

DEEP LEARNING MODELS FOR DETECTING ONLINE HARMFUL CONTENT

Feng Wei

A DISSERTATION SUBMITTED TO THE FACULTY OF GRADUATE
STUDIES IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR
THE DEGREE OF

DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN DEPARTMENT OF ELECTRICAL
ENGINEERING AND COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

August 2025

© Feng Wei, 2025

Abstract

Deep learning (DL) has emerged as a transformative technology with substantial impact across various domains, including cybersecurity. This dissertation leverages deep learning methods and their applications to address increasingly sophisticated cyber threats. DL methods are capable of learning complex and abstract features from large-scale data, making them well-suited for identifying and mitigating cyber threats that traditional methods might miss. This dissertation focuses on the practical implementation and evaluation of DL models for detecting real-world cybersecurity threats, namely, clickbait, Twitter bots and SMS spam. Specifically, we propose:

- a novel attention-based neural network model named Knowledge-Enhanced Clickbait Detector (KED) that uses linguistic knowledge graphs built from WordNet to guide the attention mechanisms. The proposed neural network can effectively capture discriminative features from local and global similarities via the proposed knowledge-enhanced attention mechanisms. Moreover, we incorporate human semantic knowledge into the neural network and its attention mechanisms to better capture semantic correlations of headline-article word pairs.
- a novel recurrent neural network (RNN) model to distinguish Twitter bots from human accounts based on textual content of their tweets. We use several types of linguistic embeddings to encode tweets, namely, word embeddings, character embeddings, part-of-speech embeddings, and named-entity embeddings. We avoid using handcrafted features, which require time-consuming and labor-intensive feature engineering. This advantage allows for faster and easier implementation and deployment of the bot detection scheme.
- a novel lightweight deep neural model called Lightweight Gated Recurrent Unit (LGRU) for SMS spam detection. We incorporate enhancing semantics retrieved from external knowledge to assist in understanding SMS text inputs for more accurate detection.

In addition, the lightweight model illustrates a method to minimize unnecessary complexity in training recurrent models without compromising the performance, which we believe is applicable to many other complex recurrent models for other applications.

Experimental results show that the above models outperform their counterparts, including state-of-the-art models/systems and other baseline models, in terms of predictive performance and/or running time. The proposed models provide robust, scalable, and real-time security solutions that can adapt to the rapidly changing landscape of cyber threats.

Acknowledgments

I would like to express my deepest gratitude to my thesis advisor, Professor Uyen Trang Nguyen, whose unwavering support and expert guidance have been invaluable throughout my Ph.D. journey. Her insightful feedback, profound knowledge, and patience have significantly shaped my research and personal growth. This dissertation would not have been possible without her mentorship and encouragement.

I am also deeply grateful to Professor Suprakash Datta for his insightful comments and continuous support throughout my Ph.D. studies. His guidance has been invaluable in shaping this research and enhancing its rigor.

I am also sincerely thankful to the members of my thesis examination committee, Professors Charlie Obimbo (external examiner), Mokhtar Aboelaze (supervisory committee), Richard Hornsey (supervisory committee), and Regina Lee (internal member), for their constructive feedback and thoughtful suggestions. Their expertise and advice have been instrumental in refining my work and ensuring its quality.

I would like to extend my heartfelt thanks to our current and former graduate program directors and all administrative staff for their constant support and assistance, which have been crucial in enabling me to complete this PhD program.

I am extremely grateful to all technical staff for their invaluable technical support, which was instrumental in the progress of my research.

I am also greatly thankful to my excellent lab mates for their collegiality, collaboration and assistance.

I would like to thank my friends who have always pushed me to go further. I would like to express my gratitude to my colleagues at work

Last but not least, I would like to thank my parents for their unconditional love.

