana Paula Appel

IBM Research, Rio de Janeiro, Brazil

{mairacg, apappel, csantosp, cicerons, dlemes, pcavalin, sbneto}@br.ibm.com

Abstract. Online Social Networks (OSN) are self-organized systems with emergent behavior from the individual interactions. Microblogging services in OSN, like Twitter and Facebook, became extremely popular and are being used to target marketing campaigns. Key known issues on this targeting is to be able to predict human behavior like posting, forwarding or replying a message with regard to topics and sentiments, and to analyze the emergent behavior of such actions. To tackle this problem we present a method to model and simulate interactive behavior in microblogging OSN taking into account the users sentiment. We make use of a stochastic multi-agent based approach and we explore Barack Obama's Twitter network as an egocentric network to present the experimental simulation results. We demonstrate that with this engineering method it is possible to develop social media simulators using a bottom-up approach (micro level) to evaluate the emergent behavior (macro level) and our preliminary results show how to better tune the modeler and the sampling and text classification impact on the simulation model.

Keywords: Online Social Network · Microblogging · Sentiment analysis · Modeling · Simulation

1 Introduction

Online social networks (OSNs) have become very popular in the last years, not only for users but also for researchers. Twitter, for example, is just a few years old, but has already attracted much attention from the research community [1, 2]. Through an OSN, users connect with each other, share and find content, and disseminate information. As example we can cite networks of professionals and contacts (e.g., LinkedIn, Facebook, MySpace) and networks for content sharing (e.g., Flickr, YouTube).

Information diffusion consists of a process in which a new idea or action widely spreads through communication channels. OSNs are the most used means for this nowadays [3]. This area is widely studied by sociologists, marketers, and

epidemiologists [4–6]. Large OSNs consist of a useful way for studying information diffusion as topic propagation. Blogspace [7], linking patterns in blog graphs [8], favorite photo marking in a social photo sharing service [9], and so forth, report on large OSNs. Besides, it is important to understanding how users behave when they connect to these sites for a number of reasons. For instance, in viral marketing one might want to exploit models of user interaction to spread their content or promotions quickly and widely. There are numerous models of influence spread in social networks that try to model the process of adoption of an idea or a product. However, it is still difficult to measure and predict how a market campaign will spread across an OSN if one or a set of users post, forward or reply a particular message, or if her or them don't post at all for a period of time about a particular topic, for instance. Agent-Based Simulation (ABS) provides a modeling paradigm that allows to perform what-if analysis to explore these kind of analysis.

ABS looks at agent behavior at a decentralized level, at the level of the individual agent, in order to explain the dynamic behavior of the system at macro-level. A multi-agent-based system is composed of many software agents interacting with one another and with the environment over time. The concept of an agent is more general than that of an individual, object or simulation process. An agent is an interactive computer system that is situated in some environment and is capable of accomplishing autonomous actions in order to meet the goals for which it has been designed [10]. Their collective behavior can be unpredictable, surprising, hence novel and emergent. In this way, this style of modeling is quite consistent with the sciences of complexity [11]. In addition, feedback loops can be achieved in ABS, since the result of agents actions stimulates other actions and eventually re-stimulate the first actions. Thus, a prerequisite for a multi-agent system to exhibit self-organization is feedback loops in which agents get feedback on their actions and stimulate or inhibit each other.

ABS has been successfully applied to a large number of works published in the literature [12]. However, to the best of our knowledge, this type of simulation has not yet been applied to information diffusion in large OSNs. Most works related to agent-based information diffusion models deal with synthetic networks for deriving the agents' behavior, making these works less realistic [13, 14]. Nevertheless, there is plenty of OSNs from which we can gather real data to conduct this kind of investigation.

In the light of this, the main contribution of this work is to propose an engineering method to simulate user behavior using a real-world OSN. The ultimate main goal lies in modeling and simulating what users post on this OSN network, to analyze how information spreads across the network. Several challenges are faced: sampling the network from the real-world OSN, performing text classification (natural language processing) to predict topic and sentiment from the posts, modeling the user behavior to predict his/her actions (pattern recognition), and large-scale simulation - for 10,000 seeds, it can easily reach 10^8 users in the network [15]. The proposed method makes use of a stochastic multi-agent based approach where each agent represents a user in the OSN. As a case-study,

a political campaign in the Twitter microblogging service is examined, more specifically, Barack Obama’s Twitter network during the 2012 United States presidential race. For doing so, we built an egocentric social network, i.e. Obama is considered as the central user (the seed) and only his immediate neighbors and their associated interconnections are examined, to help us model how individuals correspond with their connections within a social network. This paper is organized as follows. In Sect. 2 we describe the proposed method. Next, in Sect. 3 we present the experimental protocol and the results for some sensitive analysis we performed. Finally, in Sect. 4 we discuss our main remarks and point out the future work.

2 Proposed Method

The proposed method is based on a stochastic multi-agent based approach where each agent is modeled from the historical data of each user in the network as a Markov Chain process and a Monte Carlo simulation. The method has six phases and is iterative as illustrated in Fig. 1.

The first phase consists of sampling the OSN. After cleaning the data, the second phase consists of performing topic and sentiment classification on the posts extracted from the sampled data. Then in phase three, from the previously classified data we create sets of samples for each user. Each set contains the user’s posts and the posts of whom he/she follows. We build each user behavior model (fourth phase) from these sets and the models are used as input for the stochastic simulator (fifth phase). The models are validated by running the simulation and applying the validation method. We performed this cycle several times until we found the modeling strategy presented in this paper. Once the model is accurate enough, forecast on information diffusion can be performed. Next, we describe these phases in greater detail (except the Dataset Partitioning phase which is quite straightforward).

