Report: Graph Structuring for Multimodal Misinformation Detection

Jing Liu
jing.liu6@mail.mcgill.ca
McGill University School of Computer Science

Kellin Pelrine
kellin.pelrine@mila.quebec
McGill University School of Computer Science
Mila - Quebec AI Institute

ABSTRACT

As social media has become increasingly prominent in our day-to-day lives, the ability to separate information from misinformation has become vital. Existing misinformation detection models often fail to outperform transformer-based language model baselines. In this project we examine critical components for establishing a new framework to use multimodal data for better misinformation detection.

We explore two main directions: (1) heterogeneous graph neural network models classifying individual tweet graphs (i.e. a tweet node with edges to its replies, images, etc.), and (2) linking multiple tweets together to create a single global graph. For the first direction, we implemented Heterogeneous Graph Attention Networks (HAN). For the second, we examined multiple ways of creating a global graph and classifying the nodes. Results so far have been negative, still failing to outperform baselines. However, we have identified likely causes and ideas to conquer them in future work.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; **Natural language processing**;

KEYWORDS

graphs, multimodal data, misinformation, social media, natural language processing, COVID-19

ACM Reference Format:
Jing Liu and Kellin Pelrine. 2021. Report: Graph Structuring for Multimodal Misinformation Detection. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACKNOWLEDGMENTS

This project builds on previous and in progress unpublished work with Jacob Danovitch and Reihaneh Rabbany, which in turn built on work published in [22].

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Today's societal challenges often come with a broad array of relevant data. In detecting misinformation on Twitter (e.g. classifying tweets as true or false), researchers have shown that not only is the tweet itself relevant, but also the replies, network context, images, and more [3, 9, 34, 36]. Uniting these disparate data types is a challenging problem: many existing methods that have reported state-of-the-art results on various datasets [3, 9, 17, 34, 36–38] fail to outperform strong text-only baselines [23]. This project examines essential components needed to combine multimodal (i.e. multi-typed, heterogeneous) data effectively using graph structures, with the aim of (1) developing a new method that outperforms both baselines and previous work for misinformation detection, and (2) providing a starting point for developing a general framework to handle multimodal data.

Using a graph structure facilitates encoding contextual relationships between different data points in the dataset, and different data types (modes) within each data point. For example, a reply on Twitter saying "this story is false" has an entirely different meaning depending on if it refers to the original tweet, or a reply that contradicts the original tweet. Simply combining a tweet and its replies, without encoding the structure, will lose this critical context. Additionally, tweets made by the same user might be classified similarly; one can imagine a malicious user writing multiple false tweets. By connecting data points and modes in graphs, we can preserve and learn to use this information, and also have the flexibility to incorporate new types of data as they become available.

Prior to this class project, we set up embedding methods for individual data modes, a heterogeneous graph framework to combine them, and experiments on various Twitter datasets. We refer to this model as HeteroGapformer, building on Gapformer [22], which works similarly but on a homogeneous graph. HeteroGapformer treats each data point as one graph, with the features encoded as nodes in the graph, and the type of node corresponding to the type of feature (see Figure 1). For example, the source tweet is assigned one node, and the replies another node each with a different type from the source tweet. This turns the original problem into a graph classification problem. After applying the embedding method for each node individually, HeteroGapformer uses GCN [11] layers which operate on each type of edge (corresponding to the combinations of connected node types), followed by aggregation and pooling operations to combine everything together. Also, we tried concatenating the pooled output to a language model embedding of the source tweet, in order to preserve more information from the most important mode. However, the performance is mediocre, approximately on par with a text-only baseline.

