Tackling Fake News Detection by Continually Improving Social Context Representations using Graph Neural Networks

Nikhil Mehta, Maria Leonor Pacheco, Dan Goldwasser
Department of Computer Science
Purdue University
West Lafayette, IN, USA
{mehta52, pachecog, dgoldwas}@purdue.edu

Abstract

Easy access, variety of content, and fast widespread interactions are some of the reasons making social media increasingly popular. However, this rise has also enabled the propagation of fake news, text published by news sources with an intent to spread misinformation and sway beliefs. Detecting it is an important and challenging problem to prevent large scale misinformation and maintain a healthy society.

We view fake news detection as reasoning over the relations between sources, articles they publish, and engaging users on social media in a graph framework. After embedding this information, we formulate inference operators which augment the graph edges by revealing unobserved interactions between its elements, such as similarity between documents' contents and users' engagement patterns. Our experiments over two challenging fake news detection tasks show that using inference operators leads to a better understanding of the social media framework enabling fake news spread, resulting in improved performance.

1 Introduction

Over the last decade an increasing number of people access news online (Amy Mitchell, 2016), and use social networking platforms to engage, consume and propagate this content in their social circles. Social networks provide easy means to distribute news and commentary, resulting in a sharp increase in the number of media outlets (Ribeiro et al., 2018), representing a wide range of perspectives and ideologies. However, despite this diversity, content is often shared only among people that hold similar beliefs and ideologies, leading to the formation of highly segregated information communities, often referred to as *"echo chambers"* (Gentzkow and Shapiro, 2011; Quattrociocchi et al., 2016; Dubois and Blank, 2018; Garimella et al., 2018). An unfortunate consequence of this process is the rapid proliferation of "fake news" (Lazer et al., 2018), content which resembles news in form but lacks the journalistic standards ensuring its quality. Social media platforms are now flooded with inaccurate, incomplete, and intentionally misleading information, which propagates at lightning speed between users sharing an echo-chamber. According to a recent study (Vosoughi et al., 2018) false stories spread six times faster compared to real news stories. Given the volume and speed of fake news spread, manual fact checking organizations cannot be used in real-time to stop it. An alternative, which is arguably easier to scale, is to jointly model the claims and their source and ask *who and what can you trust?*

Answering this question requires modeling the complex information landscape on social media, consisting of news sources, the articles they release and their social context, corresponding to social media users engaging and sharing information in their networks. Similar to previous work (Baly et al., 2020b, 2018) we formulate the problem as associating a discrete factuality level (*high, low*, or *mixed*) with news content and news sources. These works treat news factuality level assessment as a traditional classification problem, using features based on social media data.

Our goal in this paper is to explore a different approach, driven by the principal of *social homophily* (McPherson et al., 2001), referring to the tendency of individuals to form social ties with others who share their views and preferences. We follow the observation that the political perspectives and biases expressed in the text will be reflected in the behavior of users engaging with it. Together they form *information communities*, connecting users with each other based on their content preferences, and with sources that provide that content. In this highly connected structure, even a little evidence connecting users' preferences to false narratives can be propagated and help inform the judgements about the sources they follow and

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 1363 - 1380

engage with. Fig. 1 demonstrates the interactions between users, articles and their sources.

Unfortunately, much of this rich social information is not directly observed, or due to the volume of these interactions, cannot be fully sampled. To help alleviate this problem and capture this knowledge, we propose a set of *inference operators*, each augmenting the information graph with different relationships beyond what was initially seen. By iteratively applying these inference operators, we are able to capture more of the hidden relationships that enable the spread of fake news through social media, and are crucial for detecting it.

From a technical perspective, we view fake news detection as a reasoning problem over information graphs. We use the evidence provided by our existing knowledge of high vs. low factuality content (i.e., the training data), to assess the factuality of unknown content based on observed and predicted links capturing their connections. This transductive process is done using a Relational Graph Neural Net (R-GCN) (Schlichtkrull et al., 2018), which creates distributed representations of nodes contextualized by the graph structure, allowing us to transfer information from observed evidence nodes to unknown source nodes using graph embedding tasks. We use inference operators, which build on the similarity metric defined by the learned graph embedding, to increase the number of edges connecting the two node types. These two interdependent steps are done iteratively.

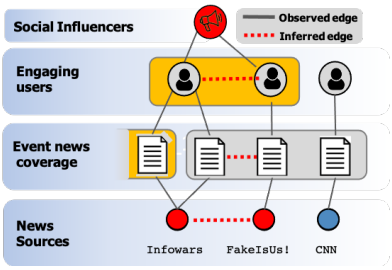


Figure 1: Information Graph capturing interactions between sources, articles, engaging users, and influencers. Dashed lines correspond Inference Operators outputs

For example in Fig. 1, observed user relationships are shown with black lines, such as users interacting with articles that are published by sources. Based on the observed data in the information graph, we can create an initial *graph-contextualized* representation for each node via graph embedding training. We can see that based on the current trained model, there are three articles that are similar in content and embedding, and are represented in the figure by sharing a gray background. Two are "fake news" articles, published by red background low-factuality news sources (*"FakeIsUs"* and *"InfoWars"*), while the right most one is published by a high-factuality source. Assuming the model is not familiar with their source factuality level, *then based on the observed graph information, it may not be able to distinguish between them*. Thus, in this work, we propose to augment the graph based on learned knowledge, via inference operators. Intuitively, the goal of our inference operators is to provide additional graph edges (shown as dashed red lines), such that the graph-contextualized embeddings would capture the similarity between the two low-factuality articles and the difference compared to the high-factuality one. For example, users engaging with the left two articles follow the same social influencer, who is a high activity user. In the initial graph training, this observed relationship would increase these users' learned node similarity (yellow background) allowing our inference operators to connect them into a strong information community of like-minded users, that was not initially observed, and thus not easily represented by the graph embedding. This newly inferred relationship can be propagated through the information graph, allowing us to have more strong information about other articles/sources/users these newly connected users interact with, thus detecting fake news better. In summary, we make the following contributions:

- We formulate fake news detection as a reasoning problem over an information graph.
- We suggest an inference-based graph representation learning approach, which incrementally augments the graph with inferences about users' social information and content preferences.
- We perform extensive experiments in source-level (Baly et al., 2020a) and content-level (Nguyen et al., 2020) settings, demonstrating our inference-based graph representation approach leads to performance improvements in both cases, even in weakly supervised settings.

2 Related Work

Fake News Detection: Detecting fake news using social media has been a popular research topic recently. It's typically studied as a supervised learning task, in which a classifier is trained using representations of news and their social context to predict factuality of the content (Hassan et al., 2017; Shu et al., 2017; Shao et al., 2018; Pérez-Rosas

et al., 2017; Volkova and Jang, 2018; Shu et al., 2019a; Kim et al., 2019). Unfortunately, these methods cannot capture the interactions between the users and sources that share fake news on social media, which is necessary to better understand the way fake news propagates, and ultimately detect it.

Due to the above mentioned limitations, researchers have recently started using Graph Neural Networks (GNNs) (to model graphs), for this task. As they contain social media entities as nodes and link them through edges based on their observed interactions, graphs are able to better capture social context. More specifically, through edge interactions, nodes in graphs can reinforce other nodes' representations, strengthening the overall information quality. Shu et al. was one of the early works, and more recently, Han et al. utilized continual learning with GNNs to capture the propagation cascade of fake news on Twitter. However, unlike our work, these and other graph models do not uncover or model hidden relationships in the data.