Table of Contents

Abstract	II
Acknowledgments	IV
Table of Contents	V
List of Tables	IX
List of Figures	XIII
List of Symbols	XV
List of Abbreviations	XVI
Publications from the Dissertation	XVII
1 Introduction	1
1.1 Background and Context	1
1.2 Motivations and Contributions of the Dissertation	3
1.2.1 Motivations	3
1.2.2 Detailed Contributions	10

1.3	Organization of the Dissertation	14
2	Literature Review	15
2.1	Applications of Deep Learning in Cybersecurity	15
2.1.1	Applications of Recurrent Neural Network to Cybersecurity	16
2.1.2	Applications of Convolutional Neural Network to Cybersecurity	20
2.2	Related Work	23
2.2.1	Twitter Bot Detection	24
2.2.2	Clickbait Detection	27
2.2.3	SMS Spam Detection	30
3	Twitter Bot Detection Using Neural Networks and Linguistic Embeddings	34
3.1	Motivations and Contributions	34
3.2	The Proposed Model	35
3.2.1	Linguistic Embeddings	36
3.2.2	BiLGRU Neural Networks	40
3.3	Experimental Results	44
3.3.1	Datasets	45
3.3.2	Existing Systems for Comparison	47
3.3.3	Neural Network Model Setup	48
3.3.4	Evaluation Metrics	49
3.3.5	Results and Discussion	50
3.4	Effectiveness of Linguistic Embeddings	52
3.4.1	Linguistic Embeddings: Detection Performance	52

3.4.2	Linguistic Embeddings: Running Time and Memory Usage	52
3.5	Performance Comparison of Different Recurrent Variants	55
3.5.1	RNN Variants: Detection Performance	56
3.5.2	RNN Variants: Running Time and Memory Usage	57
3.6	Chapter Summary	58
4	An Attention-Based Neural Network Using Human Semantic Knowledge and Its Application to Clickbait Detection	59
4.1	Motivations and Contributions	59
4.2	The Proposed Attention-based Neural Network	60
4.2.1	Human Semantic Knowledge Extraction Using WordNet	61
4.2.2	The Proposed Knowledge Enhanced Attention Mechanisms	62
4.3	Applying the Proposed Attention-based Neural Network to Clickbait Detection	65
4.4	Evaluations	70
4.4.1	Experiment Settings	71
4.4.2	Validations	75
4.4.3	Performance Comparison with Existing Works	77
4.4.4	Quantitative Analysis of Data Scarcity	81
4.4.5	Incorporating Human Semantic Knowledge into Pre-trained Models in the Semi-supervised Domain	86
4.4.6	Execution Time	89
4.5	Chapter Summary	91
5	A Lightweight Deep Neural Model for SMS Spam Detection	93
5.1	Motivations and Contributions	93

5.2	The Proposed Model with Lightweight gated recurrent unit (LGRU)	94
5.2.1	Task Definition	94
5.2.2	Lightweight Gated Recurrent Unit	95
5.2.3	Enhancing Semantics with External Knowledge	97
5.2.4	The Proposed Lightweight Model	97
5.3	Experimental Results	98
5.3.1	Datasets	98
5.3.2	Baseline Models	101
5.3.3	Evaluation Metrics	102
5.3.4	Implementation Details	103
5.3.5	Results and Discussions	104
5.4	Chapter Summary	113
6	Conclusion and Future Research Directions	114
6.1	Conclusion	114
6.1.1	Twitter Bot Detection Using Neural Networks and Linguistic Embed- dings	114
6.1.2	An Attention-Based Neural Network Using Human Semantic Knowl- edge and Its Application to Clickbait Detection	115
6.1.3	A Lightweight Deep Neural Model for SMS Spam Detection	116
6.2	Future Research Directions	116
	Bibliography	119

List of Tables

1.1	Examples of clickbait and non-clickbait from the FNC Challenge dataset [32].	6
2.1	List of applications of RNN approaches on intrusion detection surveyed highlighting the method and the task.	17
2.2	List of applications of RNN approaches on clickbait detection surveyed highlighting the method and the task.	18
2.3	List of applications of RNN approaches on SMS spam detection surveyed highlighting the method and the task.	18
2.4	List of applications of CNN approaches on intrusion detection surveyed highlighting the method and the task.	20
2.5	List of applications of CNN approaches on malware classification surveyed highlighting the method and the task.	21
2.6	List of applications of CNN approaches on traffic classification surveyed highlighting the method and the task.	21
2.7	List of applications of CNN approaches on clickbait classification surveyed highlighting the method and the task.	21
2.8	List of applications of CNN approaches on SMS spam detection surveyed highlighting the method and the task.	21
2.9	Classifiers evaluated by Almeida et al. [23].	31