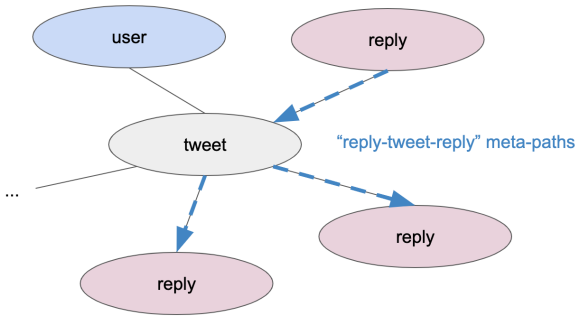

Figure 1: A graph consisting of one tweet and its features (denoted as "individual graphs" below). Different node types are represented by different colors.

Therefore, in this project we reconsider how the graph encoding is done and how the graph is structured. In particular, we consider Heterogeneous Graph Attention Networks (HAN) [33]. These are designed to work on heterogeneous graphs, in contrast to the previous method which essentially applied a homogeneous method to each edge type independently and then summed everything together. We also examine methods for constructing a global graph that connects examples together, such as connecting tweets that share users.

The main contributions discussed in this report include:

- Experimental evaluation of the ideas above. We test multiple implementations and hyperparameter settings. Unfortunately, results so far are negative, again failing to outperform text-only baselines.
- Insights into causes for poor performance. We suggest the graph classification with HAN is failing due to lack of sufficient structure in each individual graph, and the global graph is failing due to lack of homophily.
- Directions for future work. Building on the issues identified above, we outline potential fixes for future testing.

In the following sections, we begin by discussing the literature in Section 2, followed by our methodology and experimental setup in section 3. In section 4, we discuss the results. Finally, we conclude with directions for future research.

2 BACKGROUND AND COMPONENTS

2.1 Misinformation Detection

Misinformation detection, particularly on social media, is a serious and important societal problem. Negative impacts of misinformation span politics [18], economics [26], and public health [4, 24]. As highlighted by recent surveys [1, 2, 19, 27, 28, 40], misinformation detection is a broad field with a great deal of prior work, using many different types of models and data.

However, [23] showed that many sophisticated and often multimodal models in the recent literature, despite achieving state-of-the-art results compared to previous methods, are often not evaluated against the strongest possible baselines. In particular, papers often report results with older text-only baselines such as SVMs, but recent language models based on transformer [31] architectures, such as RoBERTa [15] and COVID-Twitter-BERT [20] "are competitive with and can even significantly outperform recently proposed state-of-the-art methods" for misinformation detection.

Nonetheless, because it is clear that a great deal of useful misinformation detection data is contained in modes beyond text, it should be possible to create multimodal methods that give clearly superior performance to language models. In this project, we attempt to develop such a method by adding graph structure on top of individual mode embeddings. This enables us to leverage language models to get effective embeddings for text modes, while also enabling use of more modes than a language model can handle. For example, it lets us treat a source tweet and replies differently, and use information about named entities contained in the text such as people and places.

2.2 Structure

The relationships between different modes of data are complex and varied. Many problems with graphs have a relatively clear structure imposed by the real-world problem, such as analysis of follower or citation networks. But here there is no such simplifying constraint, and there are many ways to configure the graph. For example, if a tweet cites a news article, should the replies to that tweet connect to the article? On the one hand they are clearly related, while on the other, they may not link the article themselves. It is thus up to us to choose the best way to link the graph to maximize the effectiveness of the model.

In HeteroGapformer, the node for the tweet itself was central, with replies, articles, features related to the user who posted it, and entities it mentioned linking to it. Entities and user features for replies were linked to directly to the respective reply. Each example was its own graph, with no connection to other examples beyond what the model learns in training.

A number of variations can be considered. For example, one could modify how the tweet and replies are connected together. However, this is likely to only result in a small improvement, because those nodes are already "close" in the graph to begin with. So in this project we focus on connecting nodes from different examples in order to create a more "global" graph. Because different examples were not connected at all in HeteroGapformer, this has the potential to create a much bigger improvement. Details are discussed in subsequent sections.

2.3 Interactions

"Interactions" loosely refers to how different node types interact while maintaining the graph representation. The following section, pooling, discusses how these separate nodes are combined into a single overall vector, from which we obtain the label.

