WoMG: A Library for Word-of-Mouth Cascades Generation

Federico Cinus
Sapienza University, Rome, Italy
ISI Foundation, Turin, Italy
federico.cinus@isi.it

Francesco Bonchi
ISI Foundation, Turin, Italy
Eurecat, Barcelona, Spain
francesco.bonchi@isi.it

Corrado Monti
ISI Foundation, Turin, Italy
corrado.monti@isi.it

André Panisson
ISI Foundation, Turin, Italy
andre.panisson@isi.it

ABSTRACT

Studying information propagation in social media is an important task with plenty of applications for business and science. Generating realistic synthetic information cascades can help the research community in developing new methods and applications, testing sociological hypotheses and different *what-if* scenarios by simply changing few parameters.

We demonstrate WoMG, a synthetic data generator which combines topic modeling and a topic-aware propagation model to create realistic information-rich cascades, whose shape depends on many factors, including the topic of the item and its virality, the homophily of the social network, the interests of its users and their social influence.

CCS CONCEPTS

• **Human-centered computing** → *Social networks*; *Social network analysis*; • **Computing methodologies** → *Agent / discrete models*;

ACM Reference Format:
Federico Cinus, Francesco Bonchi, Corrado Monti, and André Panisson. 2021. WoMG: A Library for Word-of-Mouth Cascades Generation. In *Proceedings of the Fourteenth ACM International Conference on Web Search and Data Mining (WSDM '21), March 8–12, 2021, Virtual Event, Israel.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3437963.3441693

1 INTRODUCTION

The analysis of information cascades in social media has received a great deal of attention recently due to its many applications such as viral marketing [6, 14] and social advertising [8, 9, 12]. Furthermore, the analysis of the spread of information through the users of a social network enables studying important sociological phenomena such as homophily and social influence [1, 2, 11], the evolution of social networks [16] and the formation of communities [3, 13]; it can also help to tackle important issues of modern democracies, such as understanding the role played by social bots [10] and misinformation [15] in influencing public opinion and tempering the political debate. In this context, having synthetic data generators enables one to produce an unlimited amount of data, overcoming the limitation imposed by the social media platforms and by privacy regulations.

More importantly, synthetic data generation enables studying, *in vitro*, specific phenomena of interest by fine-tuning the parameters of the models and controlling how each set of assumptions affects the resulting propagations.

In this demo, we introduce WoMG (Word-of-Mouth Cascade Generator), a synthetic data generator of information cascades in social networks. WoMG combines many different ingredients to produce realistic information-rich cascades: the structure of the social network, the interests of each individual (which can exhibit more or less homophily w.r.t. the structure of the network), the strength of influence that individuals can exert on their peers, the specific items that propagate in the network (described explicitly by a set of keywords and implicitly as a probability distribution in a topic space). WoMG, whose concept is depicted in Figure 1, builds on top of recent results in network embeddings, topic models, and social influence analysis [7], and it is distributed as a Python 3 library available in GitHub[1] and pypi[2].

Competitors. To the best of our knowledge, WoMG is the first Python library for information diffusion which incorporates a topic model. Nevertheless, it shares some common features with other libraries. Nepidemix[3] and GEMFsim[4] deal with the epidemiological diffusion processes. Other packages, such as nxsim[5] and NDlib[6], provide a more general approach to diffusion modeling and implement information and opinion diffusion processes. None of these libraries, however, integrate topic models to describe people's interests and items' topics in a coherent way.

Roadmap. Next section provides a high-level overview of the conceptual model behind WoMG, while Section 3 describes its software architecture. Section 4 provides a preview of the usage of WoMG and the demo outline.