Most similar to us, Nguyen et al. proposed the Factual News Graph (FANG) also modeling the relationship between sources, articles, and users in a graph framework, by training the model to better capture social context. However, rather than iteratively adding new explicit edges to uncover hidden interactions in the graph as we do by using inference operators, FANG (Nguyen et al., 2020) modified the loss function they used when training the graph to better capture user-user and user-article interactions that already exist. Despite this being effective, it does not model graph interactions that were not observed in the original data, while our approach can uncover these hidden relationships as well, and thus more strongly capture the fake news propagation landscape on social media (the information communities we make explicit can help model other content better). Moreover, our framework allows the graph to be continually enhanced, so we can capture more relationships than were built into the original graph (like source-source), and this leads to us achieving performance improvements over their work.

Iterative Graph Learning: Recently, there has also been work on learning to augment graphs, such as by using end-to-end neural models optimized for the final task (Jiang et al., 2019; Chen et al., 2020). While these works do iteratively augment the graph similar to us, by doing it end-to-end they can be prone to be task specific (edges may be created solely for achieving higher classification accuracy), rather than learning a high quality social media representation. This may lead to issues at test time or in inductive settings. In our case, as we are adding edges based on learned graph similarities, we are strengthening the information communities that already exist, while uncovering hidden ones. Further, we can easily control for the relations and amounts of them that are added.

3 Model

We view fake news detection as reasoning over the relations between sources, articles, and engaging users in an information graph. We hypothesize that due to the principle of homophily, social ties leading to the formation of online communities will capture similarities and differences in content preferences within and across communities.

We capture the interaction between social information and news content using a heterogeneous graph defined in Sec. 3.1, and use a Relational Graph Convolutional Network (R-GCN), to create vectorized node representations for factuality prediction. The R-GCN defined in Sec. 3.2 allows our model to capture the different social communities, by creating contextualized node representations. For example, an article node is represented using its contents, source, and relationships with users engaging with it (which are also represented using their relationships to other nodes).

The success of us capturing the social communities through the R-GCN hinges on having strong social information (i.e., graph edges) to characterize them. Providing this information might not be straight-forward, as collecting social information at scale can be costly and noisy. Instead, we propose inference operators, defined in Sec. 3.3, which augment the graph with new edges, using the similarity between learned nodes representation to assess their compatibility. This allows the R-GCN to enrich each newly connected nodes' contextualized representation, improving factuality classification. In Sec. 3.4 we describe a reasoning framework, which iteratively enriches the graph using inference operators and computes the updated node representations based on the updated graph. The framework is depicted in Fig. 2.

3.1 Graph Creation using Social Context

Our graph consists of the following nodes: (1) S, the news *sources*. Each sources' (s_i) vec-

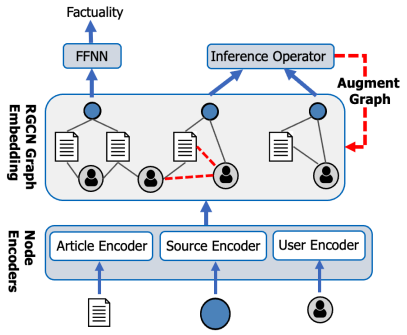


Figure 2: Factuality Prediction as Graph Reasoning

tor consists of its Twitter and YouTube profiles embeddings (numerical + LM features - details in App. A.2.1). Prior work (Baly et al., 2020b) showed that these features provide a strong signal. (2) A, the *articles* published by these sources. An article a_i vector captures its contents using a SBERT (Reimers and Gurevych, 2019) RoBERTa (Liu et al., 2019) model, as it provides strong, meaningful sentence embeddings. (3) U, the Twitter *users* that interact with articles and sources, and provide the social context for them. The description is applicable to the source-level (Baly et al., 2020a) and content-level (Nguyen et al., 2020) settings, where elements in S or A, res., are our classification targets. The user vector is identical to the Twitter embedding mentioned above.

The graph is formed by first adding the sources as individual nodes. Then, connecting each source with up to 300 articles ($e = \{s_i, a_j\}$). Next, we add social context to the graph via Twitter users that interact with sources: **(1) Following Sources:** We add up to 5000 users that follow sources, connecting each user to new sources they follow ($e = \{s_i, u_j\}$). These are likely to indicate a positive relationship. **(2) Discussing Articles:** We connect each article with users that tweet its title/link within a 3 month period of publication ($e = \{a_i, u_j\}$). These users provide the means for fake (and real) news spread, allowing us to model this process. Finally, social interactions, a crucial component for analyzing fake news propagation, are captured by scraping up to 5000 followers of each Twitter user, and connecting existing users with edges if they one follows another.

3.2 Graph Embedding

Given the observed interactions in the graph, we train a GNN to learn an embedding function, which will be used by the inference operators (Sec 3.3).

As a node embedding function, we utilize Relational Graph Convolutional Networks (R-GCNs) (Schlichtkrull et al., 2018), that generalize traditional GCNs to handle different relationship types, thus allowing us to better capture their interactions and improve their representation. Intuitively, R-GCNs create contextualized node representations by considering the graph structure through graph convolutions and learn a composition function:

$$\mathrm{h}_i^{l+1} = ReLU\left(\sum_{r\in R}\sum_{u\in U_r(v_i)}\frac{1}{z_{i,r}}W_r^l h_u^l\right),$$

where h_i^l is the hidden representation for the i-th node at layer l and $h_i^0 = v_i$ (output of the node encoder); $U_r(v_i)$ represents v_i's neighboring nodes connected by the relation type r; $z_{i,r}$ is for normalization; and W_r^l represents trainable parameters.

To obtain meaningful node representation used for capturing factuality, we optimize the Node Classification (NC) objective of Fake News Detection. After obtaining the source representations from the R-GCN, we pass them through the softmax activation function and then train using categorical cross-entropy loss, where the labels are factuality.

3.3 Inference Operators

We define multiple *inference operators* that enable the creation of new edges based on learned information graph inferences. The different operators capture our intuition about how connecting node pairs of different types would contribute to trustworthiness propagation. For example, pairs of users that are not explicitly connected in the graph (i.e., do not follow each other) but share articles with similar factuality levels may have similar levels of non-trustworthiness. Connecting them would provide more information to the nodes they connect to. One of our inference operators - adding user-user edges based on their node similarity in the embedding space - captures this situation.

For each inference operator type discussed below, we make edge connections based on the node representations we have learned, by computing similarity scores between all pairs of nodes (using the graph node embedding - efficiently with FAISS (Johnson et al., 2017)), and connecting the nodes with the top k similarity scores based on our model.

3.3.1 Social Information Based Operators

The first broad inference operator type adds edges between graph entities, in a similar way as a recommendation engine, suggesting entities to interact with each other, based on their graph relationships.

User-Source: We add edges between users and sources ($e = \{u_i, s_j\}$), using the top k most similar source/user pairs in the embedding space.

User-User: Pairs of users that interact with news in a similar way are connected ($e = \{u_i, u_j\}$). These users are likely to have the same beliefs and may even want to follow each other if they became aware of each others' profiles.

User-Article: We add edges between articles and users likely to be interested them ($e = \{u_i, a_j\}$)). This inference can be based on the target users' interactions with similar articles, or with other users sharing these articles.

3.3.2 News Content Based Operators

The second broad type connects entities based on content similarity. Unlike the previous set, these types of edges are not initially observed in the graph, which is one of the benefits of our setup, *allowing us to add inferences about latent relationships that underlie how information propagates*, such as coordination between different sources, information flooding by publishing similar content in multiple articles and "bad influencers", consistently propagating low-quality content.

Sources-Sources Sources likely to publish similar content at an equivalently factual level are connected ($e = \{s_i, s_j\}$).

Articles-Articles Articles that could be similar to each other in content are connected ($e = \{a_i, a_j\}$). To do this effectively, we first identify articles pairs that discuss the same event, approximated using the publication date and entity mentions overlap (using Flair (Akbik et al., 2018)) in their title. Second, we use an entailment model (Parikh et al., 2016; Gardner et al., 2017) to only connect articles that entail each other, as they are more likely to be talking about similar content.