In HeteroGapformer, after embedding each node's features individually, we applied a stack of generic convolutions using the default Deep Graph Library (DGL) implementation for heterogeneous graphs.[1] The model applies a convolution first within each relation type, then sums the results at the node level (i.e. for each node, adding together the representations for all the relations that connect to it). In [22] where we created a model with a homogeneous graph, we used more sophisticated convolution and attention

[1] https://docs.dgl.ai/guide/nn-heterograph.html

techniques [8, 32]. Unfortunately, these do not work out-of-the-box on heterogeneous graphs.

To optimize this component, we wanted to explore different methods of convolution, such as [25, 39]. We chose to go with HAN [33], for reasons detailed in Section 2.5.

2.4 Pooling

To move from a graph composed of many nodes to an overall representation that can be classified, we need a pooling mechanism. Previous versions of our model used attention-based pooling by first applying attention within each node type, and then between the node types. This two-stage process may be suspect because aggregating first within each type could make it more difficult to identify key information in the relationships between different types—it might be gone by the time it gets to the cross-type stage.

Work on this component is closely related to work on interactions. Again, a key challenge is to find an effective method that works with the heterogeneity.

2.5 HAN

HAN is the first heterogeneous graph neural network to use attention. To start, we define a meta-path as a sequence of relations that exists between two nodes, e.g. "reply-tweet-reply" or "user-tweet-user" [29]. Node-level attention learns the importance of the relation between a node and its meta-path neighbors. As an example, consider that certain replies are more relevant to the original tweet, e.g. if the original author responds. Second, semantic-level attention learns the importance of different meta-paths. In other words, since different meta-paths in a heterogeneous graph may capture different semantic information, attention is used to learn the most important meta-paths and their appropriate weights. Mechanically, HAN works as follows: a type-specific transformation matrix projects different types of node features into a given space. The two types of attention values are then combined to get an optimal, hierarchical combination of neighbors and meta-paths that represents the complexity in both the structure and semantic information stored in the original graph.

We chose HAN for two main reasons. First, HAN outperforms Graph Convolution Networks (GCN) on node classification for the heterogeneous ACM, DBLP, and IMDB datasets. Though those graphs are much larger than ours, they have a similar number of node types, and so HAN's increased performance on node classification may translate to graph classification as well. Second, HAN proposes pooling techniques different from our initial attempts. After projecting the different node types into a shared feature space, HAN pools neighboring nodes regardless of node type, which in theory prioritizes the graph's local structure more so than in our current method of pooling first within each node type, then between node types.

3 METHODOLOGY AND EXPERIMENT SETUP

3.1 Data

We used the PHEME dataset [12, 41]. This misinformation detection dataset contains 6425 source tweets about 9 varied news events. The tweets were collected as the events unfolded, and labels were given by professional journalists. There are four possible labels: true, false, unverified (for a tweet whose veracity was not confirmed one way or another during the data collection period) and non-rumor.

This dataset comes out-of-the-box with replies, user IDs, and other information that makes it much easier to work with and avoids any issue with deleted tweets or accounts that can no longer be accessed (a problem in some other Twitter datasets, because Twitter has limitations on what content can be shared, and tweets by suspended or banned accounts cannot be accessed through the API).

We split this data in two ways. First, for the experiments where each data point is turned into its own graph, we follow [38] by subsampling to balance classes and only examining data points with 4+ replies. This avoids potential confounders in our results (imbalanced classes, tweets with no replies, etc.). Despite this subsampling, it still preserves the difficulty of the dataset compared to other splits in the literature – state-of-the-art performance is actually lower than others such as [3, 34, 36].

Second, for all but one of the experiments where we construct a global graph, we take the entire dataset as is. To our knowledge, this alternative is not explored in recent literature. While this does make comparisons more challenging, it is important to have more data when forming the global graph, because the model needs enough connections to learn how they should influence the classification.

