A Self-attention Mechanism based Model for Early Detection of Fake News

by

Bahman Jamshidi

Bachelor of Software Engineering, Technical and Vocational University, 2008

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Computer Science

In the Graduate Academic Unit of Computer Science

Supervisor(s):	Saqib Hakak, Ph.D., Computer Science
Examining Board:	Hung Cao, Ph.D., Computer Science, Chair
	Roozbeh Razavi-Far, Ph.D., Computer Science
	Clodualdo Aranas, Ph.D., Mechanical Engineering

This thesis is accepted by the
Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

February, 2023

Abstract

Extensive studies have indicated that fake news has become one of the major threats to our social system (e.g., influencing public opinion, financial markets, journalism, and health system), and its impact cannot be understated, particularly in our current socially and digitally connected society. In the past years, this problem has been investigated from different perspectives and various disciplines, such as computer science, political science, information science, and linguistics. Even though such efforts have proposed many helpful solutions, it remains challenging to detect fake news in its early phases of dissemination. Based on previously-reported studies, detecting fake news early after its propagation is a very tough task due to the unavailability of context-based features within the first hours of spreading and the ineffectiveness of merely content-based features methods. To address this challenge, we propose a new framework for detecting fake news in the early stages of its propagation. The first three components of the proposed framework convert each news article's propagation network into a sequence of nodes after preprocessing and feature extraction. The last module of our framework leverages a self-attention mechanism based encoder. Self-attention technique is the core of the well-known Transformer model, which has achieved promising results in different areas, especially in complex tasks like language translation. In this module, a new representation of the input sequence is generated, which is mapped to a label for the news article in the proposed model's last layer. We evaluated our method on two datasets to show its effectiveness. The

achieved F1 scores by the proposed model on GossipCop and PolitiFact datasets are higher than the best baseline model by 9% and 6%, respectively.

Dedication

I dedicate my thesis to my mother.

Acknowledgements

I sincerely thank my professor Dr. Saqib Hakak, for his continuous support and guidance. He has motivated and encouraged me throughout the process.

I would like to thank the University of New Brunswick, Faculty of Computer Science, for the opportunity and support and all my Professors who taught me during the program.

Table of Contents

List of Tables

List of Figures

Abbreviations

NLP	Natural Language Processing
ML	Machine Learning
DL	Deep Learning
SVM	Support Vector Machine
DT	Decision Tree
DTC	Decision Tree Classifier
J48	A statistical decision tree algorithm based on ID3 and C4.5
OneR	One Rule
NB	Naïve Bayes
BNB	Bernoulli Naïve Bayes
MNB	Multinomial Naïve Bayes
RF	Random Forest
XGBoost	eXtreme Gradient Boosting
AdaBoost	Adaptive Boosting
SMO	Sequential Minimal Optimization
KNN	K-Nearest Neighbors
POS	Part Of Speech
TF	Term Frequency
TF-IDF	Term Frequency–Inverse Document Frequency
NN	Neural Network
MLP	Multi Layer Perceptron
CNN	Convolutional Neural Network
GAN-printR	GAN-fingerprint Removal
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
BLSTM	Bidirectional Long Short-Term Memory
CSI	Capture Score Integrate
LIWC	Linguistic Inquiry and Word Count
NELA	News Landscape
LDA	Latent Dirichlet Allocation
BERT	Bidirectional Encoder Representations from Transformers
GPT-3	Generative Pre-trained Transformer 3

Chapter 1

Introduction

1.1 Introduction

Social networks such as Twitter, Facebook, and other similar tools provide their users with a rich and convenient platform for receiving news and sharing them with others. However, the excellent features (e.g., convenience, quickness, and free) and a large community of users can motivate malicious users to take advantage of such platforms to spread unreliable news. For example, Facebook removed over seven million posts merely in the second quarter of 2020 that mostly were misinformation or unverified news about COVID-19[1].

Our society can be affected by fake news in different ways [10, 19, 4]. For example, research conducted by CHEQ, a University of Baltimore economist, artificial intelligence (AI), and cybersecurity company, has shown the amount of monetary damage caused by false information spreading websites annually[2]. The finding of this research is depicted in Fig. 1.1. As seen, around $39 billion is lost merely in the stock market, and the total economic loss is almost $78 billion. Similarly, fake news

[1] https://www.nbcnews.com/tech/tech-news/facebook-removes-seven-million-posts-sharing-false-information-coronav-rcna77

[2] https://www.odwyerpr.com/story/public/13448/2019-11-26/cost-fake-news-78-billion.html

can impact the health sector (e.g., the COVID-19 vaccine) [21, 72], political affairs (e.g., the 2016 US Presidential Election) [5, 16], journalism and democracy [48], and other social/political sectors. Note that the propagation of such news is done mainly on social networks. More than 40% of visits to websites that spread fake news is driven by links on social media platforms such as Facebook, Instagram, and Twitter [2]. This is partly due to the widespread availability and rapid dissemination of these links on these platforms [50].

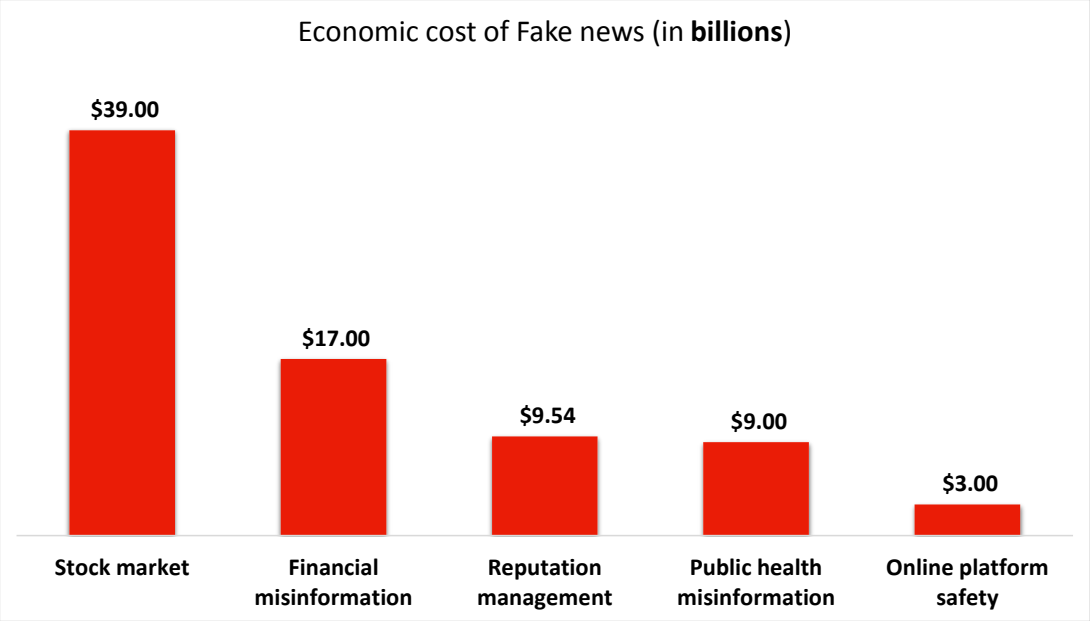

Figure 1.1: Potential economical impacts.

Hence, it is essential to design solutions for detecting and mitigating fake news dissemination. Even though there are fact-checking websites like PolitiFact[3] for detecting fake news, it is impossible to solve this issue using manual methods as they require a large number of experts, and due to the large volume of fake news, they cannot be classified in a timely manner. Therefore, automatic detection approaches based on AI(machine learning and deep learning models) and NLP can be effective as they are extremely fast and less expensive compared to manual solutions. Additionally, fake news must be detected before it becomes widely circulated, as it

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

can be difficult to change people's perceptions once they have been formed, even if those perceptions are based on incorrect information [44]. There have been many attempts to provide automated solutions for detecting fake news early using content [40, 64, 25, 68, 74], social context [24, 59, 67, 55, 56], approaches based on NLP methods [27, 74], etc. However, one of the main limitations of existing approaches is that the detection deadline to detect fake news early takes more time. In this thesis, we present a new self-attention mechanism based model that uses different features extracted from news articles' content and their social context. It detects fake news in a lesser amount of time compared to the state-of-the-art.

1.2 Definition of Fake News

Fake news refers to content that is intentionally created to deceive or mislead users [53]. It may be used to harm the reputation of individuals, groups, organizations, or governments by spreading false information.

In this work, we define Fake news as verifiable false news published intentionally by a news outlet [75]. Also, similar definitions for fake news have been used in [56, 49, 38, 46].

1.3 Problem Statement

Let $R = \{r_1, r_2, ..., r_n\}$ be a set of news instances. $r \in R$ is formed as a tuple $\langle t^r, C^r, P^r, y^r \rangle$, where (1) t^r is the posting time of r in online social networks for the first time; (2) C^r is the content (text, photo, video, etc.) of r; (3) P^r is the propagation network of r constructed within $t^r + \Delta t$(detection deadline); (4) y^r is the binary label of r (1 for fake news, 0 for real news).

Each propagation network $P^r = \{s_1^r, s_2^r, ..., s_m^r\}$ is a set of cascades where $s_j^r \in P^r$ is represented again as a set of nodes $\{N_1^{rj}, N_2^{rj}, ..., N_k^{rj}\}$. Nodes are different types

of user reactions, such as a tweet, re-tweet, reply, etc., towards r. As shown in Fig. 1.2, when all cascades of a propagation network are connected to a central point, it looks like a tree. The creation of the propagation tree and selected features for each node are explained in detail in the next sections.

Given an unseen news article r, its content C^r, and its propagation network P^r created during $t^r + \Delta t$, the problem is to assign the label y^r.

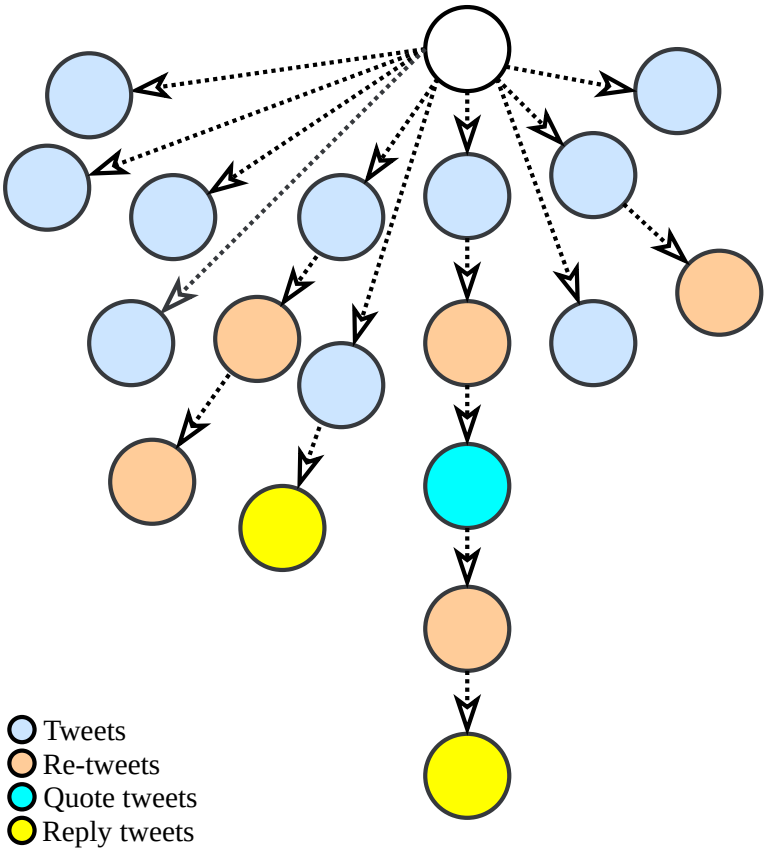

Figure 1.2: Propagation tree of a news article with multiple cascades having several nodes with different types, i.e., tweets, re-tweets, quote tweets, and reply tweets (depicted by different colors).

1.4 Summary of Contributions

The main purpose of this research is to detect fake news on social networks early. In summary, the following are the contributions of this thesis:

1. We introduce a new propagation network-based model which can detect fake news considering the corresponding initial reactions of users on social networks.

2. We test the proposed model on two widely-used available datasets. The evaluation results show that our model can detect fake news much early and more accurately than the state-of-the-art models.

3. We explain the differences between our model and the state-of-the-art baseline models and how it can perform well in the early stages of fake news dissemination with the help of the self-attention mechanism and a practical design for the input sequence.

1.5 Thesis Organization

The rest of the thesis is organized as follows:

1. Chapter 2. *Literature Review* discusses the most recent works on fake news detection into seven detection categories. This chapter also elaborates on the specifications of the fake news features from multiple points of view. Also, it presents fundamental concepts of fake news, such as its types and the life cycle of fake news.

2. Chapter 3: *Proposed Method* discusses the different categories of used features, provides a detailed discussion of the model's architecture, and explains each of the main components.

3. Chapter 4: *Experiments & Results* represents the results of the research implementation, and in this chapter, we discuss the dataset, the experimental setup, the metrics, and the results of the proposed approach.

4. Chapter 5: *Conclusion & Future Work* concludes the thesis with the contribution summary, challenges, and some remarks on future work.

Chapter 2

Literature Review

2.1 Overview

In this chapter, we discuss the fundamental concepts of fake news, such as its life cycle, types, and features from different perspectives. Then, we organize fake news detection models and techniques explored in the studied papers into categories and subcategories to understand their differences and similarities better. After that, the most recent research on fake news detection is summarized and organized into seven categories based on the main focus of the research. Also, we provide a taxonomy of these studies and a table including the details of each category.

2.2 Fundamentals of Fake News

In this section, we discuss the fundamental concepts of fake news. The major fundamental concepts discussed are the life cycle, types, and features of fake news.

2.2.1 Life Cycle of Fake News

An example life cycle of fake news is presented in Fig. 2.1, which explains how fake news can be generated and propagated through various platforms. Motivations for

such generation and dissemination vary, ranging from financial (e.g., monetary) to political (e.g., left/right wing) to terrorism (e.g., seeking to create societal panic/unrest) to defamation and so on. Fake news can also exist in different formats, such as text, image, audio, and video.

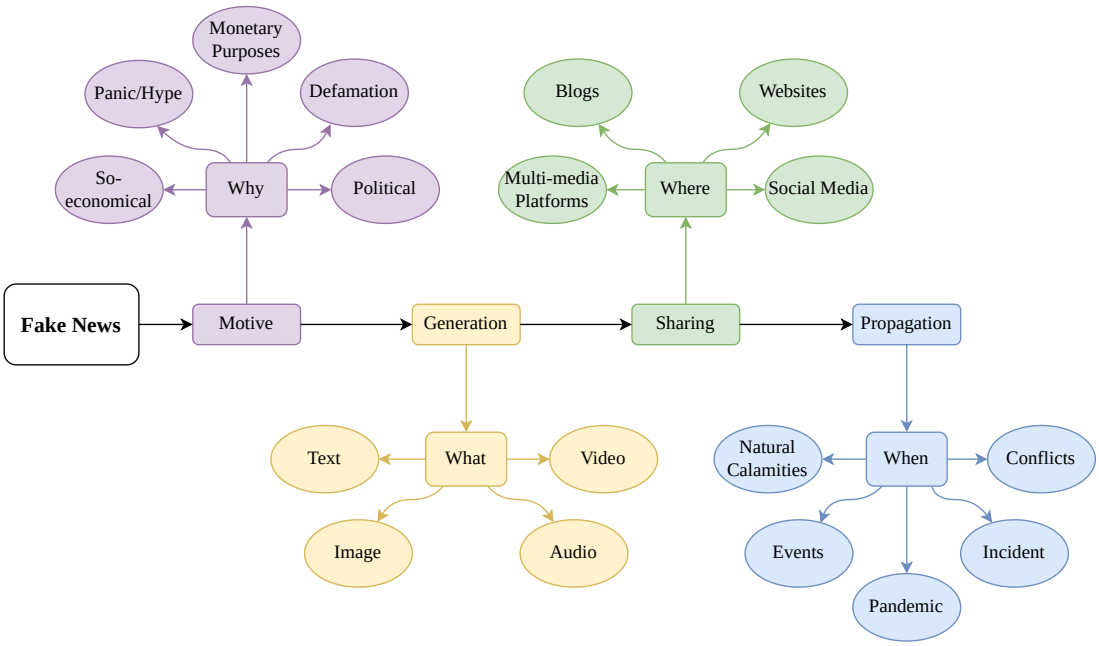

Figure 2.1: Fake news life cycle.

2.2.2 Types of Fake News

There are various types of fake news, including stance, satire, multi-modal, deep fake, and disinformation. Stance can be divided into four categories: agree, disagree, discuss, and unrelated [61]. An agreeing stance is consistent with the headline of fake news, while a disagreeing stance presents information that conflicts with the headline. Satire is a form of fake news that uses humor and mockery to convey a political message or criticism [3]. It often employs sarcastic language and is meant to be humorous. Multi-modal fake news refers to using multiple mediums to spread false information, including videos, images, audio, and text [58]. Deep fake is a form of fake news that uses deep learning techniques to generate manipulated video clips,

images, and recordings [17]. According to a startup called Deep Trace, the number of deep fake videos increased from 7964 in 2019 to twice that amount within just nine months, and this trend is expected to continue to grow rapidly [29]. Disinformation is false and misleading information that is disseminated with the intention of deceiving people [60]. It can have serious social and political consequences, as sources may spread manipulated information to achieve political goals or create chaos in society [15].