2 MODEL

WoMG is based on the model described in our previous paper [7]. Its main inputs are as follows:

($I1$) **a directed social graph** $G = (V, E)$ where the nodes V represent the set of individuals involved in the social network, and a directed link $(u, v) \in E$ represents v being a *follower* of u. As

WSDM '21, March 8–12, 2021, Virtual Event, Israel

ACM ISBN 978-1-4503-8297-7/21/03.
https://doi.org/10.1145/3437963.3441693

[1] https://github.com/FedericoCinus/WoMG
[2] https://pypi.org/project/womg
[3] http://nepidemix.irmacs.sfu.ca/
[4] http://www.ece.k-state.edu//netse/software
[5] https://github.com/kentwait/nxsim
[6] https://ndlib.readthedocs.io

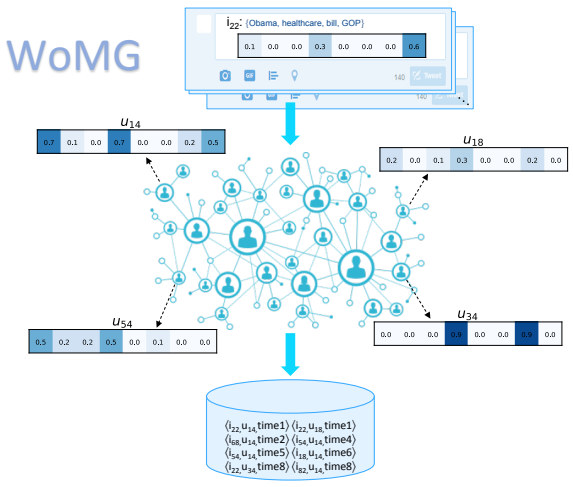

Figure 1: Bird's-eye view of WoMG: the items propagating in the network are described by a bag of words and a distribution over a topic space; the users of the social network are described by a vector of interests defined over the same topic model.

such, v has visibility of the items shared by u; therefore, v can be influenced by u to share further, thus allowing the propagation of information. The social graph can be any directed graph and can be loaded from disk in a variety of formats or, alternatively, it can be generated thanks to the integration with NetworkX.

($I2$) **a topic model** defined over a space of k topics, as produced by, e.g., the Latent Dirichlet Allocation (LDA) method [5]. The topic model is used by WoMG to describe users' interests and to create new items: for each new item i a distribution over topics γ_i is sampled from a Dirichlet distribution, then a bag of words describing i is sampled from γ_i and the topic space. If a topic model is not available, it can be created by running LDA over a corpus of documents (this functionality is provided directly by WoMG). Alternatively to creating synthetic items, WoMG allows one to use directly the real-world documents of the corpus, after they have been processed by LDA.

The first element that is generated by WoMG is the interests vector for each individual $v \in V$: i.e., a k-dimensional vector describing how much each individual is interested in each of the k topics. Generating realistic interests vectors, given a known network structure and a *tunable level of homophily* is a complex task, that we tackle by means of a method based on *non-negative matrix factorization*, which is shown experimentally to outperform non-trivial baselines [7]. Besides the interests vectors, other factors playing a key role in shaping propagations in a multi-topic setting, are the virality of the items, the influence capability of each node, or how the initial activations are generated. All parameters are described in more detail in Section 3.1.

Once generated the interest vectors, WoMG starts producing the information cascades by feeding items into the social graph and letting them propagate according to a propagation model inspired by the *topic-aware linear threshold* model of Barbieri et al. [4]. When a new item enters the network, it starts propagating: nodes can *activate* on it based on their interests. Once a node u activates on the item (i.e., they like or repost the item), their followers become aware of the item, and based on their interests and u's strength of influence, they might activate and propagate the item further. The output of this process includes the following elements:

($O1$) the vector of interests for each node $v \in V$;
($O2$) a set of items I, where each item $i \in I$ is described by a bag-of-words and a distribution in the topic space;
($O3$) a propagation trace for each item $i \in I$, where a propagation trace is a relation (i, v, t) representing the fact that the item i was adopted by node v at time t.

A concrete example of WoMG's output is given in Section 4.

3 SOFTWARE ARCHITECTURE

WoMG is distributed as a Python library through the package-management system `pip`. It offers two main interfaces to its users: a command-line tool and a Python function. Those two interfaces are functionally equivalent.