3.1	Example illustrating named-entity identification.	40
3.2	Statistics of the datasets used in the experiments.	46
3.3	The numbers of handcrafted features used in Twitter bot detection methods/models.	48
3.4	Performance comparison of various bot detection methods/models using the <i>cresci-2017</i> dataset. The proposed model BOTLE uses LGRU and four linguistic embeddings (word, character, POS, and NE). The best and second best results are highlighted in green and blue , respectively. The results of human annotators and of the baseline models BotOrNot? [7], Yang <i>et al.</i> [8], Miller <i>et al.</i> [9], Ahmed <i>et al.</i> [12], and Cresci <i>et al.</i> [11] are taken from [11]; the results of Wei & Nguyen [91] are from [91].	51
3.5	Empirical assessment of different linguistic embeddings: word, character, part-of-speech (POS), and named-entity (NE) embeddings. All models use LGRU. The best results are highlighted in green . The performance of the bot detector improves as more embeddings are added to the model.	53
3.6	Empirical assessment of running time and memory usage incurred by the linguistic embeddings: word, character, part-of-speech (POS), and named-entity (NE) embeddings. All models use LGRU.	55
3.7	Performance of four recurrent variants: LSTM, GRU, MGU, and LGRU. Model structures are all bidirectional RNNs, but each model uses a different recurrent variant. The models are enhanced with all four embeddings: word, character, POS and NE. The best results are highlighted in green	56
3.8	Empirical assessment of running time and memory usage incurred by four recurrent variants LSTM, GRU, MGU, and LGRU. Model structures are all bidirectional RNNs, but each model uses a different recurrent variant. The models are enhanced with all four embeddings: word, character, POS and NE. The best results are highlighted in green	57

4.1	Visualization of a sample from the FNC Challenge dataset. The proposed KED model is implemented with knowledge enhancement to the attention mechanisms and NKED is without knowledge enhancement. The most important phrases in self-attention are highlighted in red where the intensity of the color indicates the degree of contribution. The waves indicate the key phrases with high attention scores in knowledge-enhanced joint-attention. . .	69
4.2	Statistics of the datasets.	71
4.3	Model settings: vocabulary size and maximum sequence length.	72
4.4	Performance comparison using the Clickbait Challenge and FNC Challenge datasets. The best result in each column is highlighted in green ; our previously published best results in [18], from KED_R , are shown in blue ; other researchers' previously published best results are shown in red . The results in the first to fourth block (from DSSM to LSDA) come from [17]. The results in the fifth block are taken from [38, 39]. NKED: no knowledge enhancement to attention mechanisms. KED_R : with knowledge enhancement and randomly initialized word embeddings. KED: with knowledge enhancement and pre-trained GloVe word embeddings.	74
4.5	Effectiveness of human semantic knowledge. The hyper-parameter τ selected for the best result is marked in bold	78
4.6	Incorporation of human semantic knowledge into pre-trained models BERT, RoBERTa, and XLNet. The symbols “✓” and “✗” denote whether the proposed human semantic knowledge extraction scheme is incorporated into a model or not, respectively. The best result in each column is highlighted in orange . The second best result in each column is highlighted in green . . .	88
4.7	Training time (in minutes) on the two training datasets	90
4.8	Average inference time (in milliseconds) of one sample	91