2.2.3 Different Aspects of Features Used in Detecting Fake News

Based on our study, the researchers have put much emphasis on feature identification in detecting fake news. Features have played the most important role in classification models specifically in fake news detection as real and fake classes have very similar characteristics. We can look at the features from the following points of view:

1. *Being imitated:* Some features are difficult to be mimicked by malicious users (topographical features), while most of them can be imitated easily.

2. *Sensitive to time:* For instance, [27] shows that Word n-gram features may provide information which is less relevant at that point of time.

3. *Required level of computational resources:* Some features are readily available, while some others like layer ratio in [71] need processing sources to calculate.

4. *Applicability:* Some features cannot be applied to all situations and platforms. For example, domain reputation related features can not be utilized in social network platforms.

5. *Necessary time to emerge:* As an example, reactions of other users to a news article need a time period to be revealed.

6. *Explainable:* Features achieved by deep learning methods act like a black-box and are not interpretable in contrast to hand-crafted ones.

7. *Pre-processing steps:* Unlike features such as n-grams, some characteristics like user profile based ones do not need much pre-processing stages.

8. *Potential to be transferred:* Pre-trained word embedding can be used and updated in new situations.

9. *Source:* Features can be extracted from different sources such as contextual content, images, videos, profiles, etc.

10. *Required size of corpus:* Features gained from LIWC, NELA, different embedding need large corpus.

11. *Language-independent:* Features such as word embedding, word-grams, character-grams are dependent on the language, while profile based are independent.

12. *Domain-independent:* Most of the content-based features and features extracted from images cannot apply to all domains. To illustrate, vectors gained from word embedding need a corpus that contains sufficient data for the underlying domain.

2.3 General Methods to Detect Fake News

In the studied papers, a large number of models and techniques have been used to do pre-processing steps and detect fake news. In this section, we present them in different categories and subcategories to understand their differences and similarities. Figure 2.2 shows all the methods used by the studies chosen for this research categorized into two main detection techniques, Machine Learning (ML) and Deep

Learning (DL). Also, Figure 2.3 represents all reviewed preprocessing techniques applicable in Natural Language Processing (NLP). ML is a field of artificial intelligence that involves training algorithms to learn patterns and relationships in data to make decisions or predictions based on new, unseen data [37]. DL is a subfield of machine learning that uses artificial neural networks with multiple layers to learn patterns and relationships in data. It is particularly effective at tasks that require processing large amounts of unstructured data, such as image and speech recognition [31]. NLP is a field that combines linguistics, computer science, and artificial intelligence to study how computers can understand and use human language. NLP has many applications, including language translation, chatbots, and text classification [28]. It can be seen that the researchers have spent a significant time in machine learning models and have now started to explore deep learning models to build a robust fake news detector.

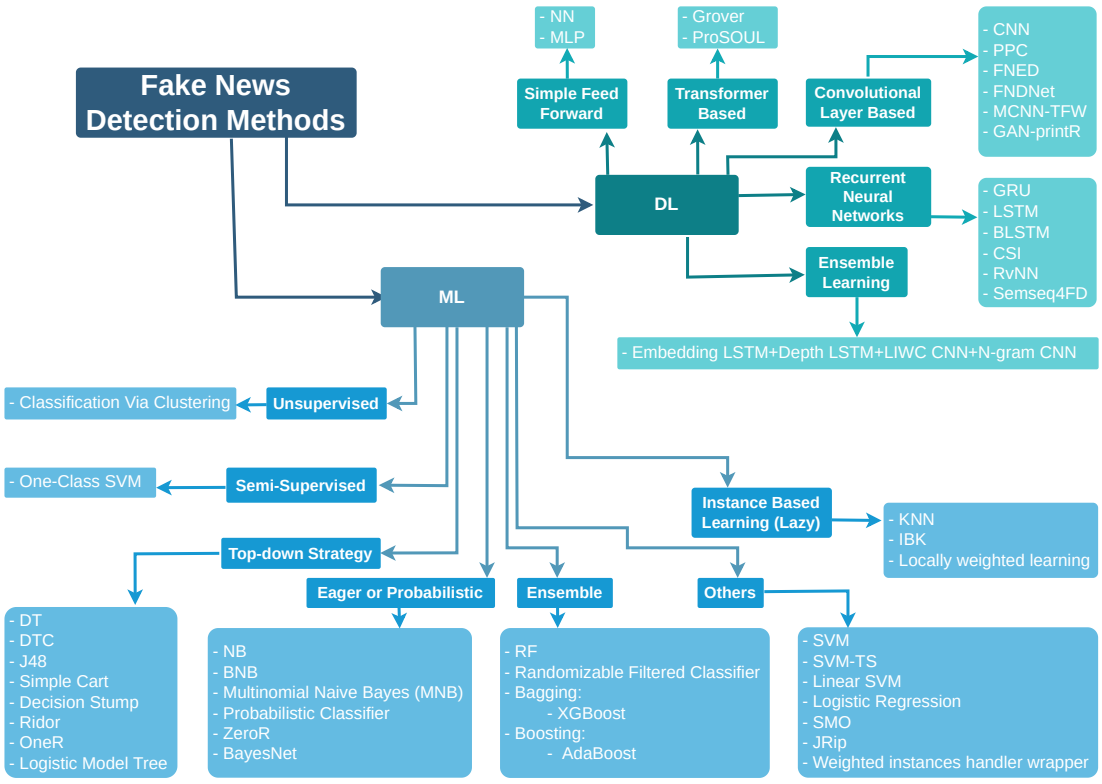

Figure 2.2: Detection methods used in literature to predict fake news categorized under DL and ML. Each of these two categories are divided into smaller groups.

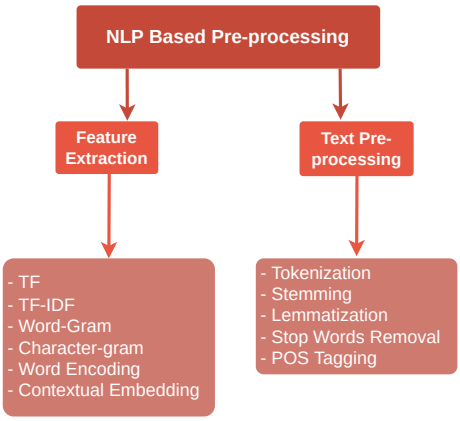

Figure 2.3: NLP based pre-processing methods used in literature.

2.4 State-of-the-art Fake News Detection

In this section, we briefly summarize the most recent works on fake news detection into seven detection categories, as shown in Fig. 2.4. The classification is done in four layers. In the first layer, the studies are sorted based on the focus of the research. Each color code represents one detection-based research focus and based on that, we have divided the work of researchers based on the type of fake news content in layer 2, fake news features in layer 3, and dataset categories in layer 4, respectively.

We have described the datasets used in these studies in Tables 2.1 - 2.7. In the following sections, the summaries of the existing works are briefly highlighted.

2.4.1 Automatic Detection

Studies proposing models that automatically capture discriminatory features from fake news and classify the news based on the hidden layers of the deep learning model comes under the category of Automatic Detection.

Alatas et al. [41] used a two-step method for fake news detection based on converting unstructured data into a structured dataset and then implementing twenty-three supervised artificial intelligence algorithms on the structured dataset by text mining methods.

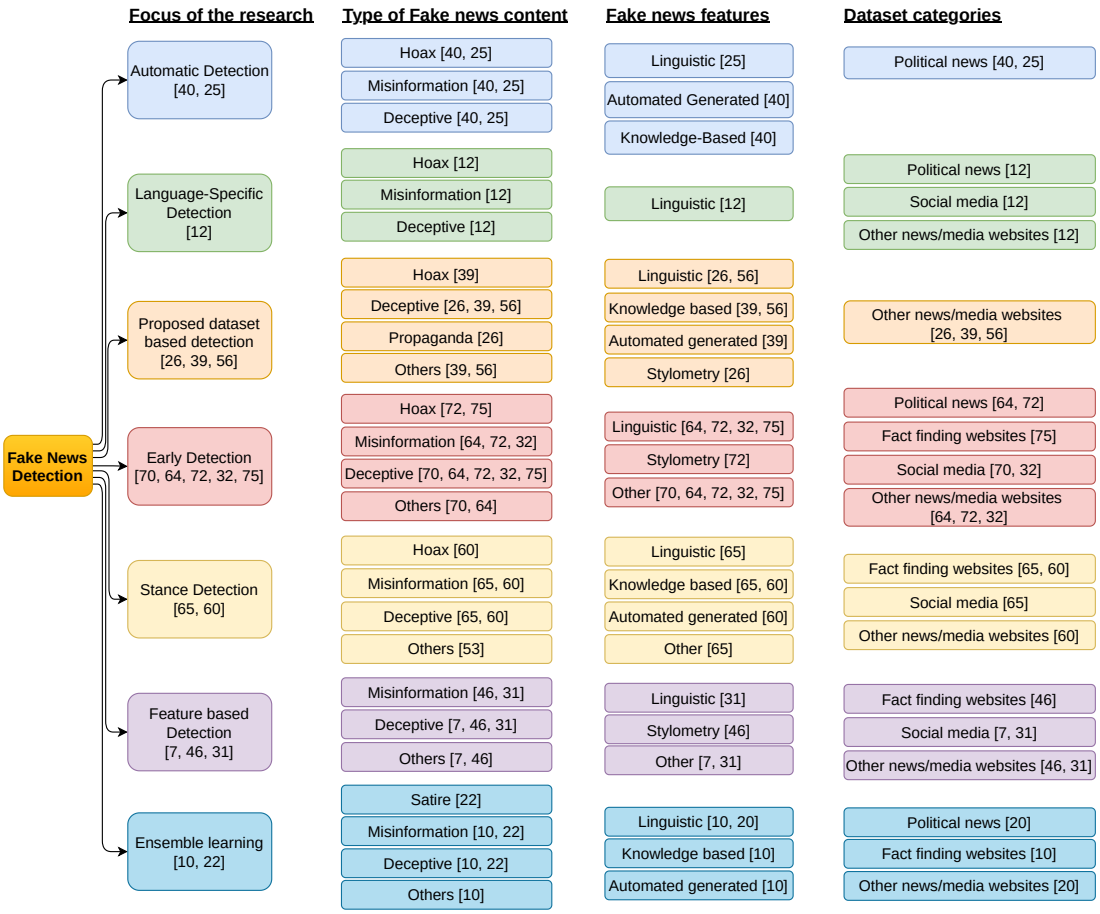

Figure 2.4: Taxonomy of fake news detection

To detect fake news, Kaliyar et al. [26] used a deep convolutional neural network (FNDNet). The proposed model is designed to automatically detect the features of fake news and differentiate them from those of real news. Moreover, the study leverages binary class datasets instead of using a hybrid approach. The hybrid approach detects fake or real news based on the combination of the news content, its context, and temporal level information and can create more impact when using multi-label datasets. In this way, the use of binary class dataset results in limiting the analysis and generalization of results obtained from a broader perspective.

Table 2.1: Qualitative Analysis of Datasets in Automatic Detection Schemes

Study	Year	Dataset used	Limitation	Contribution	Scope
[41]	2019	BuzzFeed Political News dataset	Limited dataset	Twenty-three supervised artificial intelligence algorithms adapted for the first time	Hybrid Political news containing persuasive information
		ISOT Fake News dataset	Needs transformation of textual data into a more suitable representation for analysis		Hybrid Political news containing real and fake news
		Random Political News dataset	Redundant and unrelated features must be removed from text mining for better accuracy		Hybrid political news containing real and fake news and misreporting
[26]	2020	BuzzFeed Political News dataset	large dataset for training	Better and competitive results with text-based dataset	Hybrid Political news containing persuasive information
		Random Political News dataset	use of binary dataset instead of hybrid		Hybrid political news containing real and fake news and misreporting
		ISOT Fake News dataset			Hybrid Political news containing real and fake news

2.4.2 Language-specific Detection

Language-specific detection defines all those papers dealing in languages other than or more than just English while incorporating multiple language datasets to test their detection model.

Henrique et al. [13] has proposed a text-feature-based, language-independent methodology for fake news detection. The generated text is independent of the source platform and language. The proposed methodology was evaluated on five datasets in three language groups giving satisfactory results in comparison to the benchmark. Four algorithms are applied in each set i.e., KNN, random forest, Gaussian Naïve Bayes, and SVM. Out of these four, the best results were obtained by applying Random Forest and SVM algorithms. Random Forest gave the highest F1 score of 95% and attained the highest accuracy of 95%.

Table 2.2: Qualitative Analysis of Datasets in Language-Specific Detection Techniques

Study	Year	Dataset used	Limitation	Contribution	Scope
[13]	2020	FakeBrCorpus	Nonavailability of large dataset	Proposed a text-feature based, language independent methodology for fake news detection	Mix news containing fake news and persuasive information
		TwitterBR			Social media news containing Citizen Journalism
		FakeOrRealNews			Hybrid Political news containing real and fake news
		FakeNewsDataset			Political news containing fake news
		TwitterBR LTO			Social media news containing Commentary, Feature Writing and Citizen Journalism

2.4.3 Dataset-based Detection

Studies that contribute majorly towards presenting datasets that can be widely used by other researchers to test their proposed detection model comes under the category of dataset-based detection. While other researchers use only publicly identified datasets, these studies have made their own dataset to detect fake news as part of their contribution.

Neves et al. [40] introduce a novel approach (GAN-fingerprint Removal autoencoder - GANprintR) to spoof facial manipulation detection systems. This approach removes the GAN fingerprints without compromising the image quality. Thus, machines, like humans, will not be able to distinguish fake images from real ones. GANprintR is trained with face images of real persons instead of synthetic face images with GAN-fingerprint. To carry out this study, three state-of-the-art manipulation detection approaches are used: *XceptionNet*[7], *Steganalysis*[39] and *Local Artifacts*[36]. Also, three different scenarios are designed: 1) *controlled scenarios* 2) *in-the-wild scenarios* 3) *GAN-fingerprint removal*. From this study, it can be concluded that as long as facial manipulation detection systems show poor performance, it is unfeasible to detect fake news from real ones by comparing their images.

Contents with biased messages, intentional or unintentional, are defined as propaganda [12]. In [27], a framework is proposed for propaganda Spotting in Online Urdu Language (ProSOUL). Due to the lack of dataset and LIWC dictionary in Urdu language, authors developed a labeled dataset via translating the English dataset of QCRI' s propaganda (Qprop) and also an English LIWC dictionary into Urdu language. The main contribution of this paper is analyzing several different feature sets, so following preprocessing steps, they extract NEws LAndscape (NELA) features to analyze news articles from stylistic and psycho-linguistic perspectives.

To address the issue of fake news detection in languages other than English, Silva et al. [57] formulated a semi-automated corpus named "Fake.Br" to overcome this barrier of language. This study answers the undiscussed areas of fake news detection like what set of features must be included for automatic fake news detection and the best classification strategies for this purpose.

2.4.4 Early Detection

The Early Detection category contains all those researches that focus primarily on the early detection and propagation of fake news and to efficiently learn by means of the classifier when the news is first posted on the Internet.

Wang et al. [65] has presented SemSeq4FD which is a graph-based neural network model for the task of fake news detection. The model is designed to detect fake news early and is based on enhanced text representation.

Zhou et al. [73] leveraged established social and psychological theories and supervised classification for the automatic detection of fake news. The model was developed for the early detection of fake news to prevent its propagation on social media platforms. However, as in many previous studies, the experimental analysis was limited to only text-based news articles.

Liu et al. [33] used deep learning for the early detection of fake news. The major

Table 2.3: Qualitative Analysis of Datasets in Proposed data-set based detection schemes

Study	Year	Dataset used	Limitation	Contribution	Scope
[57]	2020	Fake. Br (Brazilian Portuguese)	Nonavailability of noisy document to train the model	Comprehensive analysis on best feature and best machine learning method for fake news detection	Mix news containing fake news and persuasive information
				Formulated a reference corpus for real and fake news in Portuguese language	
[27]	2020	Qprop and its translation into Urdu Language	Translating a dataset might have a negative impact on the results.	Developing a new dataset in Urdu language by translating Qprop from English	Mix news containing Commentary
				Developing a LIWC for the Urdu language by translating the original English LIWC	
				Analyzing several different feature sets extracted from datasets	
[40]	2020	CASIA-WebFace	It is about only face images and neglect other sorts of images.	Developing iFakeFaceDB dataset with the help of GANprintR approach.	Real images
		VGGFace2		proposing an approach to spoof facial manipulation detection systems	Real images
		TPDNE			Fake and edited face images
		100K-Faces			Fake photos of male models
		PGGAN			Fake mix face images
		iFakeFaceDB			Fake and edited face images

components of the model include a feature extractor, a CNN-based news classifier, and a Positive Unlabeled (PU) learning framework. The datasets used in this study described in [34, 35] were generated from Twitter and Weibo.

Zubiaga & Jiang [76] proposed a method for hoax detection in social media by using a free collaborative knowledge base and class-specific word embedding. A logistic regression classifier was used to measure the effectiveness of the class-specific word representations.

Zhao et al. [71] applied a network analysis method to understand the distinguishing features between real news and fake news based on their propagation network. The main research goal was to study the evolution and topology of the network propa-

gation in different settings. The experimental analysis focused on the network with the largest component and topological features such as the shape of a network.

2.4.5 Stance Detection

In Stance Detection, we have categorized studies that have reflected on the phenomenon of understanding the fake news while checking the stance of all news reporters that are reporting about the same incident in other mediums and utilizing the stance to build a stance detector.