The typical usage of WoMG requires to provide, through its main interface, the input social graph, and the topic model. Both inputs can be specified in different ways. The graph can be provided as an external file with the edge list or—if the user is using WoMG as a Python function—as a NetworkX graph instance. The topic model can be provided as a CSV file, as a Numpy matrix, or it can be automatically generated by WoMG from a given file containing textual documents. In this case, WoMG uses LDA to find a suitable topic distribution, and then WoMG employs it to generate textual representations of the generated items that are coherent with their propagations on the network. In order to provide this textual functionality, WoMG relies on the Gensim library.[7] To accommodate users that wish to generate propagations but are not interested in the textual aspect, we also provide a separate Python package (`womg-core`), in order to avoid them the unnecessary large dependency on Gensim. In a transparent way for the user, `womg-core` is implemented as a dependency for the main WoMG package.

WoMG's software architecture is designed in a modular and extensible way. In this way, it is possible to implement different alternative approaches for the main factors of the propagation: different topic modeling choices, diffusion processes, and interplays with the underlying network. As presented in Figure 2, WoMG's architecture is characterized by three class hierarchies, each corresponding to one of these aspects. The two inputs (graph, topic model) and the main output (diffusion trace) are managed by each respective abstract class. The class `tlt` (topic-aware linear threshold model) provides the default diffusion process. It inherits from the general `diffusion_model` class initialization, iteration of the process, and the conditions to stop.

Since the diffusion model is based on a topic-aware model, it is necessary to represent both nodes and items in the same topic space. In other words, we need to express which topics characterize an item, and which topics appeal to a user. These aspects are handled by the two abstract classes which inherit from `network_model` and `topic_model`, respectively. The first provides the representation of

[7] https://radimrehurek.com/gensim/index.html

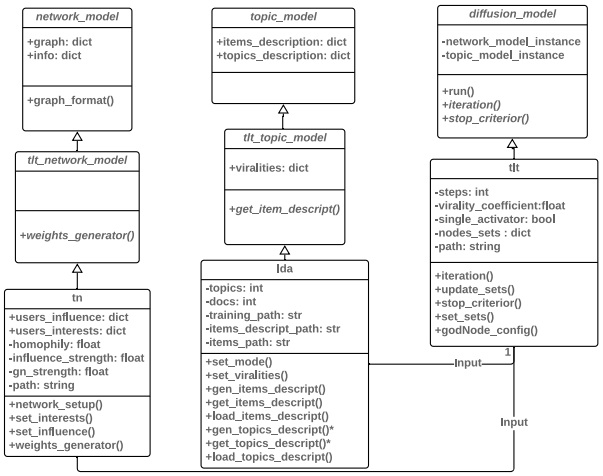

Figure 2: UML of WoMG's software architecture. The methods marked with an asterisk are part of the minimal `womg-core` package.

the network for the propagation model. Our concrete implementation, `tn` represents an interests-based network: in other words, it defines the topic-dependent weights of the network—how strong a node can influence another a specific topic.

The abstract class `topic_model` provides the representation of items in the topic space. Its concrete implementation is `lda`, which wraps LDA, bridging it to the propagation model. It models the distribution of topics in items, relying on the given topic model (possibly learning it from a given corpus of documents with Gensim).

3.1 Parameters

Despite the simplicity offered by a single-function interface, WoMG presents many ways to shape the generated cascades. By manipulating this small set of easily interpretable "knobs", we can test different assumptions about the propagation process. However, each of the parameters has a default value that is able to produce realistic cascades.

The two main parameters are the *number of items* (i.e., the number of cascades), and the *number of steps*. The latter is by default infinite: the spreading process continues as long as needed until each node is either sharing the item or not interested. The number of topics k instead is defined by the topic model.

One of the main ingredients of WoMG's model is the *interests* of each node in each topic, which must be consistent with a given level of homophily and with the network structure. The *homophily* parameter ranges from 0 to 1, controlling how much the resulting interests vectors are similar among connected individuals: a value of 1 results in high homophily in interests vector, while a value of 0 results in them being closer to random.

The parameter called `gn_strength` is a positive real number that allows one to set how much the items are successful in the environment in convincing an individual to share them. A low `gn_strength` results in items propagating mostly from neighbors, while a strong one means that items are often arriving at a node from exogenous factors.