Finally, we also do an experiment with an event-based split, where we take the 5 events with the most examples, train on 4, and test on the largest. This is a much more difficult scenario compared to splitting randomly, because the model has to predict on an event whose content has not been seen at all in training. Unlike the other settings, language models alone perform poorly here [23].

3.2 Individual Graphs

HeteroGapformer initially trained both the language model and graph component end-to-end, meaning all examples (both the language model and graph components) are processed simultaneously. This step was done using BERT-Tiny [30], implemented with HuggingFace Transformers [35].

We implemented HAN using an existing DGL implementation [6]. Rather than modifying HAN to perform graph classification, we incorporated the HAN encoder and layers into the HeteroGapformer workflow, replacing the GCN. First, we added reverse edges to the PHEME dataset, such that we would be able to construct meta-paths that start and end at the same node. This is crucial for the node-level attention, which runs separately on each meta-path's node pairs.

Each HAN layer consists of one graph attention network (GAT) layer per meta-path [32]. Then, a semantic attention layer is applied across all GAT layers. This semantic attention consists of semantic-specific embeddings that are sent through a nonlinear transformation, and whose importance is measured as the similarity with a semantic-level attention vector q. The importance measures of each meta-path are normalized and aggregated into a final embedding. The results of the HAN encoder are passed into a global attention pooling layer following [14], which incorporates the relevant features extracted from the overall set of features. A linear classification head outputs the predictions. The full model is illustrated in Figure 2.

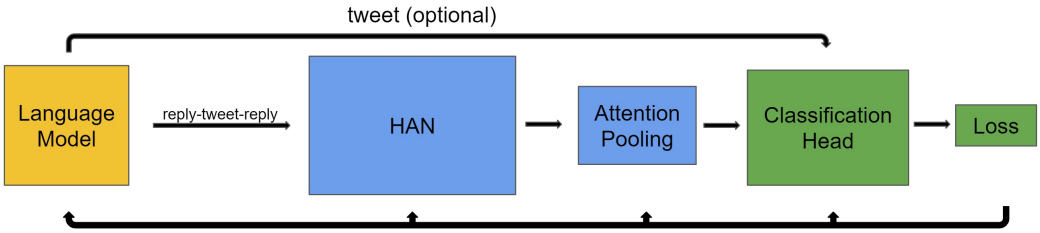

Figure 2: Individual graphs model.

We tested HAN in HeteroGapformer using two different metapath types: the first, self-loops (i.e. "tweet-tweet-tweet"), do not leverage graph structure and so we do not report their results. The second, "reply-tweet-reply", make use of reply features, and we report their results below. The non-baseline results are reported from one run each, using a learning rate of 1e-4, a precision of 16 bits, and trained over 50 epochs.

3.3 Global Graph

Constructing a global graph is motivated in two ways. First, it can facilitate the model learning information that is shared between data points. For example, a connection between two tweets that mention the same famous people and places can help the model learn that the topic is likely the same and can be approached similarly. Second, it enables label smoothing between related examples, which is a key component in algorithms like GCNs which have shown strong performance on many graph datasets [10]. For example, if the model sees 1000 tweets connected to a particular set users making replies that are all false, then that would be a strong sign that another tweet with the same connections is also false.

Therefore, instead of treating each data point as its own graph and doing graph classification, it could be beneficial to connect everything together in a "global graph". There are two challenges: determining how to connect this global graph, and designing and implementing classification models on it.

Considering first the connections, there are several pieces of data that are shared between examples. First, we can make connections based on the users posting tweets and replies. To determine if this connection could be beneficial, we examine the graph formed formed from the PHEME dataset when examples are connected when a user is shared in either the source tweet or reply. Note that connecting source tweet alone would not dramatically help the small graph problems, as less than 6.5% of the users in the dataset post 5 or more source tweets. But connecting examples that share a reply user as well gives a promisingly global graph - the giant connected component (GCC) size is 5978, out of 6425 total examples. Similarly, in Figure 1 below, we see a reasonable degree distribution - similar to many meaningful real-world graphs with power law distributions, and with a significant number of nodes having 10+ connections.