5.1	Sample spam (T) and ham (F) pairs from the two datasets used in this work.	95
5.2	Token statistics of the dataset used.	100
5.3	Confusion matrix used for this work.	102
5.4	Performance comparison with state-of-the-art algorithms on the SMS Spam dataset: traditional machine-learning-based (<i>traditional</i>) and deep-learning-based (<i>deep</i>). Bold font indicates best systems. Results in the first and third blocks come from [23], [26] respectively, and results in the second and last blocks come from [30].	106
5.5	Class-wise and overall performance, in terms of precision, recall (spam capture rate), and F1-score on the SMS Spam dataset. Bold font highlights the best values across each category. The overall metrics are reported as weighted averages across the two classes: “Ham” (non-spam) and “Spam”.	108
5.6	Overheads of existing SMS spam detection models and the proposed model evaluated on the SMS Spam dataset. Number of recurrent hidden units (Column 2) and performance accuracy (Column 5) are reported from [29, 30], while the number of parameters and training time per epoch are obtained by reproducing the original models based on their experimental setup.	109
5.7	Performance comparison with popular traditional machine-learning and deep-learning-based state-of-the-art algorithms on the ExAIS SMS Spam dataset. Bold font indicates the best-performing systems.	110
5.8	Performance comparison with the existing state-of-the-art SMS spam detection model evaluated on the ExAIS SMS dataset. Bold font indicates the best-performing systems.	113

List of Figures

1.1	An example illustrating the importance of human semantic knowledge to clickbait detection. Without human semantic knowledge, this example will be incorrectly classified as clickbait. In the example, the proposed model can find the correct answer because it knows “ringtone” is a synonym of “song”, and “fisherman” is a hyponym of “man” thanks to the use of human semantic knowledge.	7
2.1	The architecture of SSCL.	33
3.1	Plot of character n-gram representations via PCA. Colors correspond to prefixes (blue), suffixes (grey), hyphenated (green), and all others (red). Prefixes refer to character n-grams that start with the start-of-word character. Suffixes likewise refer to character n-grams which end with the end-of-word character.	39
3.2	(a) Recurrent neural network and (b) Recurrent neural network unfold. Adapted from [45].	41
3.3	Comparison of data flow and operations between GRU and our LGRU. The differences are highlighted in red.	43
3.4	Bidirectional LGRU.	44
3.5	An illustration of the proposed BiLGRU neural model with word embeddings for distinguishing Twitter bots from human accounts.	45

3.6	Word-cloud comparison of (a) human account dataset and (b) social-bot-3 dataset.	47
3.7	Performance trends across linguistic embeddings (LGRU).	54
4.1	Our proposed end-to-end neural model.	66
4.2	The diagram of NKED model.	68
4.3	Illustrations of the results given in Table 4.5.	79
4.4	Illustrations of the quantitative analysis of data scarcity	82
5.1	Class-wise distribution of the two SMS spam datasets used.	99

List of Symbols

σ	sigmoid function
ϕ	hyperbolic tangent
w	weight vectors of the network
W	weight matrices of the network
$\odot$	element-wise multiplication
$\cdot$	scalar multiplication
m	annotated text messages
M	a collection of annotated text message
$\text{Pr}(\cdot)$	posterior distribution
$\tilde{h}$	candidate output state
h	actual output state
b	bias
f	forget gate
z	update gate
r	reset gate
$\overrightarrow{h}$	hidden unit function of a hidden forward layer
$\overleftarrow{h}$	hidden unit function of a hidden backward layer
E	embeddings
J	similarity matrix

List of Abbreviations

AE	autoencoding
AR	autoregressive
BERT	bidirectional encoder representations from transformers
BiLSTM	bidirectional long short term memory
CNN	convolutional neural network
DL	deep learning
DNN	deep neural network
EDR	endpoint detection and response
GloVe	global vectors for word representation
GPT	generative pre-trained transformer
GRU	gated recurrent unit
LSTM	long short term memory
ML	machine learning
RNN	recurrent neural network
RoBERTa	robustly optimized BERT approach
SMS	short message service
SVM	support vector machine
URL	uniform resource locator

Publications from the Dissertation

Peer-reviewed Journal Papers:

- [J1] Feng Wei, and Uyen T. Nguyen, “Twitter Bot Detection Using Neural Networks and Linguistic Embeddings,” *IEEE Open Journal of the Computer Society*, vol. 4, pp. 218–230, 2023.
- [J2] Feng Wei, and Uyen T. Nguyen, “An Attention-Based Neural Network Using Human Semantic Knowledge and Its Application to Clickbait Detection”, *IEEE Open Journal of the Computer Society*, vol. 3, pp. 217–232, 2022.