In order to effectively detect, describe, and model fake news spread on online social media [46], Xu et al. [66] proposed a framework that characterizes the Web sites and reputations of the fake and real news publishers, analyzes the similarity between the real and fake news through TF-IDF and Latent Dirichlet Allocation (LDA) topic modeling, and also with the help of Jaccard similarity measures, document similarity between real, fake and hybrid news articles are analyzed. The experiments are conducted on the BuzzFeed News dataset [1].

In this article [61], the authors have proposed a stance detection-based approach to detect fake news. They have focused on determining the relative relevance between the headline and the news body based on four stance labels i.e., either agree, disagree, unrelated, or discuss.

2.4.6 Feature-based Detection

Almost all reviewed studies focus on extracting the features of fake news and then utilize them to train their classifier, however, some of the researchers have worked on novel and newer features extraction like topological features, and semantic features as a major contribution to their research. It has helped these researchers make a robust detection model, these studies have been sorted under the category of Feature-based detection where as a major part of contribution the authors have presented various

Table 2.4: Qualitative Analysis of Datasets in Early Detection Techniques

Study	Year	Dataset used	Limitation	Contribution	Scope
[71]	2020	Weibo dataset	Five hours to detect fake news cannot be considered as early detection.	Proposing a technique to design propagation networks of fake and real news	Mix news containing Commentary, Feature Writing, persuasive information and fake news
		Twitter datasec	The propagation networks are applicable solely on platforms like Twitter.	studying topological features	Social media news containing real and fake news
[65]	2020	Weibo	Performance of model decreases with lengthy news	Global semantic relationship and local sequential are included	Mix news containing Commentary, Feature Writing, persuasive information and fake news
		RCED	Less accurate than BERT model	Enhanced sentence representation using self-attention graph convolutional network and 1D cnn	Social media news containing Commentary, Feature Writing, persuasive information, real news and citizen journalism
		LUN		global sequential order document representation	Hybrid political news containing Commentary, Feature Writing, persuasive information, real news and citizen journalism
		SLN		using four datasets in two languages	Hybrid political news containing Commentary, Feature Writing and persuasive information
[73]	2020	BuzzFeed	The features are extracted solely from contextual content of news while images and videos can also be used.	Proposing a theory-driven model for fake news detection	Hybrid Political news containing persuasive information
				Investigating news content at lexicon-level, syntax-level, semantic-level, and discourse-level	
				Analyzing the relationships among fake news, deception/ disinformation, and click baits	
[33]	2020	Weibo	Representing a news article with k status-sensitive crowd responses is not applicable for websites	Proposing a status-sensitive crowd response feature extractor	Mix news containing Commentary, Feature Writing, persuasive information and fake news
		Twitter	the current publicly accessible experimental datasets adopted are small	Using a position-aware attention mechanism	Social media news containing real and fake news
				Applying a multi-region mean-pooling mechanism	
				Investigate the application of PU-Learning algorithm	
[76]	2020	Twitter	Existing datasets pose limitations when one wants to investigate the earliness of veracity classification models	Proposing a semi-automated method to build a large-scale dataset	Fake and real death reports
			Proposed model is limited to a specific kind of hoax triggered by death reports	Leveraging class-specific word embeddings to represent instances	

Table 2.5: Qualitative Analysis of Datasets in Stance Detection Studies

Study	Year	Dataset used	Limitation	Contribution	Scope
[66]	2020	BuzzFeed News dataset	It is not suitable for early detection.	Analyzing domain reputations related features	Hybrid Political news containing persuasive information
			Experiments are conducted Only in English language.	Analyzing the similarity and dissimilarity of the fake and real news via (TF-IDF) and (LDA) topic modeling	
			Experiments are conducted Only in one platform (websites).	document similarity analysis via Jaccard similarity measures	
[61]	2020	FNC dataset	Experiments are conducted Only in English language.	Using stance detection to improve fake news detection	Headlines and article bodies containing the element of commentary
			Using only pretrained word embedding (word2vec) and neglecting updating method during training.		

features for detection.

Oliveira et al. [8] conducted a stylistic-computational analysis, which is based on NLP and used a one-class SVM technique for detecting fake news on social media. The dataset used in this study contains 33000 tweets. Methods discussed in this paper can be beneficial for early detection as the required features for the detection are available as soon as a news is released.

Schuster et al. [47] developed a fake news detection model that label textual content as "real" or "fake" according to its truthfulness score. The study uses Grover[1], a model for neural fake news generation and detection. A similar setup described in Zellers et al. [70] was used in this work. The goal of the setup is to generate a large fake corpus using language model so that the classifier can classify it as real.

Li et al. [32] leveraged deep learning by proposing a multi-level convolutional neural network (MCNN) and a method of calculating the weight of sensitive words for fake news detection in textual news articles. The method provided deeper semantic anal-

[1] `https://github.com/rowanz/grover`

ysis and understanding of news article text and its veracity through the relationship between the article text content and the corresponding weight of sensitive words it invokes. The study utilizes the following benchmark fake news datasets (i) LIAR, compiled by Wang et al. [63] (ii) microblog datasets from Twitter and Sina Weibo compiled by Ma et al. [34] and Ma et al. [35] respectively (iii) NewsFN[2] and (iv) KaggleFN[3].

Table 2.6: Qualitative Analysis of Datasets in Feature based Detection Techniques

Study	Year	Dataset used	Limitation	Contribution	Scope
[8]	2020	33.000 tweets	It is not a real-time method.	Using TF-IDF to represent documents	Real and fake tweets collected from reliable accounts and the profile "Boatos.org"
			Experiments are conducted Only in English language.	Using LSA technique and Matrix Transformation Methodology to reduce dimensions	
			Experiments are conducted Only in one platform (Twitter).	Using Radial Limit Methodology to show samples	
[47]	2020	newsQA	Other datasets that represent a wide range of LM applications, from whole-article generation to forms of hybrid writing and editing are required.	Proving the inability of stylometry against machine-generated misinformation	Mix news containing both real news and commentary.
		New York Times articles		Utilizing LMs to generate synthetic fake and true news	mix news articles containing real news
		Modified New York Times articles			mix news articles containing fake news and commentary
[32]	2019	LIAR	Both datasets were relatively small, so the deep learning network are liable to be overfitted.	Proposing a method for calculating the weight of sensitive words	Hybrid Political news containing persuasive information
		Twitter		Designing and evaluating Multi-level convolutional neural network	Social media news containing real and fake news
		Weibo			Mix news containing Commentary, Feature Writing, persuasive information and fake news

[2]https://github.com/joolsa/fake_real_news_dataset
[3]https://www.kaggle.com/mrisdal/fake-news

2.4.7 Ensemble-based Detection

Ensemble Learning based Detection contains all papers that have utilized the approach of ensemble learning model in their detection to achieve better results. Elhadad et al. [11] developed a model for detecting misleading information related to the COVID-19 pandemic in English language by applying an ensembled learning approach, which combines different machine learning classifiers. In this study, a technique is introduced to collect ground-truth from credible and unbiased information sources.

Huang et al [23] has used a deep learning model for fake news detection. He used a combination of four deep learning techniques namely LSTM, depth LSTM, LIWC CNN, and N-gram CNN and formulated an ensemble learning model. Furthemore, he applied Self-Adaptive Harmony Search (SAHS) algorithm to optimize model weights and attained accuracy as high as 99.4%. Similarly, in [20], the authors have proposed another similar ensemble based approach to detect fake news.

Table 2.7: Qualitative Analysis of Datasets in Ensemble Learning based Detection Techniques

Study	Year	Dataset used	Limitation	Contribution	Scope
[11]	2020	7,486 ground-truth data instances collected from reliable sources	It covers only one language (English), one topic (COVID-19) and one platform (website).	Developing a new Dataset via a semi-automated approach	Ground-truth data related to COVID-19
				Proposing a voting ensemble classifier to detect misinformation related to COVID-19.	
[23]	2020	Fake_or_Real_news (FOR) dataset	Auxiliary information not included	Formulated a new model using four techniques	Hybrid Political news containing real and fake news
		Snopes_Fake_Legit_news (SFL) dataset	No feature analysis	SAHS algorithm for model weights optimization	Political news containing fake news
		Fake News Detection (FND) dataset	Does not work for early detection of false news propagation	Comparison with other methods on different domain-oriented datasets	Political news containing fake news
				Explored the Cross-domain intractability issue	

2.5 Concluding Remarks

This chapter discussed the life cycle of fake news and the definition of its types. Also, we presented various aspects of features used in fake news detection studies. Moreover, we organized the fake news detection models and techniques presented in the studied papers into three categories (ML, DL, NLP). Also, each category is divided into subcategories to better understand their similarities and differences. Following that, we presented a summary of the most recent research on fake news detection, grouped into seven categories based on the primary focus of the study. Each group of these studies is analyzed from multiple perspectives, and the details are provided in their tables. In the next chapter, we present the proposed model and discuss its modules.

Chapter 3

Proposed Method

3.1 Overview

Analyzing existing works shows that an effective fake news detection method must be applicable in the early stages of dissemination and prevent fake news from spreading. In this chapter, we propose a new framework for detecting fake news in the early stages of its propagation. It has four modules: 1) Preprocessing, 2) Extracting features, 3) Constructing input sequence, and 4) Self-attention based classifier. The first three modules of the proposed framework convert each news article's propagation network into a sequence of nodes after preprocessing and feature extraction. In these modules, different groups of features (both social context-based and content-based) from news articles are extracted. The last module of our framework leverages a self-attention mechanism based encoder to classify the news article. Self-attention technique is the core of the well-known Transformer model [62], which has achieved promising results in different areas, especially in complex tasks like language translation. In the following sections, the details of each module are presented. Also, we explain the Transformer's structure, especially its encoder and self-attention mechanism. Moreover, the differences between our encoder and the Transformer's encoder

are also discussed.

3.2 Proposed Method

As it is shown in Fig. 3.1, the proposed framework has four modules: 1) Preprocessing, 2) Extracting features, 3) Constructing input sequence, and 4) Self-attention based classifier. These modules work together to prepare input data for the encoder, and at the end, a label is going to be produced for each piece of news by the classifier. In the following sections, each component of this framework is introduced.

3.2.1 Preprocessing

The evaluation of the model is conducted using a comprehensive available fake news detection dataset named FakeNewsNet [51], which is collected from PolitiFact[1] and GossipCop[2] fact-checking websites. This dataset contains news content, labels, and the corresponding social context information, i.e., tweets, re-tweets, etc. It is an ideal dataset for this study because: 1) The labels are accurate as they are manually checked and assigned by experts; 2) It includes social context information which are required for creating the propagation network for news articles. Table 3.1 represents some statistics of this dataset. As can be seen, the number of instances in the GossipCop source is considerably higher than the PolitiFact that makes it more appropriate to train and test a deep learning model like the proposed one in this study.

Propagation of a news article is conducted when users start tweeting about it, and other users react to these tweets by re-tweeting (or quote tweeting, reply tweeting). For modeling such propagation, we consider each tweet and its re-tweets (quote tweets, reply tweets) as one cascade. Then, by connecting all cascades in the central

[1] https://www.Politifact.com/
[2] https://en.wikipedia.org/wiki/Gossip_Cop

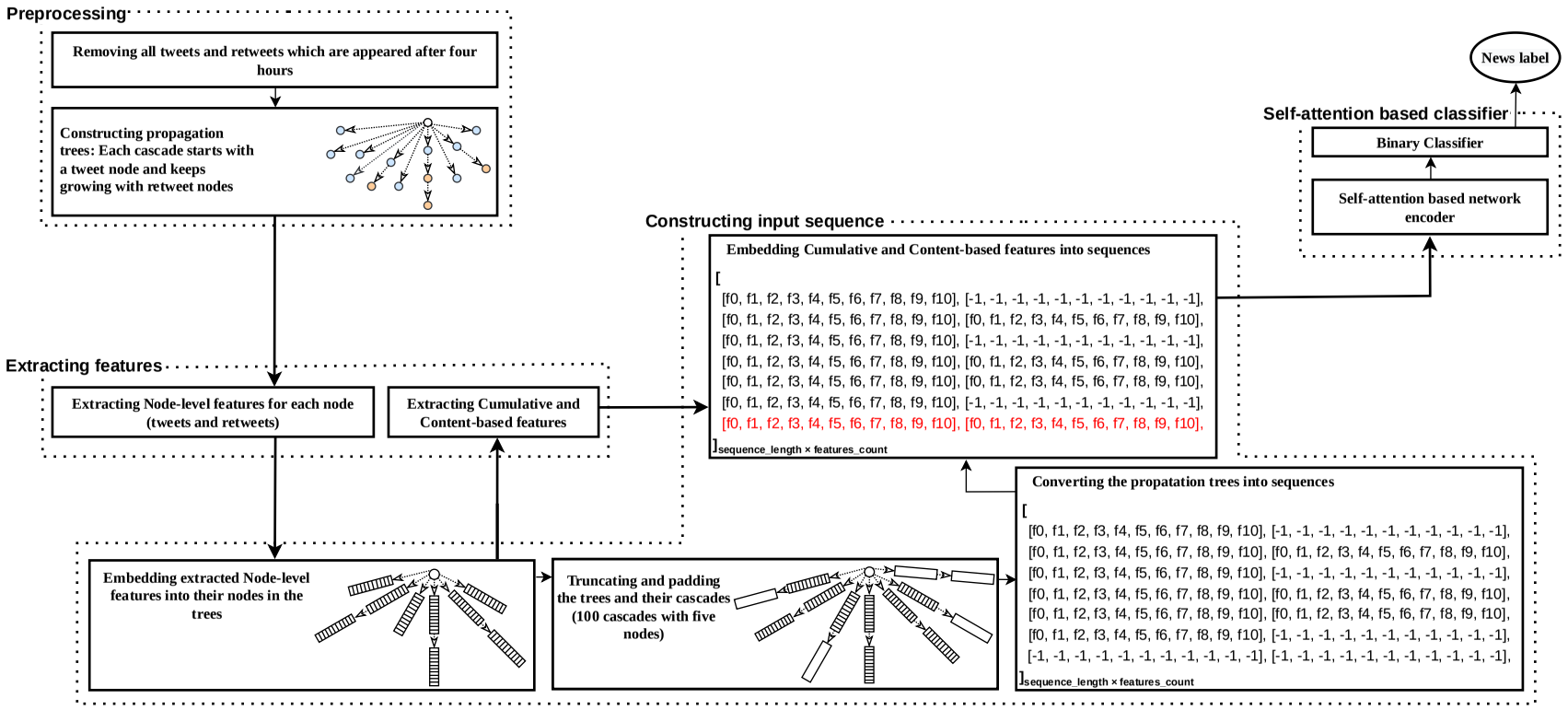

Figure 3.1: The overall architecture of the proposed method. It consists of four modules: 1)Preprocessing, 2)Extracting features, 3)Constructing input sequence, and 4)Self-attention based classifier.

Table 3.1: Statistics of the FakeNewsNet dataset. Classifiable means the news article gets user engagement in the first four hours and, the corresponding propagation network is built.

Dataset	PolitiFact		GossipCop	
	Fake	Real	Fake	Real
# Samples	385	548	4,849	15,155
# Classifiable Samples	340	355	4,615	14,119
# Tweets	100,496	266,622	356,183	682,388
# Re-tweets	71,012	341,000	180,454	130,652
# Users	114,844	363,206	302,542	143,044

point, a shape similar to a tree is produced that shows the dissemination process of the corresponding piece of news (Fig. 3.2).

As we mentioned, each cascade starts with a tweet object and continues with its re-tweet objects. However, we need a way to arrange the re-tweets. In this work, they are sorted based on the time of creation. So, the first cascade of the propagation network will be as follows: $\{tweet_0, re-tweet_0, re-tweet_1, ..., re-tweet_n\}$.

For creating such a sequence, we have to form it based on time because a re-tweet object retrieved by Twitter API merely indicates the source tweet and has nothing to do with the intermediate successors. So, this Twitter API's limitation does not allow us to determine the intermediate re-tweets. In previous works [55, 56], the user-ids mentioned in the tweets are also used for creating this sequence. In this work, as we considered the detection time shorter (4 hours), the formed cascades are mostly comprised of a tweet and two re-tweet nodes that represent almost 93% of the total cascades of all news items. As a result, there is no significant loss of information. Also, note that in [55, 56], the whole propagation network is required in the training phase, wherein the cascade's length is long and consists of many nodes. However, in this work, we only need the partial propagation network. The follower and following relation is explored neither in other works nor this work as it is not accessible in real-time.