Another option is connecting the examples based on shared named entities within the text. Shared entities could help the model find examples that discuss similar things and thus should have similar labels. Entities in PHEME tweets and replies were extracted

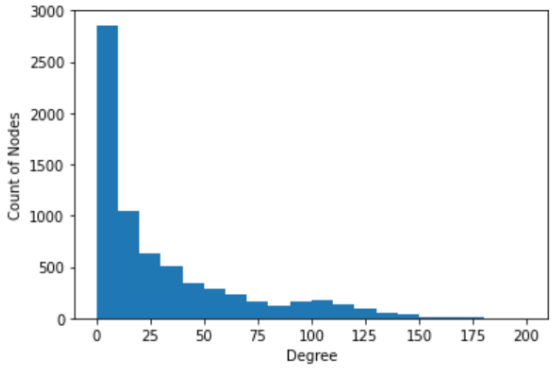

Figure 3: Degree of nodes connected by shared users.

in previous work using REL.[2] Connecting examples when they share one entity gives a GCC with 6077 examples, and the remaining 348 are all singletons. We see in Figure 2 that this leads to a very high degree distribution, with many nodes having hundreds or even thousands of connections.

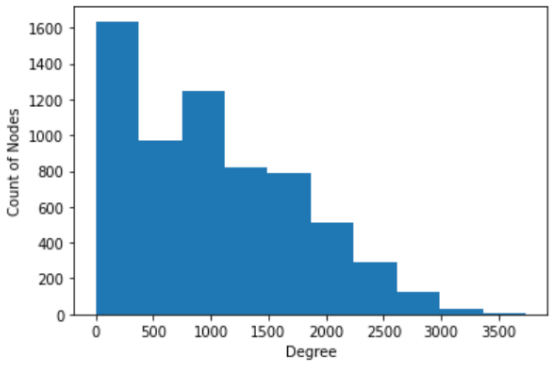

Figure 4: Degree of nodes connected by shared entities.

We examine multiple combinations of these options, discussed in Section 4.2.

[2]https://github.com/informagi/REL

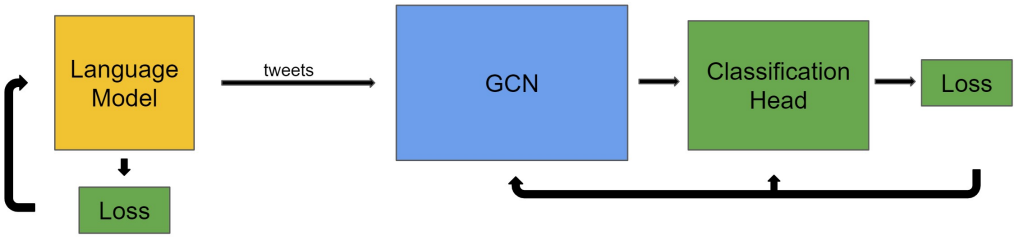

Figure 5: Global graph model.

The classifier architecture is shown in Figure 5. We first fine-tune BERT-Tiny [30] alone to classify the tweets. Although HeteroGAP-former trained both the language model and graph component end-to-end, this is not possible with a direct implementation of a global graph because all examples would have to be processed simultaneously, imposing prohibitive memory requirements. So first training the language model alone enables us to get meaningful representations for all the data points. This step was done through HuggingFace Transformers [35] and AllenNLP [7]. We use cross-entropy loss, optimized by AdamW [16] with learning rate 1e-5 for 50 epochs. Following [23], we do not tune the hyperparameters of the language model.

We get the embedding vectors from the last layer's CLS token, and pass them as the node features into a standard GCN implemented in PyTorch.[3] The aim with this is to obtain a simple model that improves on the baseline at least slightly, before examining more complicated methods to further improve performance. We again use cross-entropy loss here. We tested a variety of hyperparameter configurations: learning rates 1e-2, 1e-3, and 1e-4; weight decay 0 and 5e-4; dropout 0.1 and 0; hidden layer sizes between 1024 and 10240, and between 1500 and 15000 epochs.

