Universiteit Leiden

Master Computer Science

Content-agnostic cascade-wise ranking of
Fake News on Online Social Media

Name:	Guus Toussaint
Student ID:	s1805819
Date:	27/09/2022
Specialisation:	Computer Science: Data Science
1st supervisor:	Dr. J.N. van Rijn
2nd supervisor:	Prof.dr. H.H. Hoos
3rd supervisor:	Dr. F.W. Takes
4th supervisor:	Dr. J.P. Burger

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Fake news, i.e., misinformation spread intentionally or unintentionally, has a growing impact on our society. While fake news content is rising, fact-checkers are understaffed and overloaded. Thus a practical system for the efficient identification of check-worthy content is desired. Previous attempts to combat fake news on online social media with automated systems have presented high accuracy scores but are rarely deployed in a real-world setting. We identify three potential causes that make real-world adoption infeasible with the current methods: (1) mostly using *textual features*, (2) the placement of potentially faulty *labels*, and (3) the use of *all tweets concerning the same topic* as a single data point. An automated system for combating fake news on online social media with the aim of practical applicability must be devoid of these three causes. Therefore, we introduce a *content-agnostic cascade-wise fake news ranking system*. The system is trained in a classification setting using the AUC metric on the FakeNewsNet benchmark dataset and evaluated in a ranking setting using the top-k precision metric on the MuMiN dataset. The best performing model, i.e., a random forest with hand-crafted features, achieves an AUC of 79%, which is higher than the current state-of-the-art, i.e., a graph neural network based approach, which achieves an AUC of 73%. The random forest model achieves a top-50 precision of 55% on the evaluation dataset. The random ordering baseline, which is the current practice, achieves a top-50 precision of 40%. We show that, even though the results on a real-world task pale in comparison to the results on the content-agnostic variant of the FakeNewsNet dataset, a content-agnostic cascade-wise fake news ranking system does improve upon the current practice in terms of top-k precision and therefore is a valuable addition to a fact-checkers toolbox.

Contents

1 Introduction

As long as humans have lived in groups, *fake news*, i.e., the spread of false information, has impacted society [11]. With the rise of online social media, the general public started to get their information from less reputable sources, and as a result, the amount and impact of fake news was on the rise. In recent years the term fake news has become somewhat politically charged and can refer to many forms of misinformation. In this work, the term *fake news* refers to misinformation that is spread either intentionally or unintentionally.

Fake news content is often surprising and shocking; Henkel et al. [30] argue that people are more likely to engage in a compelling narrative than to interact with a factually backed story. The main objective of many social media platforms is to increase user engagement. To this end, they have developed sophisticated personalization algorithms that create echo chambers, i.e., an environment where a person only encounters beliefs that coincide with their own ideas, and, as a side-effect, allow disinformation to thrive [6].

Throughout the last century, countless stories have been told where fake news has directly impacted real-life events. For example, Iosifidis et al. [34] describe how the rise of fake news on online social media has profoundly impacted the political landscape in many countries. Additionally, Petratos [57] describes the risks of fake news for businesses. These are just a few examples of the impact fake news can have on our society as a whole. Recently, with COVID-19 and the invasion of Russia in Ukraine, we once again see the importance of identifying fake news quickly and accurately [13, 28]. The impact of fake news on our lives, in combination with the spread accelerated by online social media platforms, make fake news, in our opinion, one of the most significant challenges we face today.

To combat this problem, there has been an influx of fact-checking organisations. However, these organisations are overloaded with work [52]. Since fact-checking in itself is not a lucrative business, the funding of these fact checking organisations is largely dependent on donations. The limited funding and inherently time-consuming task of identifying and fact-checking potential fake news calls for a more automated approach.

Computer science researchers have tried to fill this gap by developing automated fake news detection systems. While impressive performance statistics are often presented, these systems have rarely been usefully employed in a real-world fake news detection setting. We identify three possible explanations for this apparent gap between academic results and practical application; (1) the heavy use of *textual features*, (2) the placement of *labels*, and (3) the use of *all tweets concerning the same topic* as a single data point.

First, the use of textual features in the context of fake news comes with two drawbacks. Textual features make the models inherently dependent on the language in the training data. With fake news on online social media, new topics emerge constantly and are not confounded to a single set of languages. At the same time, training data is scarce and mainly focused on the English language and political content. By restricting the use of textual features, this dependency is alleviated, and a more robust and general approach to fake news detection can be taken. Furthermore, textual features are at potential risk for adversarial attacks, as shown by Zhou et al. [77]. Propagation features, which arise from a group of actors' natural behavior, are less vulnerable to adversarial attacks since it would require sophisticated collaboration between actors to introduce an adversarial attack.

Second, a fake news detection system devoid of textual features still has its flaws. Namely, creating a fault-proof fake news detection system is unrealistic, and placing faulty labels on items can lead to distrust of fact-checking in general. Moreover, supplying fact-checkers with the raw probability scores of an item being fake might result in rubber stamping, i.e., blindly reusing the label presented by the system without giving it a second thought. However, creating a system that can rank social media posts on the likelihood of the contents being fake can significantly increase the efficiency of fact-checkers while keeping the negative influence of the system on the fact-checker to a minimum. Therefore, we

believe that a more practical and realistic approach is a hybrid one, where the artificial intelligence (AI) system is a tool in the toolbox of fact-checkers.

Third, many previous attempts to combat fake news on online social media via automatic detection use a so-called URL-wise classification approach. In this approach, all tweets concerning the same topic are grouped as a single data point. While such an approach might result in a higher performance metric, we firmly believe this limits the models' practical usability because, at the time of writing, it is infeasible to collect all tweets about a single topic reliably. We note that it would be of great benefit to fact-checkers if there were a method of collecting all tweets on the same topic; however, this is beyond the scope of this work. In this work, we will use the cascade-wise approach; with this approach, a single tweet and its corresponding retweets are treated as a single data point.

Thus, by addressing the three previously mentioned shortcomings in previous approaches, we aim to combat the existing gap in academic performance metrics and practical application of automated fake news detection systems; we present a content-agnostic cascade-wise fake news ranking system. To achieve a ranking system for fake news, we approach this problem from a binary classification point of view during training and evaluate it from a ranking point of view. During training, the performance is measured in terms of AUC, and during evaluation, the performance is measured in terms of top-k precision. In Section 2.5 a more detailed explanation and reasoning of these metrics is presented. In short, our approach can be captured in the following research question:

How does a content-agnostic cascade-wise fake news ranking system perform on a real-world task in terms of top-k precision?

To answer our main research question, we have devised the following three subquestions:

1. How does the current state-of-the-art, on cascade-wise classification of fake news on online social media using propagation structures, perform on the content-agnostic variant of the FakeNewsNet dataset in terms of Area under the ROC curve (AUC)?
2. How does a traditional model with hand crafted features perform on the content-agnostic variant of the FakeNewsNet dataset in terms of AUC?
3. How can we improve upon the state-of-the-art method, by leveraging knowledge from the hand-crafted features method?

By answering these subquestions, we can identify the best-performing fake news cascade-wise detection method on a content-agnostic variant of the FakeNewsNet dataset. Furthermore, we evaluate a more traditional method, which has the benefit of being more explainable with respect to the state-of-the-art, which can provide fact-checkers with relevant information on *why* an item was marked as potentially *fake news*. After having identified the best-performing model on a common benchmark, i.e., FakeNewsNet, we can proceed to evaluate it in a real-world setting. By evaluating the best-performing models in a real-world setting, we answer our main research question and subsequently gain insight into how such a system would contribute to the problem of fake news on online social media.

To create a content-agnostic cascade-wise ranking system for fake news detection on online social media, we have developed code and insights for practitioners and future researchers. These can be summed up in the main contributions of this work, which are as follows:

- A pipeline that can convert any tweet, given its tweet ID, to either a spreading graph used in graph neural networks or a fixed-sized feature vector used in traditional machine learning approaches. This pipeline enables future researchers to easily construct a dataset from a list of labeled tweets. Furthermore, it enables an incoming stream of tweets to be converted into a format suitable for a machine learning system, thus accelerating practitioners' adoption of fake new detection or ranking techniques.

- Insights into the performance of the state-of-the-art cascade-wise classification on the publicly available content-agnostic variant of the FakeNewsNet dataset. This highlights the fragile nature of performance metrics once the conditions change.
- Insights into the performance of a traditional machine learning approach with hand-crafted features on the publicly available content-agnostic variant of the FakeNewsNet dataset, which shows that, in a noisy environment, a more straightforward approach can outperform the state-of-the-art.
- Insights into the performance deficit when evaluating a fake news ranking system on a real-world task compared to evaluating on a benchmark dataset. This highlights the main problem that should be addressed when creating a system to be used in the real world.
- A content-agnostic cascade-wise fake news ranking system which improves upon the current practice. Thus, when used in practice, improving the efficiency of fact-checkers when fact-checking a list of posts on online social media.

This thesis is structured as follows. Section 2 provides the fundamentals used thoughout this work. More specifically, we give an overview of the workflow of a fact-checker, provide an overview of the field of fake news detection on social media, describe the inner workings of graph neural networks, describe the automated machine learning techniques used in this work, and describe and motivate the metrics used to evaluate our models. Section 3 will cover the training and evaluation dataset used in this work. In Section 4, we will describe the creation process of the feature vectors and graphs used in this work alongside the pipeline, which can be used to generate a data format suitable for machine learning systems. Then, in Section 5 we outline the methodology and experiments used to answer the subquestions and the main research question presented in this work. In this section, we also present the results of our experiments. In Section 6 we discuss the results in the context of our proposed research question and highlight the limitations of our work. Finally, in Section 7 we conclude our work and provide possible directions for future work.

2 Background

This section provides an overview of the background related to this work. First, we will describe the workflow and tools used by fact-checkers and highlight some of the complications of this process. Second, we provide an overview of previous approaches to fake news detection and ranking on online social media and describe how our work positions itself among the existing literature. Third, we explain the inner workings of the graph neural network approach used in this work. Fourth, we explain the automated machine learning techniques used to optimize our models. Finally, we describe the two metrics used to evaluate our models.

2.1 Fact Checking

A fact-checker is tasked with determining the veracity of a piece of content. By identifying that a piece of content is *fake news*, the negative impact can be significantly reduced; moreover, it can lead people to revise and update their existing beliefs, as stated by Carnahan et al. [12]. The fact-checking organizations often work independently from one another, resulting in a plethora of fact-checking methodologies. However, every fact-checking process has to go through the following three stages, as stated by Mantzarlis [48]: (1) Selecting which claims to fact-check. (2) Fact-checking these claims. (3) Communicating the veracity and convincing people of said veracity. To properly understand the process of a fact-check, the possible complications, and our solution to some of the complications, we will describe each step in detail.

The selection of which claims to fact-check can be made in two different ways. One approach is to review content flagged by users. This approach is only applicable when the fact-checkers have this information available. Another approach is for fact-checkers to go on social media platforms and look for misinformation themselves. Both approaches suffer from biases. For the first approach, i.e., reviewing flagged content, one can imagine the following scenario. A piece of content that clearly contains a false claim for an *outsider* is passed around in a single community. Within that community, no one argues its content, and thus the item is never flagged, never reported to fact-checkers, and never fact-checked. The community exposed to this message is unaware of the falsehoods they have encountered and perceives its content as facts. This process is related to the concept of an *echo chamber*, where a community is not exposed to diverse perspectives, and subsequently reinforces their own narrative, regardless of the veracity of that narrative as described by Cinelli et al. [14]. The second approach for identifying which claims to check suffers from the same bias since it is unlikely that a fact-checker is engaged in said community. Moreover, a fact-checker has limited time, and scrolling through online media to find potentially fake news is in itself a time-consuming task, let alone fact-checking the found content.

The second step of a fact-checkers workflow, i.e., determining the veracity of a piece of content, is the core of a fact-checking organization. When determining the veracity of a piece of content, numerous tools are used by fact-checkers. For example, to identify previous appearances of images, a Google Reverse Image search is used, or to verify a factual claim, a simple online search is used. Recently more advanced tools have entered the toolbox of a fact-checker, ranging from deep fake detection tools to social network analysis tools. The combined aim of these tools is to reduce the time of a single fact-check. Though the process of a fact-check can be a long and winding road, we believe a fully automated system does not offer the solution. A human must always operate a fact-checking system.

The final step in a fact-checkers workflow consists of communicating the results. Having fact-checked a piece of content without disseminating the true veracity of that piece of content does not change the views of the people affected by the false claim. Therefore, it is crucial to disseminate the result of a fact-check to the people that read and supported the original fake news content. This part is often the hardest; people do not like to be told that they have been exposed to a false claim as described by Hannak et al. [26]. If this message is not delivered with careful discretion, people might dig their heels in and discard the fact-check as fake news itself as described by Brandtzaeg et al. [8]. Shin et al. [63]

show that fake news content is often emotionally or politically charged, creating a 'us versus them' mentality for everyone involved. Communicating the 'truth' is a delicate process and by no means trivial. Because of this, we do not believe in a *stand alone* fake news detection system and argue that any fake news detection system should have a human in the loop.

In this work, we explore the possibility of assisting fact-checkers in step one of the fact-checking workflow, i.e., identifying potentially false content, by designing a system that can identify potentially fake content. We believe that with this approach, identifying tweets requiring fact-checking can be more efficient and less prone to biases.

2.2 Fake news detection

Due to the growing impact of fake news on our society, in combination with the limited time of fact-checkers, there has been an influx of research regarding automatic fake news detection. These works can be divided into three categories: Content-based, context-based, and propagation-based, as shown by Shu et al. [66]. The categories are based on the features used when determining the veracity of a piece of content.

We note that *most* previous approaches have tackled the problem of fake news detection from the standpoint of an autonomous system. We aim to create a tool that fact-checkers can use in step one of the fact-checking workflow. Therefore, we create a ranking system. However, we approach the problem from a classification point of view during training. Subsequently, we can learn from all previous approaches to fake news detection.

The content-based approaches use techniques from the Natural Language Processing (NLP) domain in computer science. Content-based approaches for fake news detection aim to distinguish true from fake via writing style, keywords, or other lexical cues by using techniques that can convert a piece of text into trainable features. A substantial amount of research has shown that there often are lexical differences between true and false statements [1, 61, 59, 58, 56]. For example, by analyzing the ratio of hand-crafted lexicon markers, Raskin et al. [59] show that first-person and second-person pronouns are used more often in fake news messages. However, these approaches are language and topic dependent. Thus when no training set is available for a specific language, a model cannot be properly trained. Furthermore, when a new topic arises, which was the case with the COVID-19 pandemic, these models require retraining, which in turn requires a set of labeled data.

Many researchers incorporate social context features in their models to increase performance when distinguishing true from fake news. These approaches fall into the category of context-based approaches. Shu et al. [67] show that the user context, i.e., followers, account activity, relations among users and publishers, can improve the performance of a fake news detection system. They created a system that incorporated these social context features and showed significant improvement when compared to a system that did not have access to the social context features. Other approaches have extended this idea to incorporate *reactions* and *likes* in the feature set for detecting fake news [62, 70], and again, found significant improvements when compared to methods that did not make use of these features.

The research conducted by Vosoughi et al. [73], Shu et al. [65], and Bodahi et al. [3] identifies the diffusion patterns of fake news on online social media as a promising direction. Showing that fake news content on online social media spreads akin to a virus [40]. Moreover, the spreading patterns appear different for fake news than for true news. Since then, many new approaches have tried to incorporate the spreading of an item in the detection pipeline [33, 68, 75, 50], these approaches fall into the propagation-based approaches of fake news detection. Monti et al. [50] use geometric deep learning on the diffusing graphs and achieve state-of-the-art performance for cascade-wise classification of fake news on online social media.

This work will use the context and propagation patterns for our fake news ranking system. A text agnostic system was chosen due to the language and topic dependencies that arise when incorporating

textual features. Furthermore, we believe context and propagation features capture sufficient predictive power for a fake news detection system.

