Tackling Fake News Detection by Interactively Learning Representations using Graph Neural Networks

Nikhil Mehta
Department of Computer Science
Purdue University, West Lafayette, IN
mehta52@purdue.edu

Dan Goldwasser
Department of Computer Science
Purdue University, West Lafayette, IN
dgoldwas@purdue.edu

Abstract

Easy access, variety of content, and fast widespread interactions are some of the reasons that have made social media increasingly popular in today's society. However, this has also enabled the widespread propagation of fake news, text that is published with an intent to spread misinformation and sway beliefs. Detecting fake news is important to prevent misinformation and maintain a healthy society.

While prior works have tackled this problem by building supervised learning systems, automatedly modeling the social media landscape that enables the spread of fake news is challenging. On the contrary, having humans fact check all news is not scalable. Thus, in this paper, we propose to approach this problem *interactively*, where human insight can be continually combined with an automated system, enabling better social media representation quality. Our experiments show performance improvements in this setting.

1 Introduction

Over the last decade, an increasing number of people access news online (Amy Mitchell, 2016), often using 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), and a rapid spread of content. In particular, false news stories tend to spread at lightning speeds, and due to the volume, cannot be checked manually. An alternative to fact-checking claims, which is arguably easier to scale, is to focus on their source, and ask *who can you trust?*

Prior works have formulated this as a traditional classification problem using techniques such as feature-based SVM's (Baly et al., 2018, 2020), and more recently Graph Neural Networks (GNNs) (Li and Goldwasser, 2019; Shu et al., 2019; Han et al., 2020; Nguyen et al., 2020), which create a better representation of social media interactions. Graphs often consist of nodes corresponding to news sources (associated with a discrete factuality level - *high, low*, or *mixed*), the articles they release, and their social context, corresponding to social media users engaging and sharing information in their networks. GNNs can utilize this information by using edge interactions to create node representations contextualized by their graph neighbours. This leads to a stronger representation of the complex information landscape on social media that enables fake news to spread, allowing it to be better detected. For this reason, we adopt graphs as our automated framework (1).

Despite the success of these works, fake news detection is still a challenging research problem and human performance is significantly higher than fully automated systems (Shaar et al., 2020). Clearly, having humans fact check every information source is not scalable. Thus, our goal in this paper is to explore a different form of interaction with humans, where they can provide *advice* (Mehta and Goldwasser, 2019) to the automated system. Advice corresponds to localized judgements (provided through natural language) that help characterize the content and social interactions associated with sources. These judgements, associated with article and social media user nodes, are then propagated through the information graph using the GNN, allowing the system to take advantage of it to improve it's representation. As advice is not providing source labels directly, which is a time-consuming process requiring a global view of the source's interactions, it is scalable.

For example, one challenging aspect of the problem is that low-factuality ("fake news") sources may not always propagate false information (some of the articles they publish may be factual), and

Proceedings of the First Workshop on Interactive Learning for Natural Language Processing , pages 46–53

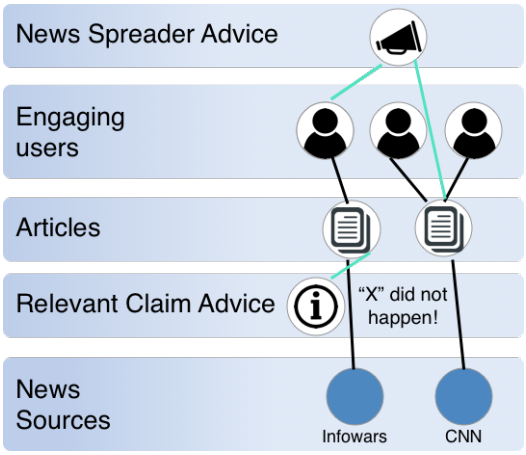

Figure 1: Information Graph capturing interactions between news sources, articles, and engaging users. Advice is added to the information graph by adding new nodes/edges(teal) based on the advice type (news spreader or relevant claims). Advice then provides information that can useful to clear up the complex social space the graph is modeling.

vice-versa (leading to model confusion). Human interaction, in the form of advice, can help clean up some of this uncertainty, by identifying claims containing egregious falsehoods. The model could then use this information and trust sources making these claims less. We refer to this form of advice, mapping a specific article to known falsehoods as *relevant claim advice*. In another case, referred to as *news spreader advice*, a human could inform the system that a user that is spreading a sources' articles frequently spreads lies, which would increase the likelihood that that source and any other source this user spreads articles from are fake. Fig 1 shows how both of these advice types can be seamlessly added to an information graph.