We also examine different ways of splitting the dataset between the language model and GCN. The simplest way is to give the same training set to both pieces of the model. For this we randomly split the data 70-30, Besides this, we also consider a stacking-like setup, where we split the data 35-35-30, train the language model on the first 35%, the GCN on that plus the second 35%, and test on the remaining 30%. The idea here is to encourage the GCN to learn to correct the errors the language model makes on unseen data, similar to the stacking ensemble method. We also use a setup like this for the experiment with an event-based split, training the language model with three events, training the GCN with those three plus one more, and finally testing on the remaining one. In all cases we reserve 500 tweets from the training set during the GCN part to use as a validation set for GCN hyperparameter tuning.

In all experiments but one, we in the GCN part we only use the GCC. As reported above, this generally includes most of the nodes. While this means some data points are left out, they have few if any connections, so the performance of the global graph classifier is not likely to be good on them anyways, and could suffer overall from including them. Furthermore, we have readily-available predictions for these left out data points – the language model prediction alone. So this does not create any significant limitation in the model.

[3]https://github.com/tkipf/pygcn

4 RESULTS AND EVALUATION

4.1 Individual Graphs

In Table 1 we present the results of HAN, compared with the BERT-Tiny baseline. RTR refers to the reply-tweet-reply metapath. RTR + Tweet includes that and the skip connection for the tweet embedding.

Table 1: HAN results.

Method	Accuracy
BERT-Tiny	0.726
HAN RTR	0.63
HAN RTR + Tweet	0.657

4.2 Global Graph

We summarize the configurations tested in Table 2. We denote connections using the user of the source tweet by UserS, and by users in the replies by UserR. For the ones with notes in the other column:

- "Non-GCC" refers to using the entire graph instead of the GCC. This is due to sparsity of connections with users only - the GCC has only 102 nodes out of 6425 originally.
- "Logits" refers to using logits from the language model instead of embeddings.
- "All Entities" refers to using all entity connections. For the others with entities we require at least 3 connections to add an edge, with the aim of minimizing noise from entities that are essentially stopwords - frequently used but uninformative.
- "Event Split" refers to splitting the training and test data using the different events, as discussed in Section 3.1.

In all cases, performance was at least 5 percentage points worse than using the language model alone.

5 DISCUSSION

HAN's classification accuracy is worse than the text-only baseline. One reason for this might be that we were only able to run using

Table 2: Global graph configurations tested.

Connections	Stacking	Other
UserS	Yes	Non-GCC
UserS + UserR	Yes	
UserS + UserR	No	
UserS + UserR	No	Logits
Entity	Yes	
Entity	No	
Entity	No	All Entities
Entity + UserS	Yes	
Entity + UserS + UserR	Yes	
UserS + UserR	Yes	Event Split

one set of meta-paths, namely "reply-tweet-reply". For reference, the DGL HAN implementation uses both existing meta-paths in their graph dataset, and so we hypothesize that using additional meta-paths might yield greater performance. Additionally, it may be possible that HAN suffers from the same pitfalls as GCN did.

The current global graph structure and classifier are not effective. The degree and GCC numbers look reasonable, as discussed in Section 3.3. However, we found that the assortativity coefficient with respect to the labels[4] is extremely low. It ranges from at most 0.147 with UserS + UserR, down to below 0.065 when connecting all entities. This can be compared with the Cora dataset, for instance, which has 0.76 assortativity coefficient [5].

This indicates connected nodes have a relatively low chance of sharing the same label. A significant portion of GCN functionality is likely through label smoothing [10], which was one of the reasons to create a global graph in the first place. Therefore, we suspect the poor performance is due to this lack of homophily.