A parameter called `infl_strength` (a positive real number) controls how much the *influential capabilities* of an individual have an effect on convincing in neighbors: a high influence strength comports that influential individuals are able to affect their neighbors (regardless of the topics); a low influence strength results in propagation being guided mostly by their individual topic-based interests.

Finally, two parameters are provided to control the virality of each item: `virality_exp` and `virality_resistance`; these are again positive real numbers. The former shapes the distribution of the virality of the set of items. Specifically, a larger `virality_exp` results in a more unequal distribution of item virality. The latter, `virality_resistance`, multiplies the threshold used by the linear threshold model, thus directly controlling how hard is for users to be influenced by their peers.

WoMG also provides a `seed` parameter in order to make all the simulations perfectly reproducible.

4 DEMONSTRATION

In the demonstration, the audience will be guided through the functionalities offered by WoMG. The library will be used through a Jupyter notebook. Jupyter provides, in fact, an easy way to experiment with different inputs, parameters, and to inspect the obtained propagations on the fly. WoMG is seamlessly integrated with Jupyter, thanks to the use of appropriate libraries (such as the NetworkX output format, and graphical progress bars).

In the first phase, we show the audience an immediate example of WoMG usage. After showing how to install WoMG easily thanks to `pip`, we demonstrate how to generate propagations on a synthetic graph that are coherent with a given news corpus in one line of code. As reported in Listing 1, the `womg` function can be imported from the homonym module and can take as input a folder path containing the items and a NetworkX graph instance.

```
from womg import womg
import networkx as nx

g = nx.random_geometric_graph(100, .3)
prop = womg(graph=g, docs_path='demo_corpus/')
```

Listing 1: WoMG command to generate cascades from a NetworkX random graph and a corpus of documents provided in a local folder.

The `womg` function returns the outputs and saves the default files in the `Output/` folder. The `Propagations` file contains the tuple `(time, item, node)` which completely describes one activation. Moreover, `Topics_description` and `Items_description` files represent, respectively, the linear combination of words which defines each topic, and the topics-dimensional vector of each item.

```
$ head -n 2 Output/Propagations0.txt
time item user
0 0 1
0 0 4

$ head Output/Topics_descript0.txt
[(0, '0.021*"said" + 0.015*"percent" + 0.011*"million" +
     0.008*"new" + 0.008*"year"  + ...'), ...]
```

```
plt.imshow(WordCloud().generate(prop.text[0]))
```

```
activations = np.zeros(N)
for _time, node in prop.propagations[0]:
    activations[node] = 1
nx.draw_networkx_nodes(g,
    node_color=activations,
    cmap=plt.cm.binary)
```

```
nx.draw_networkx_nodes(g,
  node_color=prop.interests[0])
```

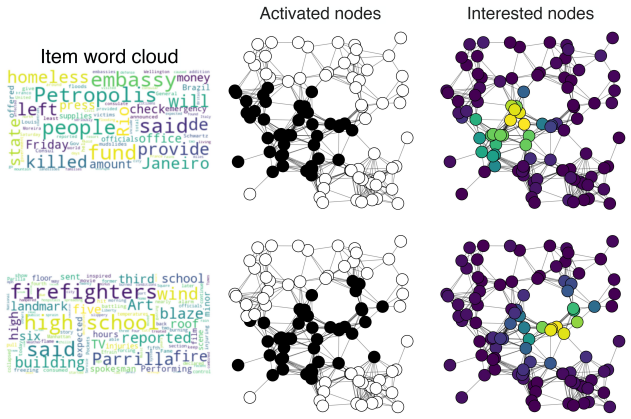

Figure 3: Visualization of WoMG output. Above the figure, we report the code used to produce each figure from WoMG's output `prop`. From left to right: we show an item words list; we display the nodes that reshared that item in WoMG's simulation; finally, we show the most interested nodes (in yellow) in the main topic of that item.

```
$ head -n 1 Output/Items_descript0.txt
0 [0.06345216, 0.20021512, 0.6291118, 0.10482439,...]
```

Listing 2: Example of WoMG output files.