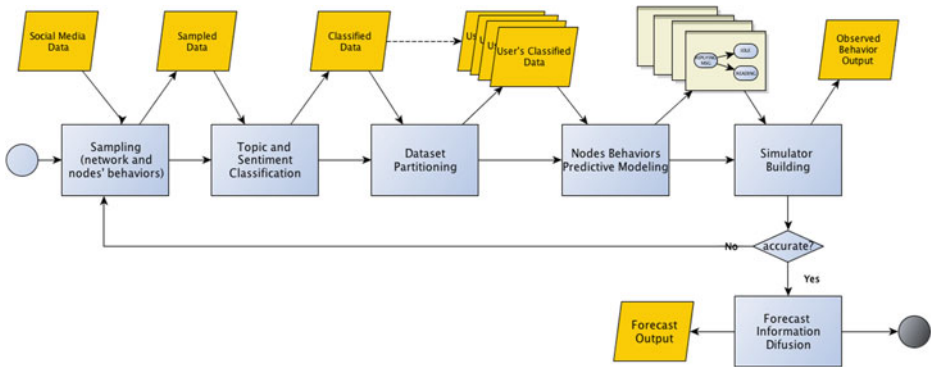


Fig. 1. Engineering method workflow.

2.1 Sampling

The sampling phase is the starting point of the method herein proposed and consists of crawling real data from an OSN. The crawling process must extract both network and user’s actions. There are several network sampling methods: node sampling, link sampling, and snowball sampling [16]. In node or edge sampling, a fraction of nodes or edges is selected randomly. On the other hand, snowball sampling randomly selects one seed node and performs a breadth-first search (hence the name, snowball sampling), until the number of selected nodes reaches the desired sampling ratio. Only those links between selected nodes are included in the final sample network. The snowball sampling method is more feasible than node and edge sampling to crawl the OSN, since it is difficult to have access to node and edge randomly and also they have a high probability to produce a network with isolated clusters [17].

Depending on the OSN, the crawling method may vary due to constraints on the API or OSN policies. Twitter offers an Application Programming Interface (API) that is easy to crawl and collect data, being the OSN used in this work. However, it is not possible to retrieve the entire data from Twitters and our sample method needs to deal with the limits imposed by Twitter’s API version 1.0 (the newest version of the API available when the data was extracted), such as the limited number of requests per hour that we could perform. Before we describe in detail how we collected users, tweets and the relationship among users, we defined some states of the nodes that are related to the information we have already obtained through the API.

We categorized the nodes depending on the type of information we have about them. For *Identified* nodes, the only information we have from these nodes are their IDs, a unique long value that identifies each user. Some of these nodes were *Consulted*, which means that we have acquired metadata about these nodes, for instance the number of followers, user language and so on. For some of the consulted nodes we *Visited* them and acquired the IDs of some or all of its followers. For nodes we visited we also *Extracted* some or all of its tweets. The Twitter’s API limited the number of consulted nodes by up to 100 nodes per request, the number of retrieved IDs by up to 5000 IDs per request and the number of retrieved tweets by up to 200 tweets per request. For tweets, we could retrieve up to 3200 of the latest tweets of a user using this process, that is, we repeat this process 16 times at most. We would stop when the API limit was reached, when all tweets were extracted or when the oldest extracted tweet reached a specific date.

The goal of this work is to analyze the diffusion of information having a unique node as its source. For analyses of this kind, we use an egocentric network, that is, a network that considers a node (person) as a focal point and its adjacency. In this context, we used Barack Obama’s egocentric network. therefore we *Visited* Barack Obama’s node and *Identified* all of its followers. Afterwards, we randomly *Consulted* 40k of his followers (distance 1 from Obama), from which we randomly choose 10k to *Visit* among the ones with English as a language. For each of these 10k nodes, we *Identified* up to 5k of its followers.

Among them there are nodes with distance 2 from Obama, from which we randomly *Consulted* up to 200 k. From these 200 k, we *Visited* up to 40 k randomly selected nodes whereas their profile language was set to English. In the end of this process, the network that we obtained contained approximately 32 million nodes and the extraction process took place in the week starting on October 26th 2012.

After that, we started extracting the tweets. Our threshold date was September 22nd and we started crawling after October 26th from 2012, therefore we tried to gather information from more than one month (sometimes the user did not have a whole month of tweets or tweeted so much that the API limit would not let us paginate further backwards before our threshold). We *Extracted* Obama's tweets and also the tweets from the 10 k *Visited* nodes with distance 1 and the 40 k *Visited* nodes with distance 2. In the end, we had approximately 5 million tweets and 24,526 users that posted in that period.

Previous work, known as "six degree of separation" [18], states that with 6 steps we are usually able to reach most of network's nodes starting in any node. On the other hand, it is reasonable to think that with distance 3 we are able to reach most of the network if we start at a high degree node. Since Obama has 23 millions of followers and the average number of followers in Twitter is around 100, performing a naive count without considering transitivity, we would have around 2 billion users in the second level (distance 2 from Obama), which is more than the number of Twitter users (1 billion). Thus, using distance 3 should be enough to analyze how information spread in this network, since it is reasonable to assume it should be possible to reach most of Twitter's users with 3 steps from Obama. Of course, if we choose other seed (user) for an egocentric network or if we use other OSN, this statistic may change a little, but it holds true in most cases for high degree seeds.