Influencers Fake news is often spread by "bad influencers" that have a large following. Over the years, Twitter has launched campaigns intended to reduce fake news spread by suspending such users. This inference operator aims to do the same, by following these steps: (1) Using the training data, we label users by counting the paths to sources with a given factuality label. (2) Identify users without significant label variation in their followers group, as potential "news influencers". We avoid users with mostly highly factuality followers. (3) At inference time, we connect new users to influencers in this initial set, with a special edge type, indicating similarity to an influencer. We add the top k users, experimentally set to 500 (App. A.2.5).

Model	Performance		
	Acc	Macro F1	# Edges
`M1` : Majority class	52.43	22.93	-
`M2` : (Baly et al., 2018)	66.45	61.08	-
`M3` :(Baly et al., 2020b)	71.52	**67.25**	-
`M4` : Replication of (Baly et al., 2020b)	69.38	63.63	-
`M5` : Node classification (NC)	68.90	63.72	-
`M6` : `InfOp` Best Model	**72.55**	66.89	32K

Table 1: Results on (Baly et al., 2020b). Our best model (`M6`) achieves a 3.17% acc improvement compared to our re-implementation (which uses the same data -> `M4` vs `M6`), and the state-of-the-art on this dataset (which used different data - outside of train/test source labels - that was not released). Further, applying the inference operators (`InfOp` with 32K added edges) improves acc by 3.65% compared to Node Classification

3.4 Joint Inference and Representation

The inference operators we defined use the graph embedding function to identify new relationships that would potentially improve the embedding quality and allow for better information propagation during learning. The two steps are clearly interdependent. Now, we describe our iterative graph learning framework that builds on this dependency, and continually learns better social context representations in the graph by applying the inference operators, and then retraining the graph. It can be seen in Algorithm 1 and runs the following steps:

(1) **Initial Representation** In this step, we train the graph G using the framework described in Sec 3.2 to get an initial graph representation.

(2) **Inference Step** Apply inference operators (Sec 3.3) based on the learned representation.

(3) **Learning Step** After, we continue the training process for the graph.

We continually apply the two steps until convergence, based on development set performance. Additional details about the process are provided in Appendices A and C. When done, we retrain the model based on the final graph uncovered by applying the inference operators. Through this iterative approach, we continually improve our representation of the social media framework that enables fake news propagation, and reveal hidden relationships critical to understanding fake news spread.

4 Experiments

We evaluate our model's ability to predict fake news better on two challenging tasks: Fake News

Algorithm 1 *Joint Representation Learning and Reasoning for Characterizing Information Sources*

Input: $G_0 = S, A, U$
Output: L_s (labels of sources), R-GCN parameters ϕ
Initialization: $\phi_0 \leftarrow$L$_{final}$ over G_0
`Initial graph embedding uses Node Classification (NC)`
while not converged **do**
  Infer: $G_i = G_{i-1} \cup$ `InferOperators`(G_{i-1}, ϕ_{i-1})
  `Use inference operators to augment the graph`
  Learn: $\phi_i \leftarrow L_{nc}$ over G_i
  `Retrain R-GCN over new graph`
end while
Final Training: $\phi_{final} \leftarrow$L$_{nc}$ over G_{final}
`Reset parameters and train final graph using NC`
return $L_s \leftarrow \phi_{final}$ over G_{final}
`Predict unknown sources, using the final R-GCN`

Source Classification, and Article Classification.

4.1 Dataset and Collection

To evaluate our model's ability to predict the factuality of news medium, we used the Media Bias/Fact Check dataset (Baly et al., 2018, 2020b). The public dataset consists of 859 sources, each labeled on a 3-point factuality scale: *low*, *mixed*, and *high*. Using the Twitter API[1], we gather an average of 27 user engagements for each articles (Sec 3.1). Our final graph consists of 69,978 users, 93,191 articles, 164,034 nodes, and 7,196,808 edges. Details about the setup we used when training our graph (chosen using the dev set), and our scraping protocol are in Appendix A. Our code is available[2].

To evaluate fake news article detection, we used the dataset released by (Nguyen et al., 2020), put together from Twitter data using related work on rumor classification (Kochkina et al., 2018; Ma et al., 2016) and fake news detection (Shu et al., 2018). For each article, the dataset provides its source and a list of engaged users. We also collected the followers for each user, leading to a graph with 48,895 users, 442 sources, and 1,050 articles.

4.2 Fake News Source Classification

Table 1 shows our results on source classification. We evaluate our models on the average of all 5 data splits released by (Baly et al., 2020b), using 20% of the training set sources as a development set. We report results on accuracy and Macro F1-score. We compare to (Baly et al., 2020b, 2018) (M2, 3), who to the best of our knowledge achieve the strongest performance on this dataset. As (Baly et al., 2020b) did not release the article and social media data they used, we replicate their setup using the data we scraped (and their code), and compare to that as well (M4). Despite us optimizing their model, our results are worse than their released performance, so we hypothesize that their data on our setup may lead to better overall performance.

When training our initial graph with only observed data using the Node Classification (NC) fake news loss and the *same data* as our replication of (Baly et al., 2020b), we obtain similar performance to their approach (M5 vs M4). When we apply our inference operators, and then train the graph identically (as in M5), we notice a 3.65% acc. improvement (M5 vs M6), showing the *clear benefit of our inference operator setup on this task*, and answering our research question that the added information helps. Further, this setup achieves the state-of-the-art on (Baly et al., 2020b), exceeding both our replication with the same data (by 3.17% acc.) and their published results (by 1.03% acc.).

Model	Split	Performance	
		AUC	# New Edges
FANG	90%	75.18	-
SVM	90%	75.89	-
NC	90%	83.48	-
InfOp	90%	**85.89**	10,000
FANG	70%	72.32	-
SVM	70%	59.18	-
NC	70%	73.15	-
InfOp	70%	**77.76**	10,000
FANG	50%	71.66	-
InfOp	50%	**73.88**	10,000
FANG	30%	70.36	-
InfOp	30%	**72.63**	10,000
FANG	10%	66.83	-
InfOp	10%	**67.51**	10,000

Table 2: On (Nguyen et al., 2020), we achieve the SOTA on all data splits (% of data used for training). Our model beats the strong FANG model (Nguyen et al., 2020), SVM, and the Node Classification (NC).

4.3 Fake News Article Classification

Our results for article classification are in Tab 2. We compare to (Nguyen et al., 2020) (FANG), who to the best of our knowledge have the best performance, and compared to several competitive baselines in their work (Ruchansky et al., 2017). Nguyen et al. are also the most similar to us (as said in Sec 2, they also train GNN's), but they do not make unobserved interactions explicit, rather they modify the loss function they used when training to better capture them. Our setup is identical to the strong (Nguyen et al., 2020) setup (we use their

[1] https://developer.twitter.com/en/docs
[2] https://github.com/hockeybro12/FakeNews_Inference_Operators

released data and data splits) except we use different Twitter and Article representations, and we also consider Twitter follower edges. In addition, FANG (Nguyen et al., 2020) considered temporal aspects of how tweets propagate, which we do not, and we hypothesize that this may improve our performance. For this reason, we are using *less data* compared to FANG, apart from the fact that we consider Twitter user followers. For proper comparison, we also evaluate our representations by training a SVM, and in App. Tab 8, we evaluate our model with the same representations as FANG. We evaluate all of their data splits in Tab 2 (90% -> 90% of data for training, 10% for test, etc.). NC evaluates our model performance on the observed data. We show the best results (extended results and details in the App. C), and as can be seen, applying inference operators also improves performance on fake news article classification on all data splits (as much as 4.61% AUC), reinforcing that explicitly learning and creating unobserved relationships in the graph enables us to detect fake news content better. Also, we achieve SOTA by average 4.26%.