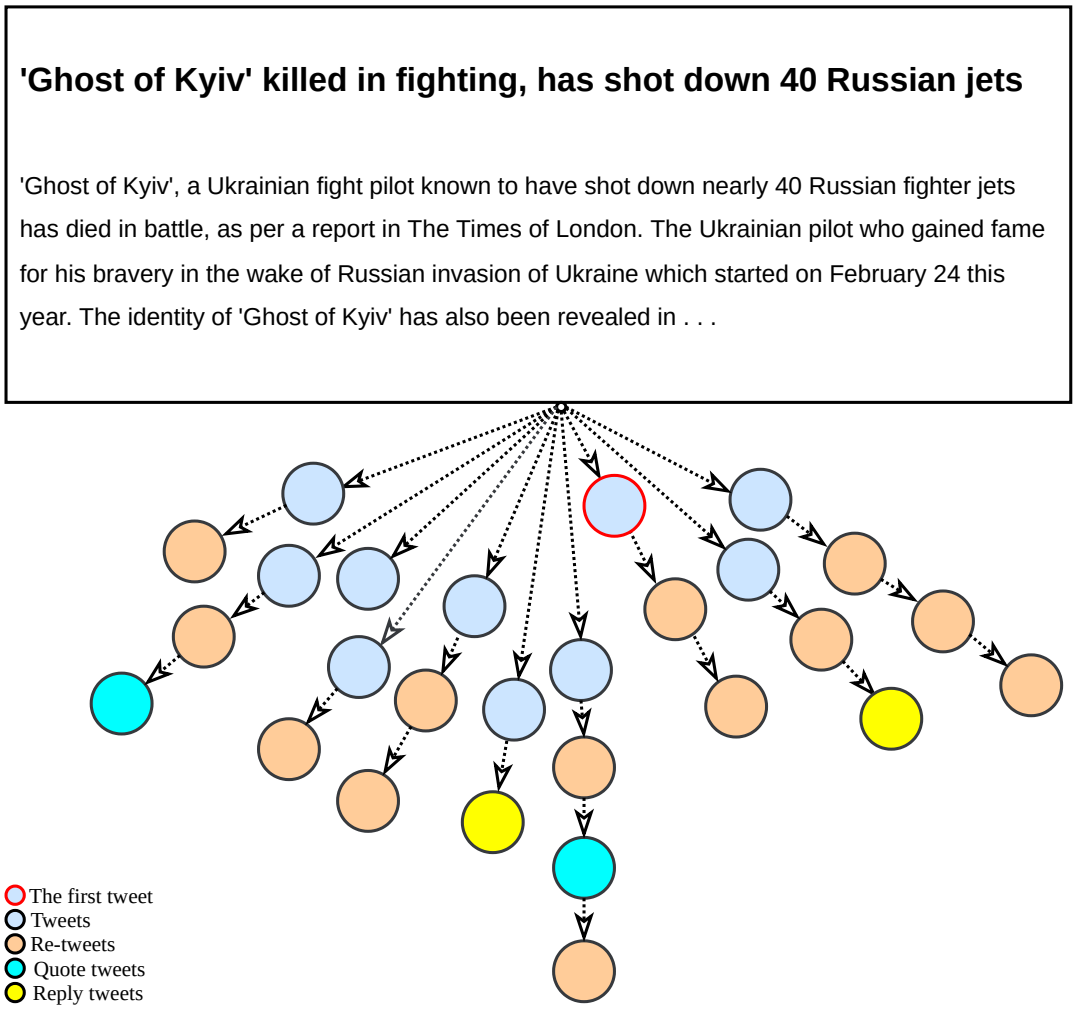

Figure 3.2: Propagation tree of a news article with multiple cascades having several nodes with different types, i.e., tweets, re-tweets, quote tweets, and reply tweets (depicted by different colors). The node with a red border is the first tweet about the shown news article. We consider the creation time of this tweet as t_0. The remaining tweets and re-tweets happen within four hours (detection deadline) after t_0.

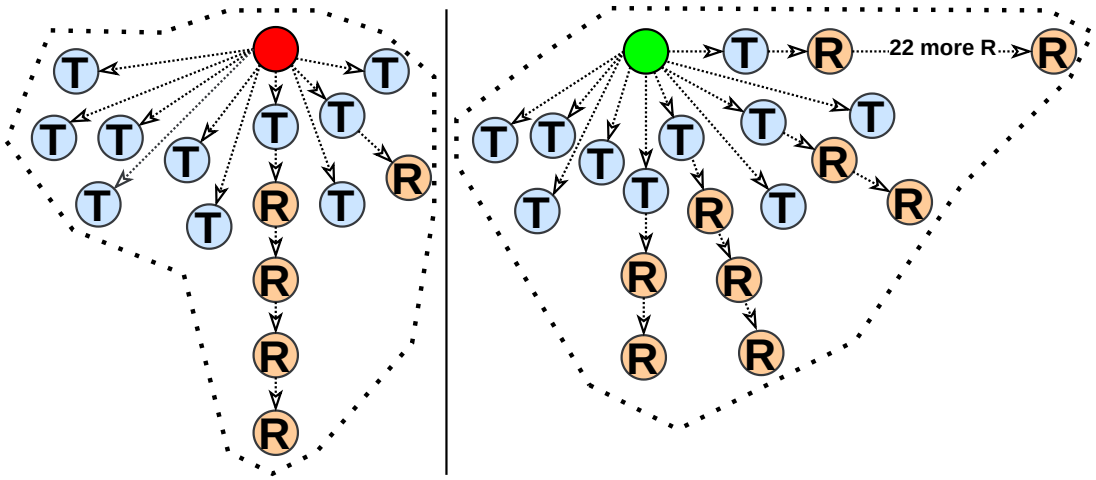

Figure 3.3: The partial propagation network of a piece of fake news with id 'Politifact14063'(left) and a real one with id 'Politifact11208'(right). These propagation networks are built from the tweets and retweets created in the first four hours of dissemination.

3.2.2 Extracting Features

In this section, we are going to explain the features used in this study. They fall into two categories: Content-based and context-based. The content-based features are collected from the news content (title and body), while the context-based features are extracted from the propagation network of the news records (Fig. 3.3). In the following sections, the details of these features are provided.

3.2.2.1 Context-based Features

The context-based category includes node-level and cumulative feature groups. As depicted in section 3.2.1, the network propagation of news comprises nodes representing tweets, re-tweets, quote tweets, and reply tweets objects. Node-level and cumulative features provide the proposed model with information about nodes and the whole propagation network, respectively.

A. Node-level Features This group of features includes information on each node that falls into three subcategories: *User-based*, *Tweet-based*, and *Temporal*. In the

Table 3.2: The list of Node-level features.

Type	Feature	Title
User-based	nf_0	Whether the user is verified
	nf_1	Number of followers
	nf_2	Number of followings or friends
	nf_3	Number of lists
	nf_4	Number of total favourites
	nf_5	Number of total tweets
Tweet-based	nf_6	Length of tweet or re-tweet
	nf_7	Number of re-tweets
	nf_8	Number of favorites
Temporal	nf_9	Registering time of the user
	nf_{10}	Tweeting time of the tweet or re-tweet

next sections, each group of these features are further discussed:

I. Node-level user-based features: These features are extracted from each user's profile. This group comprises six features depicted in Table 3.2. Features such as-Is the user verified or not (nf_0), the number of followers (nf_1), the number of followings (nf_2), the number of lists (nf_3) are capable of determining the validity of a user profile (especially the first three ones). Measuring the validity of users profiles is helpful as invalid users, compared to valid ones, are more liable to spread fake news [53].

II. Node-level tweet-based features: This group of features is derived from tweet content. As shown in Table 3.2, this category consists of three features and are useful to show differences in the users' content writing (nf_6) and how they react to the fake and real news (nf_7, nf_8).

III. Node-level temporal features: This group of features contains information about the user's registration time (nf_9) and the time of tweeting or re-tweeting (tweets being created) (nf_{10}). They can show how users react to fake news during specific time periods and whether it spreads with different time patterns. As an example, feature

(nf_9) can indicate if the users participating in a cascade(even a whole network) have been created in a short time interval. This, in turn, can show whether the fake news has been spread by a team of malicious users (e.g., user accounts created in a short time by a person or a small group of people to act as a team to spread fake news).

B. Cumulative Features As we have to truncate the propagation tree and its branches to change it to a fixed-length input sequence for the proposed model, some details about its shape, like the number of branches, long branches, etc., are lost in this process of simplification. To remedy this information loss, we can give such information to the model in the form of cumulative features. Furthermore, with the help of cumulative features, we collect some information about the whole propagation network instead of only one node, i.e., the focus is on the network's specifications. This group falls into three sub-groups: *User-based*, *Text-based*, *Temporal*. These features are explained in the following sections:

I. Cumulative user-based features: These features are collected from user profiles, and as the name suggests, they contain information about all the users engaged in the propagation of a news article. They consist of the number of users with high impact (leader users), the number of verified users, the number of not verified users, and the number of all unique users, i.e., after removing duplicate users who have tweeted or re-tweeted a piece of news several times. By leader user, we mean a user who has the conditions as per the equations (3.1, 3.2, 3.3):

$$\frac{Numberoffollowers}{Numberoffriends} > 1 \tag{3.1}$$

$$Numberoffriends > 0 \tag{3.2}$$

$$Numberoffollowers > 1000 \tag{3.3}$$

II. Cumulative text-based features: This group of features contains information about the linguistic and written style of all tweets in the propagation network, such as the total number of hashtags, the frequency of positive and negative words, etc. Table 3.3 represents the list of these features.

III. Cumulative temporal features: These features, similar to their counterparts in the Node-level group, represent the properties of temporal spreading patterns for fake and real news. They consist of the average time interval between each user registration(cf_{12}), the average time interval between each tweeting (all types of tweets)(cf_{13}), and the average time interval between each tweeting (only tweets)(cf_{14}).

Table 3.3: The list of Cumulative features.

Type	Feature	Title
User-based	cf_0	Number of verified users
	cf_1	Number of not verified users
	cf_2	Number of leader users
	cf_3	Number of all users
Text-based	cf_4	Number of hashtags
	cf_5	Number of mentions
	cf_6	The frequency of positive words
	cf_7	The frequency of negative words
	cf_8	Number of original tweets
	cf_9	Number of re-tweets
	cf_{10}	Number of quote tweets
	cf_{11}	Number of reply tweets
Temporal	cf_{12}	The average time interval between each user registration
	cf_{13}	The average time interval between each tweeting(all types of tweets)
	cf_{14}	The average time interval between each tweeting(only tweets)

3.2.2.2 Content-based Features

Unlike other features, this group has nothing to do with the propagation network and is extracted from news content (title and body). Similar to cumulative text-based features, they represent the linguistic differences between fake and real news. Table 3.4 shows the list of these features.

Table 3.4: The list of Content-based features.

Type	Feature	Title
Content-based	gf_0	The length of news title
	gf_1	The length of news text
	gf_2	The frequency of positive words in title+text
	gf_3	The frequency of negative words in title+text
	gf_4	The sentiment score computed using VADER for title+text

3.2.2.3 Removing Ineffective Features Using Wrapper Method

The introduced features in previous sections are selected from a larger group of features. We studied all features using the wrapper feature selection method [6] and eliminated the features that did not improve the output result of the model. By removing these features (Table 3.5), the number of the model's trainable parameters is reduced, and consequently, less time and computation are required to train the model.

3.2.3 Constructing Input Sequence

In the database setup stage, three tables are created for Tweets, Re-tweets, and Users (Fig. 3.5). All tweets of a news article are sorted based on their tweeting time, and the creation time of the earliest one is considered as t_0. After this step, all tweets and re-tweets of a corresponding news article that happened after $t_0 + 4\ hours$ are removed. The propagation tree of the news article can be constructed from the

Table 3.5: The list of eliminated features.

Group	Feature
Node-level	The sentiment score computed using VADER for tweet's text
	The frequency of positive words in tweet's text
	The frequency of negative words in tweet's text
	Number of hashtags in tweet's text
	Number of mentions in tweet's text
Cumulative	Number of cascades with a length of 1
	Number of cascades with a length of 2
	Number of cascades with a length of 3
	Number of cascades with a length of 4
	Number of cascades with a length of 5
	Number of cascades with a length of 6
	Number of cascades with a length of 7
	Number of cascades with a length of 8
	Number of cascades with a length of 9
	Number of cascades with a length of 10
	Number of cascades with a length greater than 10

remaining tweets and re-tweets using the method introduced in section 3.2.1. In this stage, each tree node contains only the tweet's ID. So in the next step, each node is filled with its extracted node-level features (11 features in Table 3.2).

Since the length of the model's input is fixed, the number of tree branches, as well as their length, need to be the same using the truncating and padding techniques. In padding, empty nodes are added at the end of the short cascades. In contrast, truncating method trims long cascades. In our implementation, the number of cascades (branches) is 100, and the length of each cascade is considered five nodes. These numbers are determined based on multiple experiments and evaluating the model with different sizes of input sequence considering both the average of the number of cascades and the average of cascade length per news article. After conducting the previous steps, the network propagation depicted in Fig. 3.4 has 100 cascades

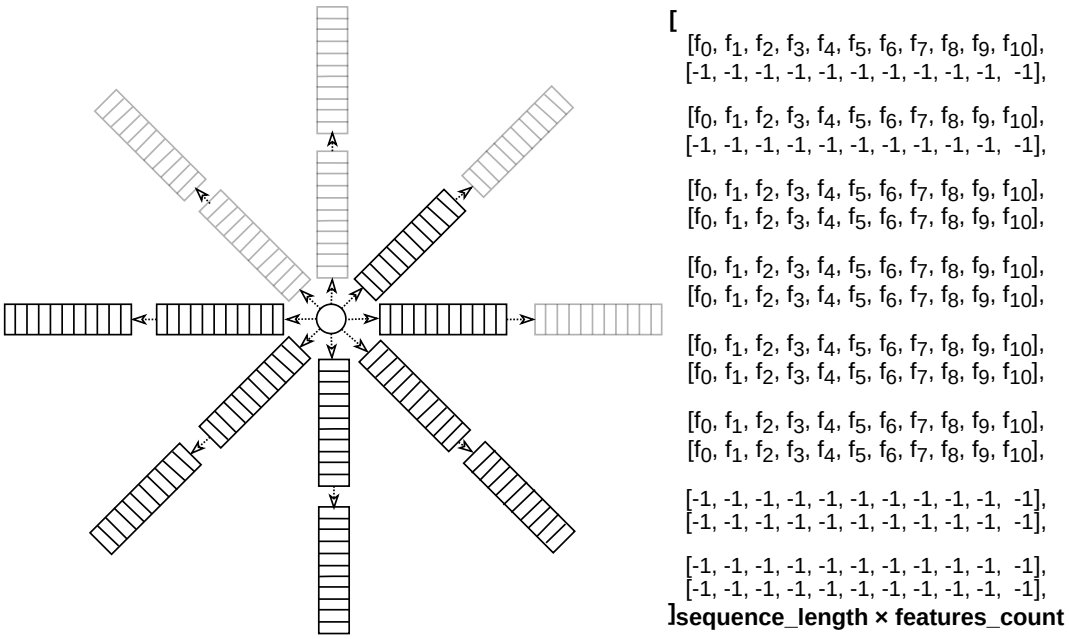

Figure 3.4: The propagation tree of each news is converted into a sequence. In our implementation, each tree has 100 cascades with five nodes making the resulting sequence 500 nodes. As can be seen, 11 features are selected for each node.

of five nodes. In this figure, the pale nodes are the result of applying the padding technique. Similarly, elements filled with (-1) denote padded nodes in the sequence. Also, cascades consisting of more than five nodes are trimmed, and if the propagation network has more than 100 cascades, they are sorted from shortest to longest, and the first 100 short cascades are chosen. In [55, 56], the shortest cascades are removed, and the 100 longest ones are selected with the justification that they may contain more information, but our model represents the contrary probably because the first nodes of each cascade compared to its following nodes have more valuable information.

In the end, the resulting propagation network turns into a sequence with a length of 500 elements, where each element has 11 values and represents one node of the propagation network. Then, each news article's cumulative and content-based features are formed as two vectors with 11 values that will replace the last two elements of the corresponding sequence. Note that the replaced elements are mostly the padded

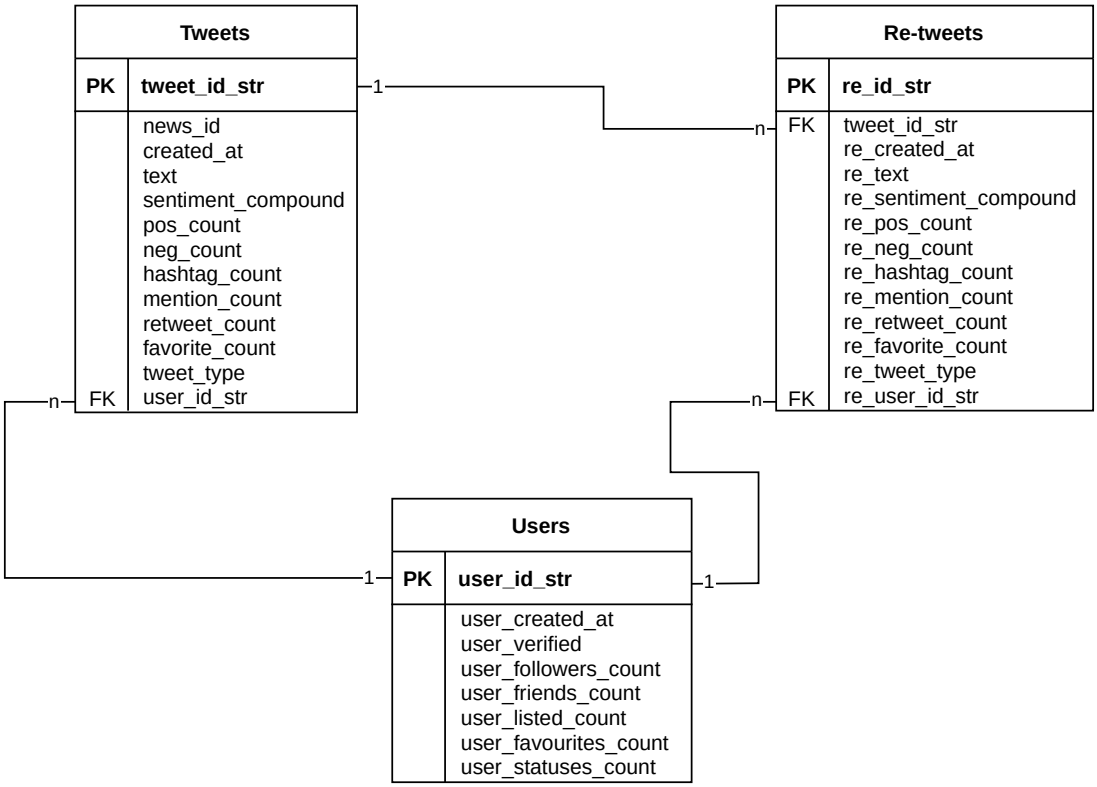

Figure 3.5: The Entity Relationship (ER) Diagram of our database. These tables maintain the information extracted from the JSON files. PK and FK represent the primary key and foreign key, respectively.

elements filled with the value of (-1) (Elements with red color in Fig. 3.1).