2.2 Topic and Sentiment Classification

We use text mining strategies to perform two important tasks of our modeling approach: topic classification and tweet sentiment analysis. The topic classification task consists in classifying a tweet as related to a certain topic or campaign (about politics, marketing, etc.). In the sentiment analysis task the objective is to classify a tweet as a positive or negative sentence. The *topic classification* task is performed using a keyword based approach. First, we select a list of keywords to represent each topic. Next, each tweet text is split into tokens using blank spaces and punctuation marks as separators. Then, the tokenized tweet is discarded or classified as belonging to one of the interesting topics, as follows:

- If the tweet contains keywords from more than one topic, it is discarded;
- If the tweet does not contain any keyword from any topic, it is classified as Other topic;
- If the tweet contains at least one keyword from a topic, it is classified as belonging to that topic.

The *sentiment classification* task is performed using a machine learning approach. We train a Naïve Bayes classifier using the training data created by Go et al. [1]. This training set contains examples of positive and negative tweets only. Therefore, the learned classifier predicts new tweets using these two classes only.

The first step when training or using the classifier is preprocessing. In preprocessing, we tokenize the tweet and perform the three strategies presented in [1] to reduce the feature space: (1) substitute all user names by the word `USER-NAME`; (2) substitute all urls (tokens starting with `http:`) by the word `URL`; and (3) replace any letter occurring more than two times in a token with two occurrences (e.g. `coooooool` is converted into `cool`). In order to train the Naïve Bayes classifier, we use a feature set composed of token unigrams and bigrams. The final classifier achieves 82 % accuracy when applied to Go et al.’s [1] test set.

2.3 Behaviors Predictive Modeling

The novelty of our approach consists of learning each user’s behavior in order to explore the power of interactions. In a microblogging like Twitter there are several *actions* that we can observe in the data like *posting*, *forwarding*, *liking* or *replying* a message, for instance. For each action to be modeled, the sampling phase must take into account that the user to be replied or that will have his/her message forwarded must be in the sampled graph. In this paper we describe the most straightforward model that can be learned from the data, whereas only the *posting* action is modeled. Hence this modeling approach can be used as a foundation to create more complex behavior models.

To learn this behavior we designed the modeler which receives the list of users in the OSN as input and, for each user, a document containing his/her posts and the posts of whom he/she follows. From this merged document, the user’s state change transitions are modeled as a Markov Chain, where the current state depends only on the previous state. Therefore the following assumptions are considered in the current version of the modeler:

- Time is discrete and we consider a Δt time interval to define action time windows;
- User actions like posting are performed on these time windows and states are attached to it. Therefore, current state on the modeler means what the user posted in the current time window, while previous state means that the user posted and/or read in the previous time window.
- Messages are interpreted as two vectors: a bit vector¹ which contains bits representing if the topic and sentiment appear in the message and an integer

¹ Suppose a user posted a positive message about *Obama*, a negative message about *Other* topic in the Δt time interval and there are only these two topics and two sentiments (positive and negative) being observed; if the first 2 positions of the vector are for positive and negative *Obama* index, and the other two for *Other* in that order; the vector is [1, 0, 0, 1].

Table 1. List of observed states and transitions. Empty sets represent vectors not observed

Θ	Transitions
1	$R_{t-\Delta t}, W_{t-\Delta t} \neq \emptyset \rightarrow W_t \neq \emptyset$
2	$R_{t-\Delta t}, W_{t-\Delta t} \neq \emptyset \rightarrow W_t = \emptyset$
3	$R_{t-\Delta t} = \emptyset, W_{t-\Delta t} \neq \emptyset \rightarrow W_t \neq \emptyset$
4	$R_{t-\Delta t} = \emptyset, W_{t-\Delta t} \neq \emptyset \rightarrow W_t = \emptyset$
5	$R_{t-\Delta t} \neq \emptyset, W_{t-\Delta t} = \emptyset \rightarrow W_t \neq \emptyset$
6	$R_{t-\Delta t} \neq \emptyset, W_{t-\Delta t} = \emptyset \rightarrow W_t = \emptyset$
7	$R_{t-\Delta t} = \emptyset, W_{t-\Delta t} = \emptyset \rightarrow W_t \neq \emptyset$

vector containing the number of messages that appeared in the position where the bit has value 1.

We modeled two possible actions: reading and posting. Therefore, let R be the vector representing what the user read and W the vector representing what the user wrote, then $A_{t-\Delta t} = \{R_{t-\Delta t}, W_{t-\Delta t}\}$ and $A_t = \{W_t\}$. In this case $NT = 7$ and Table 1 describes the transitions and/or states that can be observed in the data and that will be used in the simulator. $R_{t-\Delta t} = \emptyset$, $W_{t-\Delta t} = \emptyset$ or $W_t = \emptyset$ represent non-observed data.