5 Discussion

In this section, we analyze our best model with inference operators (Table 1 `M6`) for fake news source detection (Baly et al., 2020b) by answering the following research questions:
(1) *Ablation study: What is the contribution of each inference operator?*
(2) *Can our model learn on limited data? Does our inference-based representation help?*
(3) *Can we learn meaningful user communities?*
(4) *What type of inferences does our model make for each inference operator?*
(5) *Can we detect the factuality of new content?*
(6) *What embeddings do we learn?* (App. B.1)
(7) *How many edges should we add for each inf. operator?* (App. B.2)
(8) *How long does running inference operators take?* (App. B.3)

5.1 Ablation Study

In Table 3, we evaluate each of our inference operators, trained using our joint learning and inference algorithm for up to two iterations. To evaluate the accuracy of the edge connections we make when applying the inference operators ("Inf. Acc"), we compare the labels of the two nodes connected by an inferred edge (i.e., accurate decisions connect nodes with similar labels). Since labels are only associated in our data with sources, we define a heuristic for computing labels for article and user nodes based on the most common gold label in all the sources they were directly connected to in the initial graph (ex: a user that follows 3 high factuality sources is assigned a high factuality label). We also report the number of edges connected in each setup (dev set for all params).

We note that almost all of our models with inference operators result in performance improvements over the baselines (Tab 1 `M4, 5`), showing that capturing these hidden relations and making them explicit with new edges helps in fake news detection. Moreover, several of our inference operators (users-users/sources-sources) achieve high accuracy, while all perform better than random, showing that we can make useful edge connections after learning the initial information graph. Furthermore, applying multiple inference operators in multiple iterations through our setup (`InfOp` Users-Users and Users-Articles) leads to the strongest performance on this task. To evaluate the potential of our approach, we also evaluate our performance if we had no inaccurate edge predictions (i.e. 100% Inf. Acc), and see significant improvements. Note that this is a potential of our setup, as it involves using all the data (including the training set) to determine the user labels and then filtering out inaccurate edge predictions.

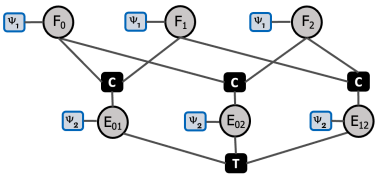


Figure 3: Inf. over Social and Factuality Information

Global Inference Operators An interesting direction for future work is to capture the interdependency between inference operators applications. We suggest a first step, based on probabilistic inference (Pacheco and Goldwasser, 2021), described in the factor graph in Fig. 3, applied to the Users-Users operators. We define two decision variable types, F associated with a user's factuality prediction, and E associated with the inference operator outcome on a user pair. Each is associated with a scoring function, ψ_1 scoring users factuality assignments, and ψ_2 scoring user pairs based on embedding similarity. The assignments are con-

Model	Performance				
	Acc	Macro F1	Std Acc.	Inf. Acc	Edges
InfOp Users-Sources	69.02	63.21	2.41	41.90	5,000
InfOp Users-Users	71.97	66.34	2.24	82.79, 96.28	30,000
InfOp Sources-Sources	69.84	64.48	2.21	80.18, 73.26	100
InfOp Articles-Articles	68.09	61.39	2.50	51.99	2,000
InfOp Users-Articles	70.63	63.76	2.48	34.92	2,000
InfOp News Influencers	70.42	62.59	1.20	-	1,000
InfOp Users-Users and Users-Articles	**72.55**	**66.89**	1.70	64.70	32,000
InfOp All 100% Acc.	75.19	70.84	4.29	100.00	30,080
InfOp Global Users-Users	72.17	64.95	1.90	79.21	6,732

Table 3: Ablation study on (Baly et al., 2020b). The strongest performing model uses the users-users and user-articles inference operators. Inf. Acc. evaluates the accuracy of the edge connections inference operators make, based on gold user/article labels determined by the source they are most often connected to.

Model	% Train	Acc	F1	Inf. Acc
NC	10%	61.04	49.64	-
InfOp	10%	66.27	56.01	67.08
NC	30%	62.79	49.48	-
InfOp	30%	67.44	60.29	65.96
NC	50%	65.11	48.20	-
InfOp	50%	68.60	63.83	66.36
NC	100%	66.86	61.49	-
InfOp	100%	72.55	66.89	64.70

Table 4: Weakly supervised settings (data split 0).

Model	k=17	k=55	k=200
B1 : User Trained	33.3, 50.9	35.4, 54.4	44.9, 60.1
B2 : User *InfOp*	33.3, 51.8	38.2, 57.2	47.5, 62.9
C1 : PF RoBERTa	33.98	34.03	36.21
C3 : PF Connect	34.21	39.02	50.76
C3 : PF *InfOp*	34.38	41.22	52.03

Table 5: Cluster purity of users (factuality, bias) and PolitiFact (PF) (factuality)

nected using two sets of constraints: **C**, ensuring factuality label consistency in users connected via a predicted edge, and **T**, ensuring transitivity across pairs of edges, sharing a node. We use MAP inference to identify the solution edge set. The results in Tab. 3, show a modest improvement (72.17) compared to local inference (71.97), obtained using significantly less edges (6.7K compared 30K). This experimentally shows a benefit of using global probabilistic inference to more intelligently determine edges to connect, rather than only using embedding similarity as we did before (here we also considered user factuality, and other decision variables/scoring functions can be added in the future). Details and other potential benefits of this setup are provided in Appendix D.

5.2 Weakly Supervised Training

Next, in Table 4 we evaluate our model on using limited training data for fake news source classification, by training on a smaller set of sources (still using the entire graph and full test set). Here, we see the ability of our inference operators to strongly improve performance (NC vs *InfOp*), as they reveal relationships the model could not learn otherwise in the weakly supervised setting, which shows how our system could be useful for detecting recently published news.

5.3 Learned Information Communities

Now, we analyze how well our user-user inference operator allows us to learn information communities of users (Tab 5). To do this, we cluster (K-means, Tab 5 shows different values of K) users before (B1) and after the inference operators are applied (B2), and evaluate the cluster purity based on user labels. To compute purity, each cluster is assigned to the class which is most frequent in the cluster, and then the accuracy of this is measured. We assigned labels to users using the same heuristic described in Sec. 5.1. We see that the users cluster better after the inference operators are applied (even via bias labels), showing our ability to use them to form information communities.

5.4 What Does our Model Learn to Connect?

Here, we analyze the inference operators by analyzing specific edge connections that are made. We see that the model makes smart choices in connecting nodes that may be part of the same information community. For example (more in Appendix B.4), a low factuality article discussing Democrats as 'dangerous open border fanatics' was connected to a user with bio 'BuildtheWall ... DEMONRATS'.

5.5 Incorporating New News Content

Finally, we evaluate how well our model can incorporate unseen news content, by clustering (like before) 1500 fact-checked claims from PolitiFact[3]. In Table 5, we first cluster the initial RoBERTa em-

[3]https://www.politifact.com

beddings of these claims (C1) and then add them into the graph by connecting them to five graph articles that have similar embeddings to them (C2). Next, we use our user-article inference operator to connect each of these articles to users (C3). It's clear that the RoBERTa embeddings statements don't cluster well by factuality. However, once they are added into the graph (C2) and after they are connected via inference operators (C3), they do. This further shows how our framework, especially through inference operators, allows the better detection of unseen news content (in this case claims). Not only can we determine its factuality, but we can also determine what other users are likely to interact with it.

6 Summary and Future Work

We propose an approach for tackling fake news detection by continually improving social context representations. To achieve this, we developed an iterative representation learning and inference framework that learns an initial graph embedding, and then applies different *inference operators* to reveal hidden relationships in the graph. We continually capture more knowledge about the social dynamics that allow fake news to propagate. We showed strong performance on fake news detection, across several datasets and settings.

Our current work looks at increasing the accuracy of the inference operators by adding external knowledge. We began exploring this direction by using an entailment model to infer article relationships using content similarity. We also explore additional ways to jointly model inference operators and capture the dependencies between them.