The previous approaches described above achieve impressive performance statistics, sometimes upwards of 95% accuracy. To place these performance numbers into context, we must describe the data differences between previous approaches and ours. Almost all of the mentioned approaches use URL-wise classification. In such a setting, all tweets related to the same story are bundled together as a single data point. We believe this gives an algorithm an unfair advantage since it is relatively easy to bundle all tweets from a dataset, while it is a hard and error prone task to do the same for *new* tweets. Therefore, in this work, we will use the cascade-wise classification approach. With this approach a single tweet serves as a single data point. We note that by doing so, our algorithm will have a much more challenging task since a single cascade-wise data point contains very little information compared to an URL-wise data point, which contains several cascade-wise data points. We argue that impressive performance metrics have little impact when practical applicability is non-existent. To the best of our knowledge, only two previous approaches have tackled the problem of fake news detection from a cascade-wise point of view, i.e., Monti et al. [50], and Ma et al. [46]. Both approaches use different datasets, and subsequently, their approaches can not be compared on a one-to-one basis. In this work, we will focus on the method introduced by Monti et al. [50] since it showed superior performance and leave the one-to-one comparison to the approach presented by Ma for future work.

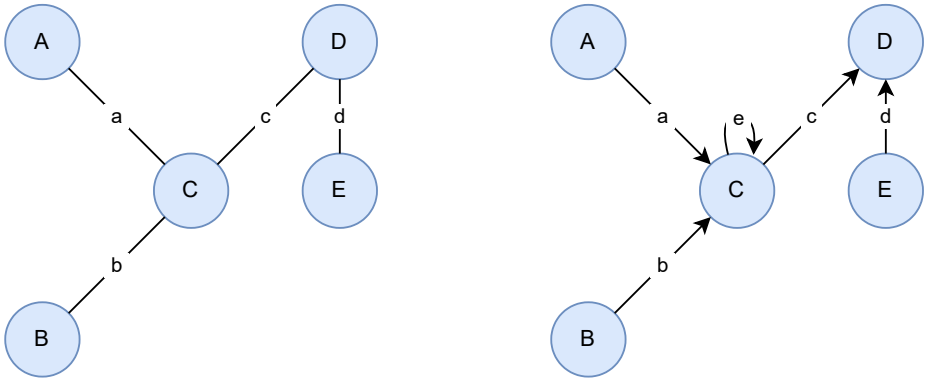
The methods mentioned above tackle the problem from a fake news *detection* standpoint. The notion of keeping a human in the loop and constructing a fake news *ranking* system has also gained recent attention, as described by Leblay et al. [41]. Tools such as ClaimRank [36], ContentCheck [47], FactRank [2], and Chequeabot [25] provide a 'claim identification' step in their fact-checking pipeline. However, all methods rely heavily on textual features and, thus, are bound to a single language. While FactRank achieves impressive results with an accuracy of $74,8\%$ when identifying check-worthy claims, the system only looks at textual content. This comes with the downside that tweets using images or videos as a way of propagating false claims cannot be identified, and the system can only operate on the Dutch language since the training data consists of Dutch sentences. With our approach, we aim to alleviate these constraints and create a more generally applicable fake news ranking system. We note that these systems do not have to *compete* with each other when applied in a real-world setting. A combination of tools, e.g., combining the output of FactRank with the output of our ranking approach, could provide a more usable system than the two tools would provide individually.

2.3 Graph Neural Networks

Graph neural networks (GNNs) are a subset of the geometric deep learning domain described by Bronstein et al. [9]. The geometric deep learning framework attempts the geometric unification of a broad class of machine learning problems. Graph neural networks aim at using the *symmetry prior* of a graph, i.e., the additional structure captured in the input signal. In this work, we will provide a GNN with a graph representation of a tweet, and the GNN calculates the probability of the tweet being fake news. The graph representation of a tweet contains information about the user, which is embedded in the nodes as node features, and information about the relationship between users, which is embedded in the edges as edge features; the exact representation is described in detail in Section 4.1 First, the relation to convolutional neural networks is described in Section 2.3.1. Second the method used in this work is described in Section 2.3.2.

2.3.1 Relation to convolutional neural networks

If we look at traditional machine learning approaches for images, the symmetry prior is the euclidean geometry of the input image. With the symmetry prior obtained from a euclidean geometry, we can leverage the knowledge that neighboring pixels are related to each other. This knowledge allows a convolutional neural network to apply convolutions, as described by LeCun et al. [42], to the input



(a) Simple undirected graph

(b) Simple directed graph

Figure 1: Example of a simple graph $G = (V, E)$ with nodes $V = \{A, B, C, D, E\}$ and edges $E = \{a = (A, C), b = (B, C), c = (C, D), d = (D, E), e = (C, C)\}$. Figure a shows the undirected variant, Figure b shows the directed variant with the addition of a self loop for node C

image and gradually shift these convolutions, along the euclidian geometry, across the whole image. If we were to apply such a geometric convolution to a graph input, the structure of the graph would be interpreted incorrectly. The following example illustrates why a geometric convolution cannot be used when using graphs. A graph is a set of nodes V connected by a set of edges E, so every graph G can be described as follows: $G = (V, E)$. The nodes and edges of a graph have no canonical ordering, unlike the pixels of an image. Thus, two graphs ordered differently, but containing the same nodes and edges, are still considered the same graph. However, if a traditional convolution were to be applied, the two graphs would produce different outcomes. To overcome this issue, a learning algorithm on graphs should be permutation invariant while still being able to capture the symmetry prior. These problems are addressed by local aggregation of neighboring nodes, i.e., updating a node's feature vector based on the neighboring nodes, where neighboring refers to two nodes sharing an edge. In essence, the euclidian geometry is replaced with the geometry of the graph. It is important to note that the euclidean geometry of an image can be encoded into a graph geometry creating a grid structure this is done by treating every pixel as a node and connecting every pixel with its neighboring pixels. Hence, the geometric deep learning framework is a unification of machine learning systems, not an extension. Graph neural networks have proven themselves to not only be a great theoretical framework but also a practical one with many applications such as antibacterial discovery [69], recommendation systems [17], traffic prediction [37], etc.

The local aggregation function for graph neural networks takes the following form:

$$f(x_i) = \phi(x_i, \sum_{j \in \mathcal{N}_i} \psi(x_j)) \tag{1}$$

Here ϕ and ψ denote learnable functions, e.g., a multi layer perceptron. The node features for node $i \in \mathcal{V}$ are captured in the vector x_i. The set of all nodes connected to i is denoted as $\mathcal{N}_i$. In this function the permutation-invariant aggregator is the sum, i.e., $\sum$, but can be replaced with any permutation-invariant aggregation operator. Finally, $f(x_i)$ represents the new features for node i.

To illustrate the workings of the local aggregation function, we provide an example. Figure 1(a) shows graph $G = (V, E)$ and its corresponding nodes and edges. For this example, we will use simple learnable functions for easy calculation. These functions are more complex in practice, but the approach is equivalent. In this example our learnable function ϕ equals $\phi(x_1, x_2) = \frac{x_1}{x_2}$, and our learnable function ψ equals $\psi(x) = \frac{x}{2}$. The feature values for our nodes are as follows: $A = 4, B = 6, C = 5, D = 8, E = 7$.

Now we want to calculate the new feature value of node D, i.e., we want to calculate $f(x_D)$. $\mathcal{N}_\mathcal{D}$ consists of all neighbors of D, which, from Figure 1(a), we know to be C and E. Thus $\sum_{j \in \mathcal{N}_D} \psi(x_j) = \psi(x_C) + \psi(x_E) = \psi(5) + \psi(7) = \frac{5}{2} + \frac{7}{2} = 6$. Now we can calculate $f(x_D) = \phi(x_D, 6) = \frac{8}{6} = \frac{4}{3}$.

After applying this local aggregation function to all nodes in the graph, we obtain a new set of node features. However, we still have a feature vector for every node in the graph. Thus the size of the resulting matrix, which contains all newly calculated node features for all nodes, depends on the total number of nodes in the graph. When the goal is a classification of a whole graph, the set of newly obtained node feature vectors needs to be converted into a single representation of all nodes present in the graph. This can be achieved by applying a global pooling of node feature vectors to the graph. The most commonly used technique is called *global mean pooling*.

$$g = \frac{\sum_{i \in \mathcal{N}} x_i}{|\mathcal{N}|} \tag{2}$$

Where $\mathcal{N}$ is the set of all nodes, and x_i is the feature representation of node i. Finally, g holds the mean of the features from all nodes.

2.3.2 Graph Attention Networks

Graph attention networks are used in the current state-of-the-art on cascade wise detection of fake news on online social media and provide a framework for implicitly specifying different weights to different nodes in their neighborhood, as presented by Veličković et al. [72]. This is important since it allows the network to apply different importance to the type and direction of the relation between users. They present an alteration to the local aggregation function presented in Equation 1.

$$x'_i = \phi(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} x_j) \tag{3}$$

Note that x'_i is different from x_i, x'_i indicates that the newly obtained feature vector has a potentially different cardinality. In other words, with the graph attention mechanism, it is possible to create a feature vector of a higher order than the original feature vector. Also, note that the learnable function ϕ now only takes in one argument. In Equation 1 ϕ took as arguments the aggregated node features for its neighbors aswell as its own node features. With graph attention networks, ϕ only considers the neighboring node features. To still use the original node features, a self-loop is introduced. Thus, each node is essentially becoming its own neighbor. This is illustrated in Figure 1(b), where node C denotes the node for which the graph attention mechanism is executed. We note a new edge (e) which is the self-loop for node C. $\mathbf{W}$ denotes a matrix $\mathbb{R}^{\mathrm{F}' \times \mathrm{F}}$, where F and F$'$ are the cardinality of the original and the newly created node vectors respectively. $\mathbf{W}$ is the weight matrix for the linear transformation from the original node features to the new, higher-order node features. The function α_{ij} represents the learnable attention vector for node x_i and x_j. This vector can be interpreted as the importance of node x_j with respect to node x_i and is defined as follows:

$$\alpha_{ij} = \mathrm{softmax}_j(\mathrm{LeakyReLU}(\mathbf{a}^T[\mathbf{W}x_i || \mathbf{W}x_j])) \tag{4}$$

Where $\mathbf{a}$ is the weight vector of a single layer feed-forward neural network, and takes the shape $\mathbf{a} \in \mathbb{R}^{F'+F'}$. The $||$ operator represents concatenation, thus the resulting feature vectors for x'_i and x'_j are concatenated and multiplied with the transposed weight vector $\mathbf{a}$. A LeakyReLU non-linearity is applied to the resulting vector. Finally all coefficients are normalised across all choices of j via the softmax function, this is done in order to make the coefficients comparable across different nodes.

A graph can also incorporate the information in its edges; an edge feature vector describes the relation's characteristics, similar to how a node feature vector describes the characteristics of a node. In a social

network context, an edge feature vector can describe how two users are related to each other, e.g., does A follow B, does B follow A, does A have previous interactions with B, etc.

We note that in this formulation, the edge features are completely ignored. Wang et al. [74] propose an addition to the original Graph Attention Networks, where edge features are also included in the attention mechanism. The proposed addition simply concatenates not only the transposed node features of node x_i and x_j, but also the edge features of e_{ij}. The weight vector $\mathbf{a}$ now takes the following shape: $\mathbf{a} \in \mathbb{R}^{F'+F'+E}$, where E is the cardinality of the edge features. The resulting equation for α_{ij} can now be described as follows:

$$\alpha_{ij} = \text{softmax}_j(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}x_i||\mathbf{W}x_j||e_{ij}])) \tag{5}$$

To recapitulate, graph attention networks solve the problem of a non-euclidean data structure of varying size by executing a learnable local aggregation function, based on the node and edge features, for all nodes. Finally, the mean of all newly calculated node features is taken as the resulting graph representation structure.

2.4 Automated Machine Learning

The area of automated machine learning focuses on automating the process of algorithm selection and hyper-parameter optimization (HPO). Work by Henderson et al. [29] shows that the values for hyper-parameters can significantly affect the performance of a model. Furthermore, Hanussek et al. [27] show that an automated machine learning system can often outperform a system designed by a human. With the recent developments in machine learning, many new algorithms and approaches have come to light. Finding the approach best suited for the task at hand is by no means a trivial task. Bearing in mind the *no free lunch theorem*, that states that no specific algorithm beats all other algorithms on all tasks, in other words, when searching for an optimal solution for a new task, it is essential to take multiple different algorithms into account. The field of automated machine learning provides solutions to the problem of finding an optimal algorithm and set of hyper-parameters for the task at hand. The objective of any HPO method is to find a solution for the following equation, which is a simplified version of the equation presented by Thornton et al. [71]:

$$\lambda^* \in \underset{\lambda \in \Lambda}{\text{argmin}}\, \mathcal{L}(f_\lambda, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{valid}}) \tag{6}$$

Where f_λ refers to the black box method, initialized with the single set of hyper-parameters λ. The space of all possible hyper-parameter configurations, also known as the search space is denoted as Λ. The loss function, which calculates the *fit* of the model based on the train dataset ($\mathcal{D}_{\text{train}}$) and the validation dataset ($\mathcal{D}_{\text{valid}}$), is denoted as $\mathcal{L}$. Finding the optimum configuration requires a model to be trained for multiple different arrangements of hyper-parameters.

Multiple approaches can be taken to assess whether an arrangement of hyperparameters is a suitable choice for the task at hand. In cases where the training of the full model is expensive, a multi-fidelity approach to HPO is desired. The core idea of a multi-fidelity approach is to find the optimum configuration λ within a hyper-parameter space Λ while keeping the number of executions of the *full* model to a minimum. Promising configurations must be identified on a small budget when trying to keep the executions on the full model to a minimum. This can be achieved in several ways; Mohr et al. [49] show that modeling the learning curve on a small budget can be a good indication of the model's performance on the full model, and Falkner et al. [18] introduce a method called *bayesian optimized hyperband* (BOHB).

Both of the approaches used in this work use BOHB; therefore, in Section 2.4.1, we will introduce the BOHB algorithm. In Section 2.4.2 we describe the HPO method to optimize our graph neural networks, and in Section 2.4.3 we describe the automated machine learning system AutoSklearn 2.0 which is used to optimize our hand-crafted features methods.

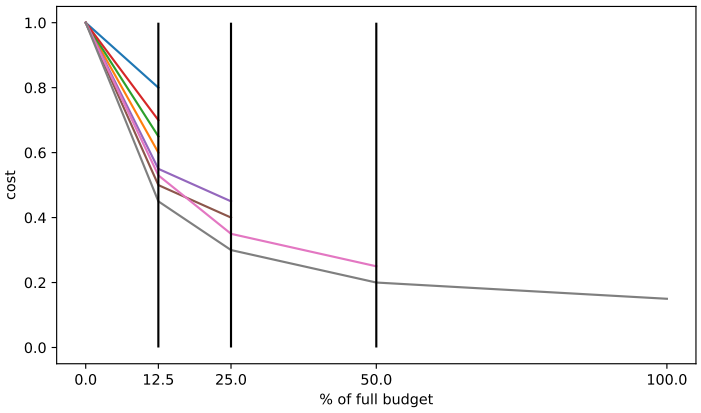


Figure 2: Example of a successive halving run of 8 different hyper-parameter configurations. We observe that all 8 configurations are evaluated on 12.5% of the full budget, then only the best performing half are selected to be evaluated on twice the original budget. This process repeats until the full budget is reached.

2.4.1 BOHB

To explain BOHB an explanation of successive halving (SH) and hyperband (HB) is required.

Successive halving, presented by Jamieson et al. [35], aims to give the most of the available budget to promising configurations and subsequently avoid wasting valuable resources on poorly performing configurations. Figure 2 shows an example run of successive halving with 8 configurations. We can clearly see the process of successive halving, i.e., allowing promising configurations to be evaluated on a larger budget while stopping configurations that perform poorly. This simple approach to multi-fidelity optimization suffers from a tradeoff between the number of configurations selected at the start and the number of cutoff points.

To overcome this issue, Li et al. [43] introduced hyperband. Hyperband places successive halving as a subroutine and thus performs successive halving with different budgets. This approach alleviates the tradeoff between the number of configurations and cutoff points since successive halving is run multiple times with a different number of configurations and cutoff points.

BOHB provides an addition to the hyperband algorithm presented by Li et al. [43]. Where hyperband selects the configurations to be evaluated based on a random sample of the complete search space, BOHB proposes to use a bayesian optimization approach. This is done by training a surrogate model to predict the cost of a new hyper-parameter configuration based on the previously tested configurations. In BOHB, the surrogate model is a *tree parsen estimator* (TPE). Li et al. [43] show that this approach provides significant speedups when compared to vanilla hyperband.

