Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

Dynamical opinion clusters exploration suite: Modeling social media opinion dynamics

Henrique Ferraz de Arruda [a,*], Kleber Andrade Oliveira [b], Yamir Moreno [c,d]

[a] Geography and Geoinformation Science, College of Science, George Mason University, Fairfax, VA, USA
[b] Department of Mathematics, Munster Technological University, Cork, Ireland
[c] Institute for Biocomputation and Physics of Complex Systems (BIFI), University of Zaragoza, Zaragoza, Spain
[d] Department of Theoretical Physics, Faculty of Sciences, University of Zaragoza, Zaragoza, Spain

ARTICLE INFO

Keywords:
Opinion dynamics
Social networks
Opinion polarization
Echo chamber

ABSTRACT

The escalating use of social media in recent years has made the study of opinion dynamics a crucial area for understanding societal trends. As digital communication platforms continue to shape collective consciousness, understanding the evolution, interaction, and spread of opinions has become imperative. Researchers have approached this phenomenon from a variety of perspectives, ranging from sociology to data analytics to computational simulation. To address the challenges posed by the multifaceted and multidisciplinary nature of this research, coupled with the recent scarcity of data, computational simulation has emerged as a key tool for understanding opinion dynamics in social networks. This paper presents a Python library, DOCES, designed to simulate essential features of real-world social networks. The library includes the simulation of a social network algorithm, user prioritization, and the ability to model changes in friendships.

Code metadata

Current code version	v0.0.7
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-24-00611
Permanent link to Reproducible Capsule	
Legal Code License	MIT License
Code versioning system used	git
Software code languages, tools, and services used	CPython and Python (NumPy)
Compilation requirements, operating environments & dependencies	Python 3.6 or above
If available Link to developer documentation/manual	https://pypi.org/project/doces/
Support email for questions	h.f.arruda@gmail.com

1. Introduction

The use of social media has been increasing in recent years. Consequently, studies on opinion dynamics have become a pivotal aspect of understanding our society. As digital communication platforms can shape our collective consciousness, it is crucial to understand the mechanisms behind how opinions evolve, interact, and propagate [1,2]. To deal with these multifaceted phenomena, scholars have been studying opinion dynamics from different points of view, including sociological perspectives, data analysis, and computational simulations.

To understand these complex social dynamics via computational simulations, an extensive range of studies tackled the mechanisms underlying social networks [3–11]. In this context, several types of opinion dynamics have been modeled, which include the voter model [12], the majority rule model [13], the bounded confidence model [14] and so on. A realistic ingredient often added to models is the friendship structure, represented as a network. Furthermore, since individuals can change their friendships over time in the real world, some models also include time-evolving (or adaptive) networks [15–17].

Agent-based models can simulate interactions between agents in a social network, including the mechanisms behind polarization [18]. For

* Corresponding author.
E-mail address: h.f.arruda@gmail.com (Henrique Ferraz de Arruda).

https://doi.org/10.1016/j.softx.2025.102136
Received 15 November 2024; Received in revised form 12 March 2025; Accepted 17 March 2025
Available online 1 April 2025

example, a model can incorporate different elements of the dynamics of online social media, such as posting, reposting, and friendship adjustments [19]. Other agent-based models represent specific underlying mechanisms behind opinion dynamics [13,20–22]. For example, the Sznajd model captures persuasion in social networks [23,24], while friendship change and adaptive networks help explain the formation of echo chambers [10,16,25].

The challenges posed by the scarcity of data underscore the significance of employing simulators to study opinion dynamics within online social networks. In this context, this paper introduces DOCES, a Python library designed to facilitate the study of opinion dynamics in social networks by providing an accessible and efficient simulation framework. The class of models the library implements considers continuous values of opinions for agents who are social media users.

Unlike existing frameworks (e.g., NetLogo [26], GAMA [27], and MESA [28]), DOCES brings a more opinion-dynamic focus and is specifically designed to simulate the social network environment. This includes the ability to test different social network connectivity, algorithms, and user behavior. Specifically, DOCES incorporates comprehensive features such as configurable user behaviors, the social network algorithm that can be used to simulate priority users, a friendship update mechanism based on opinion alignment, and a memory list for posts, which is a proxy for the social media feed.

With DOCES, one can control various parameters, including the feed size and the frequency of new posts or re-posts. The models the library is able to simulate have been extended and tuned for different applications; for a more comprehensive understanding of the dynamics simulated, see detailed descriptions in [4,29,30].