There are two directions for solutions. First, we plan to search for different ways of constructing the graph to increase homophily. These include:

- Connections based on domain of news articles. Some domains may be frequent sources of misinformation.
- More sophisticated filtering of connections. Besides just requiring a certain number of connections as we did here with entities, we can apply filters based on content. These could be based on distance between embeddings of tweets or replies, or sentiment analysis (for example, perhaps replies should only be connected if they share the same sentiment, suggesting they are taking the same stance towards the original tweet, e.g. saying it is wrong).
- Other datasets. While we hoped to develop a more universal method, a method that works on some datasets that have higher homophily would still be useful.

Second, in future work we plan to test models that are more appropriate for heterophilic data than standard GCNs, such as Distance Encoding [13] and Geometric GCN [21]. These might be more effective with the current global graphs.

[4]Calculated using NetworkX's `attribute_assortativity_coefficient()` with labels as the attribute

6 CONCLUSION

Overall, our project aimed to find critical components for establishing a new framework for multimodal data classification, especially for misinformation detection. We approached this problem using two strategies: classifying individual graphs using HAN, a method designed for heterogeneous graphs, and building a global graph to encourage productive sharing of information between related data points. In tests so far, HAN did not outperform baselines. We have plans to perform further testing using additional meta-paths. Likewise, the current versions of a global graph classifier only hurt performance compared to baselines. This is likely due to a lack of homophily with respect to the labels. We also plan to examine ways to deal with this in future work.

REFERENCES

[1] Alessandro Bondielli and Francesco Marcelloni. 2019. A survey on fake news and rumour detection techniques. *Information Sciences* 497 (2019), 38–55.

[2] Juan Cao, Junbo Guo, Xirong Li, Zhiwei Jin, Han Guo, and Jintao Li. 2018. Automatic Rumor Detection on Microblogs: A Survey. *CoRR* abs/1807.03505 (2018). arXiv:1807.03505 http://arxiv.org/abs/1807.03505

[3] Mingxi Cheng, Shahin Nazarian, and Paul Bogdan. 2020. VRoC: Variational Autoencoder-aided Multi-task Rumor Classifier Based on Text. In *Proceedings of The Web Conference 2020*. 2892–2898.

[4] Matteo Cinelli, Walter Quattrociocchi, Alessandro Galeazzi, Carlo Michele Valensise, Emanuele Brugnoli, Ana Lucia Schmidt, Paola Zola, Fabiana Zollo, and Antonio Scala. 2020. The covid-19 social media infodemic. *arXiv preprint arXiv:2003.05004* (2020).

[5] Salvatore Citraro and Giulio Rossetti. 2020. Identifying and exploiting homogeneous communities in labeled networks. *Applied Network Science* 5, 1 (2020), 1–20.

[6] Dmlc. [n.d.]. dmlc/dgl. https://github.com/dmlc/dgl/tree/master/examples/pytorch/han

[7] 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. arXiv:arXiv:1803.07640

[8] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.

[9] 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* (2020).

[10] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. 2020. Combining Label Propagation and Simple Models Out-performs Graph Neural Networks. *arXiv preprint arXiv:2010.13993* (2020).

[11] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[12] Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2018. All-in-one: Multi-task learning for rumour verification. *arXiv preprint arXiv:1806.03713* (2018).

[13] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. 2020. Distance Encoding: Design Provably More Powerful Neural Networks for Graph Representation Learning. *Advances in Neural Information Processing Systems* 33 (2020).

[14] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2017. Gated Graph Sequence Neural Networks. arXiv:1511.05493 [cs.LG]

[15] Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv* abs/1907.11692 (2019).

[16] Ilya Loshchilov and Frank Hutter. 2017. Decoupled Weight Decay Regularization. arXiv:1711.05101 [cs.LG]

[17] Yi-Ju Lu and Cheng-Te Li. 2020. GCAN: Graph-aware Co-Attention Networks for Explainable Fake News Detection on Social Media. *arXiv preprint arXiv:2004.11648* (2020).