2.4.2 SMAC4MF

In this work, we will use the method proposed by Lindauer et al. [44] SMAC4MF which demonstrated superior performance on the HPOBench benchmark described by Eggensperger et al. [16]. In SMAC4MF, the surrogate model is a random forest that replaces the TPE of vanilla BOHB. Furthermore, Lindauer et al. [44] provide a complete python package that makes the integration with our graph neural networks possible.

2.4.3 AutoSklearn 2.0

In this work we will also use the AutoSklearn 2.0 framework introduced by Feurer et al. [20], which is a follow-up of the original AutoSklearn 1.0 framework [21]. In contrast to the previously described SMAC4MF approach, AutoSklearn is a complete automated machine learning framework. Where SMAC4MF is an algorithm where each component that needs to be optimized needs to be defined beforehand, AutoSklearn has predefined a set of feature pre-processors, algorithms, and hyper-parameters for these algorithms. Thus creating a complete pipeline for the specific task at hand. With AutoSklearn 2.0 the authors provide two improvements on the existing AutoSklearn 1.0 framework. First a new meta-learning technique, in other words a more efficient way of warmstarting the bayesian optimisation. Second, the inclusion of Successive Halving, as described earlier.

The AutoSklearn 2.0 framework works in two stages: The training and testing stages. It is important to note that the authors do all the processing in the training stage beforehand. The testing stage requires processing on our part. The *training stage* aims at creating a static *portfolio* and constructing a *policy selector*. These components are necessary for the portfolio-based successive halving (PoSH) algorithm.

First, we will discuss the need for a static portfolio. The authors found that calculating meta-features that describe the dataset and can be used to identify promising pipelines was too time-consuming; therefore, they proposed a new approach. First, they construct a portfolio consisting of high-performing and complementary machine learning pipelines, aiming to perform well on as many datasets as possible. Then, when a new dataset is presented, all pipelines in the portfolio are evaluated. When there is time left, the pipelines suggested by bayesian optimization, warmstarted with the evaluated pipelines from the portfolio, are evaluated. With this approach, there is no need to calculate any meta-features for a new dataset. The portfolio is built up in a greedy fashion, starting with pipelines that perform well on average, i.e., across a diverse set of tasks, and then moving towards more specialized pipelines.

Second, we discuss the need for selecting a model selection and budget allocation strategy. The authors found that assigning an equal budget to all pipelines does not scale well to large datasets. To combat this issue, they implemented successive halving as described in Section 2.4.1, which allows more budget to be allocated to promising pipelines. However, they found that SH can be too aggressive for small datasets and potentially cut away good pipelines that require a higher budget to perform well. Subsequently, a choice has to be made to include SH or not. They also observed different tradeoffs for the model selection strategies, i.e., using 3, 5, or 10-fold cross-validation or using holdout. So in total, the PoSH algorithm requires two hyper-parameters, i.e., the choice of model selection and the choice of using SH.

The aim of AutoSklearn 2.0 is to provide a framework that can achieve state-of-the-art performance without any low-lever decision-making from the user. To that end, a system needs to be designed that chooses the best values for the two hyper-parameters of PoSH, given a new task. This is done by constructing a *policy selector* model, which can give the best model selection and budget allocation scheme given a dataset. The policy selector is trained on a diverse set of datasets and pipelines and evaluates its performance across all choices of model selection and budget allocation schemes. By calculating the meta-features of a new dataset, the policy selector can provide the best values for the PoSH algorithm.

With the AutoSklearn 2.0 framework, a new dataset is evaluated to determine the best possible model selection and budget allocation schemes. Then that information, in combination with the predefined static portfolio, is used to execute the PoSH algorithm. Resulting in an optimal machine learning pipeline, devoid of any hyper-parameters, for the task at hand.

For a detailed overview of all algorithms and configurations in the search-space defined by AutoSklearn we refer to the work by Feurer et al. [20].

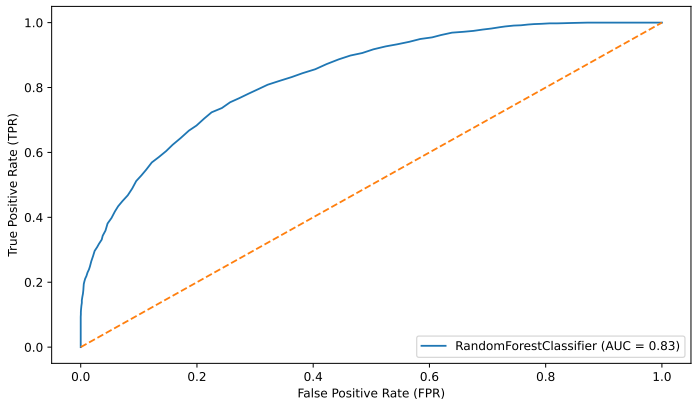


Figure 3: Example of a Area Under the Receiver Operator Characteristics Curve (AUC). The solid blue line is the ROC-curve. The dotted orange line is the base line, in other words it resembles a model which is not capable of making a distinction between the two classes. The AUC value is determined by calculating the area under the ROC-curve (blue solid line).

2.5 Metrics

To evaluate the different models presented in this work, we use two metrics; Area Under the Receiver Operator curve (AUROC) and top-k precision. Here we describe how these metrics operate and the rationale behind choosing them.

The AUROC - often simplified to AUC - measures the classification performance at several threshold settings. Where a threshold setting refers to the value used to separate true from false, in classification, 0.5 is often used as a default threshold value. AUC provides a clear metric for the degree of separability between classes, i.e., in our case, it quantifies how well a model can distinguish between true and fake news. AUC can be used as a proxy for a ranking system where the order within a class is of no importance. It does this by plotting the true positive rate (TPR) on the y-axis and the false positive rate (FPR) on the x-axis; it then calculates the area under this curve. Figure 3 illustrates an example of a Receiver Operator Curve (ROC). The AUC metric calculates the area under the solid blue line, thus giving an average of the classification performance along all threshold values. AUC is widely regarded as the industry standard for evaluating learning algorithms; furthermore, it provides a more detailed metric with respect to the accuracy metric, as described by Huang et al. [32]. The AUC metric also has its downsides - as does every single value metric - as described by Lobo et al. [45]. They mention that AUC should only be used when discrimination capacity is the main objective. Since we aim to discriminate between true and fake news, and the raw probability scores are of no importance, we feel that using the AUC metric in this work is justified.

The top-k precision metric - also known as precision@k - is used to measure the precision of a ranking algorithm. Imagine we have a collection of 200 tweets, of which 50 contain fake news. If we would order them at random and calculate the Top-10 precision, we would expect - on average - a Top-10 precision of 25%. If we had a system that could rank the tweets based on the likelihood of being fake, we would expect our Top-10 precision score to be significantly higher than the 25% baseline. It is important to note that this metric, if not presented correctly, can provide misleading results. First, having a testing set that is very unbalanced, e.g., 80% of the data points containing fake news, would lead to a baseline top-k precision of 80%. Second, it is important to have testing sets of equal size when comparing top-k precision. Having a larger 'pool' to pick from would allow a model a significant advantage compared to a model which has a smaller test set. Therefore, in our experiments, the test set is equal in size and

class balance to our evaluation set. With these restrictions, we believe a fair comparison in terms of top-k precision between the test and evaluation set can be made.

Though we evaluate our models as a classification problem using the AUC metric, we can still gain insight into their performance on a ranking system by looking at the local dominance of the ROC curve. A model that shows dominance in the first part of the ROC curve compared to another would perform better in a top-k ranking setting if the value for k were to be small. This gives rise to the following situation. There exists a ROC curve that is dominated by another ROC curve, but that has a higher top-k precision. In our situation, this could lead to the following; a model is discarded because, based on the AUC, it is outperformed by another; however, in a ranking situation discarded model might actually perform better than the model which achieved a higher AUC score. To mitigate this risk, we will evaluate all models based on the shape of the ROC curve and not the AUC value alone.

Table 1: Statistics of the FakeNewsNet dataset, with a minimum cascade-size of 7. A story can consist of multiple cascade, e.g. when multiple people tweet about the same story. A cascade consists of a single root tweet and its corresponding retweets.

Fact-checking source	# Storys	# Cascades
Gossipcop	3 623	13 072
PolitiFact	438	16 628

3 Data

In this section, we describe the data used in this research. First, in Sections 3.1 we describe the data used to train our models. In Section 3.2, we quantify our concerns about the quality of the FakeNewsNet via a small exploratory analysis. Finally, in Section 3.3, we describe the dataset used for the evaluation of a real-world task.

3.1 FakeNewsNet

In this work, we obtained data from the FakeNewsNet dataset as described by Shu et al. [64]. The FakeNewsNet dataset was designed as a means to accelerate research in the domain of fake news detection. It is publicly available and contains news content, social context, and dynamic information. The tweets/claims are labeled using two fact-checking organizations, Gossipcop and PolitiFact. Gossipcop reports on fake news relating to celebrities in the United States, and PolitiFact reports on fake news about political statements in the United States. Since our goal is to create a content-agnostic cascade-wise fake news ranking model, which can work across all languages and topics, we remove all textual information from the dataset.

The FakeNewsNet dataset is built up as follows: The authors of the FakeNewsNet dataset collected 3 623 stories from Politifact and 438 stories from Gossipcop. Tweets are linked to a story by calling the Twitter API and identifying tweets, comments, or quotes that contain a link to the fact-checked story. When comments or quotes are found, the source tweet is returned. The source tweet takes the label from the fact-checked story. This process results in a list of several cascades, i.e., tweets, where the original story is mentioned either in the tweet itself or in a quote or comment related to that tweet.

We note that this process of collecting data is inherently noisy. Tweets that mention a fact-checked story might be unrelated or disagree with the claim. Since these tweets are not cleaned in the dataset, the dataset suffers from noisy labels. To better illustrate how this process leads to noisy labels, we take a look at the following example. Imagine a story posted on Politifact where they debunk the following claim: "The Senate Democrats' reconciliation bill will strip 300 billion from Medicare.".[1] Politifact has labeled this claim as *false*, and thus, following the data collection protocol used by the FakeNewsNet dataset, all tweets mentioning the claim are labeled as textitfalse. Now imagine the following tweet: "The Senate Democrats' reconciliation bill won't strip 300 billion from Medicare as shown by a fact-check conducted by Politifact (link to fact-check)". Since this tweet mentions the original link to Politifact, the tweet has inherited the label from the fact-check, which was *false*. However, we notice that the tweet actually disagrees with the original claim and agrees with the fact-check. Therefore the label should be the inverse of the label from the fact-check, namely *true*.

This vital step of identifying the stance of the tweet in relation to the original claim was not done when constructing the FakeNewsNet dataset. This is a severe shortcoming when compared to the work presented by Monti et al. [50], where they hire human annotators to hand filter the returned tweets. However, we do not have the resources or expertise to hand filter all the tweets, and the FakeNewsNet

[1] https://www.politifact.com/factchecks/2022/aug/05/american-prosperity-alliance/no-senate-bill-wont-strip-300-billion-medicare/

dataset is widely used in fake news detection on online social media. Therefore, we keep the data as is and present the results accordingly. In Section 3.2 we provide our results of a small exploratory analysis on the noise in the FakeNewsNet dataset.

For each story, multiple tweets can be linked, resulting in a dataset that contains many individual tweets which are categorized per story. The number of stories and cascades per fact-checking source are shown in Table 1. We note that for political news, fact-checked by PolitiFact, it is far more common to have a large number of tweets about a single story in comparison to celebrity gossip, fact-checked by Gossipcop. As mentioned in Section 2 we will be using cascade-wise classification, and thus we will look at the individual tweets and not at all the tweets about a single story.

Since we use the diffusion pattern of a tweet to determine its veracity, we require a minimum amount of interaction. Following the approach of Monti et al. [50], we discard any tweet with less than 7 retweets. Though we note this limits the applicability of the method, we make a case that tweets with less than seven users have generated little impact on the public domain and thus are of little importance to fact-checkers.

3.2 FakeNewsNet data quality

To quantify the noise in the FakeNewsNet dataset, we conducted a small exploratory analysis of the FakeNewsNet dataset. The details of this analysis are shown in Appendix A. We observe a label mismatch for 24.14% of the cascades in our random sample, thus solidifying our concerns regarding the inherently noisy collection protocol used in the FakeNewsNet dataset. We note that all of the faulty labels are items labeled as *fake* while in reality, they are *true*. This is particularly problematic for our approach since this heavily affects the precision, and we are evaluating our models on top-k precision.

We note that a cascade-wise approach is particularly vulnerable to this noise. For a cascade-wise approach, the machine learning system only has that one cascade available for determining its veracity, while an URL-wise approach has all cascades concerning that topic available. Thus if the majority of the cascades in the URL-wise collection are labeled correctly, the machine learning system can still make an informed decision. This is not the case for a cascade-wise approach that has to determine the veracity on a per-tweet basis. Though we note this is a significant shortcoming in our approach, no other datasets of this size exist. Furthermore, the FakeNewsDataset is widely used in fake news detection research. Therefore we use the dataset as is but identify the noise in the dataset as a potential for worse performance metrics reported. Further exploration of the noise present in the FakeNewsNet dataset is left up to future work and is beyond the scope of this work.

3.3 Evaluation Dataset

We aim to create a content-agnostic cascade-wise fake news ranking system that would be applicable in the real world. To that end, we require an evaluation dataset that closely represents a real-world task and allows us to answer our main research question: How does a content-agnostic cascade-wise fake news ranking system perform on a real-world task in terms of top-k precision?

A real-world task consists of ranking a list of tweets in such a way that the tweets which are likely to be fake are placed on top of the list. By doing so, a fact-checker can systematically work its way through the list and first tackle the tweets whose veracity might be fake. Especially when faced with a considerable amount of tweets, the ranking can significantly improve the effectiveness of a fact-checker. To this end, our evaluation dataset should have the following characteristics: First, the dataset should be multi-lingual, allowing us to test the hypothesis that a content-agnostic fake news ranking system work across multiple languages. Second, the dataset should consist of multiple topics, allowing us to test the hypothesis that a content-agnostic fake news ranking system works across multiple potentially unseen topics. Finally, the distribution of the labels should be equal to that of our train set. As mentioned in Section 2.5 this is required for a fair comparison of top-k precision.

The MuMiN dataset described by Nielsen et al. [53] fits our first two requirements, i.e., multi-lingual and multiple topics. The class distribution of the labels is heavily skewed, with 95% of the labels being *fake* news. To overcome this imbalance, we subsampled the fake news items such that the class imbalance is equal to the FakeNewsNet, which is 61.14%. Following our approach on the FakeNewsNet dataset, we removed tweets that were retweeted less than 7 times. Resulting in 3 077 cascades, of which only 139 contain true news. All 139 true news cascades are selected alongside a random sample of 89 fake news cascades, resulting in a total of 228 cascades with a majority class of 60.69%.

Both the FakeNewsNet [64] and MuMiN [53] datasets are publicly available, thus making our results verifiable. Each cascade consists of the source tweet, the corresponding retweets, and the friends, i.e., the list of users followed by a specific user are regarded as the users *friends*, of all users involved in the cascade. The friends are collected by leveraging the Twitter API.[2]. In the next section, we describe our methodology of creating a content-agnostic cascade-wise fake news ranking system and a pipeline for converting the raw data of these datasets, i.e., the cascades, into data structures used for our machine learning approaches.

[2]https://developer.twitter.com/en/docs/twitter-api/users/follows/api-reference/get-users-id-following

4 Methods

In this work, we aim to create a practical approach for fake news ranking on online social media. To that end, we create a content-agnostic cascade-wise ranking system for fake news.

To alleviate the dependency on textual features and thus create a more generally applicable fake news detection system, an alternative approach to fake news detection is required. Since the work by Vosoughi et al. [73], interest in leveraging the diffusion patterns of fake news on online social media has sparked. The core idea behind this approach is that fake news spreads like a virus. Vosoughi et al. [73] showed that the spreading patterns for fake news are different compared to real news. Thus, modeling the spread of a news item on social media could indicate that it contains fake news without using any textual features. By only leveraging the information of users involved in the spread and the spreading pattern itself, we eliminate the use of textual features in our approach. Therefore we present a system that is not dependent on the topic and language on which it is trained.