Peer-reviewed Conference Papers:

- [C1] Feng Wei, and Uyen T. Nguyen, “Twitter Bot Detection Using Bidirectional Long Short-Term Memory Neural Networks and Word Embeddings, ” *IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, 2019.
- [C2] Feng Wei, and Uyen T. Nguyen, “A Neural Attentive Model Using Human Semantic Knowledge for Clickbait Detection”, *IEEE International Conference on Big Data and Cloud Computing (BdCloud)*, 2020.
- [C3] Feng Wei, and Uyen T. Nguyen, “A Lightweight Deep Neural Model for SMS Spam Detection”, *IEEE International Symposium on Networks, Computers and Communications (ISNCC)*, 2020.

Chapter 1

Introduction

1.1 Background and Context

Cybersecurity is a collection of strategies, policies, processes, and technology, working together to safeguard the privacy, confidentiality, and availability of attack-proof computer infrastructure, networks, software applications, and records [1]. At the application, network, host, and data stage, cybersecurity systems operate. There are plenty of technologies that operate in silos to deter attacks and detect security breaches, such as antivirus applications and firewalls [1]. Nonetheless, several rivals are still at an advantage because they only need to identify one vulnerability in the systems requiring protection. The threat is also growing as the number of internet-connected systems grows, leading to a higher probability of attack. In addition, attackers are getting more advanced, designing zero-day exploits and malware that avoid security monitor, allowing them to sustain without warning for long periods [1,2].

It is urgent to develop advanced and intelligent security protection techniques that can deal with sustainable and diversified risks, in order to optimize the security level of core system infrastructure. More importantly, the framework established can acquire the preceding and existing security state data to not merely develop modern cybersecurity protection and defense, yet offer intelligent decisions that can provide adaptive security management and

control [3].

Deep learning (DL) is a fast-growing machine learning (ML) division that explores and develops methods, theories, techniques, and application frameworks for the simulation, extension, and advancement of human intelligence. DL technology has made tremendous strides during the past decade, due to the growth of ultra-performance computing technology and the advent of big data. DL technology has particularly allowed individuals to benefit from more data, achieve better performance, and gain more opportunities. It transforms the lives of people drastically and refashions traditional ML technologies. ML has a broad variety of applications, such as computer vision, speech recognition, and natural language processing. Yet its breadth of application goes way beyond the three aspects. In the area of cybersecurity, it still has several other outstanding tasks, such as the detection of bots, spam, malware, and clickbait [3].

Traditional ML methods, such as random forests, and support vector machines (SVM), played a critical role in meeting cyberspace challenges, in the early stage of ML research. While traditional ML methods are very effective, the extraction of features is time-consuming and labor-intensive. When it is extended to the field of cybersecurity, this weakness is particularly noticeable. For instance, the different features (e.g., ‘MachineIdentifier: Individual machine ID’, ‘ProductName: Defender state information, such as win8defender’, ‘AppVersion: Defender state information, such as 4.9.10586.0’) associated with malware are necessarily and manually compiled to enable an ML model to identify malware. Therefore, it certainly restricts the efficiency and performance of threat detection [3].

This is because traditional ML algorithms operate according to particular pre-defined patterns, indicating that characteristics that are not pre-defined can resist detection and may not be identified. It can be inferred that most ML algorithm performance relies on the quality of the detection and extraction of features [4,3]. During the past decade, researchers have devoted great effort to research deep neural networks (DNN), in light of apparent shortcomings in standard ML. It is also known as DL, which is a sub-domain of ML. Recurrent neural networks (RNN) and convolutional neural networks (CNN) are two popular DNN

variants. A major difference in principle between traditional ML and DL is that without extracting its characteristics or patterns, DL can be used to train the original data directly.