We believe this work helps pave the way for further research connecting text analysis along with its social context (Pujari and Goldwasser, 2021; Hovy and Yang, 2021; Pacheco and Goldwasser, 2021; Yang et al., 2016), a natural fit for many NLP tasks.

Acknowledgements

We thank the anonymous reviewers of this paper for all of their vital feedback. This work was partially supported by an NSF CAREER award IIS-2048001.

7 Ethics Statement

To the best of our knowledge no code of ethics was violated throughout the experiments done in this paper. We reported all hyper-parameters and other technical details necessary to reproduce our results, and release the code and dataset we collected. We evaluated our model on two different datasets and tasks (source and article fake news classification), but it is possible that results may differ on other datasets. However, we feel our methodology is solid and applies to any social media fake news setting. For space constraint we moved some of the technical details and discussion to the Appendix section. The results we reported supports our claims in this paper and we believe it is reproducible. Any qualitative result we report is an outcome from a machine learning model that does not represent the authors' personal views. For any results that we discuss on the data we use, we will not include account information and all results will be anonymous.

In our dataset release for (Baly et al., 2020b), we include sources, users, and articles. Sources are public information provided in (Baly et al., 2020b), and we map each to an ID. We scraped up to 300 articles for each source (as many as we could), which we map to an ID. We also scraped users that interact with articles, which we also release. Each user is given by their Twitter ID (which may be invalid or not provided if the user deleted their profile), and their graph ID. The Twitter ID of the Tweet the user propagates about the article is also given. We also scraped users that follow sources, and this information is released by providing the user ID's that interact with each source ID's. Finally, we provide the representations for each user, article, and source we used as our initial embedding in the graph. Our data is meant for academic research purposes and should not be used outside of academic research contexts. All our data is in English.

Our framework in general does not create direct societal consequence and is intended to be used to defend against fake news. While our model could be used to build better fake news spreaders, our approach of identifying information communities through inference operators, could also prevent against that.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

Michael Barthel Elisa Shearer Amy Mitchell, Jef-

frey Gottfried. 2016. The modern news consumer. *Pew Research Center*.

Ramy Baly, Giovanni Da San Martino, James Glass, and Preslav Nakov. 2020a. We can detect your bias: Predicting the political ideology of news articles. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4982–4991, Online. Association for Computational Linguistics.

Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James Glass, and Preslav Nakov. 2018. Predicting factuality of reporting and bias of news media sources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, Brussels, Belgium.

Ramy Baly, Georgi Karadzhov, Jisun An, Haewoon Kwak, Yoan Dinkov, Ahmed Ali, James Glass, and Preslav Nakov. 2020b. What was written vs. who read it: News media profiling using text analysis and social media context. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, ACL '20.

Yu Chen, Lingfei Wu, and Mohammed Zaki. 2020. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in Neural Information Processing Systems*, 33.

Elizabeth Dubois and Grant Blank. 2018. The echo chamber is overstated: the moderating effect of political interest and diverse media. *Information, Communication & Society*, 21(5):729–745.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. 2018. Political discourse on social media: Echo chambers, gatekeepers, and the price of bipartisanship. In *Proceedings of the 2018 World Wide Web Conference*, pages 913–922. International World Wide Web Conferences Steering Committee.

Matthew Gentzkow and Jesse M Shapiro. 2011. Ideological segregation online and offline. *The Quarterly Journal of Economics*, 126(4):1799–1839.

Felix Hamborg, Norman Meuschke, Corinna Breitinger, and Bela Gipp. 2017. news-please: A generic news crawler and extractor. In *Proceedings of the 15th International Symposium of Information Science*, pages 218–223.

Yi Han, Shanika Karunasekera, and Christopher Leckie. 2020. Graph neural networks with continual learning for fake news detection from social media. *arXiv preprint arXiv:2007.03316*.

Naeemul Hassan, Fatma Arslan, Chengkai Li, and Mark Tremayne. 2017. Toward automated fact-checking: Detecting check-worthy factual claims by claimbuster. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1803–1812.

Dirk Hovy and Diyi Yang. 2021. The importance of modeling social factors of language: Theory and practice. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 588–602, Online. Association for Computational Linguistics.

Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo. 2019. Semi-supervised learning with graph learning-convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11313–11320.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

Jooyeon Kim, Dongkwan Kim, and Alice Oh. 2019. Homogeneity-based transmissive process to model true and false news in social networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 348–356.

Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2018. Pheme dataset for rumour detection and veracity classification.

David MJ Lazer, Matthew A Baum, Yochai Benkler, Adam J Berinsky, Kelly M Greenhill, Filippo Menczer, Miriam J Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, et al. 2018. The science of fake news. *Science*, 359(6380):1094–1096.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444.

Van-Hoang Nguyen, Kazunari Sugiyama, Preslav Nakov, and Min-Yen Kan. 2020. Fang: Leveraging social context for fake news detection using graph representation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1165–1174.

Maria Leonor Pacheco and Dan Goldwasser. 2021. Modeling content and context with deep relational learning. *Transactions of the Association for Computational Linguistics*, 9:100–119.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2017. Automatic detection of fake news. *arXiv preprint arXiv:1708.07104*.

Rajkumar Pujari and Dan Goldwasser. 2021. Understanding politics via contextualized discourse processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1353–1367.

Walter Quattrociocchi, Antonio Scala, and Cass R Sunstein. 2016. Echo chambers on facebook. *Available at SSRN 2795110*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Filipe N Ribeiro, Lucas Henrique, Fabricio Benevenuto, Abhijnan Chakraborty, Juhi Kulshrestha, Mahmoudreza Babaei, and Krishna P Gummadi. 2018. Media bias monitor: Quantifying biases of social media news outlets at large-scale. In *Twelfth International AAAI Conference on Web and Social Media*.

Natali Ruchansky, Sungyong Seo, and Yan Liu. 2017. Csi: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 797–806.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer.

Chengcheng Shao, Giovanni Luca Ciampaglia, Onur Varol, Kai-Cheng Yang, Alessandro Flammini, and Filippo Menczer. 2018. The spread of low-credibility content by social bots. *Nature communications*, 9(1):1–9.

Kai Shu, Deepak Mahudeswaran, and Huan Liu. 2019a. Fakenewstracker: a tool for fake news collection, detection, and visualization. *Computational and Mathematical Organization Theory*, 25(1):60–71.

Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. 2018. Fakenewsnet: A data repository with news content, social context and spatialtemporal information for studying fake news on social media. *arXiv preprint arXiv:1809.01286*.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter*, 19(1):22–36.

Kai Shu, Suhang Wang, and Huan Liu. 2019b. Beyond news contents: The role of social context for fake news detection. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 312–320.

Svitlana Volkova and Jin Yea Jang. 2018. Misleading or falsification: Inferring deceptive strategies and types in online news and social media. In *Companion Proceedings of the The Web Conference 2018*, pages 575–583.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science*, 359(6380):1146–1151.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, et al. 2019. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*.

Yi Yang, Ming-Wei Chang, and Jacob Eisenstein. 2016. Toward socially-infused information extraction: Embedding authors, mentions, and entities. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1452–1461.

A Supplemental Material: Fake News Source Detection

In this section, we provide implementation details for our models for fake news source detection. The dataset we use has 859 sources: 452 *high* factuality, 245 *mixed*, and 162 *low*, and was released publicly by (Baly et al., 2020b)[4]. The dataset does not include any other raw data (articles, sources, etc.), so we must scrape our own.

A.1 Data Collection

For each source, we attempted to scrape news articles using public libraries (Newspaper3K [5], Scrapy [6], and news-please [7] (Hamborg et al., 2017)). In the cases where the web pages of the source news articles was removed, we used the Wayback Machine [8]. We attempted to scrape up to 300 articles for each source, but this was not always possible. Overall, our sources have an average of 109 articles with a STD of 36.