To achieve a ranking system for fake news, we approach this problem from a binary classification point of view during training and evaluate it from a ranking point of view. In other words, during training, we find the best possible separation between true and fake news, and during evaluation, we rank all items on their likelihood of belonging to the fake class. It is important to note that with this approach, we treat every item belonging to the same class equally; in other words, two items labeled as fake news are treated as equally damaging. While some fake news stories are arguably more damaging than others, this nuance is beyond the scope of this work.

To address the problem of grouping multiple items about the same topic as a single data point, we use cascade-wise classification, i.e., we take a single tweet as the source and model its propagation pattern to determine the content's veracity. And subsequently evade the problem of collecting all tweets concerning the same topic.

Finally, this work will use Twitter as the social network platform. The reasoning behind this choice is twofold. First, Twitter allows for the extensive collection of public data, unlike, for example, Facebook. Second, almost all previous approaches to fake news detection on online social media used Twitter as the social network platform, as shown by Mridha et al. [51], which makes our work comparable. We also note that, even though we only use Twitter as our social media platform, the insights gained in this work can be transferred to other social media platforms that share the characteristics of Twitter. A good example is the social media platform Weibo, a Chinese social media platform that shares the user features as well as the sharing functionality of Twitter, thus containing all necessary information required for our approach. Since the feature distributions can diverge significantly from Twitter, a system would have to be retrained, but the general approach is transferable.

In this section, we describe the methodology for the data preparation of our proposed ranking system. The process of data preparation consists of two steps: The first step is the creation of a pipeline that can convert any tweet into a representation suitable for either a graph neural network or a traditional machine learning method. This is described in Section 4.1. The second step combines all graphs and feature vectors in a single dataset suitable for the training, optimization, and evaluation of a machine learning method. This is described in Section 4.2.

We aim to improve the adaptation of fake news ranking systems in the real world by supplying practitioners and future researchers with an expandable pipeline for data creation. We believe that having such a pipeline is a requirement when trying to bridge the gap between academia and the real world.

4.1 Conversion of a single data point

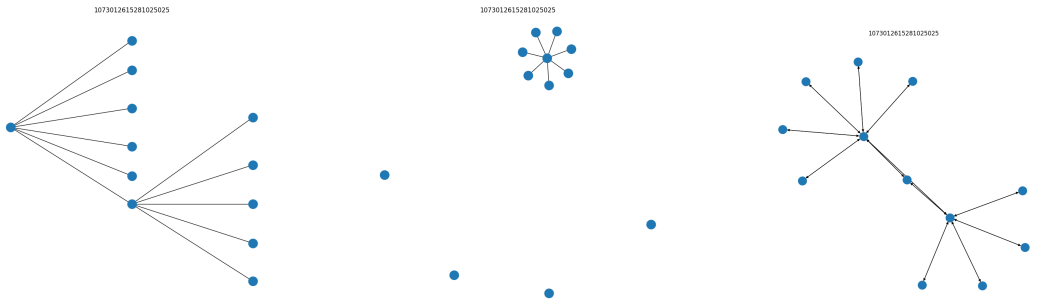
In Section 3 we described how we obtained a list of labeled tweets. To construct our cascades, we require additional information. For each source tweet, the retweets are collected. Then, the profile data

and friends list is collected for all users involved in the cascade, i.e., the author of the source tweet and all users who have retweeted.

We use three different graph structures to convert a single data point to the final output representations. The definitions of the three different types of graph structures, and output representations created by the pipeline, are explained in sections 4.1.1 and 4.1.2, respectively. Finally, Section 4.1.3 provides an overview of the complete process, from a single tweet to a representation suitable for a machine learning method.

4.1.1 Graph types

To create the data representations suitable for graph neural networks and traditional machine learning approaches, we use three different graphs related to the spread of a tweet on Twitter. Each graph can be defined as a set of vertices V, and edges E, where each edge is a pair of vertices (v, w) with $v, w \in V$. For all three graphs the set of vertices is identical, i.e. $V_s = V_c = V_p$. The features embedded in the nodes will be described in Section 4.1.2. The set of vertices will be represented as an ordered list of all users involved in the spread of the tweet. With v_0 we denote the original tweet, and $v_{(n-1)}$ (where $n = |V|$) denotes the last retweet. Figure 4 shows the three types of graphs.



(a) Spreading graph (b) Community graph (c) Propagation graph

Figure 4: The three different types of graphs that are used to represent a single cascade. In this example we took a single cascade and modeled the three different graphs; Spreading (a), Community (b), and Propagation (c).

4.1.1.1 Spreading graph

The spreading graph G_s captures the *news spreading* pattern of a tweet, i.e., it represents an approximation of the diffusion of a tweet on Twitter. The edges for G_s are defined as follows:

For every $v_i \in V_s$ where $i > 0$ we draw an edge via the following statement: If v_i follows at least one user in $\{v_0, \ldots, v_{n-1}\}$, an edge is drawn to the last user in $\{v_0, \ldots, v_{n-1}\}$ who is followed by v_i. Else an edge is drawn to the user in $\{v_0, \ldots, v_{n-1}\}$ who has the most followers.

We note that this approach does not completely mimic the diffusion of a tweet on Twitter; however, this technique is used in most works relating to diffusion paths on Twitter. With the notable exception of Zola et al. [78], who introduced a novel technique called 'Interaction Strength' to capture a tweets diffusion pattern. In their approach, they aim to combine historical interaction into the diffusion graph, thus addressing the *suggested for you* diffusion path, which with the rise of better recommendation systems, is becoming more prevalent in online social media. However, their approach requires an infeasible amount of data collection and is, therefore, beyond the scope of this work.

4.1.1.2 Community graph

The community graph G_c captures the *community* of the users spreading the tweet. The edges for G_c are defined as follows:

For every $v \in V_c$ an edge is drawn to every $w \in V_c$ if v *follows* w.

The *follows* relationship is derived from collecting all *friends* for each user $v \in V_c$. We found that of the 576 276 unique users involved in the cascades, 157 077 have been either suspended, removed, or changed their profile to private. When removing all cascades where a user involved is removed, only 932 remain; since this is too little data to train our models, all cascades are kept. Users who spread fake news on Twitter might have a higher chance of being removed from the platform. Figure 5 shows the cumulative density plot for a fraction of existing users in True and False news cascades. Though it shows there is indeed a difference between the True and Fake News cascades, we argue that this difference is too little to affect our model significantly.

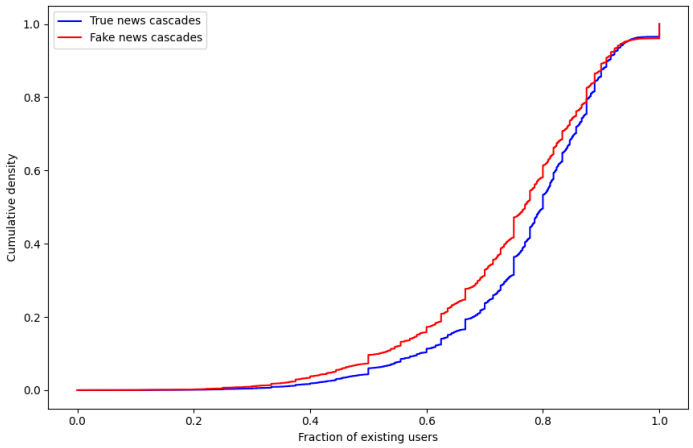


Figure 5: Cumulative density plot for the fraction of existing users in True (blue) and Fake (red) news cascades.

4.1.1.3 Propagation graph

The propagation graph G_p combines characteristics from both the spreading and community graph and serves as the input graph for our graph neural networks. The edges for G_p are defined as follows:

For every $v \in V_p$ an edge is drawn to every $w \in V_p$ if one of the following holds:

$$(v, w) \in E_s$$

$$(w, v) \in E_s$$

$$(v, w) \in E_c$$

$$(w, v) \in E_c$$

These four criteria will be encoded as edge features for the propagation graph. The spreading and community graph contains no edge features.

With this propagation graph, which will serve as the input for our graph neural network model, the graph neural neural network has access to all node features, spreading patterns, and community

structures of a single cascade. By encoding the bidirectional spreading and community relations in the edge features, a graph attention network can learn the direction and importance of these relations. The bidirectional encoding allows for a graph neural network to learn independent of the original relation direction and subsequently prescribe different importance to the type and direction of the relationship between nodes. Since the node features and edge features are both combined in the graph attention network, a system can learn what relation in combination with node characteristics indicates the presence of fake news.

4.1.2 Features

In this section, we describe the features of our approach. First, we describe the features embedded in the nodes of the graph neural network input. Second, we describe the fixed-sized feature vector used for our traditional machine learning approach.

For the graph neural network models, a set of features is embedded in the nodes and edges of the input graph. In the previous section, we described the structure and edge features for our graph neural network input. The node features are shown in Table 2. These features describe the *user* associated with the node.

A fixed-size feature vector is required to make the data suitable for traditional machine learning approaches. The complete set of features for a cascade consists of *node features* and *network features*. When using more traditional models that, unlike a GNN, do not allow a varying input size, the features need to be translated into a vector of fixed size. When combining multiple nodes for every numerical node feature, we calculate the mean, variation-coefficient, min, max, and entropy. This approach has been successfully applied by Xu et al. [76] and introduced by Nudelman et al. [54]. For every binary node feature, the sum is calculated. Another difference between the feature set of traditional models and GNNs is the network features. GNNs allow a graph structure as input; thus, the structural relations are already embedded in the input. Traditional classification models do not allow graph-based input. Therefore, features that represent the graph are also concatenated to the input vector. Table 3 shows the complete list of features in the input vector.

With the 14 network features used in this work, we aim to capture the structure from both the *community* graph, i.e., the structure social community of the users involved, and the *retweet* graph, i.e., the diffusion path of the tweet. The network feature *structural virality* can be a good indication of the veracity of the tweet, as shown by Vosoughi et al. [73]. Previous research on diffusion patterns of fake news has often focused on centrality measures; therefore, in this work, we have included the most common centrality measures as network features. We note that most of the network features are node based, resulting in a unique value for each node. Since a traditional machine learning approach does not allow for a varying input size, and the number of nodes in a cascade can differ, we calculate the mean, variation-coefficient, min, max, and entropy for all node-based network features.

4.1.3 Overview

The process of creating a graph and feature vector, which will be used as input for our models, from the raw data supplied by Twitter is described in Figure 6. This is done for every tweet in our datasets. When faced with an incoming stream of new tweets of which the veracity is to be ranked, this pipeline allows for quick conversion of the input required for a trained model.

4.2 Combining cascades into a dataset

In the previous section, we have described how a single data point is converted into a representation that is suitable for a graph neural network or a traditional machine learning approach. Now we will describe how these individual data points are combined into a single dataset suitable for the training, optimization, and evaluation of machine learning systems. In Section 4.2.1, we describe our approach for

Table 2: List of node features.

Feature	Type	Description
Friends count	Numerical	The number of users the current user follows.
Followers count	Numerical	The number of users who follow the current user.
Tweet count	Numerical	The number of tweets the current user has posted.
Listed count	Numerical	The number of lists the current users appears on.
Verified	Binary	If the current users profile is verified.
Account age	Numerical	The number of days the account has existed, at the moment of (re)tweet.
Retweet time difference	Numerical	The number of hours since the retweet in comparison to the original tweet. 0 if the tweets is the original tweet.
Distance to source node.	Numerical	The distance of the current node in the Retweet graph, as described in Section 4.1.1, with respect to the root node.

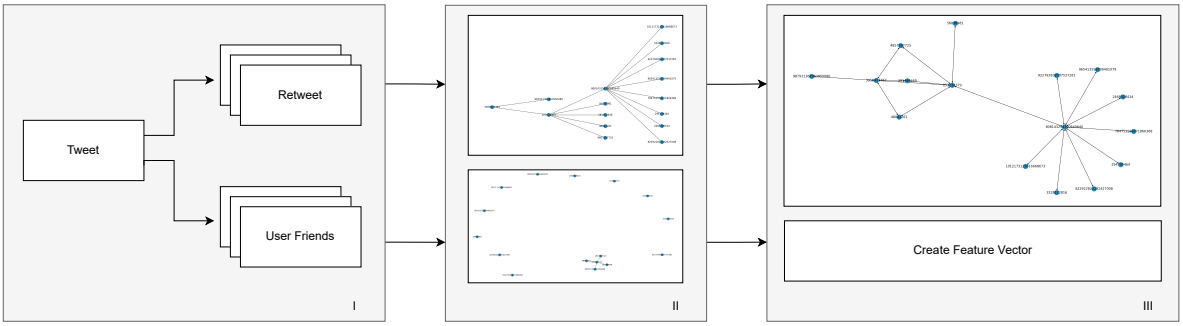


Figure 6: The whole process of collecting and creating the data for both our graph neural networks and our traditional models is captured in this diagram. The process consists of three stages, which are marked as gray boxes. Stage I consists of collecting the relevant data from Twitter. For each tweet, we collect the related retweets and the user profiles for all the users involved in the cascade. In Stage II, the community and retweet cascade are created, as defined in Section 4.1.1. In Stage III, the graph data from the community and retweet cascade is used to create the spreading cascade and the fixed-size feature vector for graph neural networks and traditional models.

combining multiple data points into a single dataset. In Section 4.2.2, we described our normalization approach. Finally, in Section 4.2.3, we provide a complete overview of the process from raw data to input for a machine learning system.

4.2.1 Grouping of data points

When combining our single data points into a dataset suitable for a machine learning system, it is essential to focus on preventing data leakage. Data leakage would not be a problem in a situation where all data points are unrelated; however, due to the collection approach for the FakeNewsNet dataset, multiple tweets are related to each other by the topic they address. As mentioned earlier, the FakeNewsNet dataset collects tweets based on stories. Thus, the tweets that belong to the same *story* are not unrelated. Therefore when combining all single data points into a dataset, we group the cascades which share a *story* such that when creating our training, validation, and test folds, all a story is only present in one of the folds. For a dataset with unrelated data points, e.g., our evaluation dataset, each cascade is treated as a story with only one cascade. This approach is known as grouped cross-validation.

Table 3: List of features used in feature vector for traditional models. When node features are returned for multiple nodes, e.g., all node features, for every binary feature, the sum is returned, and for every numerical feature, the mean, variation-coefficient, min, max, and entropy are returned. When network features are node based, i.e., a unique value is calculated for every node in the network, again, the mean, variation-coefficient, min, max, and entropy are returned. The column 'Feature' holds the name of the feature. The column 'Graph' denotes the graph type on which the feature operates. The column 'Description' provides a brief description of the feature.

Feature	Graph	Description
Source node features	Community	Node features from the source node, i.e. the node that has tweeted the original tweet. The node feature 'distance to source node' is omitted.
All node features	Retweet	Node features from all nodes.
Total number of nodes	Community	The total number of nodes.
Total number of edges	Community	The total number of edges.
Total number of triangles	Community	The total number of triangles.
Eigenvector centrality	Community	Calculates the centrality for each node, which is the summed connection to other nodes weighted by the centrality of those nodes, as described by Bonacich [5].
Degree centrality	Community	For each node, the number of connections is calculated, divided by the total number of nodes.
Closeness centrality	Community	For each node, the total number of nodes divided by the sum of the shortest paths to all other nodes is calculated, as described by Freeman [22].
Current flow closeness centrality	Community	Also known as information centrality. Alteration of the closeness centrality where the shortest paths are replaced with a model of electrical current as described by Brandes et al. [7].
Betweenness centrality	Community	For each node the sum of the fraction of all shortest paths that pass through that node is calculated as described by Freeman [22]. Thus providing a measure for the *importance* of a node.
Current flow betweenness centrality	Community	An alteration of the betweenness centrality where the shortest paths are replaced with a model of electrical current as described by Brandes et al. [7].
Harmonic centrality	Community	For each node the sum is of 1 divided by the shortests paths from all other to the node of interest is calculated as described by Boldi et al. [4].
Pagerank	Retweet	A ranking of all nodes based on the structure of incomming links is calculated as described by Page et al. [55].
Clustering coefficient	Community	Calculates the fraction of total triangles through a node.
Total number of connected components	Community	Calculates the number of connected components. A connected component is a set of nodes and edges where there exists a possible path from all nodes to all other nodes.
Structural virality	Retweet	Calculates the average difference between all pairs of nodes in a diffusion tree as described by Goel et al. [24].