We inspect the output of WoMG in different ways. We show a significative excerpt of the code and its output in Figure 3. First, we look at the text of the synthetically generated items; specifically, we show a word cloud of the item text with the `wordcloud` Python library. Then, through NetworkX, we display the activated nodes on that item by simply plotting WoMG's output. Finally, we check the underlying topical interests generated by WoMG, observing that they are coherent with the propagations; to do so, we simply plot the interests generated by WoMG relative to the topic most significative for that item.

In a second phase, we investigate, together with the audience, the different knobs offered by the WoMG package. Its parameters, in fact, can be used to simulate *in vitro* different scenarios, each one characterized by different assumptions. This exploration can be useful to investigate the properties of topic-based propagation through a what-if game we play with the audience. What will happen if the network does present a very small degree of homophily? What if the propagations are largely driven by exogenous factors? What if a small set of individuals is able to influence heavily the

```
# setting the womg parameters
womg(graph=g,
     int_mode='nmf',
     numb_docs=10,
     numb_steps=100,
     homophily=.5,
     gn_strength=13,
     infl_strength=12,
     virality_exp=8,
     virality_resistance=13)
```

Listing 3: Modifying WoMG's parameters

others? We show how WoMG allows any user to simulate these settings easily.

The generated propagations can be used both to investigate the results of a topic-aware word-of-mouth process, as well as to test any algorithm that wishes to operate on a textual propagation data set. As we show in the demonstration, WoMG gives its user the possibility to generate text and propagation, at the same time, in a coherent way. Such a data set is of immediate use and allows one to experiment thoroughly with any technique users wish to test in different settings. This can be useful for instance for propagation classifiers, network analysis techniques, or text-graph hybrid embeddings.

REFERENCES

[1] A. Anagnostopoulos, R. Kumar, and M. Mahdian. Influence and correlation in social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 7–15, 2008.

[2] E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic. The role of social networks in information diffusion. In *Proceedings of the 21st International Conference on World Wide Web*, pages 519–528, 2012.

[3] N. Barbieri, F. Bonchi, and G. Manco. Cascade-based community detection. In *Sixth ACM International Conference on Web Search and Data Mining, WSDM 2013*, pages 33–42, 2013.

[4] N. Barbieri, F. Bonchi, and G. Manco. Topic-aware social influence propagation models. *Knowledge and information systems*, 37(3):555–584, 2013.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

[6] F. Bonchi. Influence propagation in social networks: A data mining perspective. *IEEE Intell. Informatics Bull.*, 12(1):8–16, 2011.

[7] F. Cinus, F. Bonchi, C. Monti, and A. Panisson. Generating realistic interest-driven information cascades. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 14, pages 107–118, 2020.

[8] Ç. A. et al. Viral marketing meets social advertising: Ad allocation with minimum regret. *PVLDB*, 8(7):822–833, 2015.

[9] Ç. A. et al. Revenue maximization in incentivized social advertising. *PVLDB*, 10(11):1238–1249, 2017.

[10] E. e. a. Ferrara. The Rise of Social Bots. *Commun. ACM*, 59(7):96–104, June 2016.

[11] A. Goyal, F. Bonchi, and L. V. Lakshmanan. Learning influence probabilities in social networks. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, pages 241–250, 2010.

[12] T. Grubenmann, R. C. K. Cheng, and L. V. S. Lakshmanan. TSA: A truthful mechanism for social advertising. In *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining*, pages 214–222, 2020.

[13] Y. Mehmood, N. Barbieri, F. Bonchi, and A. Ukkonen. CSI: community-level social influence analysis. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013*, pages 48–63, 2013.

[14] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70, 2002.

[15] S. Vosoughi, D. Roy, and S. Aral. The spread of true and false news online. *Science*, 359(6380):1146–1151, 2018.

[16] L. Weng, J. Ratkiewicz, N. Perra, B. Gonçalves, C. Castillo, F. Bonchi, R. Schifanella, F. Menczer, and A. Flammini. The role of information diffusion in the evolution of social networks. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 356–364, 2013.