Before feeding the prepared input sequence to the model, the values of each feature are normalized to ensure that all attributes in the dataset are on the same scale. This can be done through various methods, such as Min-Max normalization [22]. Without normalization, the attribute with a smaller range of values may not function appropriately compared to the others. Normalization helps equalize all attributes' scales, producing more balanced and reliable data. Also, normalized data helps the model to converge faster. The Min-Max method is conducted by using the following formula:

$$X_{new} = \frac{X - min(X)}{max(X) - min(X)} \tag{3.4}$$

X_{new} : The new value from the normalized results

X : Old value

$min(X)$: Minimum value in the training set

$max(X)$: Maximum value in the training set

3.2.4 Self-attention Based Classifier

The main module of the proposed classifier is a self-attention based network encoder. The self-attention mechanism was introduced in the Transformer's architecture for the first time [62]. In the following sections, we first briefly explain the Transformer and self-attention mechanism and then introduce the proposed encoder.

3.2.4.1 Original Transformer Architecture

A Transformer is a deep learning model that processes sequential data such as text. It was introduced in [62] in 2017 and has become widely popular for tasks such as machine translation, language modeling, and text classification. The Transformer model consists of an encoder and a decoder (Fig. 3.6). The encoder is made up

of a stack of N (6) identical layers, each of which consists of two sublayers. The first sublayer implements a multi-head self-attention mechanism. This mechanism involves multiple heads that each receive a different version of the queries, keys, and values that have been linearly transformed. These heads generate outputs in parallel, which are then combined to generate the final result. The second sublayer within each layer of the encoder is a fully connected feed-forward network that consists of two linear transformations separated by a Rectified Linear Unit (ReLU) activation function:

$$FFN(x) = ReLU(W_1 x + b_1)W_2 + b_2 \tag{3.5}$$

The six layers of the Transformer encoder apply the same linear transformations to every word in the input sequence, but each layer uses different weight (W_1, W_2) and bias (b_1, b_2) parameters. In addition, each of these two sublayers has a residual connection, or shortcut connection, around it. This means that the output of the sublayer is added to its input, allowing the information from the input to be directly passed through to the output. This can help alleviate the vanishing gradient problem, which can occur when training deep neural networks, and improve the model's overall performance. Each sublayer is also followed by a normalization layer, which normalizes the sum of the input to the sublayer and the output produced by the sublayer itself:

$$Norm(x + sublayer(x)) \tag{3.6}$$

It's important to note that the Transformer architecture does not inherently capture information about the relative positions of the words in a sequence, as it does not use recurrence. This means that positional information must be added to the input embeddings to be considered by the model. This can be done by introducing positional encoding vectors to the input embeddings. These vectors are of the same dimension as the input embeddings and are generated using sine and cosine

functions of different frequencies. Then, they are added to the input embeddings to incorporate the positional information. This allows the Transformer to consider the order of the words in the input sequence and how they relate to each other (Fig. 3.6).

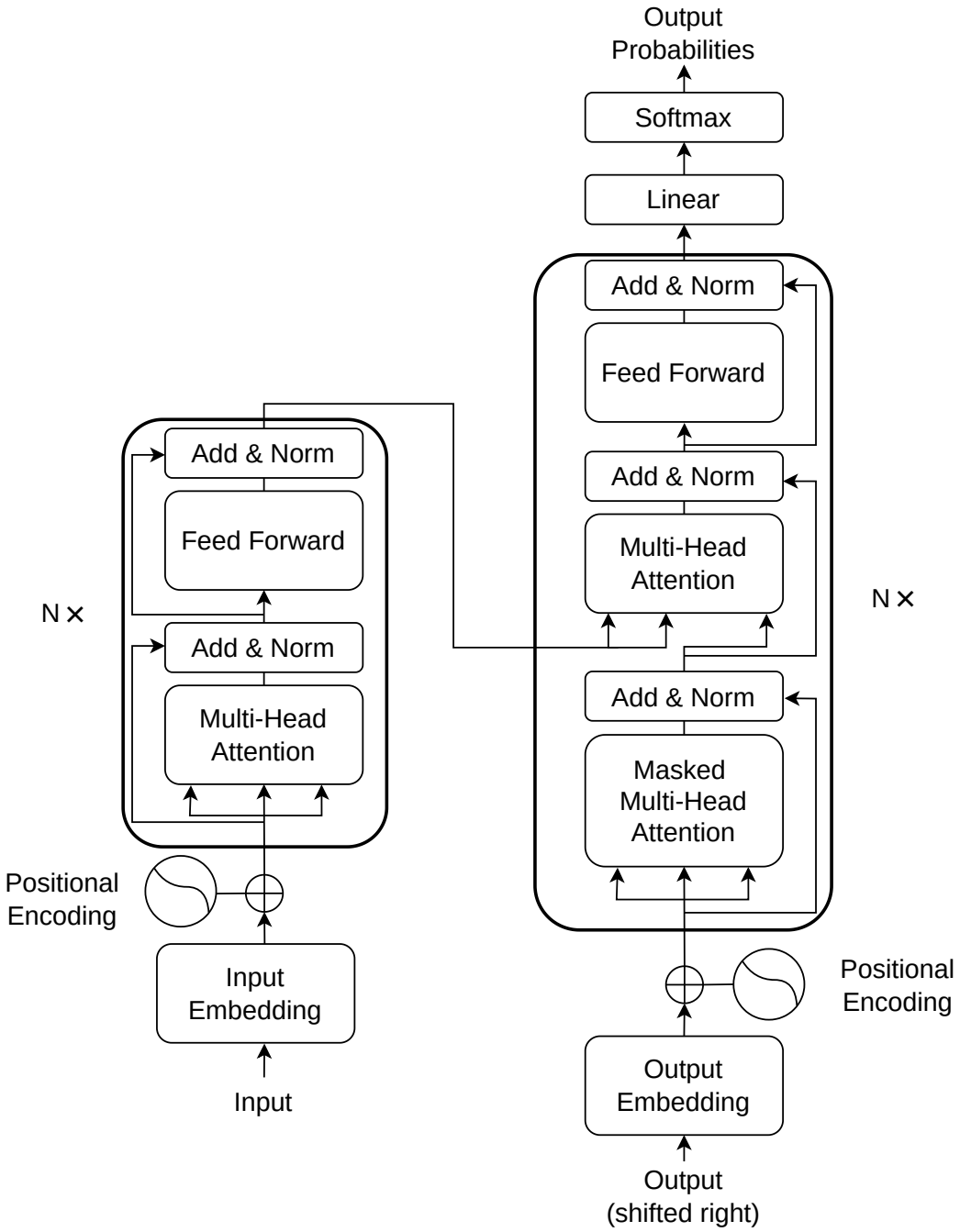

Figure 3.6: The encoder-decoder structure of the Transformer architecture Taken from [62].

As shown in Fig. 3.6, the decoder module shares several similarities with the encoder in its structure and function. For more information, refer to the main paper [62].
One of the key innovations of the Transformer model is the use of a self-attention mechanism, which allows the model to directly consider the relationships between all input tokens at every time step rather than just considering a fixed window of nearby tokens as in traditional RNNs or LSTMs. This allows the Transformer to capture long-distance dependencies and relationships in the input data. The self-attention mechanism works by first projecting the input elements into a higher-dimensional space using a linear transformation. The model then calculates the dot product of each element with all the others and applies a softmax function to obtain a set of attention weights that represent the importance of each element with respect to the others. These attention weights are then used to weigh the contributions of the different elements when generating the output. The self-attention mechanism has proven to be very effective in natural language processing tasks and has been widely used in state-of-the-art models such as BERT [9] and GPT-3 [14].

3.2.4.2 Proposed Encoder

The most important module of the proposed model is a self-attention based network encoder. It is fed an input sequence, created from all feature groups of a news article, and generates an embedding for it. This module (Fig. 3.7) is the same encoder that has been proposed in the well-known Transformer architecture [62], except for some minor differences.
In our implementation, the number of used encoder layers and heads varies. The encoder module of the original Transformer model gets a sequence of token embeddings as it was designed for handling texts, while our encoder receives a sequence of elements having 11 values. The projection part of the encoder(Fig. 3.7) projects each element to a new vector with d dimension. In other words, it is a linear layer

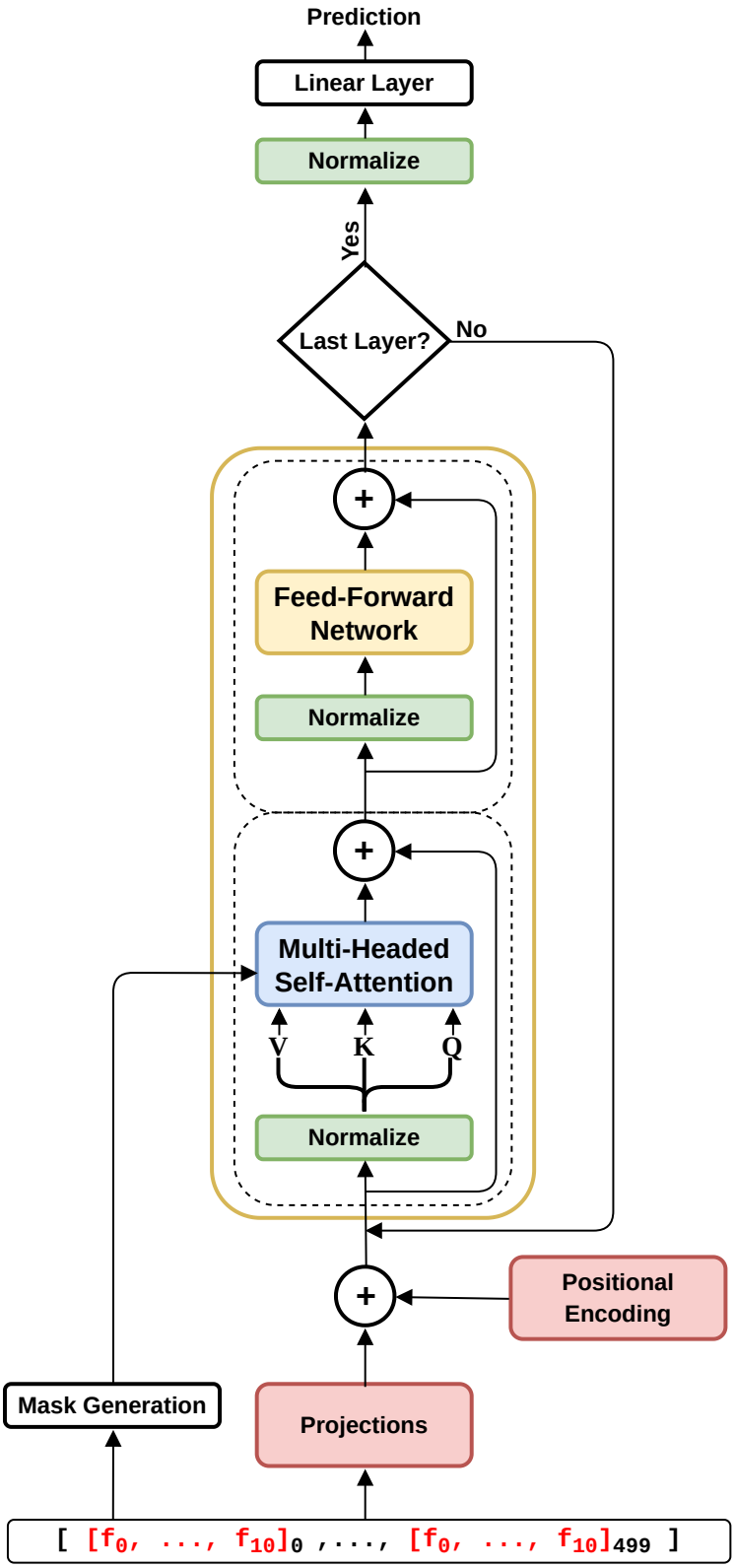

Figure 3.7: Self-attention based network encoder is the main module of the proposed classifier. It is responsible for encoding the propagation network of a news article used by the last layers of the classifier to generate a label.

that receives a list of vectors each of the size of 11 (the number of features) and maps to a list of vectors of the size of 32 (the dimension of the model). The size of the vectors arriving at the next layers of the model is 32.

In the Transformer architecture, positional encoding is used to inform the model about the position of every word in the textual input such that it knows the order of the words. Similarly, we exploit positional encoding to add a positional vector to each element's projection. Thus, the model will be able to see the temporal order existing in the input sequence. Note that the relationship among a text's sentences is not precisely similar to the relationship among the propagation network's cascades. For example, the first sentence cannot be placed in the middle of the text (they are sequential), while there is no such order among a propagation network's cascades. Thus, the positional encoding of the original Transformer needs to be changed such that the address vectors added to each cascade's nodes will start from the beginning. Before forwarding the input sequence equipped with positional embeddings to encoder layers, a mask is generated that is used later by self-attention mechanism. This mask aims the attention mechanism not to assign any attention to the padded nodes valued with (-1) (Fig. 3.1). As far as the remaining nodes are concerned, the more informative they are, the more attention they are assigned by this mechanism (Fig. 3.8).

The rest of the proposed encoder is similar to the original encoder in the Transformer model. The output of the last encoder layer is normalized using layer normalization. Finally, the hidden states generated for the first and last element of the input sequence are concatenated and forwarded to the classifier, which is a linear layer and a sigmoid function to predict the label (Fig. 3.7).

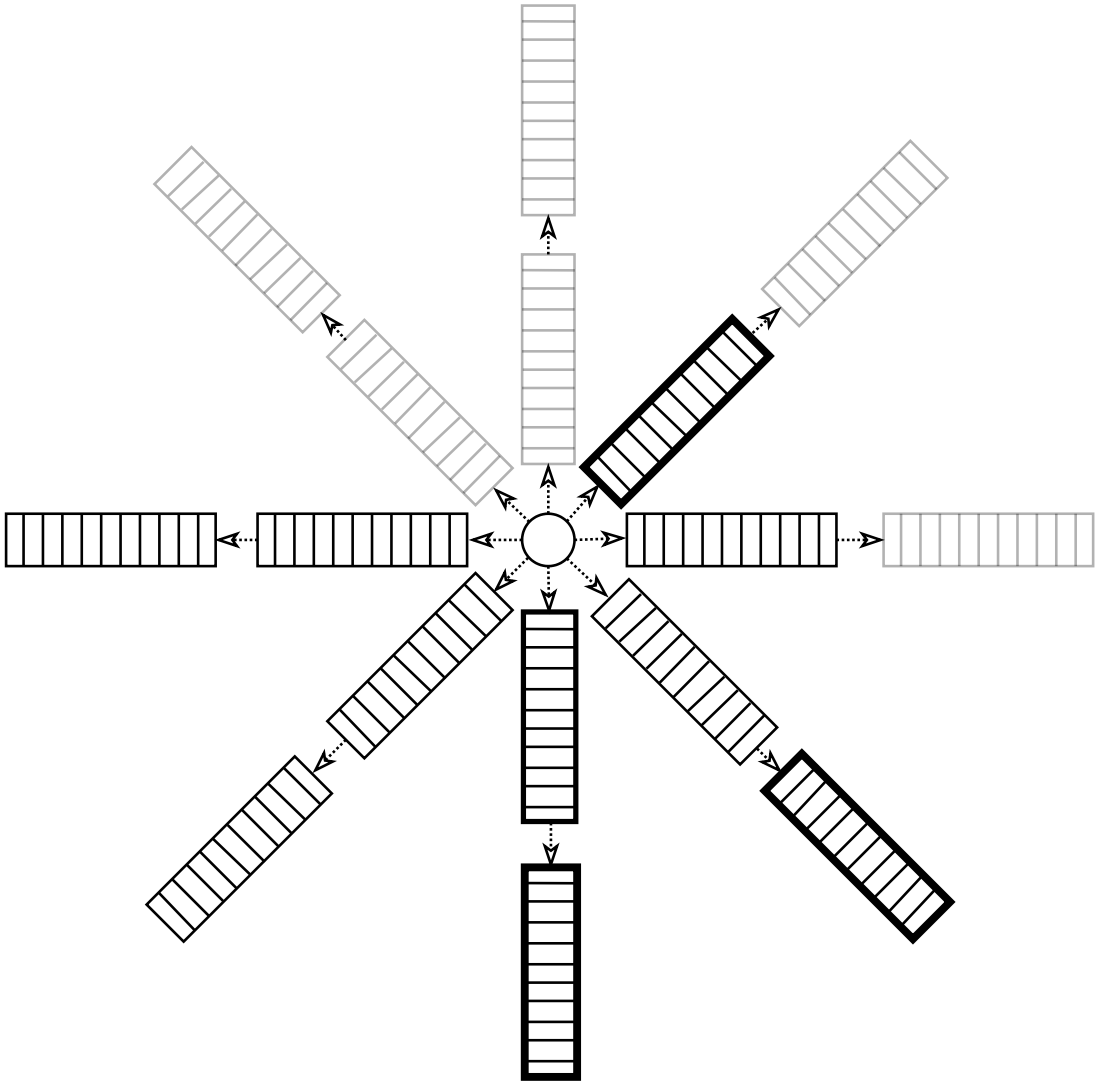

Figure 3.8: The self-attention mechanism in a piece of news's propagation network. The pale nodes have been padded to the tree, so they are masked and ignored entirely in this mechanism. The bold ones are more informative and therefore they are given more attention. Each node is assigned a score considering the values of its 11 features.