2. Software description

DOCES is written as a CPython extension and offers fast implementation and a user-friendly API. Here, we briefly describe the main features of our library and show how to use it. First, as we added new ingredients to our model, we renamed some of the model steps concerning the description on the original paper [4]. The steps previously referred to as “transmission” and “distribution” are now “posting” and “receiving”, respectively.

The first step is to define a DOCES object with the `Opinion_dynamics` constructor method (see Code Listing 1). The parameters of this object represent the network, which is given by the number of vertices (`vertex_count`), the edge list (`edges`, a Python list of 2-tuples), and whether the network is directed (`directed`, a boolean variable). There is also an optional hidden parameter `verbose`, which is set by default as True. Notice that the network structure may change as the simulation runs depending on how it is configured.

Code Listing 1: Example of how to initialize a DOCES object

```
import doces
# Initializes the network parameters
...
# Creates a Opinion_dynamics object.
od = doces.Opinion_dynamics(
    vertex_count,
    edges,
    directed)
```

The method `simulate_dynamics` in the `Opinion_dynamics` object is used to iterate over a sequence of steps that change the opinions of users and the network structure. Its primary outputs are a dictionary with the resulting opinions of users as a Python list of floats and the resulting edge list as a Python list of 2-tuples. However, it can also output statistics from the post cascades that happened during the iterations.

The method `simulate_dynamics` can be adjusted through several dynamics parameters (Code Listing 2) before the model is run, see below. The first is the number of iterations defined for the simulation (`number_of_iterations`). `phi` is the parameter that controls the starting point of the receiving filter. The innovation parameter `mu` controls the probability of re-posting a piece of information from the feed. If `mu = 0`, posts are always selected from memory lists; if `mu = 1`, all posts are newly created and the feed posts are never re-posted. The posting and receiving filters (`posting_filter` and `receiving_filter`) are arrays of integers, with values from 0 to 5. These options are equivalent to filter functions whose equations are in [4]. Table 1 describes the correspondence between options and equations. In addition to these values, 5 can also be used to indicate a customized configuration where each user can have a different option.

Table 1
Equivalence of filter options to equations in the original model from [4]. ‘Random’ is a random choice from Eqs. 2–4 to each user (different users may have different options).

Option	Posting filter	Receiving filter
0	Eq. 2	Eq. 5
1	Eq. 6 with $\phi = 0$	Eq. 6
2	Eq. 4	Eq. 7
3	Eq. 3	Eq. 3
4	Random	–

Code Listing 2: Example code to run the dynamics.

```
# Initializes the dynamics parameters
...
# Run the dynamics
output_dictionary = od.simulate_dynamics(
    number_of_iterations,
    phi,
    mu,
    posting_filter,
    receiving_filter,
    b = None,
    feed_size = 5,
    rewire = True,
    cascade_stats_output_file = None,
    min_opinion = −1,
    max_opinion = 1,
    delta = 0.1,
    verbose = True,
    rand_seed = None)

opinions = output_dictionary["b"]
edge_list = output_dictionary["edges"]
```

There are also optional parameters that are described as follows. `b` is a list of float numbers representing the opinions of the agents. The feed (or memory list) size is defined by the integer parameter `feed_size`. The rewiring condition is checked if the boolean parameter `rewire` is set to True. The argument `cascade_stats_output_file` may receive a string with a filename destination to store a CSV file with statistics of the post cascades. Alternatively, one can access the same information as a Python dictionary by using the method described in Code Listing 3. The minimum and maximum values of opinions are defined by `min_opinion` and `max_opinion` respectively. `delta` is a float parameter that determines how much the opinion changes when realigned (default is 0.1). As with the object definition, there is also the possibility to set the verbose mode with the boolean parameter `verbose`. The last parameter `rand_seed` is an integer number that defines the random seed.

Code Listing 3: Get information from the cascades.

```
# To get a dictionary with all the information
stats_dict = od.get_cascade_stats_dict()
```

```
# The contents of the output dictionary are
# stats_dict = {
# "post_id": od.post_ids,
# "theta": od.post_thetas,
# "count": od.post_posted_counts,
# "cascade_size": od.post_cascade_sizes,
# "birth": od.post_births,
# "death": od.post_deaths,
# "live_posts": od.post_live_post_counts,
# "user_opinion": od.post_user_opinions
# }
```

Notice that the method `simulate_dynamics` can be both used once with a high number of iterations to study system stability or called many times to measure dynamical changes from the initial condition. The variables that store measurements are kept in memory after the `simulate_dynamics` method is executed.