For Twitter users, we used the Twitter API[9] to scrape 5000 followers for each Twitter account we could find (72.5% of the sources, identical to (Baly et al., 2020b). In addition, we used the Twitter Search API to search articles on Twitter and find any Tweets that mention the article title or URL within 3 months of the article being published. We then downloaded the users that make these Tweets as well, and added them to our graph, linking them to the respective article they tweeted about. Finally, to increase the connectivity of the graph and accurately capture the interactions between the users, we also scraped the followers of every Twitter user. We then filtered the users to only add to our graph ones that either interact with multiple sources (through source or article connections) or another user, so that every node would be interconnected.

We did not scrape YouTube accounts, but rather used the same ones as the released (Baly et al., 2020b). They found YouTube channels for 49% of sources and published this information.

[4]https://github.com/ramybaly/News-Media-Reliability
[5]https://github.com/codelucas/newspaper
[6]https://github.com/scrapy/scrapy
[7]https://github.com/fhamborg/news-please
[8]https://archive.org/web/
[9]https://developer.twitter.com/en/docs

A.2 Experimental Settings

A.2.1 Initial Embeddings

Our initial Twitter embedding for each source and engaging user was a 773 dimensional vector consisiting of the SBERT (Reimers and Gurevych, 2019) (RoBERTa (Liu et al., 2019) Base NLI model) representation of their bio (up to the first 512 tokens) concatenated with the following numerical features: a binary number representing whether the source is verified, the number of users a source follows and the number that follow it, the number of tweets it makes, and the number of favorites/likes its' tweets have received. For, YouTube, the embedding we used was the average of the number of views, dislikes, and comments for each video the source posted. For articles, we used the SBERT RoBERTa model to generate an embedding for each article, which was a 768 dimensional vector representing the article text, up to the first 512 tokens.

A.2.2 Model Setup

Our models are built on top of PyTorch (Paszke et al., 2019) and DGL (Deep Graph Library) (Wang et al., 2019) in Python. The R-GCN we use consists of 5 layers, 128 hidden units, a learning rate of 0.001, and a batch size of 128 for Node Classification. Our initial source and article embeddings have hidden dimension 768, while the user one has dimension 773. We use a final fully connected layer for classification, of size 3 for (Baly et al., 2020b).

For our Joint Inference and Representation Learning framework, we choose parameters using the development set (20% of training sources) for one of the training data splits, and then apply them uniformly across all the splits, when training the final models. We ended up running all the inference operators for 2 iterations (except users-sources, articles-articles, and users-articles) before they converged (we determined convergence using the dev set), although we hypothesize that a higher accuracy of choosing the edges to connect would allow us to run this process longer. The number of edges shown in Table 1 were also chosen based on the development set.

When applying inference operators, we make sure that at least 50% of the nodes that are being connected are connected to sources that are not in the train set. For example, to be considered linked to a non-train set node, if we were connecting two

users, at least one of those users would need to be linked to a non-train set node directly (either by following a non-train set source or interacting with an article from a non-train set source). Not only does this setup simulate a real-world scenario when the model is deployed and inference operators are connecting new articles/sources to existing nodes, but it also makes sure that the graph does not only connect nodes that are learned better as they are closer to training set nodes, which helps performance.

Our models were trained on a 12GB TITAN XP GPU card and training each data split for Node Classification takes approximately 4 hours. The user-user inference operator phase to add 20,000 edges took 997.8724 seconds, or approximately 16 minutes.

A.2.3 Replication of Prior Work

To replicate (Baly et al., 2020b) (M4), we used their released code with our features. Specifically, we used our article, Twitter profile, Twitter Follower, and YouTube embeddings. This setup consists of all the data in our graph, and also provided the best performance in (Baly et al., 2020b).

A.2.4 Explanation of 100% Inf. Acc Approach

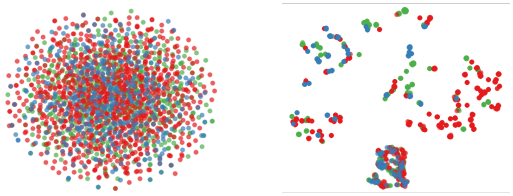
In Sec 5.1, we had mentioned results where we ran our approach with 100% Inf. Acc, and said this was a potential of our approach. Here, we explain that setup in further detail.

For our 100% accurate edges, we needed labels for the users, articles, and sources. We computed those using all the data (train, dev, and test set), which is cheating. The labels for the articles were computed based on the source they were directly connected to (even if it was a test set source), and the labels for the users were computed based on sources they interacted with or articles they followed (again, even if it was a test set). This is why we consider this a potential of our approach, because in practice we do not have access to the test set labels and thus cannot do this inference process with 100% accuracy.

A.2.5 Explanation of News Influencers

In Sec 3.3.2, we mentioned one of our inference operators as news influencers. Here, we explain how we choose those:

To choose influencers, we first looked at each user in the graph and determined how many followers it had. Then, using just the training set, we determined the labels of each user (based on the articles/sources they connect to, and what is the most common label). We consider influencers as users who mostly have one label of people following them. For example, a user with predominantly low factuality users following them is likely an influencer of low factuality information.



(a) (b)

Figure 4: TSNE plots of our article embeddings before and after the model is trained. (a) shows the embeddings before the model is trained, when we used SBERT (Reimers and Gurevych, 2019) RoBERTa (Liu et al., 2019) to encode the articles. In this case, the articles are not well separated based on factuality. In (b), after the graph model is trained and inference operators are applied, they are, showing that factuality in our graph propagates to articles as well.

Thus, now that we have the user labels and we know who follows each user, we can determine a distribution for each potential influencer. If the gap between the most common label (of all the people who follow them) and the next most common is larger than a threshold (which we experimentally set to 3000 for high/mixed factuality and 100 for low since there were less of those), then we consider a user an influencer. We added 500 influencers to begin with, and through the inference operator in Table 3, added 1000 more. When a user is connected to an influencer, it is done with a special edge type in the R-GCN.

B Fake News Source Detection Analysis

In this section, we build upon Sec 5 by providing extra analysis experiments and further detailing the ones already discussed.

B.1 Extra Analysis: Graph Embeddings

In this sub-section, we present a new analysis that attempts to answer the question of how meaningful our models embeddings are. To do this, we analyze article embeddings before (a) and after the model is trained (b), and plot them in Fig 4 (red=high factuality, blue is low, green is mixed). The embeddings show that our inference operators enable factuality

to propagate to the representation of the articles, as the articles are more cleanly clustered after the graph is trained.

B.2 Extra Analysis: Inference Operator Edge Count Experiment

Now, we provide details on how the performance changes for each of our inference operators based on how many edges we add for each. The number of added edges is a hyper-parameter, as we can decide when to stop adding edges (we added edges based on the top k similarity scores, and that k is the hyperparameter). The results are shown in Tab 6. For each inference operator, we evaluate three different number of added edges hyperparameter settings, showing results on the test and a development set (20% of sources in the train set).

B.3 Extra Analysis: How Long Does Running Inference Operators take?

Now, we evaluate how long running our inference operators take, showing that our procedure is efficient. We timed our users-user inference operator when it added 20,000 edges. This is the most amount of edges of any of our single inference operators. Thus, this is is the most computationally expensive inference operator, except for articles-articles - which did not perform well.

We timed it on a single GPU GeForce GTX 1080 Ti GPU, with 6 Intel Core i5-8400 CPU @ 2.80 GHz processors over 5 runs (all data splits averaged). The inference operator took 997.8724 seconds (just over 16 minutes).

Thus, we believe that our procedure does not involve huge computational and memory costs. This is partly because we used FAISS (Johnson et al., 2017) to do the embedding similarity search efficiently, as we mentioned in Sec 3.3. Other inference operators and cases where less edge connections need to be made will take less time, especially on stronger machines. This is also an interesting direction for future work, where we can find the right balance between the number of edges that should be added, while taking into account the impact they would have on performance.