[18] Priyanka Meel and Dinesh Kumar Vishwakarma. 2020. Fake news, rumor, information pollution in social media and web: A contemporary survey of state-of-the-arts, challenges and opportunities. *Expert Systems with Applications* 153 (2020), 112986. https://doi.org/10.1016/j.eswa.2019.112986

[19] Valeryia Mosinzova, Benjamin Fabian, Tatiana Ermakova, and Annika Baumann. 2019. Fake News, Conspiracies and Myth Debunking in Social Media-A Literature Survey Across Disciplines. *Conspiracies and Myth Debunking in Social Media-A Literature Survey Across Disciplines (February 3, 2019)* (2019).

[20] Martin Müller, Marcel Salathé, and Per E Kummervold. 2020. COVID-Twitter-BERT: A Natural Language Processing Model to Analyse COVID-19 Content on

Twitter. *arXiv preprint* arXiv:2005.07503 (2020).

[21] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287* (2020).

[22] Kellin Pelrine, Jacob Danovitch, Albert Orozco Camacho, and Reihaneh Rabbany. 2020. ComplexDataLab at W-NUT 2020 Task 2: Detecting Informative COVID-19 Tweets by Attending over Linked Documents. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*. 434–439.

[23] Kellin Pelrine, Jacob Danovitch, and Reihaneh Rabbany. 2021. The Surprising Performance of Simple Baselines for Misinformation Detection. *Under Review, Web Conference 2021* (2021).

[24] Cristina M Pulido, Beatriz Villarejo-Carballido, Gisela Redondo-Sama, and Aitor Gómez. 2020. COVID-19 infodemic: More retweets for science-based information on coronavirus than for false information. *International Sociology* (2020), 0268580920914755.

[25] Rahul Ragesh, Sundararajan Sellamanickam, Arun Iyer, Ram Bairi, and Vijay Lingam. 2020. HeteGCN: Heterogeneous Graph Convolutional Networks for Text Classification. *arXiv preprint arXiv:2008.12842* (2020).

[26] Kenneth Rapoza. 2017. Can 'fake news' impact the stock market? *by Forbes* (2017).

[27] Kai Shu, Limeng Cui, Suhang Wang, Dongwon Lee, and Huan Liu. 2019. defend: Explainable fake news detection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 395–405.

[28] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake News Detection on Social Media: A Data Mining Perspective. *SIGKDD Explorations* 19, 1 (2017), 22–36. https://doi.org/10.1145/3137597.3137600

[29] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. In *In VLDB' 11*.

[30] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-Read Students Learn Better: On the Importance of Pre-training Compact Models. *arXiv preprint arXiv:1908.08962v2* (2019).

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[32] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[33] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *The World Wide Web Conference*. 2022–2032.

[34] Youze Wang, Shengsheng Qian, Jun Hu, Quan Fang, and Changsheng Xu. 2020. Fake News Detection via Knowledge-driven Multimodal Graph Convolutional Networks. In *Proceedings of the 2020 International Conference on Multimedia Retrieval*. 540–547.

[35] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv* abs/1910.03771 (2019).

[36] Lianwei Wu and Yuan Rao. 2020. Adaptive Interaction Fusion Networks for Fake News Detection. *arXiv preprint arXiv:2004.10009* (2020).

[37] Lianwei Wu, Yuan Rao, Yongqiang Zhao, Hao Liang, and Ambreen Nazir. 2020. DTCA: Decision Tree-based Co-Attention Networks for Explainable Claim Verification. *arXiv preprint arXiv:2004.13455* (2020).

[38] Zhiyuan Wu, Dechang Pi, Junfu Chen, Meng Xie, and Jianjun Cao. 2020. Rumor Detection Based On Propagation Graph Neural Network With Attention Mechanism. *Expert Systems with Applications* (2020), 113595.

[39] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 793–803.

[40] Xinyi Zhou, Reza Zafarani, Kai Shu, and Huan Liu. 2019. Fake news: Fundamental theories, detection strategies and challenges. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 836–837.

[41] Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PloS one* 11, 3 (2016), e0150989.