We compute the Maximum Likelihood Estimation (MLE) with smoothing to estimate the parameter for each $\theta_i \in \Theta$ transition type. Therefore, for each user's sampled data u we estimate L for:

- Observed transitions $\theta_1, \theta_3, \theta_5$:

$$L(\theta|R_{t-\Delta t}, W_{t-\Delta t}, W_t) = \frac{\text{count}(\theta, R_{t-\Delta t}, W_{t-\Delta t}, W_t) + 1}{\text{count}(R_{t-\Delta t}, W_{t-\Delta t}, W_t) + |S|} \quad (1)$$

- Non-observed transitions $\theta_2, \theta_4, \theta_6$ and θ_7 :

$$L(\theta|R_{t-\Delta t}, W_{t-\Delta t}, W_t) = \frac{1}{\text{count}(R_{t-\Delta t}, W_{t-\Delta t}, W_t) + |S|} \quad (2)$$

Where $|S|$ is the number of states.

We take into account that the user may post a message related to a topic and a sentiment, which are grouped and stored in the set Ξ . For this reason, the aforementioned transitions are computed for each topic and sentiment $\xi_i \in \Xi$, so that the actions of the users are modeled according to the type of message that he/she is reading or writing.

In considering that the user might behave differently according to the period of the day, we compute the probability of posting a message at a given period $\phi_i \in \Phi$, where $1 \leq i \leq K$. This takes into account the total of messages m_i posted by the user at ϕ_i and the messages posted over all periods (the whole day), as in Eq. 3. In addition, we consider the following notation for each period ϕ_i . The

corresponding starting time is denoted $\phi'_i \in \Phi'$, and its length (in hours) is denoted $|\phi_i|$.

$$L(\text{posting } |\phi_i|) = \frac{m_i}{\sum_{\phi_j \in \Phi} m_j} \quad (3)$$

The volume of messages posted by the user is saved in a vector containing integer values, where each position corresponds to the average number of messages written for an element in the set Ξ . Equation 4 describes how to compute the transitions volume, where N represents how many W_t vectors there are for the same θ transition, L denotes the total of topics/sentiments, i.e. $|\Xi|$, and w_{lj} corresponds to the number of messages written for $\xi_l \in \Xi$ and transition θ .

$$V_{W_t}(\theta) = [\frac{\sum_{j \in N} w_{1j}}{N}, \frac{\sum_{j \in N} w_{2j}}{N}, \dots, \frac{\sum_{j \in N} w_{Lj}}{N}] \quad (4)$$

Volume vectors are computed for both transitions and periods. Equation 5 shows how to compute the average for periods:

$$V(\phi_i) = [\frac{\sum_{j \in M} w'_{1j}}{M}, \frac{\sum_{j \in M} w'_{2j}}{M}, \dots, \frac{\sum_{j \in M} w'_{Lj}}{M}] \quad (5)$$

Where M represents how many different vectors there are for period ϕ_i , and w'_{lj} corresponds to the number of messages sent for the topic/sentiment $\xi_l \in \Xi$ at a period ϕ_i .

The volume vector $V(\phi_i)$, as we will explain further, is used by the simulator to set different weights to $V_{W_t}(\theta)$, according to the current period ϕ_i . For this reason, we divide each position of $V(\phi_i)$ by the mean observed volume over all periods. As a consequence, the periods where the user posted a larger volume of messages will have greater weights than periods where he/she posted less messages. In Eq. 6 we demonstrate how this division is done.

$$V'(\phi_i) = [\frac{v_{1i}}{\bar{v}_{1j} | \phi_j \in \Phi}, \frac{v_{2i}}{\bar{v}_{2j} | \phi_j \in \Phi}, \dots, \frac{v_{Li}}{\bar{v}_{Lj} | \phi_j \in \Phi}] \mid v_{li} \in V(\phi_i) \quad (6)$$

Where v_{li} denotes the volume for the topic/sentiment ξ_l and period ϕ_i .

2.4 Simulation

The SMSim simulator herein described is a stochastic agent-based simulator where each *agent* of the system encapsulates the social media network user behavior and the environment where the agents live and interact is the followers *Graph* extracted from the social media network. Since each user is an **agent** in the simulator, hence the corresponding graph notation is $G = (A, R)$ where A is the set of agents and R is the set of followers relationships.

The SMSim is modeled as a discrete-event simulation [19] where the operation of the system is represented as a chronological sequence of events. Each event occurs at an instant in time (which is called a *time step* or just *step*) and marks

a change of state in the system. The step exists only as a hook on which the execution of events can be hung, ordering the execution of the events relative to each other. The agents and environment are events at the simulation core.

Therefore, the basic agent actions in the simulator are *To Read* or *To Post* and the agent states are *Idle* or *Posting* and in both states the agent reads the received messages from whom she follows and can write or not depending on the modeled behavior. When the agent is posting a message, at the simulator level, it is sending the message to all its followers. The message can have *positive* or *negative* sentiment about a *topic*. That's how the messages are propagated during simulation.

The agent behavior is determined by Markov Chain Monte Carlo simulation method. In the previous subsection we described how the user behavior is modeled as a Markov Chain into which we can call now the *UserModel* structure. During the SMSim initialization two important steps are performed: (i) the graph is loaded from the edges list file, and (ii) for each user in the graph, an agent instance is created and each *UserModel* file is deserialized into the agent model.

We implemented SMSim using Java and the second step is performed by translating the transitions saved in the *UserModel* by the modeler to a map where the key represents the source state id and the value is another map containing the probabilities to go from the source state id to the target state id, i.e., the key of the latter map is the target state id and the value is the *SimTransition* which contains the set of probability values. We defined these maps indexed by the states id to improve performance. Since each agent will have a set of transitions and there will be thousands of agents in the system interacting at the same time.