3.3 Concluding Remarks

In this chapter, we presented our fake news detection architecture in full detail. We discussed that the first module of the model is responsible for preprocessing operations. Also, we described that in the second module, different types of features (node-level, cumulative, content-based) are extracted and how we have modeled the propagation network of a news article. Then, in the third module, the construction of the input sequence was explained. Finally, we discussed the last module, which is a Self-attention based classifier. Also, we briefly presented the Transformer's architecture and its self-attention mechanism. Then, we discussed the differences between our encoder and the original Transformer's encoder. In the next chapter, we will discuss the experiments and results of the proposed framework.

Chapter 4

Experiments & Results

4.1 Overview

This chapter begins by giving an overview of the experimental setup and the dataset used for the experiments and obtaining the results. Later, we discuss the baseline models and divide them into three categories: content-based, propagation network-based, and mixed methods. Then, we explain the evaluation metrics utilized in this work after discussing the proposed model's parameters.

Later in the chapter, we present in detail the implementation results of the proposed model and provide a comparison of it with the baseline models. Finally, the experiments' results prove the proposed model's effectiveness in fake news detection within the first hours of dissemination. Also, we briefly discuss how the model can achieve appropriate results in the early stages of propagation.

4.2 Experimental Setup

The implementation and execution of all the modules were processed on Ubuntu 20.04. All the data preprocessing and feature extraction were implemented using Python scripts and Python public libraries. In addition, the proposed model was implemented using PyTorch version 1.12.1 (see Appendix A). We trained and tested the model on a GPU server with two NVIDIA TITAN V units with 12GB memory.

4.3 Dataset

In this study, the FakeNewsNet dataset is used to train and test the proposed model. The content and labels of the news articles can be downloaded from this repository[1]. Also, we downloaded the social context of the news articles by using a script that is provided by the same repository. The script uses Twitter's API to download tweets, retweets, user profiles, etc. The downloaded information consists of a large number of JSON files. The downloading process took about three months because of Twitter's rate limits[2]. We present some statistics about the dataset in Table 3.1, but for more information, refer to the main paper [51].

4.4 Baselines

As it can be seen in Table 4.1, three types of methods are selected as baseline approaches: **Content-based**, **Propagation network-based**, and **mixed**.

[1] `https://github.com/KaiDMML/FakeNewsNet/tree/654361e1c8d5baa751baf1dac5032df621652280`

[2] `https://developer.twitter.com/en/docs/twitter-api/rate-limits`

4.4.1 Content-based Methods

This group includes four methods that predict the label of the news articles exclusively using their content:

- RST [45]: Rhetorical Structure Theory (RST) forms a tree structure rhetorical relations among the words in the news text. It takes advantage of the frequencies of rhetorical relations to create a feature vector representing news style features.

- LIWC [42]: Linguistic Inquiry and Word Count (LIWC) is capable of collecting the lexicons that are categorized as psycho-linguistic words. The resulting feature vector is a learned representation based on psychology and deception point of view.

- Text-CNN [30]: This method uses a Convolutional Neural Network that utilizes multiple filters to encode the news text.

- HAN [69]: This method exploits a hierarchical attention mechanism to classify news records. With the help of a two-level attention technique (word and sentence), it generates a new representation for a news article.

4.4.2 Propagation Network-based Methods

Three of baseline approaches fall into this group; instead of using the content of the news records, they utilize their propagation network to predict their labels:

- HPA-BLSTM [18]: This method is a neural network consisting of a bi-directional long short-term memory component and a hierarchical attention module. It learns a representation from the user engagements on social media with a news article by assigning different levels of attention to words, posts, and sub-events.

- AE [55]: This model uses an autoencoder to predict the complete propagation network's embedding of a news article using its initial propagation network's embedding. Then, a classifier generates a label for the news article exploiting the predicted embedding using the autoencoder. In the training phase, both the initial and complete propagation networks are required, while the initial network is enough in the inference phase.

- Propagation2Vec [56]: This is a type of graph embedding method that generates an embedding for the propagation network of a news item. It utilizes a hierarchical attention mechanism to attend at two levels: node and cascade. Like the previous one, this method needs both the initial and whole propagation network for training and the initial propagation network in the test phase.

4.4.3 Mixed Methods

This group exploits both the content and context of the news records and includes three methods:

- TCNN-URG [43]: Two-Level Convolutional Neural Network with User Response Generator (TCNN-URG) has two main components: 1) TCNN, which extracts information from news content by considering it at the sentence and word level. 2)URG, a generative conditional variational autoencoder, which is trained to generate users' responses to the news article.

- CSI [46]: CSI is a hybrid deep learning model with three components Capture, Score, and Integrate. It generates a new representation for a news article by an LSTM and embeddings (Doc2Vec) of the news text and the corresponding user comments.

- DEFEND [49]: This is an explainable detection model of fake news consisting of four modules: 1) a news content encoder (word-based and sentence-based), 2) a user comment encoder, 3) a sentence-comment co-attention module, and 4) a classifier.

4.5 Parameter Settings

The main parameter to combat fake news is the detection deadline ($t^r + \Delta t$) which is determined as four hours in this work. Therefore, the proposed model predicts the label of the news utilizing its propagation network built within only the first four hours of its release. On the other hand, the baseline models make use of the whole propagation network and even those that are categorized as early detection are not able to classify the news earlier than five hours.

The second parameter, input sequence length, is set to 500 nodes after several deliberation and examination considering the depth and extent of the news propagation networks. Each node has 11 features, and its creation is introduced in section 3.2.

The hyper parameters related to the main module of the model that is a self-attention based encoder, are valued as follows: the number of heads (2), the number of encoder-layers (3), the dimensionality of the model (32), dropout (0.4), and learning rate (0.001).

The model is trained for 100 epochs with a batch-size of 64. The aforementioned values are determined after evaluating various ranges of values considering the available hardware resources. To evaluate the proposed model like [49, 52, 56], 75% of the samples are chosen randomly, and the remaining 25% are put aside for testing purposes. This process is done in five iterations, and the average of the outcomes is reported as the final result.

4.6 Evaluation Metrics

In this study, the following metrics are adopted to evaluate the performance of the proposed method as they are commonly used, and they make it possible to compare our approach with the baselines.

1. *True Negative (TN):* The number of real articles detected as real news.
2. *False Negative (FN):* The number of fake articles detected as real news.
3. *True Positive (TP):* The number of fake articles detected as fake news.
4. *False Positive (FP):* The number of real articles detected as fake news.

The ratio of correct predictions by the total number of instances is called *accuracy* and computed as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision is calculated as the number of true positives divided by the total number of true positives and false positives, i.e., the ratio of correct positive predictions out of all positive predictions.

$$Precision = \frac{TP}{TP + FP}$$

Recall quantifies the number of correct positive predictions made out of all positive predictions that could have been made.

$$Recall = \frac{TP}{TP + FN}$$

F1 combines the *Precision* and *Recall*, which makes it a reliable metric for both balanced and imbalanced datasets. It is the harmonic mean of the *precision* and *Recall*:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

4.7 Experimental Results

Table 4.1: The performance comparison for 11 different methods of fake news detection using accuracy, precision, recall, and F1. These methods fall into three groups: content-based approaches (C), propagation network-based approaches (P), and appropriate for early detection, i.e., capable of detecting using only initial propagation networks (E). The blue color of the background indicates the best results for only early detection methods, and the bold ones are the best results among all methods.

Method	Type			PolitiFact				GossipCop			
	C	P	E	Acc	Prec	Rec	F1	Acc	Prec	Rec	F1
RST	✓		✓	0.607	0.625	0.523	0.569	0.531	0.534	0.492	0.512
LIWC	✓		✓	0.769	0.843	0.794	0.818	0.736	0.756	0.461	0.572
text-CNN	✓		✓	0.653	0.678	0.863	0.760	0.739	0.707	0.477	0.569
HAN	✓		✓	0.837	0.824	0.896	0.860	0.742	0.655	0.689	0.672
HPA-BLSTM		✓		0.846	0.894	0.868	0.881	0.753	0.684	0.662	0.673
AE		✓	✓	0.874	0.878	0.870	0.871	0.889	0.859	0.805	0.828
Propagation2Vec		✓	✓	0.897	0.895	0.892	0.893	0.892	0.879	0.869	0.874
TCNN-URG	✓	✓		0.712	0.711	0.941	0.810	0.736	0.715	0.521	0.603
CSI	✓	✓		0.827	0.847	0.897	0.871	0.772	0.732	0.638	0.682
DEFEND	✓	✓		**0.904**	**0.902**	**0.956**	**0.928**	0.808	0.729	0.782	0.755
Proposed Method	✓	✓	✓	0.871	0.869	0.870	0.871	**0.911**	**0.871**	**0.908**	**0.886**

Table 4.2: The performance comparison for methods that are appropriate for early detection of fake news using F1 measurement. (DD) represents the detection deadline that is less than five hours(here four hours). The bold values indicate the best results when only the content and(or) the propagation network built in the first four hours are used.

Method	DD	PolitiFact	GossipCop
		F1	F1
RST	4(h)	0.569	0.512
LIWC	4(h)	0.818	0.572
text-CNN	4(h)	0.760	0.569
HAN	4(h)	0.860	0.672
AE	4(h)	0.786	0.743
Propagation2Vec	4(h)	0.808	0.789
Proposed Method	4(h)	**0.871**	**0.886**

In this section, the results of our experiments are presented to demonstrate the

proposed method's performance for detecting fake news within the first four hours of dissemination.

As it can be observed in Table 4.1, some of the methods are not appropriate for early detection (TCNN-URG, CSI, and DEFEND), i.e., they need the whole propagation network to be capable of predicting the label. Similarly, HPA-BLSTM uses the complete propagation network, but Propagation2Vec and AE need it only in the training phase, and at the time of inference, the partial propagation network (built during the first **five** hours) is enough. However, the introduced model uses only the partial propagation network (built within the first **four** hours) not only in the inference phase, but also when it is trained. Therefore, our model can detect fake news one hour earlier than the best baseline methods and does not need the whole propagation network.

The results for the GossipCop dataset show that the introduced model outperforms all baselines, especially the Propagation2Vec method, which is the best early detection one. Note that the Macro version of the F1 measurement is reported in this work because the fake and real classes in the GossipCop dataset are slightly imbalanced, and the Micro F1 (which is almost 2% higher than the Macro F1) can be misleading. Also, in our experiments, we found that on this dataset, the proposed method can achieve almost the same results using only context-based features.

As far as the PolitiFact dataset is concerned, the proposed method performs better than most of the baselines except for two approaches: DEFEND, which is not an early detection model, and Propagation2Vec, that attained around 2% higher values in all measurements. It seems that the insufficient number of instances in this dataset for appropriate training of a deep learning-based model is the main reason our model cannot achieve satisfactory results compared with the GossipCop dataset (Table 3.1). Also, according to Table 4.2, which represents the results for only early detection methods, the achieved F1 scores by our model on both datasets, GossipCop and

PolitiFact, are obviously higher than the baseline models, especially the best propagation based method (Propagation2Vec) by 9% and 6%, respectively. Note that the reported results in Table 4.2 are for a detection deadline of four hours.

4.8 Discussion

In the previous propagation-based work [52, 54], to generate a final embedding of the propagation network, simple averaging methods are adopted to aggregate node-level features, for example, the average time of all tweets or the average sentiment score of them. The problem with such methods is that the structure of the propagation network is ignored [55]. To solve this issue, Silva et al. [55] proposed an aggregation technique that can generate embeddings for nodes based on their neighbors, i.e., a summarization of their k-hop network. Also, in [56], a two-level (node and cascade) scoring method is introduced that generates embedding for the network by assigning scores to the neighbor nodes and cascades, considering their importance. However, the aforementioned method's limitation is that they only pay attention to its neighbor nodes while producing an embedding for a node. In practice, this means when the number of neighbor nodes is small (a short detection deadline leads to a small propagation network), the resulting embedding will not be informative enough.

In contrast, in the proposed method, each node's embedding is generated by paying varying attention to all nodes of the input sequence (propagation network). This is the main reason for this model's success when the detection deadline is considered short (that resulted in small propagation networks). This is precisely similar to the self-attention mechanism in the Transformer models like BERT [9] where the embedding of each word is produced considering the varied importance of all words in the input text. For example, in the translation task, the meaning of a word at position t in the source text is selected from the destination language's vocabulary

while varying attention is paid to all words of the input sequence instead of only the word at position t or its close neighbors in the sentence.

Another advantage of the proposed model is using cumulative features that can help to preserve information about the propagation network's structure.

4.9 Concluding Remarks

This chapter started by providing an overview of the experimental setup and the dataset used to obtain the results. It then discussed the different types of baseline models, which can be grouped into three categories: content-based, propagation network-based, and mixed methods. The chapter also explained the evaluation metrics used and the parameters of the proposed model. The evaluation results of the proposed model are presented in detail, and a comparison with the baseline models is provided. The results of the experiments demonstrated that the proposed model is effective at detecting fake news in the early stages of dissemination. The chapter also discussed how the model could achieve good results in the early hours of fake news propagation. In the next chapter, we present our conclusion and discuss our future works.

Chapter 5

Conclusion & Future Works

5.1 Conclusions

The widespread use of the internet and social media platforms has made it easy for anyone to disseminate information, regardless of its accuracy or validity. This has led to a high proportion of misinformation and fake news being shared on these platforms, which can have negative impacts on society and individuals. Many researchers have studied the fake news problem, but a small percentage of these works have aimed to detect fake news in the early stages of dissemination. In this thesis, we first review previous research on fake news detection, providing a detailed survey of the different techniques and approaches used for fake news detection. As a solution to this problem, in this thesis, we have proposed a new model exploiting the famous self-attention based mechanism for detecting fake news in the first four hours of dissemination. It means we can detect them one hour earlier than the best baseline. Also, we explained the approach used to build the input sequence for the model. The introduced framework converts the propagation network of each news article into a sequence of nodes, similar to a text consisting of tokens. We evaluated our method on two available datasets, GossipCop and PolitiFact, where on the first

dataset it outperformed all the state-of-the-art baselines by almost 2% in all measurements on average, but on the second dataset, our results were around 2% lower than Propagation2Vec method as the best early detection method. Note that the performance of the Propagation2Vec and AE models dropped by around 8.5% for detection deadlines earlier than five hours.

For future work, we plan to extract features about the users' timeline, which will probably provide our model with further helpful information on user behavior. Also, users participating in each cascade form a group and probably have some common specifications. Such information can be extracted as cascade-level features and appended to the end of each cascade as a new node. Also, some of the node-level features in this work, such as the number of re-tweets, the number of favorites, and the length of tweet are repeated in a cascade's nodes. These features can be removed from node-level features and added to the newly extracted cascade-level features.

5.2 Future Works

1. We plan to extract features about the users' timeline, which will probably provide our model with further helpful information on user behavior.

2. Also, users participating in each cascade form a group and probably have some common specifications. Such information can be extracted as cascade-level features and appended to the end of each cascade as a new node.

3. Moreover, some studies indicate that fake news spreaders produce mostly non-textual content because multi-modal fake news travels faster than just textual fake news. Therefore, we must utilize visual features in addition to social context and textual content-based features. We plan to extract such features and feed them to the proposed model or use an ensemble technique to combine them.

Bibliography

[1] *This analysis shows how viral fake election news stories outperformed real news on facebook.*

[2] Hunt Allcott and Matthew Gentzkow, *Social media and fake news in the 2016 election*, Journal of Economic Perspectives **31** (2017), no. 2, 211–36.

[3] Geoffrey Baym and Jeffrey P Jones, *News parody and political satire across the globe*, Routledge, 2013.

[4] Amirhosein Bodaghi and Jonice Oliveira, *The theater of fake news spreading, who plays which role? a study on real graphs of spreading on twitter*, Expert Systems with Applications **189** (2022), 116110.

[5] Alexandre Bovet and Hernán A Makse, *Influence of fake news in twitter during the 2016 us presidential election*, Nature communications **10** (2019), no. 1, 1–14.

[6] Girish Chandrashekar and Ferat Sahin, *A survey on feature selection methods*, Computers & Electrical Engineering **40** (2014), no. 1, 16–28.

[7] Francois Chollet, *Xception: Deep learning with depthwise separable convolutions*, 07 2017, pp. 1800–1807.