In this work, we show that our protocol in which humans iteratively provide these types of advice by interacting with the model (even after it is trained) improves overall fake news detection performance. In summary, we formulate fake news source detection as a reasoning problem over an information graph. We then suggest an interactive learning based approach for incorporating human knowledge as *advice* to clean up uncertain graph decisions, which allows us to better learn and reason on this graph. Finally, we perform experiments that demonstrate that this setup leads to performance improvements on fake news source detection.

2 Model

2.1 Graph Creation and Training

We start with defining our social context information graph. It consists of sources (S), articles they publish (A), and Twitter users that interact with sources/articles (U). Our goal is fake news source factuality classification. Each node in the graph is represented by a high dimensional feature vector, (similar to prior work (Baly et al., 2018, 2020; Nguyen et al., 2020)) to provide knowledge to the model that can be utilized when learning the graph embedding. Source and user feature vectors are created by concatenating embeddings based on their Twitter profiles (SBERT + features, details in Appendix A.2.1). Sources also can include YouTube profile embeddings. Articles are represented by the encoding text into a SBERT RoBERTa embedding.

Our graph is formed by first adding all the sources as individual nodes. We then scrape and add up to 300 articles (a_i) for each source, connecting each with an edge to the source that published it ($e = \{s_i, a_j\}$). Next, we add social context to the graph via Twitter users that interact with sources. We add up to 5000 users that follow sources, and users that tweet links to any articles in the graph within a 3 month period of the article being published ($e = \{s_i, u_j\}$, $e = \{a_i, u_j\}$). Users that follow/engage with sources are likely to be aligned with/propagating the view of the sources, and modeling this can be useful. Finally, in order to capture the social interactions between users in the graph, which is critical to capturing fake news propagation on social media, we scrape up to 5000 followers of each Twitter user and make an edge between a pair of existing users if one user follows another.

In order to learn the information captured by our information graph, we train a GNN to learn an initial embedding, on top of which we will apply the interactive protocols (Sec 2.2). As a node embedding function, we utilize Relational Graph Convolutional Networks (R-GCN) (Schlichtkrull et al., 2018) (they can handle the social media relationships well). We achieve meaningful representations and capture factuality of the different nodes in our graph by optimizing the Node Classification (NC) objective of Fake News Detection. After obtaining the source representations o_s from the R-GCN, we pass them through the softmax activation function σ and then train using categorical cross-entropy loss: $L_{nc} = -\sum_{i=1}^{C} y_i log(\sigma(o_s))$ where the C classes for y_i are either *high*, *mixed*,

or *low* factuality, and s is the current source.

2.2 Advice Protocols

We now describe the two advice protocols we utilize in this paper. As mentioned in the introduction, in this work, we define advice as a form of human provided judgement (typically provided through natural language) about intermediate relationships in the information graph, that cleans up the space of complex judgements made by the GNN, allowing us to better capture the challenging landscape on social media that enables fake news to spread (Fig 1). Advice is provided by humans interactively and continuously, so that the process is scalable (not many judgements are needed, and they can always be provided, even after the system is deployed). In this way, our advice protocols provide a mechanism for humans to interact with the automated graph system. We use two forms of advice:

2.2.1 Relevant Claim Advice

When a human provides relevant claim advice, they have some prior knowledge about a certain claim (or news statement), and are telling this information (the claim and what their belief about its' factuality is) to the model . For example, a human may know that a certain claim is not factual (perhaps many users on social media spread it and thus the human has seen it before). The human would then provide this claim and a message about its factuality through natural language.

Once a human has provided advice in the form of a claim that may be relevant, the model must decide which articles (if any) the claim is relevant for. Once it does so, it can add a *new node* in the graph for the claim (represented similar to the article text node with SBERT RoBERTa embedding), and connect it to the relevant article(s), allowing the advice knowledge to easily propagate through the graph (either by re-training the GNN or using the trained GNN to embed the advice node appropiately $\rightarrow$ we evaluate both setups in Sec 3). This automated setup allows for minimal effort needed from the human, making the advice simple to provide.