DOCES also allows the configuration of additional parameters; see Code Listing 4. Posting and receiving functions can be set using `set_posting_filter` and `set_receiving_filter`, respectively. The parameters are set as lists with a specific probability function for each user, using the integer options shown in Table 1. Stubborn users (who do not change their opinion during iterations) can also be set with `set_stubborn`. The parameter of this method is a list of integers (or boolean) with a length equal to the number of nodes in the network, where each position represents a user. The users set to 1 (or True) become stubborn, while those assigned to 0 (or False) keep changing their opinion normally.

Code Listing 4: Methods for setting additional parameters of the dynamics.

```
# Initializes the lists to be set
...
# Set the posting filter
od.set_posting_filter(posting_filter)

# Set the receiving filter
od.set_receiving_filter(receiving_filter)

# Set stubborn users
od.set_stubborn(stubborn_users)
```

Code examples are available in the GitHub repository, in the *docs/tutorials/* directory. These tutorials include a Python notebook with an example of a simulation that converges to the formation of echo chambers.

To ensure broad accessibility, DOCES is designed to be multi-platform and has been successfully tested on various operating systems, including Linux (Debian and Ubuntu), MacOS, and Windows.

3. Impact

Multidisciplinary researchers, including computational physicists and computational social scientists, have been studying opinion dynamics in social media. To ease the simulation of opinion dynamics, we implement DOCES, a Python library that can be easily used by scholars and students from different areas. The single pre-requisite is to have basic knowledge of Python programming language. For more advanced programmers, we also allow the possibility to set and handle the parameters and configurations of the dynamics to test a wide range of scenarios found in real systems, such as the formation of echo chambers or consensus. Our library combines performance and usability to impact the simulation of a broad class of opinion dynamics. In addition, DOCES contains all the code needed to implement the dynamics described in [4,29,30].

4. Conclusions and future improvements

The growing influence of social media has made the study of opinion dynamics essential to understanding its impact on society. Given the complexity of this research and the recent scarcity of data, computational simulation has become a critical tool. This paper presents DOCES, a Python library with key functionality implemented in C that simulates key aspects of real-world social networks, including algorithm-driven content distribution, user prioritization, and evolving network structures.

In summary, DOCES serves as a valuable resource for researchers and students in multidisciplinary fields, facilitating computational experiments of opinion dynamics. Its user-friendly interface, coupled with the ability for advanced users to customize parameters, ensures accessibility across different levels of expertise. DOCES allows the simulation of social network algorithms, allowing each user to be calibrated individually. Agent feeds and reposting behavior can also be calibrated. In addition, agents can exhibit different behaviors, including a zealot attitude. The network structure can evolve over time via friendship rewiring. The simulation outputs the resulting network structure, user opinions, and post dynamics, such as lifetime and reach.

In addition, future enhancements may include the ability to use arbitrary functions and the refinement of rewiring dynamics for increased realism. We also plan to implement a number of different opinion models for comparison purposes. It is also possible to use our library and create a graphical interface for users without Python knowledge.

CRediT authorship contribution statement

Henrique Ferraz de Arruda: Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Conceptualization. **Kleber Andrade Oliveira:** Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Conceptualization. **Yamir Moreno:** Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank Caroline Ceribeli for her contribution to designing the DOCES logo. Y.M was partially supported by the Government of Aragón, Spain and “ERDF A way of making Europe” through grant E36-23R (FENOL), and by Ministerio de Ciencia e Innovación, Agencia Española de Investigación, Spain (MCIN/AEI/10.13039/501100011033) Grant No. PID2023-149409NB-I00. We acknowledge the use of the computational resources of COSNET Lab at Institute BIFI, funded by Banco Santander, Spain (grant Santander-UZ 2020/0274) and by the Government of Aragón (grant UZ-164255). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

[1] Ciampaglia GL, Menczer F. Misinformation and biases infect social media, both intentionally and accidentally. Conversat 2018;20.
[2] Shao C, Hui P-M, Wang L, Jiang X, Flammini A, Menczer F, et al. Anatomy of an online misinformation network. Plos One 2018;13(4):e0196087.
[3] Valensise CM, Cinelli M, Quattrociocchi W. The drivers of online polarization: Fitting models to data. Inform Sci 2023;642:119152. http://dx.doi.org/10.1016/j.ins.2023.119152.
[4] de Arruda HF, Cardoso FM, de Arruda GF, Hernández AR, da Fontoura Costa L, Moreno Y. Modelling how social network algorithms can influence opinion polarization. Inform Sci 2022;588:265–78.