Every agent (user) is initialized in the *Idle* state. When the SMSim is started, each agent switches its behavior to *Posting* or *Idle* back depending on the activated transitions using Monte Carlo method. The transition will only be activated if the probability value calculated as described in Eq. 7 corresponds to a random value generated by the system, where $v_{w_t} \in V_{W_t}$.

$$\rho(\theta_i) = L(\theta_i | R_{t-1}, W_{t-1}, W_t) * L(posting | \phi_i) \quad (7)$$

In this case, once transition θ_i is picked, the volume of messages to be posted for each topic and sentiment ξ_i in the period ϕ_i of current time step is calculated using the weighted value of the corresponding average volume:

$$v(\theta, \phi_i, \xi_i) = v_{w_t}(\theta_i) * v'_{\phi_i}(\phi_i) \quad (8)$$

If no transition is activated, the system switches the user's state to *Idle*. Algorithm 1 describes these steps. We performed some experiments where instead of switching the state to *Idle* we switched to the most probable state according to the transitions Θ . However that approach did not result in a positive impact in the overall simulation results. The same happened if we create a uniform probability distribution for transitions where both previous and current state were not observed.

Algorithm 1 Agent states transition algorithm

```

1: if State = Posting then
2:   for each subject do
3:     message  $\leftarrow$  new Message(subject)
4:     post(message)
5:   end for
6: else
7:   State  $\leftarrow$  getNextState()
8:   if State = Null then
9:     State  $\leftarrow$  Idle
10:  end if
11: end if

12: function GETNEXTSTATE( )
13:   value  $\leftarrow$  pseudoRandomGen.getNext(0.0,1.0)
14:   prevProbabilityValue  $\leftarrow$  0.0
15:   for each stateId in TransitionsMap do
16:     probabilityValue  $\leftarrow$  TransitionsMap.get(stateId) * PeriodMap.get
       (currPeriod)
17:     if prevProbabilityValue  $\leq$  value and value  $\leq$  (prevProbabilityValue +
       probabilityValue) then
18:       return stateId
19:     end if
20:     prevProbabilityValue  $\leftarrow$  prevProbabilityValue + probabilityValue
21:   end for
22: end function

```

2.5 Validation

The Root Mean Square Error (*RMSE*) is frequently used to validate simulation models like weather predictions or to evaluate the differences between two time series. The formula to calculate this error is presented in Eq. 9.

$$RMSE(T) = \sqrt{\frac{\sum_{t=1}^T (y'_t - y_t)^2}{T}} \quad (9)$$

where y'_t represents the total of messages sent in the simulator at time t , and y_t denotes the total of messages sent at time t in the observed data.

The proposed models are validated using the Coefficient of Variation of the Root Mean Square Error CV_{RMSE} (Eq. 10), where the results of the simulator are compared with those computed from the observed data. Hence *RMSE* is applied to compare the curve of the total of messages sent by the users in the simulator, up to a time T , with the curve plotted from the observed data used to estimate the parameters of the simulator; and the CV_{RMSE} normalizes to the mean of the observed data. With this metrics we can compare both pattern and volume.

$$CV_{RMSE}(T) = \frac{RMSE(T)}{\bar{y}|_{t=1}^T} \quad (10)$$

3 Experimental Results

In this section we present the experiments carried out to evaluate the simulator. The main goal is compare the total number of messages posted by the users in the simulator with the total number of messages sent by the real users. For this task it was considered a dataset consisting of tweets extracted from Barack Obama’s Twitter network, posted during the 2012 United States presidential race. As a consequence, we modeled an egocentric network, centered on Obama, composed of 24,526 nodes. These nodes along with about 5.6 million tweets were sampled from the real network using the method described in Sect. 2.1. This dataset allows us to model and simulate the behavior of the users in the network when reading and posting messages related to the two main candidates of the 2012 elections: Barack Obama and Mitt Romney. For this reason, the topics/sentiments in Ξ are set to (‘Obama Positive’, ‘Obama Negative’, ‘Romney Positive’, ‘Romney Negative’, ‘Other’), where the two main topics are ‘Obama’ and ‘Romney’ and the two main sentiments are ‘Positive’ and ‘Negative’. Note that ‘Other’ corresponds to a message whose topic is neither Obama nor Romney, and whose sentiment is not relevant in this work. All tweets were then classified into a topic/sentiment of Ξ using the two-step procedure described in Sect. 2.2. In this case, 17,853 tweets were classified as ‘Obama positive’ and 8,766 as ‘Obama negative’. Most of the remaining tweets were considered as ‘Other’. More details about the sampled dataset are presented in Table 2.

Next, we describe in greater detail the topic and sentiment classification, the scenarios and results obtained in these experiments, and the performance of the simulator in terms of time.