4.2.2 Normalization

When creating the dataset for our traditional machine learning approach, we do not normalize the data. In this work, we evaluate two approaches to traditional machine learning systems: random forests and the AutoSklearn 2.0 framework. Random forests are inherently scaling invariant because random forests are a tree-based method. Scaling is only problematic when something reacts to that scale; tree-based methods only make a cutoff at a certain point; thus, the scaling of features does not affect performance, as shown by Ferreira et al. [19]. The AutoSklearn 2.0 framework, as described in Section 2.4.3, incorporates feature preprocessing in its optimization process; therefore, it is not required to do normalization

prior to calling the AutoSklearn 2.0 framework.

The graph neural network approach used in this work does require a normalization step for optimal performance. We deploy the *z-score* normalization technique for all node features. The z-score normalization technique work by applying the following formula to the input features:

$$x' = \frac{x - \mu}{\sigma}$$

Where μ denotes, the mean and σ denotes the standard deviation of the feature. Since we do not know the true mean and standard deviation of the features, these values are calculated based on the training sample, i.e., the train set. Resulting in a feature set where each feature has a mean of 0 and a standard deviation of 1. The edge features are not normalized since all edge features are binary, i.e., either a 1 or a 0.

4.2.3 Overview

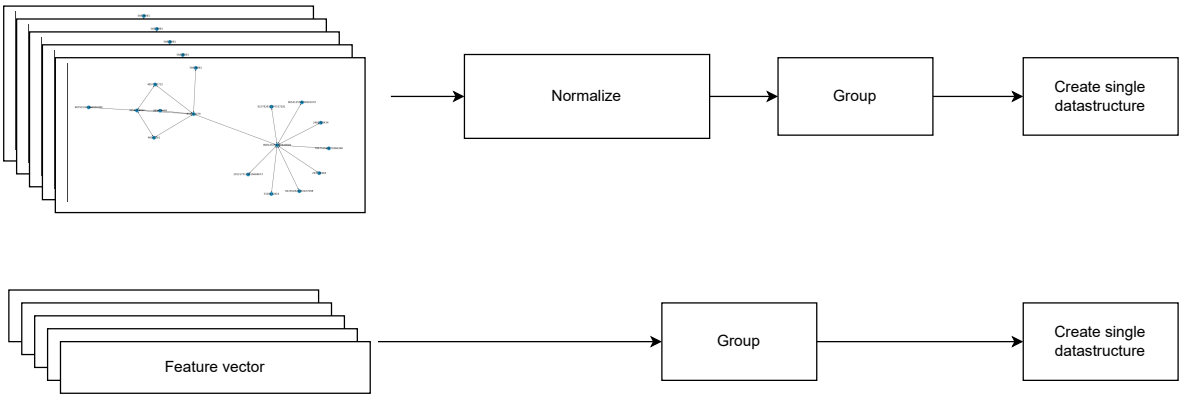


Figure 7: This figure provides an overview of combining the single data points into a dataset suitable for a machine learning approach. The pipeline for a graph neural network approach is shown in at the top. We identify the following four steps. First, we collect all graphs from all individual tweets, as described in Section 4.1. Second, we apply *z-score* normalization as described in Section 4.2.2. Third, all data points are grouped according to their *story*, as described in Section 4.2.1. Finally, a single data structure is created. The pipeline shown at the bottom of the figure represents the process of combining single data points for a traditional machine learning approach. First, all feature vectors are collected as described in Section 4.1. Second, all data points are grouped according to their *story*, as described in Section 4.2.1. Finally, a single data structure is created.

Figure 7 shows the pipeline of combining single data points into a dataset suitable for machine learning approaches. We note that the first part of the pipeline is the single data points obtained from Figure 6. This complete pipeline, i.e., the combination of Figures 6 and 7, results in a pipeline that can convert any list of tweets into a format suitable for machine learning techniques.

We note that this whole pipeline is available online as open-source code.[3] With this pipeline, any list of labeled tweets can be converted into a dataset used for training and evaluating machine learning models. The codebase is created in a modular fashion, which enables the easy addition or alteration of features. We refer to the documentation on the GitHub page for an explanation of how to use this pipeline.

In the following section, we describe the models used in this work, which aim to use the features and structures described in this section to distinguish true from fake news on online social media.

[3]https://github.com/GuusToussaint/MasterThesis

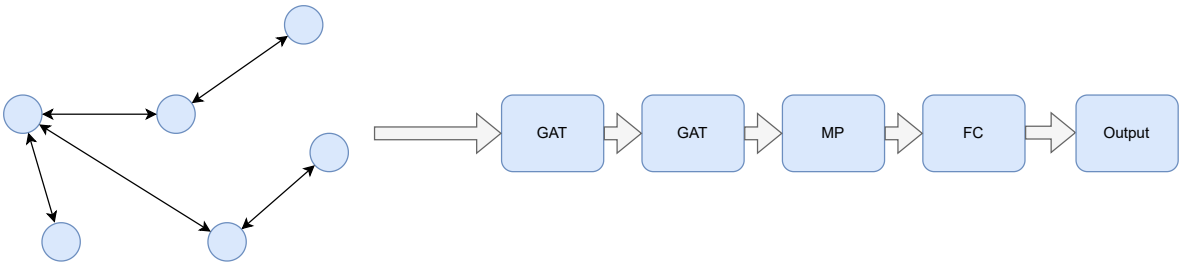


Figure 8: Diagram of a Graph Neural Network as presented by Monti et al. [50] From left to right; The input cascade as described in Section 4.1.1 is fed into the Graph Neural Network. First the input is passed through two Graph Attention (GAT) layers, with an output dimension of 64, resulting in an $N \times 64$ Matrix for a single graph, where N denotes the number of nodes. After the global mean pooling layer (MP), a vector of 64 remains. This vector is fed into the a fully connected layer with 32 neurons, finally an output is created by the two output neurons.

5 Setup and Results

This section describes the setup and results of our various experiments related to the three subquestions and the main research question. In the previous section we have described the pipeline for converting raw data into data that is usable for a machine learning system. In this section we will describe the methods and models used to exploit the data obtained from our pipeline. In Section 5.1 we implement the state-of-the-art and evaluate its performance on a publicly available dataset, in Section 5.2 we compare said method to a more traditional approach with handcrafted features, and finally, in Section 5.3 we use the knowledge obtained from our handcrafted features method to improve upon the state-of-the-art method. Once we have obtained the results for the models mentioned above, we can evaluate their performance in a real-world setting and subsequently answer our main research question in Section 5.4.

For all models, GNN and traditional, which are evaluated on a hold-out test set, we use 5-fold cross-validation to report the results. The data is split up into 80% training data, 10% evaluation data, and 10% test data.

5.1 Performance of current SOTA on FakeNewsNet

With this experiment, we aim to answer the first subquestion of this work: *How does the current state-of-the-art cascade-wise classification of fake news on online social media using propagation structures perform on the content-agnostic variant of the FakeNewsNet dataset in terms of Area under the ROC Curve (AUC)?* We first describe the experimental setup used to conduct our experiments and then provide the results.

5.1.1 Setup

Our experiment consists of two steps to evaluate the state-of-the-art method on a new dataset properly. First, we implement and train the method exactly as described in the original work. Second, we tune the structure and hyper-parameters associated with the method using the SMAC3 HPO framework. The HPO is done to allow the model and hyper-parameters to adapt to the new dataset.

As said, for step 1 of the experiment, we will implement the state-of-the-art according to the original work. The current state-of-the-art method of cascade-wise classification of fake news on social media is presented by Monti et al. [50].

The structure of the model can be described as follows: They present a graph neural network model

Table 4: hyper-parameter Optimisation search space. The 'Monti et al. [50]' column shows the values used in the original work. The 'HPO Configuration Space' column shows the range and options used in the HPO search for numerical and categorical hyper-parameters respectively.

Name	Monti et al. [50]	HPO Configuration Space
# convolutional layers	2	1 - 5
# output dimensions per convolutional layer	64	16 – 256
# fully connected layers	1	1 - 5
# output neurons per fully connected layer	32	16 – 256
learning rate	5×10^{-4}	1×10^{-4} – 1
loss function	Hinge loss	Hinge loss, Cross Entropy Loss
Optimizer	AMSGrad	AMSGrad

with two graph convolutional layers and two fully connected layers. A graph attention network (GAT), as described in Section 2.3.2, with one head of graph attention and a 64-dimensional output map was used for each graph convolutional layer. Global mean pooling was used for dimensionality reduction. Throughout the network, scaled exponential linear unit (SELU) [39] was used as non-linearity. The two fully connected layers have 32 and 2 output neurons, respectively. Finally, the softmax function was used to scale the two output neurons to probabilities for true and fake news. A diagram of the model is shown in Figure 8.

In the original work, Monti et al. [50], note the following training settings and hyper-parameters: During training, AMSGrad [60] was used as an optimizer, and hinge loss was deployed as the loss function. A batch-size of 1 is used, the rationale for this choice of batch-size is not mentioned in the paper. Since our training dataset is significantly larger, 3 586 versus 29 700 data points, and the slow training time associated with a batch-size of one, we set the batch-size to 256 in all of our experiments, this number was chosen as a trade-off between efficiency and quality. We note that it is possible to set a batch-size higher, however, as stated by Keskar et al. [38] having a large batch-size might result in a significant degradation in the quality of the model. All other hyper-parameter values can be found in Table 4; the column 'Monti et al. [50]' refers to the values presented in the original work and correspond to the values that will be used in step 1 of the experiment.

The second part of the experiment consists of tuning the structure and hyper-parameters of the *base*-model to allow it to adapt to the new dataset. The configuration space for our HPO framework is shown in column 'HPO Configuration Space' of Table 4. The hyper-parameters '# convolutional layers' and '# output dimensions per convolutional layer' denote the number of GAT layers and the size of the output map, respectively. The '# fully connected layers' and '# output neurons per fully connected layer' hyper-parameters denote the size of the fully connected network, excluding the final output layer, which is of size 2. The learning rate is sampled from a log scale between the ranges 1×10^{-1} and 1. The loss function is chosen from Hinge loss [23] and Binary Cross Entropy Loss [15]. We note that it is possible to incorporate more tunable features in the set of hyper-parameters that are to be optimized, such as the size of the learnable function ϕ as described in Section 2.3.2. However, we limited ourselves to the aforementioned six tunable hyper-parameters since we believe these are the most important for performance and to allow the HPO method for sufficient exploration with the limited compute budget.

More specifically, we will use the `SMAC4MF` method from the `SMAC3` package, which is described in Section 2.4.2, with a minimal budget of 7 epochs, a maximum budget of 60 epochs, and an η of 3. The HPO framework will be deployed on a RTX 2080TI GPU with an execution time of 48 hours.

The threshold of 60 epochs was chosen since our initial results show that over-fitting occurs around the 60 epoch mark, as shown in Figure 9. To make a fair comparison between parts 1 and 2 of our experiment, we also train the *base*-model for 60 epochs.

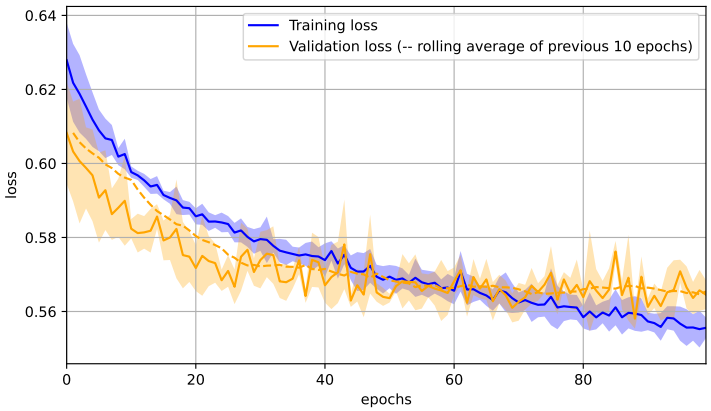


Figure 9: Training progress of the Graph Neural Network as presented by Monti et al. [50], we notice no further improvement of the validation loss around the 60 epoch mark. The average is taken of 5 fold cross validation.

Table 5: Configurations for the 10 best performing models of the hyper-parameter Optimisation search.

AUC	Learning rate	Loss function	# conv. layers	# output dim. GAT	# FC layers	# out dim. FC
73.40%	0.00195	cross entropy loss	2	244	5	184
73.30%	0.00193	cross entropy loss	2	244	5	212
73.00%	0.00195	cross entropy loss	3	144	3	184
73.00%	0.00195	cross entropy loss	2	146	4	184
72.97%	0.00195	cross entropy loss	3	144	3	184
72.96%	0.00195	cross entropy loss	2	144	4	184
72.89%	0.00195	cross entropy loss	3	144	3	184
72.84%	0.00190	cross entropy loss	3	144	2	183
72.83%	0.00130	cross entropy loss	3	237	1	239
72.83%	0.00332	cross entropy loss	3	19	5	116

From now on, the *base*-model, from step 1 of the experiment, will be denoted as 'Monti et al.', and the *tuned*-model, from step 2 of the experiment, will be denoted as 'Monti et al. (HPO)'.

5.1.2 Results

In the first part of our experiment, we reimplemented the state-of-the-art as presented by Monti et al. [50]. Their original work obtained an AUC score of 88% for cascade-wise classification. Figure 10 shows the ROC curves we obtained, and we note an AUC of 70% for the base-model, i.e., 'Monti et al.'. The cause of this performance deficit can be one of the following: First, the data used in the original work was cleaned by human annotators, while the data used in this work was not. Second, the data used in the original work has a majority class of 82%, while the data used in this work has a majority class of 62%. Third, since we aim to create a practical system, we have removed the textual features from our feature set. Since neither the code nor the data from the original work are available, we conclude that our implementation of the state-of-the-art, which obtained an AUC of 70%, resembles the current state-of-the-art of cascade-wise fake news classification.

In the second part of our experiment, we tuned the structure and hyper-parameters of the state-of-the-art to allow the model to adapt to the new dataset. Shown in Table 5 are the results of the top-10

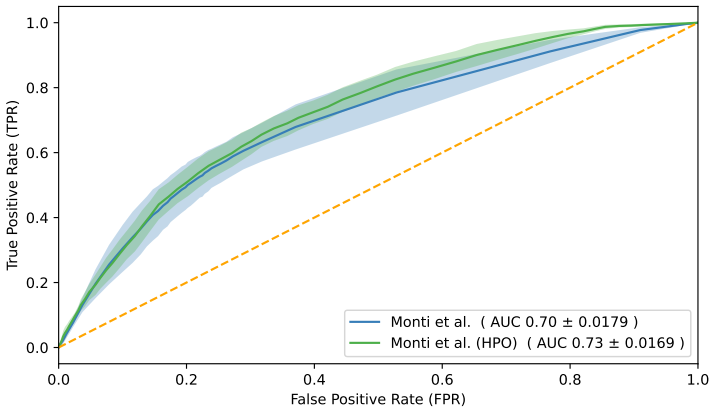


Figure 10: Receiver operator characteristics curve for the graph neural network based model presented by Monti et al. [50] (blue), and the hyper-parameter Optimised version (green). The lower right shows the AUC of both curves and the standard deviation across 5-folds of cross validation.

best-performing models from out HPO search. We observe that all top-10 best-performing models use 'cross entropy loss' as the loss function, indicating it is a better fit for the current setup. Furthermore, we note that all models have a higher number of neurons - compared to the base-model - for the fully connected layers, indicating that 32 is not the optimal value. This trend can also be observed in the output dimensionality of the GAT layers. Except for the number of GAT layers, all other hyper-parameters obtained in our search take different values than those presented in the original work, which indicates that the proposed values for the hyper-parameters and structure of the model are not ideally suited for our current setup.

When comparing the relative performance between the base-model and the tuned-model as shown in Figure 10, we observe the following: We note a AUC of 70%($\pm$1.79%) and 73%($\pm$1.69%) for the base-model and tuned-model respectively. With a $\mu = 0.70$, a $\sigma = 0.0179$ for the base-model and a $\mu = 0.73$ and a $\sigma = 0.0169$ for the tuned-model, both with a sample size of 5 we can calculate the test statistic with a significance level $\alpha < 0.05$. We obtain a p-value of 0.026, which is smaller than α; thus, we note a significant improvement.