B.4 Continued Analysis: Inference Operator Specific Example Analysis

Next, we continue our analysis discussion from Sec 5 by analyze the inference operators by analyzing specific edge connections that are made. We see that the model makes smart choices in connecting nodes that may be part of the same information community.

(1) A low factuality article discussing Democrats as ‘dangerous open border fanatics’ was connected to a user with bio ‘BuildtheWall ... DEMONRATS‘.

(2) A user with a bio containing ‘Held Hostage by the Environmentalist Movement‘ that was connected to predominately low factuality sources was connected to an article talking about Democrats wanting to destroy fun through the ‘green movement’.

(3) A user with bio containing ‘Lets save our Republic‘ was connected to an article mentioning a celebrity tweeting a photo of a well known Democrat leader killing the President.

(4) A user with bio ‘makes memes for the masses’ was connected to another user with bio ‘here to have fun‘.

(5) Two articles talking about the Presidents first week in office were connected.

Overall, these examples show that our model can make decisions based on information communities that could exist on social media, as it connects users/articles that are talking about similar content, which is what we are trying to capture.

B.5 Continued Analysis: Learned Information Communities

Here, we provide the technical details behind our analysis in Sec 5.3. When clustering users, we evaluated purity based on user labels. The labels were computed for each user based on the most common label of the source it was directly connected to in the initial graph (ex: a user that follows 3 high factuality sources is assigned a high factuality label). We also evaluated how users clustered with respect to bias labels (second number in each cell of the first two rows of the table), using the bias labels from (Baly et al., 2020b).

B.6 Continued Analysis: Incorporating News Content

In our analysis in Sec 5.5, we wanted to see how well our model can incorporate news content that was not seen before. For this, we scraped 1500 fact-checked statements from the popular fact-checking website PolitiFact. Each statement was labeled by experts on a 1-5 scale in order of increasing

Model	Performance				
	Dev Acc	Dev Macro F1	Test Acc	Test Macro F1	Edges
InfOp Users-Sources	66.66	58.28	69.49	63.63	2,500
InfOp Users-Sources	67.24	60.32	69.02	63.21	5,000
InfOp Users-Sources	66.51	58.19	69.49	63.35	7,500
InfOp Users-Users	65.49	57.22	69.72	63.84	15,000
InfOp Users-Users	66.36	57.66	71.97	66.34	20,000
InfOp Users-Users	65.50	54.61	69.02	62.49	25,000
InfOp Sources-Sources	67.38	59.36	69.02	62.49	75
InfOp Sources-Sources	67.53	60.42	69.84	64.48	100
InfOp Sources-Sources	66.51	57.75	70.65	63.35	125
InfOp Users-Articles	67.53	61.18	69.37	62.47	1,500
InfOp Users-Articles	67.96	60.50	68.09	61.39	2,000
InfOp Users-Articles	67.53	61.13	68.09	60.90	2,500

Table 6: Experimental results on (Baly et al., 2020b) of our InfOp models for different hyper-parameter settings of number of edges added. For each inference operator evaluated, we show the performance on the development set and test set. We used the development set to select the hyper-parameter value to use in our final models. Results are averaged across all five data splits

level of factuality, which we converted to high (top two), low (bottom two) and mixed (other - # 3), to match our source labels. Then, we incorporated the statements into our graph by computing the similarity of each statements' RoBERTa embeddings to all of the graph article RoBERTa embeddings. We connected each PolitiFact statement to it's top 5 similar graph articles, generated graph embeddings for each statement, and then clustered this in Table 5 D2. Then, we applied our user-article inference operator, where each article was now a PolitiFact statement, and clustered them again in D3. When doing this, we found improved clustering at all values of k, showing that the inference operators allowed us to better capture the factuality of the PolitiFact statements. Further, the inference operator had an Inference Accuracy of 51.45% (user label inferred from user source connections matching PolitiFact statement label), showing that the graph was able to make good decisions when choosing what to connect.

C Supplemental Material: Fake News Article Detection

For Fake News Article detection, we used the dataset released publicly by (Nguyen et al., 2020). However, in our graph setup for the initial node embeddings, we use RoBERTa (Liu et al., 2019) article embeddings (See Sec 3.1) and Twitter Profile embeddings (see Sec A.2.1), which (Nguyen et al., 2020) did not. Thus, we downloaded the article text from the links released by (Nguyen et al., 2020), and encoded it using RoBERTa, with the first line of the article being the claim text (Nguyen et al., 2020) used, if it was available. As before, we encode up to the first 512 tokens. Further, we downloaded the Twitter profiles, but the users we have in our graph are the same as (Nguyen et al., 2020). Later in this section, we will also evaluate a version of our model where we use the same initial representations as (Nguyen et al., 2020), instead of the RoBERTa SBERT ones.

In addition to the initial embeddings, we also scraped followers of each social media user (Nguyen et al., 2020) used, and connected users where at least one follows the other in the graph. As mentioned in Sec 3.1, this process enables us to capture the social media landscape that enables fake news propagation, and is helpful for us to build information communities via our inference operators.

(Nguyen et al., 2020) also incorporated temporal information to their graph, which we do not. We hypothesize that adding temporal information to our setup will lead to further improvements, and leave it for future work.

Apart from these differences, our work is identical to (Nguyen et al., 2020) (FANG), except we utilize inference operators. Our graph is the same as described above in Appendix (albeit with different data), with sources, users, and articles with the same node/edge types. We train the same RGCN with the same parameters, using the same inference operator connection process.

Model	Performance	
	AUC	Edges
FANG 90%	75.18	-
NC 90%	83.48	-
SVM 90%	75.89	-
U-U 90%	**85.89**	10,000
U-U 2 Iter. 90%	85.53	20,000
U-A 90%	82.14	10,000
U-A 2 Iter. 90%	81.75	20,000
U-A + U-U 90%	74.46	20,000
FANG 70%	72.32	-
NC 70% NC	73.15	-
SVM 70%	59.18	-
U-U 70%	69.41	10,000
U-U 2 Iter. 70%	67.59	20,000
U-A 70%	**78.81**	10,000
U-A 2 Iter. 70%	48.56	20,000
U-A + U-U 70%	75.54	20,000
FANG 50%	71.66	-
NC 50%	71.35	-
SVM 50%	58.07	-
U-U 50%	69.84	10,000
U-U Iter. 2 50%	**74.75**	20,000
U-A 50%	73.88	
U-A Iter. 2 50%	70.93	20,000
U-A + U-U 70%		20,000
FANG 30%	70.36	-
NC 30%	70.98	
SVM 30%	24.68	-
U-U 30%	72.63	10,000
U-U Iter. 2 30%	71.85	20,000
U-A 30%	67.85	10,000
U-A Iter. 2 30%	**71.92**	20,000
U-A + U-U 30%	70.76	20,000
FANG 10%	66.83	-
NC 10%	62.04	-
SVM 30%	22.73	-
U-U 10%	**67.51**	10,000
U-U Iter. 2 10%	62.97	20,000
U-A 10%	66.52	10,000
U-A Iter. 2 10%	59.86	10,000
U-A + U-U 10%	67.06	20,000

Table 7: Extended Final results on (Nguyen et al., 2020). From the results, it is clear that applying inference operators leads to performance improvements. Key: FANG = (Nguyen et al., 2020), NC = Node Classification, 90% = percentage of data used for training, the rest was for test/validation, SVM = support vector machine trained on our article + Twittere user features, U-U = Inference Operator connecting users to users, U-A = inference operator connecting users to articles, Iter. 2 = running the inference operator for 2 iterations.