[8] N. R. de Oliveira, D. S. V. Medeiros, and D. M. F. Mattos, *A sensitive stylistic approach to identify fake news on social networking*, IEEE Signal Processing Letters **27** (2020), 1250–1254.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, arXiv preprint arXiv:1810.04805 (2018).

[10] Trevor Diehl and Sangwon Lee, *Testing the cognitive involvement hypothesis on social media:'news finds me'perceptions, partisanship, and fake news credibility*, Computers in Human Behavior **128** (2022), 107121.

[11] M. K. Elhadad, K. F. Li, and F. Gebali, *Detecting misleading information on covid-19*, IEEE Access **8** (2020), 165201–165215.

[12] Jacques Ellul and Konrad Kellen, *Propaganda: The formation of men's attitudes*, Vintage, 1973.

[13] Pedro Henrique Arruda Faustini and Thiago Ferreira Covões, *Fake news detection in multiple platforms and languages*, Expert Systems with Applications (2020), 113503.

[14] Luciano Floridi and Massimo Chiriatti, *Gpt-3: Its nature, scope, limits, and consequences*, Minds and Machines **30** (2020), no. 4, 681–694.

[15] Deen Freelon and Chris Wells, *Disinformation as political communication*, 2020, pp. 145–156.

[16] Nir Grinberg, Kenneth Joseph, Lisa Friedland, Briony Swire-Thompson, and David Lazer, *Fake news on twitter during the 2016 us presidential election*, Science **363** (2019), no. 6425, 374–378.

[17] David Güera and Edward J Delp, *Deepfake video detection using recurrent neural networks*, 2018 15th IEEE international conference on advanced video and signal based surveillance (AVSS), IEEE, 2018, pp. 1–6.

[18] Han Guo, Juan Cao, Yazi Zhang, Junbo Guo, and Jintao Li, *Rumor detection with hierarchical social attention network*, Proceedings of the 27th ACM international conference on information and knowledge management, 2018, pp. 943–951.

[19] Ashish Gupta, Han Li, Alireza Farnoush, and Wenting Jiang, *Understanding patterns of covid infodemic: A systematic and pragmatic approach to curb fake news*, Journal of business research **140** (2022), 670–683.

[20] Saqib Hakak, Mamoun Alazab, Suleman Khan, Thippa Reddy Gadekallu, Praveen Kumar Reddy Maddikunta, and Wazir Zada Khan, *An ensemble machine learning approach through effective feature extraction to classify fake news*, Future Generation Computer Systems **117** (2021), 47–58.

[21] Saqib Hakak, Wazir Zada Khan, Muhammad Imran, Kim-Kwang Raymond Choo, and Muhammad Shoaib, *Have you been a victim of covid-19-related cyber incidents? survey, taxonomy, and mitigation strategies*, Ieee Access **8** (2020), 124134–124144.

[22] Henderi Henderi, Tri Wahyuningsih, and Efana Rahwanto, *Comparison of min-max normalization and z-score normalization in the k-nearest neighbor (knn) algorithm to test the accuracy of types of breast cancer*, International Journal of Informatics and Information Systems **4** (2021), no. 1, 13–20.

[23] Yin-Fu Huang and Po-Hong Chen, *Fake news detection using an ensemble learning model based on self-adaptive harmony search algorithms*, Expert Systems with Applications (2020), 113584.

[24] Zhiwei Jin, Juan Cao, Yongdong Zhang, and Jiebo Luo, *News verification by exploiting conflicting social viewpoints in microblogs*, Proceedings of the AAAI conference on artificial intelligence, vol. 30, 2016.

[25] Zhiwei Jin, Juan Cao, Yongdong Zhang, Jianshe Zhou, and Qi Tian, *Novel visual and statistical image features for microblogs news verification*, IEEE transactions on multimedia **19** (2016), no. 3, 598–608.

[26] Rohit Kumar Kaliyar, Anurag Goswami, Pratik Narang, and Soumendu Sinha, *Fndnet–a deep convolutional neural network for fake news detection*, Cognitive Systems Research **61** (2020), 32–44.

[27] S. Kausar, B. Tahir, and M. A. Mehmood, *Prosoul: A framework to identify propaganda from online urdu content*, IEEE Access **8** (2020), 186039–186054.

[28] Vlado Keselj, *Speech and language processing daniel jurafsky and james h. martin (stanford university and university of colorado at boulder) pearson prentice hall, 2009, xxxi+ 988 pp; hardbound, isbn 978-0-13-187321-6,* 115.00, 2009.

[29] Jan Kietzmann, Adam J Mills, and Kirk Plangger, *Deepfakes: perspectives on the future "reality" of advertising and branding*, International Journal of Advertising **40** (2021), no. 3, 473–485.

[30] Y Kim et al., *Convolutional neural networks for sentence classification. arxxiv, doi*, arXiv preprint arXiv:1408.5882 (2014).

[31] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, *Deep learning*, nature **521** (2015), no. 7553, 436–444.

[32] Qian Li, Qingyuan Hu, Youshui Lu, Yue Yang, and Jingxian Cheng, *Multi-level word features based on cnn for fake news detection in cultural communication*, Personal and Ubiquitous Computing (2019), 1–14.

[33] Yang Liu and Yi-Fang Brook Wu, *Fned: A deep network for fake news early detection on social media*, ACM Transactions on Information Systems (TOIS) **38** (2020), no. 3, 1–33.

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

[35] Jing Ma, Wei Gao, and Kam-Fai Wong, *Detect rumors in microblog posts using propagation structure via kernel learning*, Association for Computational Linguistics.

[36] Falko Matern, Christian Riess, and Marc Stamminger, *Exploiting visual artifacts to expose deepfakes and face manipulations*, 01 2019, pp. 83–92.

[37] Tom M Mitchell and Tom M Mitchell, *Machine learning*, vol. 1, McGraw-hill New York, 1997.

[38] Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael M Bronstein, *Fake news detection on social media using geometric deep learning*, arXiv preprint arXiv:1902.06673 (2019).

[39] Lakshmanan Nataraj, Tajuddin Manhar Mohammed, B. Manjunath, Shivkumar Chandrasekaran, Arjuna Flenner, Md Jawadul Bappy, and Amit Roy-Chowdhury, *Detecting gan generated fake images using co-occurrence matrices*, Electronic Imaging **2019** (2019), 532–1.

[40] Joao C Neves, Ruben Tolosana, Ruben Vera-Rodriguez, Vasco Lopes, Hugo Proença, and Julian Fierrez, *Ganprintr: Improved fakes and evaluation of the state of the art in face manipulation detection*, IEEE Journal of Selected Topics in Signal Processing **14** (2020), no. 5, 1038–1048.

[41] Feyza Altunbey Ozbay and Bilal Alatas, *Fake news detection within online social media using supervised artificial intelligence algorithms*, Physica A: Statistical Mechanics and its Applications **540** (2020), 123174.

[42] James W Pennebaker, Ryan L Boyd, Kayla Jordan, and Kate Blackburn, *The development and psychometric properties of liwc2015*, Tech. report, 2015.

[43] Feng Qian, Chengyue Gong, Karishma Sharma, and Yan Liu, *Neural user response generator: Fake news detection with collective user intelligence.*, IJCAI, vol. 18, 2018, pp. 3834–3840.

[44] Arne Roets et al., *'fake news': Incorrect, but hard to correct. the role of cognitive ability on the impact of false information on social impressions*, Intelligence **65** (2017), 107–110.

[45] Victoria L Rubin, Niall J Conroy, and Yimin Chen, *Towards news verification: Deception detection methods for news discourse*, Hawaii International Conference on System Sciences, 2015, pp. 5–8.

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

[47] Tal Schuster, Roei Schuster, Darsh J Shah, and Regina Barzilay, *The limitations of stylometry for detecting machine-generated fake news*, Computational Linguistics (2020), 1–12.

[48] Wajiha Shahid, Bahman Jamshidi, Saqib Hakak, Haruna Isah, Wazir Zada Khan, Muhammad Khurram Khan, and Kim-Kwang Raymond Choo, *Detecting and mitigating the dissemination of fake news: Challenges and future research opportunities*, IEEE Transactions on Computational Social Systems (2022).

[49] Kai Shu, Limeng Cui, Suhang Wang, Dongwon Lee, and Huan Liu, *defend: Explainable fake news detection*, Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 395–405.

[50] Kai Shu and Huan Liu, *Detecting fake news on social media*, Synthesis lectures on data mining and knowledge discovery **11** (2019), no. 3, 1–129.

[51] Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu, *Fakenewsnet: A data repository with news content, social context and spatial-temporal information for studying fake news on social media*, arXiv preprint arXiv:1809.01286 (2018).

[52] Kai Shu, Deepak Mahudeswaran, Suhang Wang, and Huan Liu, *Hierarchical propagation networks for fake news detection: Investigation and exploitation*, Proceedings of the international AAAI conference on web and social media, vol. 14, 2020, pp. 626–637.

[53] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu, *Fake news detection on social media: A data mining perspective*, ACM SIGKDD explorations newsletter **19** (2017), no. 1, 22–36.

[54] Kai Shu, Guoqing Zheng, Yichuan Li, Subhabrata Mukherjee, Ahmed Hassan Awadallah, Scott Ruston, and Huan Liu, *Leveraging multi-source weak social supervision for early detection of fake news*, arXiv preprint arXiv:2004.01732 (2020).

[55] Amila Silva, Yi Han, Ling Luo, Shanika Karunasekera, and Christopher Leckie, *Embedding partial propagation network for fake news early detection.*, CIKM (Workshops), 2020.

[56] ———, *Propagation2vec: Embedding partial propagation networks for explainable fake news early detection*, Information Processing & Management **58** (2021), no. 5, 102618.

[57] Renato M Silva, Roney LS Santos, Tiago A Almeida, and Thiago AS Pardo, *Towards automatically filtering fake news in portuguese*, Expert Systems with Applications **146** (2020), 113199.

[58] Shivangi Singhal, Rajiv Ratn Shah, Tanmoy Chakraborty, Ponnurangam Kumaraguru, and Shin'ichi Satoh, *Spotfake: A multi-modal framework for fake news detection*, 2019 IEEE fifth international conference on multimedia big data (BigMM), IEEE, 2019, pp. 39–47.

[59] Eugenio Tacchini, Gabriele Ballarin, Marco L Della Vedova, Stefano Moret, and Luca De Alfaro, *Some like it hoax: Automated fake news detection in social networks*, arXiv preprint arXiv:1704.07506 (2017).

[60] Alexandros Tsakalidis, *Misinformation in eu elections 2019: A post analysis*, (2020).

[61] M. Umer, Z. Imtiaz, S. Ullah, A. Mehmood, G. S. Choi, and B. W. On, *Fake news stance detection using deep learning architecture (cnn-lstm)*, IEEE Access **8** (2020), 156695–156706.

[62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, *Attention is all you need*, Advances in neural information processing systems **30** (2017).

[63] William Yang Wang, *" liar, liar pants on fire": A new benchmark dataset for fake news detection*, arXiv preprint arXiv:1705.00648 (2017).

[64] Yaqing Wang, Fenglong Ma, Zhiwei Jin, Ye Yuan, Guangxu Xun, Kishlay Jha, Lu Su, and Jing Gao, *Eann: Event adversarial neural networks for multi-modal fake news detection*, Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining, 2018, pp. 849–857.

[65] Yuhang Wang, Li Wang, Yanjie Yang, and Tao Lian, *Semseq4fd: Integrating global semantic relationship and local sequential order to enhance text representation for fake news detection*, Expert Systems with Applications **166** (2020), 114090.

[66] K. Xu, F. Wang, H. Wang, and B. Yang, *Detecting fake news over online social media via domain reputations and content understanding*, Tsinghua Science and Technology **25** (2020), no. 1, 20–27.

[67] Shuo Yang, Kai Shu, Suhang Wang, Renjie Gu, Fan Wu, and Huan Liu, *Unsupervised fake news detection on social media: A generative approach*, Proceedings of the AAAI conference on artificial intelligence, vol. 33, 2019, pp. 5644–5651.

[68] Yang Yang, Lei Zheng, Jiawei Zhang, Qingcai Cui, Zhoujun Li, and Philip S Yu, *Ti-cnn: Convolutional neural networks for fake news detection*, arXiv preprint arXiv:1806.00749 (2018).

[69] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy, *Hierarchical attention networks for document classification*, Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies, 2016, pp. 1480–1489.

[70] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi, *Defending against neural fake news*, Advances in neural information processing systems, pp. 9054–9065.

[71] Zilong Zhao, Jichang Zhao, Yukie Sano, Orr Levy, Hideki Takayasu, Misako Takayasu, Daqing Li, Junjie Wu, and Shlomo Havlin, *Fake news propagates differently from real news even at early stages of spreading*, EPJ Data Science **9** (2020), no. 1, 7.

[72] Lei Zheng, Jon D Elhai, Miao Miao, Yu Wang, Yiwen Wang, and Yiqun Gan, *Health-related fake news during the covid-19 pandemic: perceived trust and information search*, Internet Research (2022).

[73] Xinyi Zhou, Atishay Jain, Vir V Phoha, and Reza Zafarani, *Fake news early detection: A theory-driven model*, Digital Threats: Research and Practice **1** (2020), no. 2, 1–25.

[74] Xinyi Zhou, Jindi Wu, and Reza Zafarani, *Safe: similarity-aware multi-modal fake news detection (2020)*, Preprint. arXiv **200304981** (2020).

[75] Xinyi Zhou and Reza Zafarani, *A survey of fake news: Fundamental theories, detection methods, and opportunities*, ACM Computing Surveys (CSUR) **53** (2020), no. 5, 1–40.

[76] Arkaitz Zubiaga and Aiqi Jiang, *Early detection of social media hoaxes at scale*, ACM Transactions on the Web (TWEB) **14** (2020), no. 4, 1–23.

Appendix A

In this appendix, some essential parts of our implementation are provided.

```
# This function extracts and calculates tweets features
def feature_extractor(sid_obj, positives, negatives, data, new_record):
    new_record['created_at'] = data['created_at']
    new_record['tweet_id_str'] = data['id_str']
    new_record['text'] = data['text']

    # Calculate the sentiment value of the tweet
    sentiment_dict = sid_obj.polarity_scores(data['text'])
    new_record['sentiment_compound'] = sentiment_dict['compound']

    # Count the frequency of positive and negative words in each tweet
    tokens = data['text'].split()
    pos_count = 0
    neg_count = 0
    for token in tokens:
        if token.strip().lower() in positives:
            pos_count += 1
        elif token.strip().lower() in negatives:
            neg_count += 1

    new_record['pos_count'] = pos_count
    new_record['neg_count'] = neg_count
    new_record['hashtag_count'] = len(data['entities']['hashtags'])
    new_record['mention_count'] = len(data['entities']['user_mentions'])
    new_record['user_id_str'] = data['user']['id_str']
    new_record['retweet_count'] = data['retweet_count']
    new_record['favorite_count'] = data['favorite_count']
    new_record['tweet_type'] = determine_tweet_type(data)
```

Figure 5.1: Code snippet for extracting and calculating tweet's features.

```
df = pd.DataFrame(columns=['news_id', 'created_at', 'tweet_id_str', 'text',
                    'sentiment_compound', 'pos_count', 'neg_count',
                    'hashtag_count', 'mention_count', 'user_id_str',
                    'retweet_count', 'favorite_count', 'tweet_type'])

# Creating a sentiment analyzer object to calculate sentiment_compound
# of each tweet and retweet's text. Retweets can be quated tweets or replys
sid_obj = SentimentIntensityAnalyzer()

# Loading two lists of positive and negative words in order to count the
# number of positive and negative words in each tweet and retweet
file_positives = open("positive.txt",'r')
positives = file_positives.read().replace("\n"," ").lower().split()
file_negatives = open("negative.txt",'r')
negatives = file_negatives.read().replace("\n"," ").lower().split()

for news in news_list:
    tweets = [f for f in os.listdir(os.path.join(source, news, 'tweets'))]

    for tweet in tweets:
        # Opening JSON file of each tweet and extracting its important features
        f = open(os.path.join(source, news, 'tweets', tweet))
        data = json.load(f)

        new_record = {}
        new_record['news_id'] = news
        feature_extractor(sid_obj, positives, negatives, data, new_record)

        f.close()

        row_df = pd.DataFrame([new_record])
        df = pd.concat([df,row_df], ignore_index=True)
```

Figure 5.2: Code snippet for creating a table for tweets by extracting their features from JSON files.

```
df = pd.DataFrame(columns=['id_str', 're_created_at', 're_id_str', 're_text',
            're_sentiment_compound', 're_pos_count', 're_neg_count',
            're_hashtag_count', 're_mention_count', 're_user_id_str',
            're_retweet_count', 're_favorite_count', 'retweet_type'])

# Creating a sentiment analyzer object to calculate sentiment_compound
# of each tweet and retweet's text. Retweets can be quated tweets or replys
sid_obj = SentimentIntensityAnalyzer()

# Loading two lists of positive and negative words in order to count the
# number of positive and negative words in each tweet and retweet
file_positives = open("positive.txt",'r')
positives = file_positives.read().replace("\n"," ").lower().split()
file_negatives = open("negative.txt",'r')
negatives = file_negatives.read().replace("\n"," ").lower().split()

for news in news_list:
    tweets = [f for f in os.listdir(os.path.join(source, news, 'tweets'))]

    for tweet in tweets:
        # Opening JSON file of each tweet and extracting its id
        f = open(os.path.join(source, news, 'tweets', tweet))
        data_tweet = json.load(f)
        f.close()

        # Opening JSON file of each retweet and extracting important features
        retweets = os.path.join(source, news, 'retweets', tweet)
        if os.path.exists(retweets):
            f = open(retweets)
            data_retweet = json.load(f)
            if (len(data_retweet['retweets'])) > 0:
                for retweet in data_retweet['retweets']:
                    new_record = {}
                    new_record['id_str'] = data_tweet['id_str']
                    feature_extractor(sid_obj, positives, negatives,
                                      data_retweet, new_record)

                    f.close()
                    row_df = pd.DataFrame([new_record])
                    df = pd.concat([df,row_df], ignore_index=True)
```