To do this, first, the model filters a subset of sources (a process which we call *filtering*), whose articles could be candidates to receive advice. As mentioned earlier, advice cleans up complexities in the information graph, so these sources are ones which the model predicts the label of with low confidence (we rank Softmax scores for this). Then, for each filtered source's article, the model decides if the claim provided by the human is relevant, by analyzing content in two ways. **(1)** First, a heuristic is used to determine if the advice and article are talking about the same event. To do this, the model extracts the entities from the advice claim and the article (we use the FLAIR tagger (Akbik et al., 2019)), and determines if any of them overlap. If they do, the model also checks the date the advice claim was made, and makes sure it is within a one week period of the article being published. **(2)** Then, to further check content relevance, we use an entailment model (Parikh et al., 2016) and a sentence selection model (Nie et al., 2019) to check if any sentences from the article (chosen by the sentence selection model) entail the advice claim. If they do, the chance that the two are talking about similar content is higher. If there is an entailment, the advice statement d node is connected to the article a with an edge. All advice is also connected to a special label node h, m, or l, representing 'high', 'low', or 'mixed' factuality, based on the advice label (which is provided by the human), so that the model can easily represent that information.

In our interactive process, which we evaluate in Sec 3.2, a human can continuously provide relevant claims (through natural language) based on knowledge they posses as advice, and through the process described above, the model can determine which articles to use it for (thus connecting the advice in the graph). In this way, the human interacts with the system to clear up potential confusion about certain articles, which propagates via the graph through sources and users, to lead to better fake news detection performance.

2.2.2 News Spreader Advice

When providing news spreader advice, the human informs the system that a certain user is a *bad actor*, meaning that they frequently spread lies. This knowledge would increase the likelihood that articles this user tweets, and other users they interact with, are also non-factual. The user is then connected via an edge to a special 'low' factuality node, signifying to the model the set of users that are deemed to not be trusted.

2.2.3 Simulating Advice

In this preliminary work, we simulate the two previous forms of human provided advice by collecting data from fact-checking websites (PolitiFact, Snopes, USA Today, The Washington Post) and Twitter (details in Appendix A.1). For relevant

Model	Performance		
	Acc	Macro F1	# of Advice
M1 : Majority class	52.43	22.93	-
M2 : Best Model from (Baly et al., 2020)	71.52	**67.25**	-
M3 : Our replication of (Baly et al., 2020)	69.38	63.63	-
M4 : Node classification (NC)	65.76	55.97	-
M5 : Relevant Claim All Advice	69.02	61.89	29,673
M6 : Relevant Claim Advice Filtering 25%	68.09	60.28	17,305
M7 : Relevant Claim Advice Selection	70.54	62.61	4,106
M8 : Relevant Claim Advice Filtering + Selection 50% + 50%	68.56	60.80	4,106
M9 : Relevant Claim All Advice Match Label	76.36	70.89	8,677
M10 : News Spreader Only Bad	67.40	59.57	2,643
M11 : News Spreader Bad 50%	65.91	58.65	1,350
M12 : News Spreader Bad 50% + 50%	66.35	58.26	2,643

Table 1: Final results.

claim advice, we scrape all claims fact-checked by these websites and their factuality scores, and use that. This simulates humans providing advice in the real world, as a claim and some factuality insight about it are given. For news spreader advice, we use the Twitter API to determine all users that have been suspended since we initially collected our dataset, and use them as our news spreaders. Twitter manually suspended most of these users after the storming of the US capitol, so using this data allows us to accurately simulate human advice.

Although in this work we did not explicitly ask users to provide advice based on our learned graph model, our approximation of human advice that we collected was provided by human experts, and is thus relatively close to real advice that a human could provide. Relevant claims advice is based on real news claims that experts have associated factuality labels with, and Twitter manually suspended the users we used for news spreader advice.

3 Experiments

3.1 Dataset and Collection

To evaluate our model's ability to predict the factuality of news medium, we used the Media Bias/Fact Check (MBFC) dataset (Baly et al., 2018, 2020) (859 sources, each labeled on a 3-point scale based on their factuality: *low*, *mixed*, and *high*). We provide graph statistics in App. A.

3.2 Fake News Classification

Table 1 shows our results. We average our models on all 5 data splits released by (Baly et al., 2020), using 20% of the training set sources as a development set, and report results on accuracy and Macro F1-score for fake news source classification. We compare our advice protocol models to the baseline-graph based model trained only on node classification (NC - no advice provided, M4). For completeness, we included the results of the SOTA (Baly et al., 2020) (M2), as well as replication of their setup using the data we scraped (and their code). Our results are worse than their released performance, so we hypothesize that their data on our setup may lead to better overall performance.