During the past decade, in the fields of computer vision, speech recognition, and natural language processing, DL has achieved 20 percent to 30 percent performance enhancement and achieved a remarkable leap in the growth of ML [5,3]. Regardless of any new data types, DL can model nonlinear and sparse correlations hidden in the data, and detect unknown attacks, a desirable advantage in cybersecurity applications. DL has made significant progress in the prevention of cybersecurity threats in recent years. Although adversaries use the most sophisticated deception tactics, DNN may model the high-level abstract features of cyber attacks [6,3].

The overarching goal of the research in this dissertation is to model and apply deep learning methods to cybersecurity tasks with new design, technologies, and algorithms. Following are the motivations behind our research and the contributions of the dissertation.

1.2 Motivations and Contributions of the Dissertation

Deep learning (DL) is a popular topic that has been widely applied in cybersecurity. The dissertation focuses on applying deep neural networks (DNN) to cybersecurity tasks such as detecting Twitter bots, clickbait, and SMS spam.

1.2.1 Motivations

We have identified gaps in existing research in several areas such as Twitter bot detection using neural networks and linguistic embeddings, attention-based neural networks using human semantic knowledge and its application to clickbait detection, and lightweight deep neural models for SMS spam detection.

Twitter Bot Detection

Twitter is a popular online social networking and micro-blogging tool. As a social network, Twitter’s defining characteristic is its simplicity: its community interacts via publishing text-based posts, known as tweets. Twitter has its own special memes, i.e., hashtag (#), mention (@), shortened URL (<http://t.co>) and retweet (RT). Hashtags, namely words or phrases prefixed with a # symbol, allow tweets to be grouped by topics. For example, *#usopen2019* and *#SheTheNorth* are two trending hashtags on Twitter in September 2019. The symbol @ followed by a user name in a tweet enables the direct delivery of the tweet to that user. Links shared on Twitter, including links shared in private direct messages, will automatically be processed and shortened to an <http://t.co> link. A retweet is a re-posting of a tweet. Sometimes people type “RT” at the beginning of a Tweet to indicate that they are re-posting someone else’s content.

The growing user population and open nature of Twitter have made itself an ideal target of exploitation by automated programs, known as bots. Automation is a double-edged sword for Twitter. On the one hand, legitimate bots generate a large volume of informative tweets, like news and blog updates. On the other hand, malicious bots spread spam or malicious content.

The Twitter bot problem has received substantial attention from researchers. Existing Twitter bot detection models generally rely on features obtained from user information such as profile [7, 8, 9, 10], timestamps [7, 8, 10], friendship [10, 7], behavior [11, 12], and network connection [13, 8, 14]. However, feature engineering is very time-consuming and labor-intensive.

This dissertation proposes a recurrent neural network (RNN) model named BOTLE (BOT detection using Linguistic Embeddings), which employs BiLGRU and linguistic embeddings to distinguish between Twitter bots and human users. The task of Twitter bot detection is formulated as a binary text classification problem, where the goal is to classify accounts as either bots or non-bots based solely on the textual content of their tweets. To the best

of our knowledge, BOTLE is the first model that does not rely on handcrafted features or require prior knowledge or assumptions regarding user profiles, social networks, or historical activity.

Clickbait Detection

Clickbait is designed to entice Internet users into clicking on a hyperlink that leads to a web page or another piece of online content such as an article, an image or a video. There are three principal uses of clickbait:

1. *Phishing attacks.* Attackers send malicious messages to victims that trick them into clicking on a hyperlink. The purpose is to steal sensitive information such as login credentials, banking information or credit card numbers, or to install malicious software such as ransomware on the victims' computers.
2. *Legitimate and illegitimate marketing.* Marketers use attention-grabbing headlines to entice users into clicking on hyperlinks that lead to online articles or advertisements. Marketers make money based on the number of clicks that an article or advertisement receives.
3. *Spreading misinformation and disinformation (fake news).* Similarly, attention-grabbing headlines are used to build suspense and sensation in order to lure users into clicking on hyperlinks that lead to articles or online content with fake news.

The focus of our work in this dissertation is on cases (2) and (3). In particular, we determine whether a headline is clickbait or not by matching its content to the textual content of the article linked to it. If the content of the headline closely matches that of the linked article, it is not clickbait. Otherwise it is classified as clickbait. An example is given in Table 1.1.