Our extended experimental results below (Table 7) show how our different inference operators (connecting users to users -> U-U and connecting users to articles -> U-A, doing them in multiple iterations -> Iter 2, doing both at once -> U-A + U-U) enable performance improvements even on fake news article classification. As before, we use the dev set to judge convergence. For a complete evaluation, we also include results on our model trained without adding inference operators (just on Fake News Article Node Classification (NC)), and us using our article and social media features in a SVM (we trained a SVM using grid-search parameter optimization where the features for each article were the RoBERTa embedding we used in the graph plus the average of all the user profiles the article was connected to in the final graph). The SVM allows us to evaluate how much performance improvements we get from our Twitter/Article embeddings/other data compared to (Nguyen et al., 2020) FANG, and we can see that the embeddings do not lead to strong performance. Thus, it can be seen from the results that our graph setup, and on top of that applying inference operators, always leads to performance improvements compared to (Nguyen et al., 2020) (FANG), showing that our setup is useful for fake news article detection. Moreover, applying inference operators on top of our initial setup (NC), always leads to further performance improvements, achieving the best results on this task (to our knowledge). This shows the benefit of using our inference operator based framework for fake news article detection. Results can be seen in Table 7.

For further analysis on the initial representations, in Table 8, we evaluate our model performance with and without inference operators when we use the same node representations as (Nguyen et al., 2020) for articles, sources, and users. Nguyen et al. released these representations publicly with their paper, and each of them is 100 dimensional (articles are TFIDF + semantic from GLOVE, sources are TFIDF + GLOVE semantic of homepage and about us, users are TFIDF + GLOVE Semantic of profile bio). Everything else in our graph framework stays the same. Thus, our setup is now even weaker compared to (Nguyen et al., 2020), as we still do not include the temporal information that they used and we do not capture stance of Twitter users with respect to articles. Even in this case, we see that inference operators enable performance improve-

Model	Performance	
	AUC	Edges
FANG 90%	75.18	-
NC Same Rep. 90%	68.57	-
U-U Same Rep. 90%	69.25	10,000
FANG 70%	72.32	-
NC Same Rep. 70%	67.15	-
U-U Same Rep. 70%	70.54	10,000
FANG 50%	71.66	-
NC Same Rep. 50%	66.68	-
U-U Same Rep. 50%	69.58	10,000
FANG 30%	70.36	-
NC Same Rep. 30%	63.65	-
U-U Same Rep. 30%	66.26	10,000
FANG 10%	66.83	-
NC Same Rep. 10%	58.16	-
U-U Same Rep. 10%	65.70	10,000

Table 8: Results on (Nguyen et al., 2020) using the same representations as (Nguyen et al., 2020). In this setup, we use the same article (TFIDF + semantic from GLOVE (Pennington et al., 2014) of the text), source (TFIDF + semantic of homepage and about us), and user (TFIDF + semantic of profile bio) representations as (Nguyen et al., 2020), that they released online.

ments when compared to the node classification graph baseline, showing that inference operators can help models even with weak initial node representations. Further, the inference operator models compete with the best results from Nguyen et al.. We hypothesize that when inference operators are combined with the extra information Nguyen et al. used (stance prediction/temporal data), overall performance would be higher and exceed Nguyen et al. even with the same embeddings as them, and leave this for future work.

D Supplemental Material: Improved Inference

D.1 Inference Process

In Sec 5.1 we discussed a potential extension of our user-user inference operator based approach, where instead of adding the top k edges based on only embedding similarity search, we used a global probabilistic inference approach. Here, we explain that process in more detail.

In our current approach in Sec 3, we had considered similarity scores between all pairs of nodes for each inference operator, and chosen the top k to connect, where k was determined from the development set. While successful, this process may not always connect the best set of edges. For example, consider a situation when there are three users: *A, B, C*. Assume the inference operators decided to connect *A* and *B* with an edge, and likewise *B* and *C*. In this case, it's clear that *A* and *C* are likely in the same information community, as they are both connected to *B* (by the property of transitivity). However, in our current approach, we may not connect *A* and *C*, if their graph embeddings are not similar enough. At the same time, if *A* and *C* are very dissimilar, or they have a very different level of factuality (as determined by the training set), then we likely don't want to connect them (despite them both being connected to *B*. Thus, having a setup that allows all these decisions to be made jointly, would be beneficial.

For this reason, we explore how we can take advantage of global relational learning, to determine what edges to connect. In this work, we used a framework called DRaiL (Pacheco and Goldwasser, 2021). DRaiL is a probabilistic learning framework for learning a relational model using weighted logical rules. We described the settings as a factor graph in Fig 3 in Sec 5.1.

In the factor graph, we define two decision variable types, F associated with a user's factuality prediction, and E associated with the inference operator outcome on a user pair. Each is associated with a scoring function, ψ_1 scoring users factuality assignments, and ψ_2 scoring user pairs based on embedding similarity (similarity comes from FAISS, as before). The assignments are connected using two sets of constraints: **C**, ensuring factuality label consistency in users connected via a predicted edge, and **T**, ensuring transitivity across pairs of edges, sharing a node (explained above). DRaiL then uses MAP inference to identify the solution edge set. The results in Tab. 3, show a modest improvement (72.17) compared to local inference (71.97), obtained using significantly less edges (6.7K compared 30K).

Each scoring function (ψ_1, ψ_2) is given a weight in the MAP inference problem. For us, as factuality is important, we weighted ψ_1 (user factuality) with weight 5000 and ψ_2 (user-user similarity scores) with weight 1.0. Each constraint (**C**, **T**) is weighted equally. We determined all the weights experimentally, using the development set.

The factuality score for each user in the graph is probabilistic, spanning across the three values

- high, low, and mixed. We determined factuality labels for users using the training set, where users that follow training set sources or interact with training set articles are assigned the label of the respective source (if there are multiple, then this determines the probability - for example a user following 5 sources, 3 of which have a high factuality label in the training set, would have a 0.6 probability of being high factuality). For users that don't interact with training set articles, we look at all their indirect connections in the graph up to two hops away (a directly connected node is one hop away) and assign the label that way (for example, a user that interacts with another user that interacts with an article would be given the label of that article). Users that aren't assigned labels to this process are ignored, but as our graph is well connected, there are only 21 of these out of 69K. In the future, if new sources were added that were not in the training set and we wanted to use them to compute user labels (for example if their users weren't connected to anything in the training set), then we could also use the model source predictions to approximate their labels.

After receiving the top k similarity scores from the model based on FAISS, which we call candidates (30,000 candidates in the case of users-users), DRaiL solves the constrained MAP problem of deciding which edges to add based on the weighted scores and constraints provided (and discussed above). Through experimental testing, we decided to add a minimum of 5,000 candidates, and then DRAIL can further add edges based on which ones solve the optimization problem. In total, on average across the data splits for (Baly et al., 2020b) and the user-user inference operator, we added 6,732 edges.

D.2 Inference Benefits

Using a system like DRaiL for global inference has several benefits, some of which we discuss here.

Theoretically, through weighted scoring functions and constraints, DRaiL allows more knowledge to be incorporated into the node connection decision making process, which can eventually improve fake news detection performance. Here, we began to explore the users-users case, where we considered model similarity score, user factuality labels, and the transitivity constraint. However, more scoring functions and constraints can also be added in the future, such as increasing the likelihood of connecting users that talk about the same events. In this case, a semantic model could be used to determine the likelihood that two users are talking about the same event, and that could be passed in as an additional scoring function to a system like DRaiL. Similarly, there could also be additional constraints.

From an experimental perspective, by using DRaiL we saw performance improvements similar to our initial approach (slightly better in Table 3) by using significantly less edges (6,732 for DRaiL vs 30,000 for our similarity score approach). Adding less edges has several benefits, such as introducing less noise, being faster (which could also help in early detection of fake news), and not making the graph too large.

In the future, as more scoring functions and constraints are added, and more inference operator types are explored via an optimization system like DRaiL, we may see further performance improvements on fake news detection. Exploring this further is a very interesting direction for our future work.