The ROC curves for the two methods follow each other in the first part of the graph, up to false positive rate 0.2, which indicates that both models perform equally well at predicting the 'obvious' fake news items. At higher values for the false positive rate, the tuned-method attains a higher true positive rate, indicating that the tuned-method is better at distinguishing the harder-to-classify fake news items than the base-method. Since our end goal is to create a fake news *ranking*-system, we are particularly interested in the first part of the ROC curve since a higher true positive rate for a small false positive rate results in a better top-k precision for small values of K. Thus, even though we note a significant improvement with the tuned-model compared to the base-model, by analyzing the ROC curve, we observe that this dominance is not present in the first part of the ROC curve. Subsequently, we conclude that this dominance will be less relevant when evaluating the top-k precision.

5.2 Traditional models with handcrafted features

With this experiment, we aim to answer the second subquestion posed in this work: *How does a traditional model with hand-crafted features perform on the content-agnostic variant of the FakeNewsNet dataset in terms of AUC?*

5.2.1 Setup

We have designed a fixed-sized feature vector to train traditional machine learning algorithms, i.e., algorithms requiring an underlying euclidean data structure with fixed input size. The hand-crafted features in the feature vector are explained in Section 4.1.2.

Using the aforementioned fixed-sized feature vector, train the following models. First, we train a traditional random forest, as described by Ho et al. [31], with the default settings as defined by `sklearn`[10]. Second, we use AutoSklearn 2.0 framework to find the best combination of feature pre-processing, learning algorithm, and hyper-parameters.

The random forest model allows us to calculate the relative feature importance using the mean decrease in impurity measure. Using all features, we calculate the top 10 most important features for our base random forest model. We acknowledge that 'feature importance' from a random forest is inherently noisy since often, not a single feature but a collection of features provides the most predictive power. That being said, it does provides valuable insight into what features *might* be informative for the task at hand.

In addition to the feature importance, the random forest model will also be used to conduct an ablation study. We compare a random forest trained with all features to a random forest trained with only the node features, thus emitting the network features altogether. More specifically, from Table 3, only the features 'Source node features', 'All node features', and 'Total number of nodes' are kept. Furthermore, Table 2 shows the feature 'Distance to source node'; since this feature requires knowledge of the structure of the propagation graph, this feature is also omitted. With this approach, we aim to identify fake news solely on the people who spread it, not how that spread occurred. By comparing the performance of these random forest models, we gain insight into the added predictive power of the propagation patterns. Furthermore, if adding the network features yields no additional performance, they can be omitted when deployed in a real-world setting. This is important since collecting the network features is time-consuming. When no additional predictive power is generated when including the network features, this process can be omitted from the pipeline.

Finally, we use the hyper-parameter optimization strategy presented by Feurer et al. [20] `AutoSklearn 2.0`. To make a fair comparison against the HPO strategy used for the graph neural network models presented in the previous section, we allow the same compute time, i.e., 48 hours, on a single CPU. We note that a GPU and a CPU are inherently different devices with different costs, both in power and purchase price. However, these differences are beyond the scope, and intent, of this work, and a CPU and GPU compute hour will be treated as equal in this work. The final ensemble obtained from AutoSklearn will be the HPO optimized hand-crafted feature model.

5.2.2 Results

Figure 11 shows the ROC curves for our models trained with hand-crafted features. We observe that removing the network features from the fixed-sized feature vector did not impact the performance of the random forest. Indicating that the structural features do not add any predictive power in the context of cascade-wise fake news classification on the FakeNewsDataset. This observation has a significant impact on the practical application of the system. As mentioned earlier, the data-collection process for the network features is the most time-consuming step of the pipeline. Thus, having found that these features add no additional predictive power, the data collection step for the network features can be omitted.

Besides the ablation study, we also investigated the feature importance obtained from our random forest model. Figure 12 shows the relative feature importance of the top 10 most important features for our base random forest model trained on all features. The feature 'account age' shows up in three of the top 10 features, the maximum account age present in the cascade is the most important feature. The mean account age and source node account age also seem to indicate whether the cascade contains true

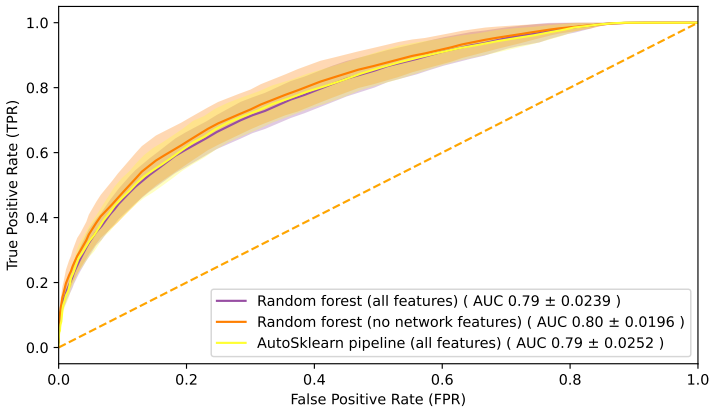


Figure 11: ROC curve for the Random Forest with all features (purple), Random Forest (structural features omitted) (yellow), and HPO optimised model (orange).

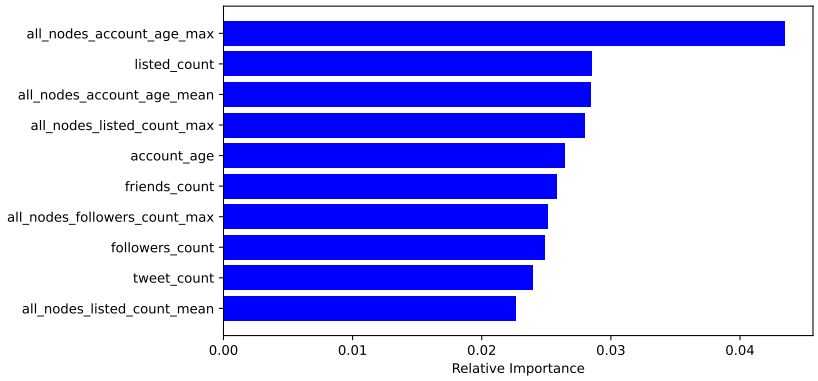


Figure 12: Relative feature importance of the random forest model with all features.

or fake news. We note that none of the structural features are present in the top 10, further solidifying the lack of predictive power associated with network features in this context, as mentioned earlier. Furthermore, source node features comprise 5 of the top 10 most important features. This indicates that when classifying the veracity of an item, the features describing the message's author provide the most predictive power.

When comparing the relative performance between the random forest models and the HPO ensemble, i.e., AutoSklearn 2.0, we note that the HPO ensemble has not improved the performance compared to the random forest models, AUC of 79%($\pm$2.52%) versus AUC of 79%($\pm$2.39%).

We now compare the results obtained from our random forest models to those obtained from the GNN-based models which are the state-of-the-art. Figure 13 shows the ROC curves of all hand-crafted feature models alongside the state-of-the-art models. We note a superior performance of the models with hand-crafted features over all areas of the ROC curve, including the first part of the curve, which is essential when evaluating the model as a ranking system. The dominance of the whole ROC curve indicates a knowledge gap between our graph neural network input cascades, i.e., the input for the state-of-the-art models and the fixed-sized input vector. Furthermore, we observe that a simple random forest trained on the same data with hand-crafted features outperforms the state-of-the-art. With a

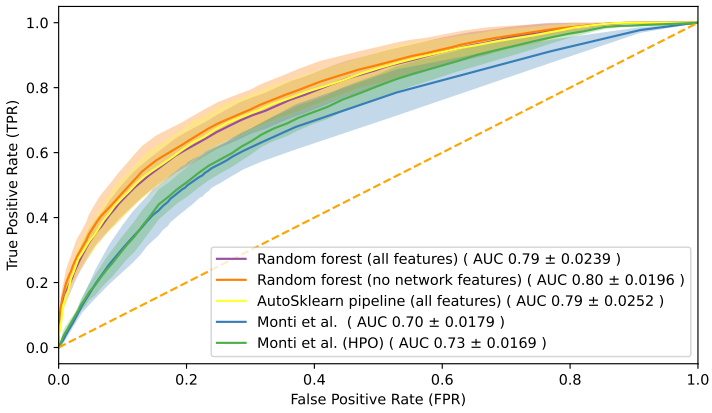


Figure 13: ROC curve for all models Monti et al. [50] (blue), Monti et al. [50] HPO optimised (green), random forest with all features (purple), random forest (structural features omitted) (orange), and the AutoSklearn pipeline (yellow).

$\mu = 0.79$ and a $\sigma = 0.0239$ for the hand-crafted features method, and a $\mu = 0.73$ and a $\sigma = 0.0169$ for the HPO state-of-the-art method, we can calculate a test statistic with a $\alpha = 0.05$. We obtain a p-value of 0.001, which is smaller than α; thus, we significantly improve upon the state-of-the-art. This tells us that, in our setup, a more straightforward approach is a better fit when compared to a graph neural network. Since a graph neural network requires the structure to learn valuable information, in combination with the lack of improvement when incorporating the network features with the random forest, we believe that for our setup, the structure is more a distraction than a valuable source of information.

5.3 Combining Source Node features with the Graph Neural Network

Figure 12 shows that 5 out of the top 10 most important features are source node features. Furthermore, in Figure 13 we can observe a significant difference in performance between the models trained on handcrafted features and the graph neural network models. We hypothesize that this is because the graph neural network cannot know the source node in the network. This section aims to test that hypothesis and bridge the knowledge gap between the models trained on handcrafted features and the graph neural network models. To that end, we provide an answer to the third subquestion posed in this work: *Can we improve upon the state-of-the-art method by leveraging knowledge from the hand-crafted features method?*

5.3.1 Setup

This hypothesis will be tested by concatenating the source node's 'node features', as described in Table 2, to the tensor received from the global mean pooling layer. Figure 14 provides a detailed schematic overview of this new method's structure and data flow. In short, we create a skip connection from the source node to the global mean pooling layer.

We rerun the experiments outlined in Section 5.1 with the new structure. That is, we first train the structure with the hyper-parameters presented by Monti et al. [50], and then we run an HPO framework to optimize the structure and hyper-parameters settings of the model.

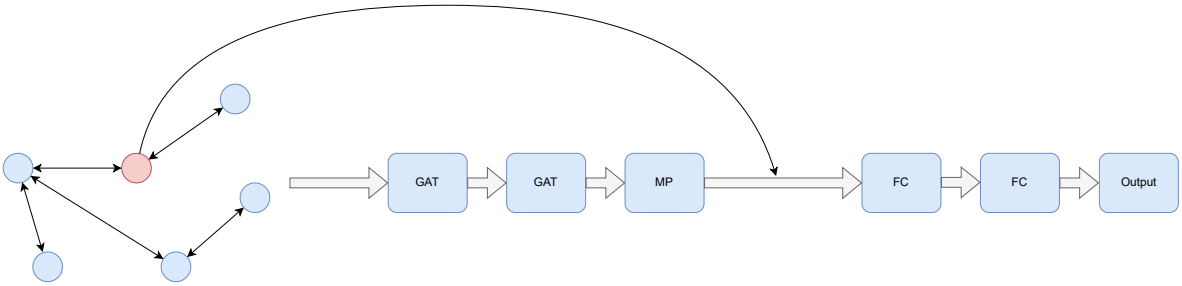


Figure 14: Diagram of a Graph Neural Network where the source node features are concatenated after the Global Mean Pooling Layer. From left to right; The input cascade as described in Section 4.1.1 is fed into the Graph Neural Network. Followed by two GAT layers, with an output dimension of 64, resulting in an $N \times 64$ Matrix for a single graph, where N denotes the number of nodes. After the global mean pooling layer, a vector of 64 remains. This vector is then concatenated with the 7 node features, as described in Table 2 (excluding the feature distance to source, which is always 0 for the source node), resulting in a vector of size 71 which is fed into the two fully connected layers.

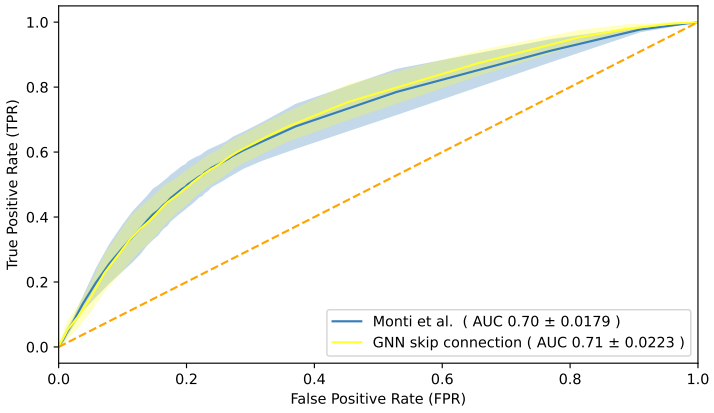


Figure 15: The ROC curve of the Graph Neural Network presented by Monti et al. [50] (blue), and the adjusted model which includes the source node features (yellow).

5.3.2 Results

Figure 15 shows the adjusted graph neural network' results for the hyper-parameters settings as presented by Monti et al. [50], in comparison to the base-model. We note a minor improvement in the AUC score. If we take a closer look at the ROC curve, we can observe that the improvement is located at the end of the curve. At the beginning of the ROC curve, we observe that the base-model outperforms the adjusted-model ever so slightly. Indicating that the slight improvement is not that relevant when we evaluate the model as a ranking system.

Since the structure of the model has changed, we have conducted another hyperparameter optimization search, with an identical search space as presented in Section 5.1. Figure 16 shows the ROC curves of the tuned-model, i.e., 'Monti et al. (HPO)', and the adjusted-model, i.e., 'Source node (HPO)'. We observe almost identical performance along the complete ROC curve. This indicates that our proposed improvement of adding a skip connection from the source node to the global mean pooling layer added no additional predictive benefit for the classification of fake news cascades.

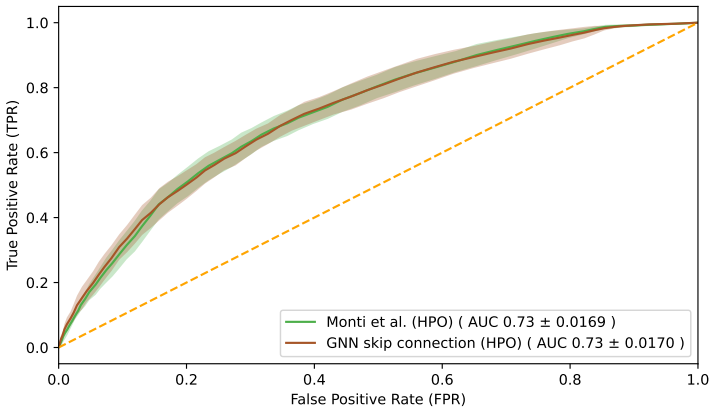


Figure 16: The ROC curve of the base and source-node methods, both optimised using hyper-parameter Optimisation.

Table 6: Configurations for the 10 best performing models of the hyper-parameter Optimisation search.

AUC	Learning rate	Loss function	# conv. layers	# output dim. GAT	# FC layers	# out dim. FC
73.61%	0.00098	cross entropy loss	3	20	5	39
73.49%	0.00098	cross entropy loss	3	20	5	42
73.33%	0.00094	cross entropy loss	3	20	5	45
73.27%	0.00098	cross entropy loss	3	19	5	41
73.03%	0.00098	cross entropy loss	3	22	5	46
72.99%	0.00098	cross entropy loss	3	20	5	40
72.94%	0.00931	cross entropy loss	4	17	5	17
72.87%	0.00097	cross entropy loss	3	22	5	21
72.81%	0.00688	cross entropy loss	1	110	2	84
72.77%	0.00515	cross entropy loss	1	141	2	76

Table 6 shows the top-10 best performing configurations of our HPO search for the adjusted-model. We note a configuration with significantly less trainable parameters when compared to the configuration found for the base-model described in Section 5.1. This indicates that adding the source node skip connection allows for a simpler configuration while having comparable performance results.

In this section, we set out to answer the following question: Can we improve the state-of-the-art method by leveraging knowledge from the hand-crafted features method? While we found that source node features are among the most important features judging by the feature importance from our random forest model, we conclude that a skip connection from the source node to the global mean pooling layer does not improve the state-of-the-art on cascade-wise classification of fake news.