3.1 Topic and Sentiment Classification

In our experiment, two topics are considered: Barack Obama and Mitt Romney. The keyword list used to the Obama’s topic includes the words: *barack*, *barack2012*, *barackobama*, *biden*, *joebiden*, *josephbiden*, *mrpresident*, *obama*, *obama2012*, *potus*. For Romney, we considered the keywords: *mitt*, *romney*, *mittromney*, *paulryan*, *governorromney*, *mitt2012*, *romney2012*. Note that we also considered hashtags (e.g. *#obama*, *#romney*, *#gobama*, *#obamabiden2012*, *#goromney*, *#romneyryan2012*) and usernames (e.g. *@BarackObama*, *@MittRomney*, *@JoeBiden* and *@PaulRyanVP*). In addition, besides the cases considered for topic classification described in Sect. 2.2, we also considered a special

Table 2. Sampled data, topic and sentiment classification results

Tweets	Active users	Direct followers	Edges	Triangles	TS classification		
					Other	OB+	OB-
5.6M	24,526	3,594 (0.017 % of real amount)	160,738	83,751	5,564,170	17,853	8,766

treatment for messages originated by Obama. That is, if a tweet is generated by Obama himself, we also consider some personal pronouns (such as *I*, *me*, *my*, *mine*) and the keyword *president* to classify the main topic of the tweet as ‘Obama’. According to this rule, retweets of Obama’s messages also consider these additional terms. In this case, though, the *RT @username* text fragment is ignored for topic evaluation to avoid that a retweet of an original negative message is classified as a negative post about the candidate.

3.2 Sensitive Analysis

There are three types of sensitive analysis that we present here for the 24,526 simulated users. First we analyze by periods, then by sliding windows time steps, and finally by sentiment. All of them we discuss on how the CV_{RMSE} varies throughout the simulation time. Recall, we are not stating that this is the only way of analyzing the results, though it gives tangible insights on the proposed method. For each scenario we run 10 simulation trials and computed the average.

3.2.1 Period Impact

We defined two distinct scenarios to consider:

Fixed: Modeler with 4 periods with equal durations: all periods $\Phi = (\text{‘Night’}, \text{‘Morning’}, \text{‘Afternoon’}, \text{‘Evening’})$ have the same length of hours, i.e. $|\phi_i| = 6, \forall \phi_i \in \Phi$, with the corresponding starting times defined as $\Phi' = (12:00\text{AM}, 6:00\text{AM}, 12:00\text{PM}, 6:00\text{PM})$.

Short Night: Modeler with 4 periods and short night: the same 4 periods in Φ as the other scenario but the ‘Night’ period is shorter with a duration of 4 h and starting later, the morning, afternoon and evening are shifted, and afternoon and evening have 1hr longer duration. The corresponding starting times are defined as $\Phi' = (4:00\text{AM}, 8:00\text{AM}, 2:00\text{PM}, 9:00\text{PM})$.

For both scenarios, $\Delta t = 15$ min. The ‘Short Night’ scenario was defined based on two observations: (i) the time observed in the data is UTC-3, Brasilia, Brazil time, however the majority of users are in the USA. Hence the minimum time zone difference is 3 h, and (ii) the night period in the observed data is shorter compared to the others periods.

In Fig. 2 the curves representing the volume of messages sent at each simulation step, for both scenarios, are shown along with the volume of messages plotted from the sampled data. In both scenarios the volume of messages results in a curve whose shape is similar to that computed from the real data. This shows that the proposed approach is promising towards an accurate modeling of users’ behavior in social media networks. In Fig. 3 we show the validation with CV_{RMSE} as described in Sect. 2.5. It can be observed that the error rate of the simulator in the ‘Short Night’ scenario is generally lower than in the ‘Fixed’ scenario. This indicates that the proper setting of the length and the starting times of the periods may improve the overall modeling of the users’ behavior.

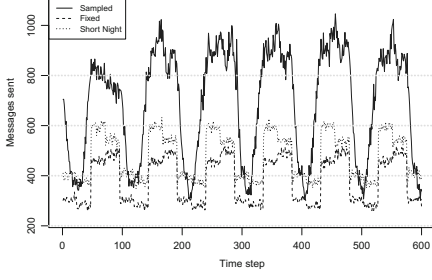


Fig. 2. Volume of messages per time step.

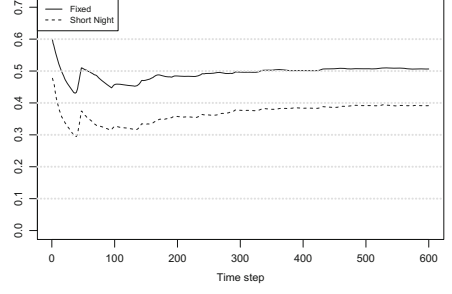


Fig. 3. CV_{RMSE} validation metric.

3.2.2 Sliding Window Time Step Impact

In our experiments, four different sliding window time steps (Δt) were applied through the simulation. The time steps were 5, 15, 30 and 60 min, and the whole simulation took place in a 9000 min time window, that corresponds to 6.25 days. By the end of each simulation, we plotted the curves corresponding to the simulated volume of messages that we sampled corresponding to the topic “Obama” and the simulated output. Also, we calculate the cumulative CV_{RMSE} of both curves and use it to analyze the quality of our simulation.

Figures 4, 5, 6 and 7 show the number of messages, simulated for ‘Short Night’ scenarios, and sampled in each time step. We can see that there is a higher agreement between the 5 min discretization and the sampled data than with the other discretizations. That is also verified when we look at the CV_{RMSE} in Fig. 8, that clearly shows lower cumulative error as time grows.

One possible interpretation for this behavior is that short time windows better capture the user behavior about a topic because there is less noisy data when considering a thinner granularity of our discretization. At the same time, we can imagine that Twitter users tend to react to stimulæ from other users relatively fast, since the approach that presented the best results was the 5 min time step.