For relevant claim advice, we evaluate settings in which we provide all the advice we scraped (29,673 statements - M5), where we provide advice only to the bottom 25% of sources that our model is not confident on during train/dev/test time (M6, all advice that passes the event filter is used $\rightarrow$ at least one entity in the articles title matches the advice claim and the dates are within one week of each other), and where we provide advice to all sources and make sure articles pass the event + entailment + sentence selection criteria (M7 - full setup in Sec 2.2.1). In all these setups, the advice is provided on the best model in M4, and then parameters are reset and the model is re-trained to learn how to incorporate the advice. M8 is different and more interactive, as advice is first provided on the bottom 50% of confident sources based on the protocol in Sec 2.2.1, then the model is re-trained. Then, the rest of the advice is provided as in M7, except this time the model isn't retrained. This simulates advice being continuously provided interactively by the human in the real world, and performance still improves. In this setting, as no re-training of the model is necessary, advice can be

quickly utilized. All setups improve performance from the baseline, and using the filtering + sentence selection approach (M7) leads to the best performance, showing that the content of the advice matching the article matters. Thus, in the future when humans provide advice that is more likely to match the content of the articles, it is likely that we will see further performance improvements. Further, it is likely that less advice will need to be provided to see improvements.

For completeness, in M9 we also evaluate an upper bound, where advice provided by the human would be 100% accurate, i.e. the human would only provide advice that matched the article label (article label based on the label of the source).

Finally, we evaluate news spreader advice, first when all news spreaders are told to the model (M10), then when only 50% are (M11), and finally when 50% are told, the model is retrained, and then the rest are provided (M12, simulating true interaction).

In all advice models, performance improves from the NC baseline, showing that these types of advice can be helpful to the model. Furthermore, once the advice is provided, we can add more (M8, M12), and still see performance improvements without having to retrain the model, demonstrating a true interactive scenario, where a human can continuously be interacting with an automated system. In addition, providing advice as a localized judgement is simple and easier than labelling an entire source, so large amounts of advice can be collected from different experts to improve results. In the future, when we experiment with humans providing advice that is more content relevant (not simulating), the amount of advice needed could also decrease.

3.3 How Does Advice Help?

In this section, we analyze a few specific cases of how relevant claim advice is used by the model to improve performance. In one case, an article from a news source labeled as spreading fake news was discussing how a Democratic leader would become Vice President if the President was impeached. Our model incorrectly predicted the factuality of this source. However, an advice claim from Snopes stating that the 25th amendment would not lead to this Democratic candidate immediately becoming Vice President was able to push the prediction of the source in the appropriate direction. In another case, advice that a specific former President was the first to speak against the current President was provided through PolitiFact with a False label, pushing a different source towards the fake news label.

4 Summary and Future Work

In this paper, we proposed an approach to tackle fake news detection interactively by designing a protocol for a graph based system to continuously solicit human advice, and take advantage of it to improve overall information quality, which enables better fake news detection performance. We showed the benefits of two forms of advice (relevant claims and news spreaders), provided either all at once or continuously. In the future, we plan to have humans actually provide this advice, and explore other advice types.

5 Acknowledgments

We thank the anonymous reviewers of this paper for all of their vital feedback. This works was partially supported by an NSF CAREER award IIS-2048001. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

6 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. For space constraint we moved some of the technical details to the Appendix section which is submitted with this manuscript. 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. Any results that we discussed on the data we used did not include account information and all results are anonymous. We anonymized the Twitter, article, and advice (Politifact, Snopes, USA Today, The Washington Post) data we collected to respect the privacy policy of the various websites and user data. While our overall approach does rely on user insights, each advice statement provided does not directly affect the final prediction, so a system receiving advice for fake news detection can not be easily manipulated.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. Flair: An easy-to-use framework for state-of-the-art nlp. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

Michael Barthel Elisa Shearer Amy Mitchell, Jeffrey Gottfried. 2016. The modern news consumer. *Pew Research Center*.

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. 2020. 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.

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*.

Chang Li and Dan Goldwasser. 2019. Encoding social information with graph convolutional networks forpolitical perspective detection in news media. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2594–2604.

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*.

Nikhil Mehta and Dan Goldwasser. 2019. Improving natural language interaction with robots using advice. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1962–1967.