Figure 5.3: Code snippet for creating a table for re-tweets by extracting their features from JSON files.

```
# This function extracts and calculates tweets features
def user_feature_extractor(data, new_record):
    new_record['user_id_str'] = data['user']['id_str']
    new_record['user_followers_count'] = data['user']['followers_count']
    new_record['user_friends_count'] = data['user']['friends_count']
    new_record['user_listed_count'] = data['user']['listed_count']
    new_record['user_created_at'] = data['user']['created_at']
    new_record['user_favourites_count'] = data['user']['favourites_count']
    new_record['user_verified'] = data['user']['verified']
    new_record['user_statuses_count'] = data['user']['statuses_count']
```

Figure 5.4: Code snippet for extracting user's features.

```
df = pd.DataFrame(columns=['user_id_str', 'user_followers_count',
            'user_friends_count', 'user_listed_count', 'user_created_at',
            'user_favourites_count', 'user_verified', 'user_statuses_count'])

for news in news_list:
    tweets = [f for f in os.listdir(os.path.join(source, news, 'tweets'))]
    for tweet in tweets:

        # Extract important features of each tweet's user
        f = open(os.path.join(source, news, 'tweets', tweet))
        data = json.load(f)
        new_record = {}
        user_feature_extractor(data, new_record)
        f.close()

        row_df = pd.DataFrame([new_record])
        df = pd.concat([df,row_df], ignore_index=True)

        # Extract important features of each retweet's user
        retweets = os.path.join(source, news, 'retweets', tweet)
        if os.path.exists(retweets):
            f = open(retweets)

            data_retweet = json.load(f)
            if (len(data_retweet['retweets'])) > 0:
                for retweet in data_retweet['retweets']:
                    new_record = {}
                    user_feature_extractor(retweet, new_record)
                    row_df = pd.DataFrame([new_record])
                    df = pd.concat([df,row_df], ignore_index=True)

            f.close()
```

Figure 5.5: Code snippet for creating a table for users by extracting their features from JSON files.

```
dataset_name = '{}_{}_counting_{}hours.json'.format(dataset, label_or_class,
                                                    hours)
df = pd.read_json(dataset_name, orient ='table', compression = 'infer')

# Counting the re-tweets of each tweet belonging to each news
df['count'] = df.groupby(['id_news',
                          'id_str'])['re_id_str'].transform('count')

df_counts = df.drop(['created_at', 're_created_at', 're_text',
    'new_re_created_at', 're_user_id_str', 're_retweet_count',
    're_favorite_count', 'diff', 'new_created_at', 're_id_str', 'zero_time',
    'diff_t', 'diff_r'], axis=1).drop_duplicates()

# Creating a propagation tree for each news article without trancating trees
# and branches, but padding is applied.
propagation_trees = {}
cascade_len = 16   # 2^3
cascade_num = 256 # 2^7

for news_id in news_ids:
    propagation_trees[news_id] = []
    # Collecting all tweets for each news
    # ascending=True means selecting the shortest branches
    t_ids = list(df_counts[df_counts['id_news'] == news_id]\
                .sort_values(['count'], ascending=True)['id_str'])

    # Collecting all re-tweets for each tweet of each news
    for t_id in t_ids:
        r_ids = list(df[(df['id_news'] == news_id) & (df['id_str'] == t_id)]\
                     .sort_values(['diff'],
                                  ascending=True)['re_id_str'])

        r_ids.insert(0, t_id)

        # adding 'pad' ids to the short cascades untill their
        # length becomes cascade_len
        r_ids += ['pad'] * (cascade_len - len(r_ids))
        propagation_trees[news_id].append(r_ids)

    # add ['pad', ..., 'pad'] cascades to the trees untill their
    # cascade number become cascade_num
    cascade_pad = cascade_num - len(propagation_trees[news_id])
    propagation_trees[news_id] += [['pad'] * (cascade_len)] * cascade_pad
```

Figure 5.6: Code snippet for constructing each news article's propagation tree or network. This step's created tree contains only the nodes' ids.

```
def retrieve_features(tweet_fields, new_cascade, t_or_re, df_users):
    # Getting the creation time and converting itno UTC
    x = tweet_fields.iloc[0][t_or_re + 'created_at']
    tweet_created_at = time.mktime(time.strptime(str(x),
                                    "%Y-%m-%d %H:%M:%S+00:00"))
    # 0- Adding created_at feature of the tweet to the node
    new_cascade.append(tweet_created_at)
    # 1- Adding retweet_count feature of the tweet to the node
    new_cascade.append(tweet_fields.iloc[0][t_or_re + 'retweet_count'])
    # 2- Adding favorite_count feature of the tweet to the node
    new_cascade.append(tweet_fields.iloc[0][t_or_re + 'favorite_count'])
    # 3- Adding tweet_length feature to the node
    new_cascade.append(len(tweet_fields.iloc[0][t_or_re + 'text']))

    # Retrieving tweet's user fields and adding the important ones
    tweet_user_id = tweet_fields.iloc[0][t_or_re + 'user_id_str']
    # The result dataframe might have multiple rows
    user_fields = df_users[df_users['user_id_str'] == tweet_user_id]
    # 4- Adding user_verified feature to the node (1:verified, 0:unverified)
    new_cascade.append(int(user_fields.iloc[0]['user_verified']))
    # 5- Adding user_followers_count feature to the node
    user_followers_count = round(sum(user_fields['user_followers_count'])/\
                                 len(user_fields['user_followers_count']))
    new_cascade.append(user_followers_count)
    # 6- Adding user_friends_count feature to the node
    user_friends_count = round(sum(user_fields['user_friends_count'])/\
                               len(user_fields['user_friends_count']))
    new_cascade.append(user_friends_count)
    # 7- Adding user_listed_count feature to the node
    user_listed_count = round(sum(user_fields['user_listed_count'])/\
                              len(user_fields['user_listed_count']))
    new_cascade.append(user_listed_count)
    # 8- Adding created_at feature of the tweet's user to the node
    x = user_fields.iloc[0]['user_created_at']
    user_created_at = time.mktime(time.strptime(str(x),
                                                "%Y-%m-%d %H:%M:%S+00:00"))
    new_cascade.append(user_created_at)
    # 9- Adding user_favourites_count feature to the node
    user_favourites_count = round(sum(user_fields['user_favourites_count'])/\
                                  len(user_fields['user_favourites_count']))
    new_cascade.append(user_favourites_count)
    # 10- Adding user_statuses_count feature to the node
    user_statuses_count = round(sum(user_fields['user_statuses_count'])/\
                                len(user_fields['user_statuses_count']))
    new_cascade.append(user_statuses_count)
```

Figure 5.7: Code snippet for retrieving 11 node-level features from the database.

```
# Creating a propagation tree for each news article without trancating trees
main_propagation_trees = {}

cascade_len = 16   # 2^3
cascade_num = 256 # 2^7
feature_num = 11
cascade_len *= feature_num

for news in propagation_trees.keys():
    main_propagation_trees[news] = []

    for cascade in propagation_trees[news]:
        new_cascade = []
        # When a tweet does not have any retweets, the first retweet
        # becomes nan (I didn't handle this issue when I was retriving
        # retweets for each tweet) and the rest retweets become 'pad'
        cascade = ['pad' if str(i) == 'nan' else i for i in cascade]

        if cascade[0] != 'pad':
            # Retrieving tweet fields and adding the important ones
            tweet_fields = df_tweets[df_tweets['tweet_id_str'] == cascade[0]]

            tweet_or_retweet = ''
            retrieve_features(tweet_fields, new_cascade, tweet_or_retweet,
                              df_users)

            for retweet in cascade[1:]:
                if retweet == 'pad':
                    continue
                # Retrieving retweet fields and adding the important ones
                retweet_fields=df_retweets[df_retweets['re_id_str']==retweet]

                tweet_or_retweet = 're_'
                retrieve_features(retweet_fields, new_cascade,
                                  tweet_or_retweet, df_users)

        # add 'pad' ids to the short cascades untill their length
        # becomes cascade_len
        new_cascade += ['pad'] * (cascade_len - len(new_cascade))
        main_propagation_trees[news].append(new_cascade)

    # add ['pad', ..., 'pad'] cascades to the trees untill their
    # cascade number become cascade_num
    cascade_pad = cascade_num - len(main_propagation_trees[news])
    main_propagation_trees[news] += [['pad'] * cascade_len] * cascade_pad
```

Figure 5.8: Code snippet for embedding 11 node-level features into the previous step's constructed propagation tree.

```python
class MultiHeadedAttention(nn.Module):
    def __init__(self, n_heads, d_model, dropout=0.1):
        super(MultiHeadedAttention, self).__init__()
        self.n_heads = n_heads
        self.d_model = d_model
        self.d_k = int(d_model / n_heads)
        self.linear_query = nn.Linear(d_model, d_model)
        self.linear_key = nn.Linear(d_model, d_model)
        self.linear_value = nn.Linear(d_model, d_model)
        self.linear_out = nn.Linear(d_model, d_model)
        self.dropout = nn.Dropout(p=dropout)
        self.alphas = None

    def make_chunks(self, x):
        batch_size, seq_len = x.size(0), x.size(1)
        x = x.view(batch_size, seq_len, self.n_heads, self.d_k)
        return (x.transpose(1, 2)) # N, n_heads, L, d_k

    def init_keys(self, key):
        self.proj_key = self.make_chunks(self.linear_key(key)) # N,n_heads,L,d_k
        self.proj_value = self.make_chunks(self.linear_value(key))

    def score_function(self, query):
        proj_query = self.make_chunks(self.linear_query(query))
        dot_products = torch.matmul(proj_query, self.proj_key.transpose(-2, -1))
        return (dot_products / np.sqrt(self.d_k))

    def attn(self, query, mask=None):
        scores = self.score_function(query) # N, n_heads, L, L
        if mask is not None:
            scores = scores.masked_fill(mask == 0, -1e9)
        alphas = F.softmax(scores, dim=-1) # N, n_heads, L, L
        alphas = self.dropout(alphas)
        self.alphas = alphas.detach()
        return (torch.matmul(alphas, self.proj_value))

    def output_function(self, contexts):
        return (self.linear_out(contexts)) # N, L, D

    def forward(self, query, mask=None):
        if mask is not None: mask = mask.unsqueeze(1)
        context = self.attn(query, mask=mask) # N, n_heads, L, d_k
        context = context.transpose(1, 2).contiguous() # N, L, n_heads, d_k
        context = context.view(query.size(0), -1, self.d_model)
        out = self.output_function(context) # N, L, d_model
        return out
```

Figure 5.9: Code snippet for implementing a self-attention mechanism class with multi head. This class is used in the proposed model's encoder layer.

```
class EncoderLayer(nn.Module):
    def __init__(self, n_heads, d_model, ff_units, dropout=0.1):
        super().__init__()
        self.n_heads = n_heads
        self.d_model = d_model
        self.ff_units = ff_units
        self.self_attn_heads = MultiHeadedAttention(n_heads, d_model,
                                                    dropout=dropout)
        self.ffn = nn.Sequential(
            nn.Linear(d_model, ff_units),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(ff_units, d_model),
        )

        self.norm1 = nn.LayerNorm(d_model)
        self.norm2 = nn.LayerNorm(d_model)
        self.drop1 = nn.Dropout(dropout)
        self.drop2 = nn.Dropout(dropout)

    def forward(self, query, mask=None):
        # Sublayer #0
        # Norm
        norm_query = self.norm1(query)
        # Multi-headed Attention
        self.self_attn_heads.init_keys(norm_query)
        states = self.self_attn_heads(norm_query, mask)
        # Add
        att = query + self.drop1(states)

        # Sublayer #1
        # Norm
        norm_att = self.norm2(att)
        # Feed Forward
        out = self.ffn(norm_att)
        # Add
        out = att + self.drop2(out)
        return out
```

Figure 5.10: Code snippet for implementing an encoder layer class with two sublayers. This class is used in the proposed model.

```
class PositionalEncoding2(nn.Module):
    def __init__(self, max_len, d_model):
        super().__init__()

        cascade_len = 5
        self.d_model = d_model
        pe = torch.zeros(max_len, d_model)

        for i in range(int(max_len/cascade_len)):
            position = torch.arange(0, cascade_len).float().unsqueeze(1)
            angular_speed = torch.exp(torch.arange(0, d_model, 2).float() *
                                      (-np.log(10000.0) / d_model))

            # even dimensions
            pe[(i*cascade_len):((i+1)*cascade_len), 0::2] = torch.sin(
                                                    position * angular_speed)

            # odd dimensions
            pe[(i*cascade_len):((i+1)*cascade_len), 1::2] = torch.cos(
                                                    position * angular_speed)

        self.register_buffer('pe', pe.unsqueeze(0))

    def forward(self, x):
        # x is N, L, D
        # pe is 1, maxlen, D
        scaled_x = x * np.sqrt(self.d_model)
        encoded = scaled_x + self.pe[:, :x.size(1), :]
        return encoded
```

Figure 5.11: Code snippet for implementing a new kind of positional encoding (explained in the thesis). This class is also used in the proposed model.

```
class ProposedModel(nn.Module):
    def __init__(self, encoder_layer, n_layers=2, max_len=500,
                 n_features=9, n_outputs=1):
        super().__init__()

        self.d_model = encoder_layer.d_model
        self.n_outputs = n_outputs
        self.n_features= n_features
        self.pe = PositionalEncoding2(max_len, self.d_model)
        self.norm = nn.LayerNorm(self.d_model)
        self.layers = nn.ModuleList([copy.deepcopy(encoder_layer)
                                     for _ in range(n_layers)])

        # Classifier to produce as many logits as outputs
        self.classifier = nn.Linear(2 * self.d_model,
                                    self.n_outputs)

        self.proj = nn.Linear(n_features, self.d_model)

    def forward(self, query, mask=None):
        # Projection
        query_proj = self.proj(query)

        mask = (query != -1).all(axis=2).unsqueeze(1)

        # Positional Encoding
        x = self.pe(query_proj)
        for layer in self.layers:
            x = layer(x, mask)

        out = self.norm(x)
        # only first and last items of cascades in
        # sequence (N, L(cascade_n*cascade_l), D)
        output = out[:, ::out.size()[1] - 1, :].reshape(out.size()[0],
                                                        2 * out.size()[2])


        # classifier will output (N, 1, n_outputs)
        out = self.classifier(output)

        # final output is (N, n_outputs)
        return out.view(-1, self.n_outputs)
```

Figure 5.12: Code snippet for implementing the proposed model's class. PositionalEncoding2 and EncoderLayer classes are used in this class.

```
# This version selects the first 100 cascades. They can be the shortest or the
# longest depending on used sorting type in propagation tree construction step
def change_shape(node_len, pad_num):
    def tree_to_list(propagation_trees, label):
        cascade_len = 5
        cascade_num = 100
        # The list of instances for passing to the model
        instances = []

        final_propagation_trees = {}
        for news in propagation_trees.keys():
            new_cascade = []
            cascade_count = 0
            for a in propagation_trees[news]:
                # replace pad with -1 in the current cascade of features
                # with the length of cascade_len * node_len
                a = [pad_num if i=='pad' else i for i in a]

                # convert a list of ints to a numpy.array of floats
                a = np.array(a)
                a = a.astype(float)

                # change the shape of current cascade from cascade_len * node_len
                # to (cascade_len, node_len)
                a = [a[i * node_len:(i + 1) * node_len] for i \
                     in range((len(a) + node_len - 1) // node_len )]
                # The length of cascade is cascade_len(8), but in line below
                # the first 5 nodes are considered as a new cascade
                new_cascade.extend(a[:cascade_len])

                # The counts of cascades is 128, but in lines below the first 100
                # cascades are considered
                cascade_count += 1
                if cascade_count == cascade_num:
                    break
            new_cascade = np.array(new_cascade)
            final_propagation_trees[news] = new_cascade

        for news in final_propagation_trees.keys():
            # (news_id, news_propagation, news_label)
            instances.append([news, final_propagation_trees[news], label])

        return instances

    return tree_to_list
```

Figure 5.13: Code snippet for converting a propagation tree into a sequence.

Vita

Candidate's full name: Bahman Jamshidi

University attended :
Bachelor of Software Engineering
Technical and Vocational University,
2004-2008

Publications:
Jamshidi B., Hakak S., Lu R.: **"A Self-attention Mechanism based Model for Early Detection of Fake News"** – Under Review, IEEE Transactions on Computational Social Systems

Shahid W., **Jamshidi B.**, Hakak S., Isah H., Khan WZ., Choo KKR.: **"Detecting and Mitigating the Dissemination of Fake News: Challenges and Future Research Opportunities"** – 2022, IEEE Transactions on Computational Social Systems

Poster presentations:
Jamshidi B., Hakak S.; **"Early Fake News Detection"**
19th Annual International Conference on Privacy, Security & Trust; August 2022; Fredericton, New Brunswick, Canada.