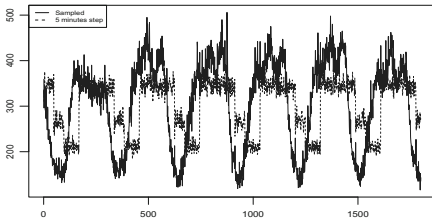


Fig. 4. Volume of messages per time step for a 5 min step.

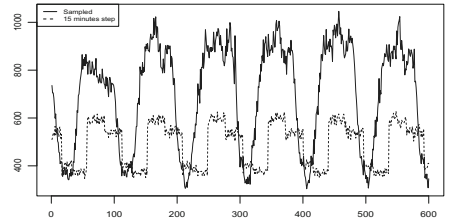


Fig. 5. Volume of messages per time step for a 15 min step.

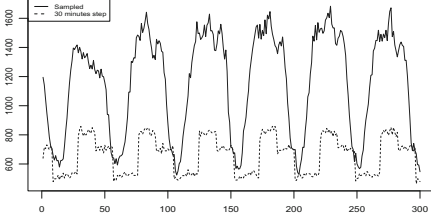


Fig. 6. Volume of messages per time step for a 30 min step.

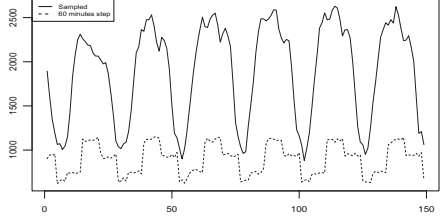


Fig. 7. Volume of messages per time step for a 60 min step.

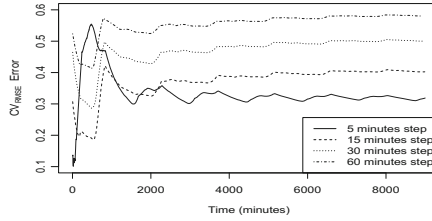


Fig. 8. Cumulative CV_{RMSE} error of the trials per time of the simulation.

It is also interesting to note how the error grows in the beginning of the simulation for the 5 min time step. It surpasses the errors of the other discretizations and then drops until it reaches the lowest error we observed. This indicates that, in spite of its higher error during the first simulation steps, the 5 min configuration can result in lower modeling error over longer simulation runs. The longer time window configurations, in contrast, might be more suitable for short simulation runs.

3.2.3 Sentiment Analysis

In Figs. 9 and 10, the CV_{RMSE} for each topic in \mathcal{E} is shown, for ‘Short Night’ scenarios with sliding window (Δt) of 5 and 15 min time steps, respectively.

From these results we can observe that either the sampling would have to be increased with regard to the sentiment of the posts related to Obama and Romney topics or these sliding windows for considering the pair topic-sentiment are not well parametrized. First because the CV_{RMSE} should be minimized and second because we would expect the range scale in Fig. 9 to be lower than in Fig. 10, since it had better accuracy as showed in Figs. 4 and 5.

3.3 Performance

We run the experiments in a Red Hat x86_64 Linux with 256 GB memory size and 16 CPU cores. For the sample used, both the modeler and simulator scaled in a linear time. However we tested some scenarios with a complementary sample

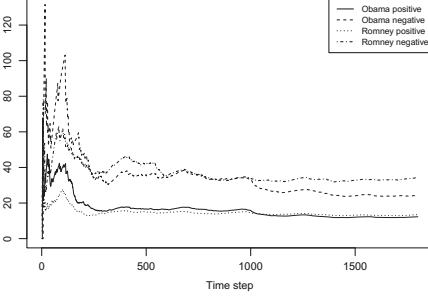


Fig. 9. CV_{RMSE} validation for each topic ('Short Night' - 5 min).

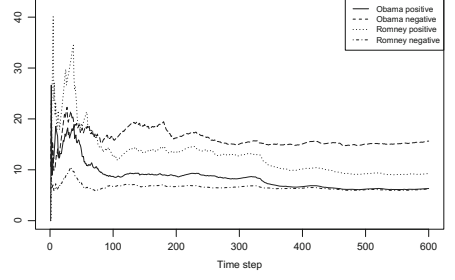


Fig. 10. CV_{RMSE} validation for each topic ('Short Night' - 15 min).

not used for the previous experiments described. If some users have more than 100 leaders the modeler was highly impacted, while the simulator had a lower increasing in the execution time. On the other hand the simulator execution time scales with the size of the network. Figure 11 shows the average simulation steps durations for 10 simulations trial with 602 steps and 24 k agents.

If we use the same host environment that we used, for instance, if we scale the network to 10^k agents, the average time would shift to 10^{k-1} ms through a naive induction.

4 Final Remarks

In this paper we proposed a method to simulate the behavior of users in a social media network, using Barack Obama's Twitter network as a case study. The data sampled from Twitter allowed us to build individual stochastic models to represent how each user behaves when posting messages related to two main topics, i.e. Barack Obama and Mitt Romney, and a positive or negative sentiment. Experiments considering different scenarios demonstrated that the proposed approach is promising for simulating the overall behavior of users in the social network. And the main contribution of using ABS technology is the possibility to explore what-if analysis where we tune user's behavior.