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.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2019. Combining fact extraction and verification with neural semantic matching networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6859–6866.

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.

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*.

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.

Shaden Shaar, Alex Nikolov, Nikolay Babulkov, Firoj Alam, Alberto Barrón-Cedeno, Tamer Elsayed, Maram Hasanain, Reem Suwaileh, Fatima Haouari, Giovanni Da San Martino, et al. 2020. Overview of checkthat! 2020 english: Automatic identification and verification of claims in social media. *Cappellato et al.[10]*.

Kai Shu, Suhang Wang, and Huan Liu. 2019. 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.

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*.

A Supplemental Material

In this section, we provide implementation details for our models. The dataset we use has 859 sources: 452 *high* factuality, 245 *mixed*, and 162 *low*, and was released publicly by (Baly et al., 2020)[1]. 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 [2], Scrapy [3], and news-please [4] (Hamborg et al., 2017)). In the cases where the web pages of the source news articles was removed, we used the Wayback Machine [5]. Overall, our sources have an average of 109 articles with a STD of 36.

For Twitter users, we used the Twitter API[6] to scrape 5000 followers for each Twitter account we could find (72.5% of the sources, identical to (Baly et al., 2020). In the graph, we then connected these users to the sources they follow. 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 talk 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 made sure to only add users to our graph 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 ones released by (Baly et al., 2020). They found YouTube channels for 49% of sources.

For collecting relevant claim advice from news sources (PolitiFact, Snopes, USA Today, and the Washington Post), we used the Google FactCheck tool[7], along with scraping the PolitiFact website. We downloaded 29,673 claims in total.

[1]https://github.com/ramybaly/News-Media-Reliability
[2]https://github.com/codelucas/newspaper
[3]https://github.com/scrapy/scrapy
[4]https://github.com/fhamborg/news-please
[5]https://archive.org/web/
[6]https://developer.twitter.com/en/docs
[7]https://toolbox.google.com/factcheck/explorer

A.2 Experimental Settings

A.2.1 Initial Embeddings

Our initial Twitter embedding for each source and engaging user was a 773 dimensional vector consisting the SBERT (Reimers and Gurevych, 2019) (RoBERTa (Liu et al., 2019) Base NLI model) representation of their bio concatenated with the following numerical features: a binary number representing whether the source is verified, the number 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. Sources that did not have a YouTube channel had a random YouTube embedding. For articles, we used the SBERT (Reimers and Gurevych, 2019) RoBERTa (Liu et al., 2019) model to generate an embedding for each article. For relevant claims advice, we used the same SBERT (Reimers and Gurevych, 2019) RoBERTa (Liu et al., 2019) model to generate an embedding for each advice claim.

We also mentioned special factuality nodes in Sec 2.2.1 that are added into the graph and connected to advice claims, to allow the model to easily represent the advice label (either the claim label or the fact that a Twitter user is spreading bad news). These nodes are initialized randomly with a 768 dimensional embedding that is then learned when the graph is re-trained after the initial set of advice is added.

A.3 Graph Statistics

We downloaded an average of 109 articles per source, with a STD of 36, and user-engagements (talking about articles, following sources/other users) via the Twitter API [8](sources have an average of 27 users directly connected to them or to their articles). Using this data we construct the graph as described in Sec 2.1, which consists of 69,978 users, 93,191 articles, 164,034 nodes, and 7,196,808 edges. Details about the model setup we utilized when training our graph (chosen using the development set), and our scraping protocol are in Appendix A.

A.3.1 Model Setup

Our models are built on top of PyTorch (Paszke et al., 2019) and DGL (Deep Graph Library) (Wang

[8]https://developer.twitter.com/en/docs

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, article, and advice embeddings have hidden dimension 768, while the user one has dimension 773.

We choose parameters using the development set (20% of train sources) for one of the training data splits, and then apply them uniformly across all the splits, when training the final models. We choose the stopping point for the best performing models on which to apply advice on top of based on the dev set. In the setups where we did not apply all the advice at once, we determined all the advice that could be relevant and then randomly chosen which ones to apply based on the percentage of the total the experiment required.

Our models were trained on a 12GB TITAN XP GPU card and training each data split for Node Classification takes approximately 4 hours, while training Link Prediction Pre-training and the combined initialization step takes 24 hours.

A.3.2 Replication of Prior Work

To replicate (Baly et al., 2020) (`M3`), 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., 2020).