5.4 Performance on a real-world task

In this section, we answer our main research question: *How does a content-agnostic cascade-wise fake news ranking system perform on a real-world task in terms of top-k precision?* In the previous subsections, we have answered our subquestions and consequently obtained models that are ready to be evaluated on a real-world task.

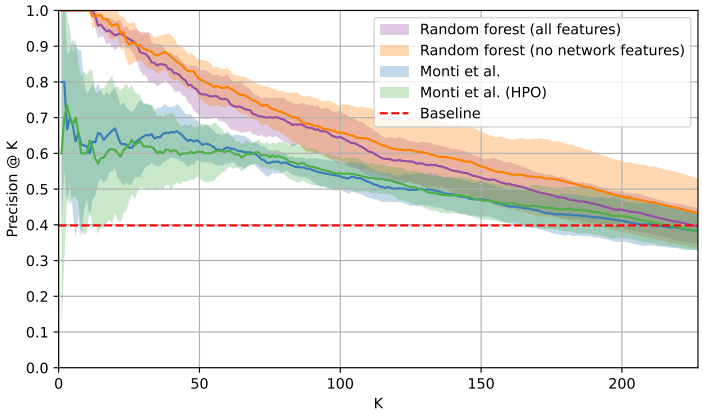


Figure 17: The top-k precision (or precision@k) plot for a subset of the FakeNewsNet test set for all values of k.

5.4.1 Setup

To mimic a real-world task, we evaluate our models on the following two datasets: The first dataset consists of a random sample of 228 cascades from the content-agnostic variant of the FakeNewsNet dataset. The second dataset consists of the 228 cascades from our evaluation set. Our evaluation set is constructed as described in Section 3.3 and thus has an equal class distribution compared to the FakeNewsNet dataset. When evaluating our models on the evaluation set, the models are retrained on the complete FakeNewsNet dataset. When evaluating our models on the random sample of the FakeNewsNet dataset, the models are trained on all other cascades of the FakeNewsNet dataset.

In this evaluation, we include the following models: Random forest with all features, random forest with no structural features, Monti et al. [50], and Monti et al. [50] HPO. We did not include the optimized pipeline obtained with the AutoSklearn 2.0 framework since it showed similar performance to our random forest while significantly increasing the complexity of the approach. The GNN with a skip connection is also not included in this evaluation since it showed no improvement over the GNN without a skip connection.

The performance will be reported in terms of top-k precision, and thus the models will be evaluated on their ability to rank fake news tweets higher than tweets containing true news. By comparing the performance in terms of top-k precision between the FakeNewsNet sample and the evaluation set, we observe the difference between a lab and a real-world setting.

5.4.2 Results

First, we will look at the results on the FakeNewsNet sample dataset, which can be observed in Figure 17. We observe a pattern similar to the results presented in the previous section: all random forest models outperform the GNN-based models. This shows that a model that performs well in terms of AUC also does well in terms of top-k precision. Furthermore, overall solid performance can be noted. The hand-crafted features methods yield a Top-50 precision of 80%, and the GNN-based methods achieve a Top-50 precision of 63%, which is above the random ordering of 40%.

In this work, we set out to evaluate a content-agnostic cascade-wise fake news ranking system on a real-world task. Figure 18 shows the results on the evaluation set, which mimics a real-world task. First of all, we note a considerable gap between the results of the FakeNewsNet dataset presented in Figure 17 and the results obtained from the evaluation set. This gap highlights and supports the gap

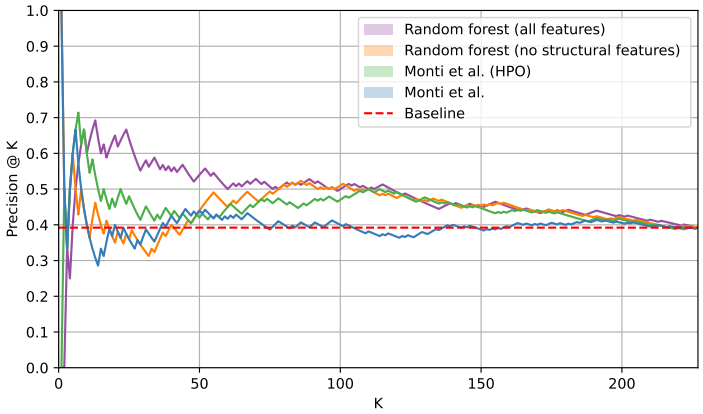


Figure 18: The top-k precision (or precision@k) plot for a evaluation dataset for all values of k.

between academic results and the real world, as mentioned in the introduction of this work. On the one hand, this shows that our approach does indeed mimic a real-world task, while on the other, we note that a content-agnostic cascade-wise ranking approach is still subject to significant performance drops when transferred from a lab setting to the real world.

We note that the random forest model with all features performs significantly better than the random baseline, with a top-50 precision of 55% compared to the baseline of 40%. All other models seem to bearly outperform the random baseline at the top-50 mark. However, if we shift our perspective to the top-100 precision mark, we note that all models, with the exception of the GNN without hyper-parameter optimization, outperform the random baseline by a 10% margin. Thus, even though the results are significantly worse when compared to the FakeNewsNet dataset, they still provide an improvement over the current approach. Because this system is built as a ranking system and not a classification system, every improvement upon the random ordering, which is the current approach, increases a fact-checkers workflow. We, therefore, conclude that, even though the results on a real-world task pale in comparison to the results on the content-agnostic variant of the FakeNewsNet dataset, a content-agnostic cascade-wise fake news ranking system does improve upon the current practice and therefore is a valuable addition to a fact-checkers toolbox.

To better understand the performance of the models on this real-world task, Figure 19 shows the ROC curve of our models on the evaluation set. Again we note that only the random forest with all features can achieve an increase in the true positive rate while keeping the false positive rate relatively low. Indicating that the random forest method outperforms other methods for small values of k. Furthermore, we note an AUC of 64% for the random forest with all features model. This is significantly lower than the AUC of 79% that was achieved on the FakeNewsNet dataset; however, considering the model was trained on another dataset, an AUC of 64% shows potential when deploying the system in the real world.

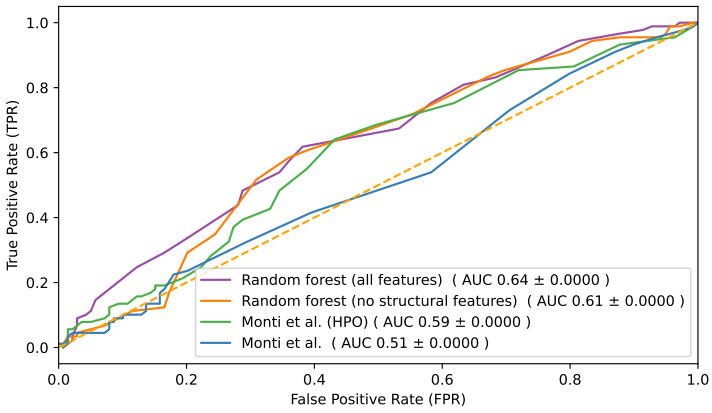


Figure 19: The ROC curves for our models on the evaluation dataset.

6 Discussion and limitations

In this work, we reimplemented the state-of-the-art of cascade-wise classification of fake news on online social media. The state-of-the-art achieved an AUC of 70% on the FakeNewsNet dataset; this is considerably lower than the AUC of 88% reported in the original work, which was achieved on a private dataset. Even though the results fall short compared to the metrics presented in the original work, an optimized variant of the state-of-the-art achieves an AUC of 73% and a top-10 precision of 70% which is well above the random ordering on the content-agnostic variant of the FakeNewsNet dataset.

When analyzing the performance of an optimized machine learning pipeline with hand-crafted features obtained from the AutoSklearn 2.0 framework, we found a significant improvement compared to the state-of-the-art in AUC (81%) and top-10 precision (90%). Furthermore, from the results obtained from our random forest ablation study, we note that adding network features does not yield any additional performance benefits; both random forest models achieved an AUC of 78%. Together with the performance gap between the graph neural network methods, this lack of improvement when adding the network features indicates that in our setup and for the FakeNewsNet dataset, the network features are more a distraction than a source of information regarding the identification of fake news cascades. Additionally, we show that the performance gap between the state-of-the-art and the hand-crafted features methods cannot be bridged by adding a skip connection from the source node to the global mean pooling layer. When comparing the graph neural network without a skip connection with the graph neural network with a skip connection, both methods achieved an AUC of 70%, and their optimized counterparts achieved an AUC of 73%.

We use a larger k for top-k precision on the evaluation dataset, i.e., 50 and 100 rather than 10, since our results on the evaluation dataset show a very volatile top-k precision line for lower values of k. We show a steep decrease in predictive performance when evaluating our models on a real-world task, with our best performing model, i.e., the random forest with all features, dropping from a top-50 precision of 80% to a top-50 precision of 55%. Though the performance falls short compared to the FakeNewsNet test set, all models, except for the non-optimized GNN, achieve a top-100 precision of 50%, which is above the random baseline of 40%. The risk of mislabelling a tweet is mitigated by not providing a fact-checker with direct labels but only a ranking. Furthermore, with the creation of a complete pipeline from raw tweet IDs to a ranking, we believe that the system presented in this work is a valuable addition to a fact-checkers toolkit.

The main limitations to our setup are three-fold. First, in our work, we only use the FakeNewsNet dataset for training. The FakeNewsNet dataset only uses subjects related to political news or celebrity gossip from the United States. Therefore, the data can not be seen as a random sample of all potential fake news on the social media platform. This is problematic since we aim to create a fake news ranking system that is not limited by topics or languages. For our evaluation set, we did change the language and topics. Therefore, having a more diverse dataset and being more aligned with the evaluation task might have resulted in a smaller gap between our training and evaluation results.

Second, our evaluation set is an approximation of a real-world task. Though we believe that by training our models in a different setting and evaluating them on a different dataset, we approximate the performance in a real-world setting, evaluating the models in a *truly* real-world setting would provide us with a complete picture.

Third, our models are trained and optimized using the AUC metric, while they are evaluated with the top-k precision metric. Though we firmly believe that a model which outperforms another model on the AUC metric also outperforms that model on the top-k precision metric, we note that optimizing for a ranking might result in a more effective pipeline. This problem can be alleviated by designing a metric focusing on the ROC curve's first part. Such a metric would eliminate the situation that a model achieves a higher AUC score while performing worse in terms of top-k precision.

7 Conclusion and future work

The increased presence of fake news on online social media, combined with the effect on our society, makes fake news on online social media one of the greatest challenges we face today. As fact-checkers are overwhelmed by the sheer volume of fake news, a more automated approach is required. To bridge the gap between high-performance metrics presented in academic works and an almost non-existent adoption in the real world, we created a content-agnostic cascade-wise fake news ranking system. This approach eliminates the heavy use of *textual features*, avoids the placement of *labels*, and does not leverage the use of *all tweets concerning the same topic* as a single data point.

First, we determined the best-performing fake news detection models by (1) implementing the state-of-the-art on cascade-wise classification of fake news, (2) comparing the state-of-the-art to a hand-crafted features approach, and finally (3) trying to improve upon the state-of-the-art by leveraging the knowledge obtained from the hand-crafted features approach. We found that a simple hand-crafted features approach outperformed the state-of-the-art and that adding a skip connection from the source node to the global mean pooling layer in the state-of-the-art had an insignificant performance improvement. The hand-crafted features approach has the additional benefit of being a machine learning model which is explainable, thus the reasoning of the machine learning system can be communicated back to the fact-checker. Furthermore, we note that the network features, i.e., features associated with the spread of an item, do not add additional predictive power to a model.

After evaluating the models on the benchmark dataset FakeNewsNet, they are evaluated on the evaluation dataset, which mimics a real-world task. This evaluation allowed us to answer our main research question: How does a content-agnostic cascade-wise fake news ranking system perform on a real-world task in terms of top-k precision? We found that the graph neural network approach for fake news detection on online social media, which is regarded as the current state-of-the-art, performs similar to a random ordering on a real-world task in terms of top-k precision. Furthermore, we show that a hand-crafted features random forest method outperforms the state-of-the-art in terms of AUC on the benchmark dataset and top-k precision on the evaluation set. We show that, even though the results on a real-world task pale in comparison to the results on the content-agnostic variant of the FakeNewsNet dataset, a content-agnostic cascade-wise fake news ranking system does improve upon the current practice in terms of top-k precision and therefore is a valuable addition to a fact-checkers toolbox.

With the system created in this work, which includes the complete pipeline from a raw tweet to a ranking, we aim to increase the adoption of automated tools for fake news ranking by fact-checkers and encourage the creation of new datasets that can be used for fake news-related research.

For future work, we suggest the following directions. We believe that evaluating the noise and bias in the FakeNewsNet dataset is a valuable direction for future work. Since most scientific work on fake news detection on online social media uses this dataset, a good understanding of its shortcomings is vital. With our small exploratory analysis of the quality of the FakeNewsNet dataset, we found a significant amount of noise in the labels. A proper quantitative analysis of the noise in the dataset would allow for a better analysis of the results obtained. Furthermore, since the dataset only contains political fake news or celebrity gossip, an in-depth analysis of the bias can be a valuable addition. In our work, and in many others, the dataset is treated as a proxy for fake news on online social media in general, not just political or celebrity focussed. Therefore, having a better understanding of the biases present in the data can create a more usable and robust fake news detection or ranking system.

Also, creating a standalone evaluation set would be an instrumental contribution to the research field of fake news detection on online social media. With the pipeline provided in this work, a researcher or practitioner can easily convert any list of fact-checked tweets into a dataset ready to train and evaluate new fake news detection or ranking methods. Having an evaluation set that consists of a large, diverse set of fact-checked claims on online social media would allow future researchers to quantify the performance of their proposed methods in a real-world scenario. When this evaluation set contains

different topics and languages compared to the training dataset, it could result in future work to focus more on the practical usability of proposed methods and speed up practitioners' adoption.

Another interesting direction for future work is the grouping of similar tweets. It would alleviate the dependency on using cascade-wise classification for a practical approach, thus allowing for a URL-wise-based system to be practically usable. In related work, we observe that URL-wise-based approaches achieve higher performance metrics, which would benefit the efficiency and impact that fact-checkers can have. Furthermore, a grouping of similar tweets system would enable fact-checkers to fact-checker a single item and then collect all related tweets. Thus, again, increasing the efficiency of a fact-checker.

Additionally, creating a better understanding of the risk of adversarial attacks on network and content features is another promising direction for future work. In this work, one of the reasons for eliminating the content-based features was the potential risks of adversarial attacks. However, no in-depth quantitative analysis has been done on the risks of using textual features in fake news detection. This would greatly benefit the research field since it would provide valuable insight into the risks of using textual features.

Finally, an analysis of the true diffusion path of online content is also an open question at the time of writing. Therefore, further exploration in creating propagation graphs is a research direction that would benefit fake news detection on online social media. A better understanding of how an item spreads on online social media could significantly increase our awareness and improve fake news detection methods in general. Furthermore, more sophisticated patterns might be discovered, which could not only detect but prevent fake news from going viral.