From the proposed method point of view, the future work is fourth fold. First, we need to enhance the modeling of the users. This might be achieved by using machine learning and optimization techniques to define both the architecture and the parameters of the models from data. Second, to re-evaluate the sampling of the data and to enlarge the dataset may help to better estimate the models. Third, by improving topic and sentiment classification the remaining phases of the simulator will be able to rely on a more accurate estimate of user opinions. And last but not least, the simulator does not scale with the sampling size. Therefore we need to address simulation distribution and parallelization in order to simulate larger samples. From the social network analysis point of view, we

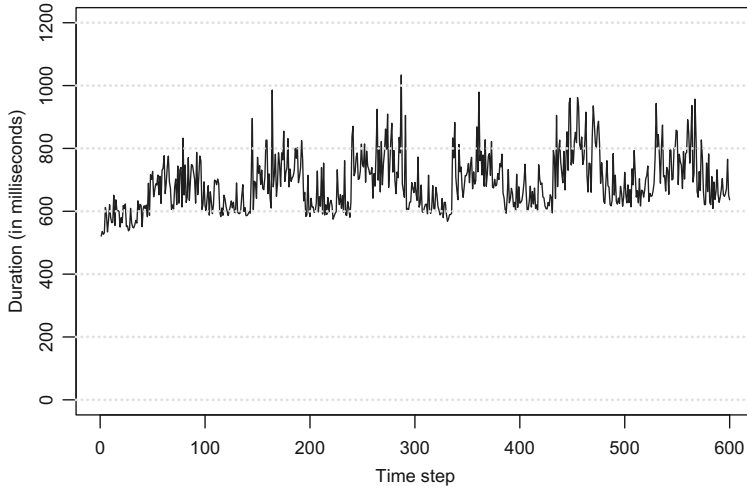


Fig. 11. Simulation steps durations.

are exploring scenarios where we change the behavior of highly influential users in order to understand their impact.

References

1. Go, A., Bhayani, R., Huang, L.: Twitter sentiment classification using distant supervision. Technical report CS224N, Stanford University (2009)
2. Kwak, H., Lee, C., Park, H., Moon, S.: What is twitter, a social network or a news media? In: Proceedings of the 19th International Conference on World Wide Web, WWW '10, pp. 591–600. ACM, New York (2010)
3. Rogers, E.M., Rogers, E.: Diffusion of Innovations, 5th edn. Free Press, New York (2003)
4. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '03, pp. 137–146. ACM, New York (2003)
5. Leskovec, J., Adamic, L.A., Huberman, B.A.: The dynamics of viral marketing. In: Proceedings of the 7th ACM Conference on Electronic Commerce, EC '06, pp. 228–237. ACM, New York (2006)
6. Strang, D., Soule, S.A.: Diffusion in organizations and social movements: from hybrid corn to poison pills. *Ann. Rev. Sociol.* **24**(1), 265–290 (1998)
7. Gruhl, D., Guha, R., Liben-Nowell, D., Tomkins, A.: Information diffusion through blogspace. In: Proceedings of the 13th International Conference on World Wide Web, WWW '04, pp. 491–501. ACM, New York (2004)
8. Leskovec, J., Horvitz, E.: Planetary-scale views on a large instant-messaging network. In: Proceedings of the 17th International Conference on World Wide Web, WWW '08, pp. 915–924. ACM, New York (2008)

9. Cha, M., Mislove, A., Gummadi, K.P.: A measurement-driven analysis of information propagation in the flickr social network. In: Proceedings of the 18th International Conference on World Wide Web, WWW '09, pp. 721–730. ACM, New York (2009)
10. Wooldridge, M., Jennings, N.R.: Intelligent agents: theory and practice. *Knowl. Eng. Rev.* **10**(2), 115–152 (1995)
11. Jennings, N.R.: An agent-based approach for building complex software systems. *Commun. ACM* **44**(4), 35–41 (2001)
12. Macal, C.M., North, M.J.: Tutorial on agent-based modelling and simulation. *J. Simul.* **4**, 151–162 (2010)
13. Janssen, M.A., Jager, W.: Simulating market dynamics: interactions between consumer psychology and social networks. *Artif. Life* **9**(4), 343–356 (2003)
14. Wicker, A.W., Doyle, J.: Leveraging multiple mechanisms for information propagation. In: Dechesne, F., Hattori, H., ter Mors, A., Such, J.M., Weyns, D., Dignum, F. (eds.) AAMAS 2011 Workshops. LNCS (LNAI), vol. 7068, pp. 1–2. Springer, Heidelberg (2012)
15. Mislove, A., Marcon, M., Gummadi, K.P., Druschel, P., Bhattacharjee, B.: Measurement and analysis of online social networks. In: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, IMC '07, pp. 29–42. ACM, New York (2007)
16. Lee, S.H., Kim, P.J., Jeong, H.: Statistical properties of sampled networks. *Phys. Rev. E* **73** (2009)
17. Ahn, Y.Y., Han, S., Kwak, H., Moon, S., Jeong, H.: Analysis of topological characteristics of huge online social networking services. In: Proceedings of the 16th International Conference on World Wide Web, WWW '07, pp. 835–844. ACM, New York (2007)
18. Milgram, S.: The small world problem. *Psychol. Today* **2**, 60–67 (1967)
19. Fishman, G.S.: Discrete-Event Simulation: Modeling, Programming, and Analysis. Springer, New York (2001)

Multi-Agent-Based Simulation XIV

International Workshop, MABS 2013, Saint Paul, MN,

USA, May 6-7, 2013, Revised Selected Papers

Alam, S.J.; Dyke Parunak, H.V. (Eds.)

2014, X, 163 p. 50 illus., Softcover

ISBN: 978-3-642-54782-9