[5] Liao H, Li X, Tang M. How to process local and global consensus? a large-scale group decision making model based on social network analysis with probabilistic linguistic information. Inform Sci 2021;579:368–87.
[6] Noorazar H. Recent advances in opinion propagation dynamics: a 2020 survey. Eur Phys J Plus 2020;135(6):1–20.
[7] Xiong L, Su X, Qian H. Conflicting evidence combination from the perspective of networks. Inform Sci 2021;580:408–18.
[8] Yu Z, Lu S, Wang D, Li Z. Modeling and analysis of rumor propagation in social networks. Inform Sci 2021;580:857–73.
[9] Sasahara K, Chen W, Peng H, Ciampaglia GL, Flammini A, Menczer F. Social influence and unfollowing accelerate the emergence of echo chambers. J Comput Soc Sci 2021;4(1):381–402.
[10] Baumann F, Lorenz-Spreen P, Sokolov IM, Starnini M. Modeling echo chambers and polarization dynamics in social networks. Phys Rev Lett 2020;124(4):048301.
[11] Galam S. Sociophysics: A review of galam models. Internat J Modern Phys C 2008;19(03):409–40.
[12] Fernández-Gracia J, Suchecki K, Ramasco JJ, San Miguel M, Eguíluz VM. Is the voter model a model for voters? Phys Rev Lett 2014;112:158701.
[13] Galam S. Minority opinion spreading in random geometry. Eur Phys J B- Condens Matter Complex Syst 2002;25(4):403–6.
[14] Lorenz J. Continuous opinion dynamics under bounded confidence: A survey. Internat J Modern Phys C 2007;18(12):1819–38.
[15] Gracia-Lázaro C, Quijandría F, Hernández L, Floría LM, Moreno Y. Coevolutionary network approach to cultural dynamics controlled by intolerance. Phys Rev E 2011;84(6):067101.
[16] Benatti A, de Arruda HF, Silva FN, Comin CH, da Fontoura Costa L. Opinion diversity and social bubbles in adaptive sznajd networks. J Stat Mech Theory Exp 2020;2020(2):023407.
[17] Su W, Wang X, Chen G, Yu Y, Hadzibeganovic T. Noise-based synchronization of bounded confidence opinion dynamics in heterogeneous time-varying communication networks. Inform Sci 2020.
[18] Noorazar H, Vixie KR, Talebanpour A, Hu Y. From classical to modern opinion dynamics. Internat J Modern Phys C 2020;31(07):2050101. http://dx.doi.org/10.1142/S0129183120501016.
[19] Sasahara K, Chen W, Peng H, Ciampaglia GL, Flammini A, Menczer F. Social influence and unfollowing accelerate the emergence of echo chambers. J Comput Soc Sci 2021;4(1):381–402.
[20] Deffuant G, Neau D, Amblard F, Weisbuch G. Mixing beliefs among interacting agents. Adv Complex Syst 2000;3(01n04):87–98.
[21] DeGroot MH. Reaching a consensus. J Amer Statist Assoc 1974;69(345):118–21.
[22] Rainer H, Krause U. Opinion dynamics and bounded confidence: models, analysis and simulation. J Artif Soc Soc Simul 2002;5(3).
[23] He M, Li B, Luo L. Sznajd model with social temperature and defender on small-world networks. Internat J Modern Phys C 2004;15(07):997–1003.
[24] Sznajd-Weron K, Sznajd J, Weron T. A review on the sznajd model—20 years after. Phys A 2021;565:125537.
[25] Holme P, Newman ME. Nonequilibrium phase transition in the coevolution of networks and opinions. Phys Rev E 2006;74(5):056108.
[26] Wilensky U. Netlogo 5.0.1. 2011.
[27] Taillandier P, Vo D-A, Amouroux E, Drogoul A. Gama: a simulation platform that integrates geographical information data, agent-based modeling and multi-scale control. In: Principles and practice of multi-agent systems: 13th international conference, PRIMA 2010, Kolkata, India, November 12–15, 2010, Revised selected papers 13. Springer; 2012, p. 242–58.
[28] Kazil J, Masad D, Crooks A. Utilizing Python for agent-based modeling: The mesa framework. In: Social, cultural, and behavioral modeling: 13th international conference, SBP-BRiMS 2020, Washington, DC, USA, october 18–21, Proceedings 13. Springer; 2020, p. 308–17.
[29] de Arruda HF, Oliveira KA, Moreno Y. Echo chamber formation sharpened by priority users. iScience 2024.
[30] Oliveira KA, de Arruda HF, Moreno Y. Mechanistic interplay between information spreading and opinion polarization. 2024, arXiv preprint arXiv:2410.17151.