References

[1] S. Afroz, M. Brennan, and R. Greenstadt, "Detecting hoaxes, frauds, and deception in writing style online," in *proceedings of 2012 IEEE Symposium on Security and Privacy*, 2012, pp. 461–475. (Cited on p. 5)

[2] B. Berendt, P. Burger, R. Hautekiet, J. Jagers, A. Pleijter, and P. Van Aelst, "Factrank: Developing automated claim detection for dutch-language fact-checkers," *Online Social Networks and Media*, vol. 22, p. 100113, 2021. (Cited on p. 6)

[3] A. Bodaghi and J. Oliveira, "The theater of fake news spreading, who plays which role? a study on real graphs of spreading on twitter," *Expert Systems with Applications*, vol. 189, p. 116110, 2022. (Cited on p. 5)

[4] P. Boldi and S. Vigna, "Axioms for centrality," *Internet Mathematics*, vol. 10, no. 3-4, pp. 222–262, 2014. (Cited on p. 22)

[5] P. Bonacich, "Power and centrality: A family of measures," *American journal of sociology*, vol. 92, no. 5, pp. 1170–1182, 1987. (Cited on p. 22)

[6] P. M. Borges and R. R. Gambarato, "The role of beliefs and behavior on facebook: A semiotic approach to algorithms, fake news, and transmedia journalism," *International Journal of Communication*, vol. 13, p. 16, 2019. (Cited on p. 1)

[7] U. Brandes and D. Fleischer, "Centrality measures based on current flow," in *STACS 2005*. Springer Berlin Heidelberg, 2005, pp. 533–544. (Cited on p. 22)

[8] P. B. Brandtzaeg and A. Følstad, "Trust and distrust in online fact-checking services," *Communications of the ACM*, vol. 60, no. 9, pp. 65–71, 2017. (Cited on p. 4)

[9] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017. (Cited on p. 6)

[10] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122. (Cited on p. 28)

[11] J. M. Burkhardt, "History of fake news," *Library Technology Reports*, vol. 53, no. 8, pp. 5–9, 2017. (Cited on p. 1)

[12] D. Carnahan and D. E. Bergan, "Correcting the misinformed: The effectiveness of fact-checking messages in changing false beliefs," *Political Communication*, vol. 39, no. 2, pp. 166–183, 2022. (Cited on p. 4)

[13] D. Carrion-Alvarez and P. X. Tijerina-Salina, "Fake news in COVID-19: A perspective," *Health promotion perspectives*, vol. 10, no. 4, p. 290, 2020. (Cited on p. 1)

[14] M. Cinelli, G. De Francisci Morales, A. Galeazzi, W. Quattrociocchi, and M. Starnini, "The echo chamber effect on social media," *Proceedings of the National Academy of Sciences*, vol. 118, no. 9, p. e2023301118, 2021. (Cited on p. 4)

[15] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958. (Cited on p. 25)

[16] K. Eggensperger, P. Müller, N. Mallik, M. Feurer, R. Sass, A. Klein, N. Awad, M. Lindauer, and F. Hutter, "HPOBench: A collection of reproducible multi-fidelity benchmark problems for HPO,"

in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. (Cited on p. 10)

[17] C. Eksombatchai, P. Jindal, J. Z. Liu, Y. Liu, R. Sharma, C. Sugnet, M. Ulrich, and J. Leskovec, "Pixie: A system for recommending 3+ billion items to 200+ million users in real-time," in *proceedings of the 2018 World Wide Web Conference*, ser. WWW '18. International World Wide Web Conferences Steering Committee, 2018, pp. 1775–1784. (Cited on p. 7)

[18] S. Falkner, A. Klein, and F. Hutter, "BOHB: Robust and efficient hyperparameter optimization at scale," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 1437–1446. (Cited on p. 9)

[19] P. Ferreira, D. C. Le, and N. Zincir-Heywood, "Exploring feature normalization and temporal information for machine learning based insider threat detection," in *2019 15th International Conference on Network and Service Management (CNSM)*, 2019, pp. 1–7. (Cited on p. 22)

[20] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter, "Auto-sklearn 2.0: The next generation," *CoRR*, vol. abs/2007.04074, 2020. (Cited on p. 11, 28)

[21] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in Neural Information Processing Systems*, vol. 28. Curran Associates, Inc., 2015. (Cited on p. 11)

[22] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social networks*, vol. 1, no. 3, pp. 215–239, 1978. (Cited on p. 22)

[23] C. Gentile and M. K. K. Warmuth, "Linear hinge loss and average margin," in *Advances in Neural Information Processing Systems*, vol. 11. MIT Press, 1998. (Cited on p. 25)

[24] S. Goel, A. Anderson, J. Hofman, and D. J. Watts, "The structural virality of online diffusion," *Management Science*, vol. 62, no. 1, pp. 180–196, 2016. (Cited on p. 22)

[25] D. Graves, "Understanding the promise and limits of automated fact-checking," 2018. (Cited on p. 6)

[26] A. Hannak, D. Margolin, B. Keegan, and I. Weber, "Get back! you don't know me like that: The social mediation of fact checking interventions in twitter conversations," in *Eighth International AAAI Conference on Weblogs and Social Media*, 2014. (Cited on p. 4)

[27] M. Hanussek, M. Blohm, and M. Kintz, "Can automl outperform humans? an evaluation on popular openml datasets using automl benchmark," in *2020 2nd International Conference on Artificial Intelligence, Robotics and Control*, ser. AIRC'20. Association for Computing Machinery, 2020, pp. 29–32. (Cited on p. 9)

[28] K. Hartley and M. K. Vu, "Fighting fake news in the COVID-19 era: Policy insights from an equilibrium model," *Policy Sciences*, vol. 53, no. 4, pp. 735–758, 2020. (Cited on p. 1)

[29] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. (Cited on p. 9)

[30] I. Henkel, "How the laughing, irreverent briton trumped fact-checking: A textual analysis of fake news in british newspaper stories about the eu," *Journalism Education*, vol. 6, no. 3, pp. 87–97, 2018. (Cited on p. 1)

[31] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282. (Cited on p. 28)

[32] J. Huang and C. Ling, "Using AUC and accuracy in evaluating learning algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, 2005. (Cited on p. 12)

[33] E. Inan, "Zoka: A fake news detection method using edge-weighted graph attention network with transfer models," *Neural Computing and Applications*, vol. 34, no. 14, pp. 11 669–11 677, 2022. (Cited on p. 5)

[34] P. Iosifidis and N. Nicoli, "The battle to end fake news: A qualitative content analysis of facebook announcements on how it combats disinformation," *International Communication Gazette*, vol. 82, no. 1, pp. 60–81, 2020. (Cited on p. 1)

[35] K. Jamieson and A. Talwalkar, "Non-stochastic best arm identification and hyperparameter optimization," in *Artificial intelligence and statistics*. PMLR, 2016, pp. 240–248. (Cited on p. 10)

[36] I. Jaradat, P. Gencheva, A. Barrón-Cedeño, L. Màrquez, and P. Nakov, "Claimrank: Detecting check-worthy claims in arabic and english," *arXiv preprint arXiv:1804.07587*, 2018. (Cited on p. 6)

[37] W. Jiang and J. Luo, "Graph neural network for traffic forecasting: A survey," *Expert Systems with Applications*, vol. 207, p. 117921, 2022. (Cited on p. 7)

[38] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," in *International Conference on Learning Representations*, 2017. (Cited on p. 25)

[39] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017. (Cited on p. 25)

[40] A. Kucharski, "Study epidemiology of fake news," *Nature*, vol. 540, no. 7634, pp. 525–525, 2016. (Cited on p. 5)

[41] J. Leblay, I. Manolescu, and X. Tannier, "Computational fact-checking: Problems, state of the art, and perspectives," in *The Web Conference*, 2018. (Cited on p. 6)

[42] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. (Cited on p. 6)

[43] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017. (Cited on p. 10)

[44] M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, D. Deng, C. Benjamins, T. Ruhkopf, R. Sass, and F. Hutter, "SMAC3: a versatile bayesian optimization package for hyperparameter optimization," *Journal of Machine Learning Research*, vol. 23, no. 54, pp. 1–9, 2022. (Cited on p. 10)

[45] J. M. Lobo, A. Jiménez-Valverde, and R. Real, "AUC: A misleading measure of the performance of predictive distribution models," *Global Ecology and Biogeography*, vol. 17, no. 2, pp. 145–151, 2008. (Cited on p. 12)

[46] J. Ma, W. Gao, and K.-F. Wong, "Detect rumors in microblog posts using propagation structure via kernel learning," in *proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2017, pp. 708–717. (Cited on p. 6)

[47] I. Manolescu, "Contentcheck: Content management techniques and tools for fact-checking," *ERCIM News*, 2017. (Cited on p. 6)

[48] A. Mantzarlis, "Fact-checking 101," *Journalism, fake news & disinformation: Handbook for journalism education and training*, pp. 85–100, 2018. (Cited on p. 4, 45)

[49] F. Mohr and J. N. van Rijn, "Learning curves for decision making in supervised machine learning–a survey," *arXiv preprint arXiv:2201.12150*, 2022. (Cited on p. 9)

[50] F. Monti, F. Frasca, D. Eynard, D. Mannion, and M. M. Bronstein, "Fake news detection on social media using geometric deep learning," *arXiv preprint arXiv:1902.06673*, 2019. (Cited on p. 5, 6, 14, 15, 24, 25, 26, 27, 30, 31, 33)

[51] M. F. Mridha, A. J. Keya, M. A. Hamid, M. M. Monowar, and M. S. Rahman, "A comprehensive review on fake news detection with deep learning," *IEEE Access*, vol. 9, pp. 156 151–156 170, 2021. (Cited on p. 17)

[52] N. Newman, R. Fletcher, A. Schulz, S. Andi, C. T. Robertson, and R. K. Nielsen, "Reuters institute digital news report 2021," *Reuters Institute for the study of Journalism*, 2021. (Cited on p. 1)

[53] D. S. Nielsen and R. McConville, "Mumin: A large-scale multilingual multimodal fact-checked misinformation social network dataset," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, 2022, p. 3141–3153. (Cited on p. 16)

[54] E. Nudelman, K. Leyton-Brown, H. H. Hoos, A. Devkar, and Y. Shoham, "Understanding random sat: Beyond the clauses-to-variables ratio," in *International Conference on Principles and Practice of Constraint Programming*. Springer, 2004, pp. 438–452. (Cited on p. 20)

[55] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," 1998. (Cited on p. 22)

[56] V. Pérez-Rosas, B. Kleinberg, A. Lefevre, and R. Mihalcea, "Automatic detection of fake news," in *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, 2018, pp. 3391–3401. (Cited on p. 5)

[57] P. N. Petratos, "Misinformation, disinformation, and fake news: Cyber risks to business," *Business Horizons*, vol. 64, no. 6, pp. 763–774, 2021. (Cited on p. 1)

[58] M. Potthast, J. Kiesel, K. Reinartz, J. Bevendorff, and B. Stein, "A stylometric inquiry into hyperpartisan and fake news," in *proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2018, pp. 231–240. (Cited on p. 5)

[59] H. Rashkin, E. Choi, J. Y. Jang, S. Volkova, and Y. Choi, "Truth of varying shades: Analyzing language in fake news and political fact-checking," in *proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2017, pp. 2931–2937. (Cited on p. 5)

[60] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *International Conference on Learning Representations*, 2018. (Cited on p. 25)

[61] V. Rubin, N. Conroy, Y. Chen, and S. Cornwell, "Fake news or truth? using satirical cues to detect potentially misleading news," in *proceedings of the Second Workshop on Computational Approaches to Deception Detection*. Association for Computational Linguistics, 2016, pp. 7–17. (Cited on p. 5)

[62] N. Ruchansky, S. Seo, and Y. Liu, "CSI: A hybrid deep model for fake news detection," in *proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, ser. CIKM '17. Association for Computing Machinery, 2017, pp. 797–806. (Cited on p. 5)

[63] J. Shin and K. Thorson, "Partisan selective sharing: The biased diffusion of fact-checking messages on social media," *Journal of Communication*, vol. 67, no. 2, pp. 233–255, 2017. (Cited on p. 4)

[64] K. Shu, D. Mahudeswaran, S. Wang, D. Lee, and H. Liu, "FakeNewsNet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media," *Big data*, vol. 8, no. 3, pp. 171–188, 2020. (Cited on p. 14, 16)

[65] K. Shu, D. Mahudeswaran, S. Wang, and H. Liu, "Hierarchical propagation networks for fake news detection: Investigation and exploitation," *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 14, no. 1, pp. 626–637, 2020. (Cited on p. 5)

[66] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake news detection on social media: A data mining perspective," *SIGKDD Explorations Newsletter*, vol. 19, no. 1, pp. 22–36, sep 2017. (Cited on p. 5)

[67] K. Shu, S. Wang, and H. Liu, "Exploiting tri-relationship for fake news detection," *CoRR*, vol. abs/1712.07709, 2017. (Cited on p. 5)

[68] A. Silva, Y. Han, L. Luo, S. Karunasekera, and C. Leckie, "Propagation2vec: Embedding partial propagation networks for explainable fake news early detection," *Information Processing & Management*, vol. 58, no. 5, p. 102618, 2021. (Cited on p. 5)

[69] J. M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N. M. Donghia, C. R. MacNair, S. French, L. A. Carfrae, Z. Bloom-Ackermann, V. M. Tran, A. Chiappino-Pepe, A. H. Badran, I. W. Andrews, E. J. Chory, G. M. Church, E. D. Brown, T. S. Jaakkola, R. Barzilay, and J. J. Collins, "A deep learning approach to antibiotic discovery," *Cell*, vol. 180, no. 4, pp. 688–702.e13, 2020. (Cited on p. 7)

[70] E. Tacchini, G. Ballarin, M. L. D. Vedova, S. Moret, and L. de Alfaro, "Some like it hoax: Automated fake news detection in social networks," *CoRR*, vol. abs/1704.07506, 2017. (Cited on p. 5)

[71] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-weka: Combined selection and hyperparameter optimization of classification algorithms," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 847–855. (Cited on p. 9)

[72] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *stat*, vol. 1050, p. 20, 2017. (Cited on p. 8)

[73] S. Vosoughi, D. Roy, and S. Aral, "The spread of true and false news online," *Science*, vol. 359, no. 6380, pp. 1146–1151, 2018. (Cited on p. 5, 17, 20)

[74] Z. Wang, J. Chen, and H. Chen, "EGAT: Edge-featured graph attention network," in *Artificial Neural Networks and Machine Learning – ICANN 2021.* Springer International Publishing, 2021, pp. 253–264. (Cited on p. 9)

[75] Z. Wu, D. Pi, J. Chen, M. Xie, and J. Cao, "Rumor detection based on propagation graph neural network with attention mechanism," *Expert systems with applications*, vol. 158, p. 113595, 2020. (Cited on p. 5)

[76] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "SATzilla-07: the design and analysis of an algorithm portfolio for sat," in *International Conference on Principles and Practice of Constraint Programming.* Springer, 2007, pp. 712–727. (Cited on p. 20)

[77] Z. Zhou, H. Guan, M. M. Bhat, and J. Hsu, "Fake news detection via nlp is vulnerable to adversarial attacks," *arXiv preprint arXiv:1901.09657*, 2019. (Cited on p. 1)

[78] P. Zola, G. Cola, M. Mazza, and M. Tesconi, "Interaction strength analysis to model retweet cascade graphs," *Applied Sciences*, vol. 10, no. 23, 2020. (Cited on p. 18)

A FakeNewsNet data quality

Table 7: Quality test of the FakeNewsNet dataset. For a random sample of 29 tweets from the FakeNewsNet dataset the veracity was manually checked. Of the 29 tweets we observed 7 cases (marked in bold) where the veracity does not resonate with the label present in the FakeNewsNet dataset.

TweetID	Label	True label
6250314634	True	True
6253632931	True	True
6503638505	Fake	Fake
7864238024	True	True
8085788358	True	True
9028658446	True	True
12644835516	Fake	Fake
12644835598	Fake	Fake
15188963167	Fake	Fake
15862822209	True	True
16868713069	Fake	Fake
18404282494	True	True
21744967247	True	True
21944644056	**Fake**	**True**
22371405872	True	True
22921381200	**Fake**	**True**
23937065774	True	True
28421922132	True	True
29617785307	True	True
29691289818	True	True
2032927370125312	**Fake**	**True**
2062949267017728	True	True
6751270341709827	True	True
8096627243876352	True	True
10206220807835648	**Fake**	**True**
11676528156868608	True	True
12940669131694080	True	True
22056508313305088	**Fake**	**True**
23420327124017152	**Fake**	**True**
23421138482757632	**Fake**	**True**

In order to get a better understanding of the quality of the FakeNewsNet dataset, we conducted a small exploratory analysis. First, we grabbed a random sample of 35 tweets. Of the 35 tweets in our random sample, 6 tweets were no longer available on Twitter. Thus those were removed from our sample. For the remaining 29 tweets, we verified their veracity by following the procedures of fact-checking explained by Mantzarlis [48]. We found that 7 tweets had a veracity that did not align with the label present in the FakeNewsNet dataset, for all misaligned tweets we found that the tweet does not match up with the story. We hypothesis that this is due to someone in the comments mentioning a fact-checker story, while the original tweet is unrelated to that story. Thus we observe a $\frac{7}{29} \times 100 = 24.14\%$ error rate in the dataset. The list of tweets used in this analysis, with the exception of the deleted tweets, is shown in Table 7. The tweet ID can be used to find the tweet on Twitter by appending the tweet to the following link: https://www.twitter.com/placeholder/status/.