Only a limited number of existing approaches have leveraged the semantic relationships between headlines and their corresponding articles for the task of clickbait detection. Biyani et al.[15] computed similarity measures between headlines and the initial five sentences of

Table 1.1: Examples of clickbait and non-clickbait from the FNC Challenge dataset [32].

♦ Headline	EXCLUSIVE: Apple To Unveil The Long-Awaited Retina Mac- Book Air At Its “Spring Forward” Event
♣ Body	Last week, Apple sent out the invites for its “Spring Forward” event, ... The Retina MacBook Air, currently codenamed “Mac- Book Stealth” internally, will also come with a modest price decrease according to previous reports.
♠ Candidate	[clickbait, non-clickbait]
♦ Headline	Next-generation Apple iPhones’ features leaked
♣ Body	When faced with the choice of feasting on a fine meal of human while listening to Justin Bieber’s music ... his cellphone rang, and the ringtone of Justin Bieber hit “Baby” startled the animal ... “I know that sort of ringtone isn’t to everyone’s taste but my grand- daughter loaded it onto my phone for a joke,” ...
♠ Candidate	[clickbait, non-clickbait]

linked articles, integrating these with handcrafted textual features such as the frequency of uppercase words and acronyms. Kumar et al. [16] employed a Siamese neural network architecture to capture similarities between textual and visual modalities, and subsequently processed these features through fully connected neural layers. Dong et al. [17] utilized bidirectional gated recurrent units (BiGRU) to model the contextual representations of headlines and associated articles, further incorporating a cosine similarity-based mechanism to quantify global semantic alignment.

However, there are two shortcomings in the above clickbait identification models. First, they do not leverage human semantic knowledge, which is essential to determine semantic correlations between a headline and its linked article for the clickbait detection task [18]. The example in Figure 1.1 illustrates the importance of human semantic knowledge to clickbait detection. Second, the methods they use to match a headline with its linked article, such as cosine similarity [15] or simple attention-based networks [17], do not adequately capture similarity information.

An example from FNC Challenge dataset.

Headline: *Justin Bieber ringtone scares bear, saves Russian fisherman*

Body: *A Russian man was recently able to fend off a bear attack with a Justin Bieber song.*

Label: ***non-clickbait***

Figure 1.1: An example illustrating the importance of human semantic knowledge to clickbait detection. Without human semantic knowledge, this example will be incorrectly classified as clickbait. In the example, the proposed model can find the correct answer because it knows “ringtone” is a synonym of “song”, and “fisherman” is a hyponym of “man” thanks to the use of human semantic knowledge.

To overcome these challenges, we propose an attention-based neural network using human semantic knowledge, which can be applied to clickbait detection and several other tasks

(word sense disambiguation and machine reading comprehension). We introduce two novel mechanisms in the proposed network. First, we incorporate human semantic knowledge into the proposed neural network, which is inspired by the work in neuroscience by Chen et al. [19]. We use WordNet [20], a lexical database of semantic relations between words, to acquire semantic correlations of headline-article word pairs. Intuitively, semantic knowledge of word pairs such as hyponyms, hypernyms, and/or synonyms can facilitate the similarity matching of words in a headline to words in the linked article. Therefore, for each headline and its linked article, the proposed model generates a linguistic knowledge graph using WordNet that represents semantic correlations of headline-article word pairs. Secondly, the attention mechanisms, joint attention and self-attention, used by the neural network are enhanced with human semantic knowledge. Specifically, the attention mechanisms use the knowledge graph generated earlier and focus on the most important parts of the graph in order to extract most meaningful headline-article word pairs. For this reason, we say that the proposed attention mechanisms are *knowledge enhanced*.

To the best of our knowledge, our work is the first that

- incorporates human semantic knowledge into a neural network for clickbait detection;
- uses linguistic knowledge graphs built from WordNet to guide the attention mechanisms for the clickbait detection task.

SMS Spam Detection

Globally, the short messaging service (SMS) is one of the most popular and also most affordable telecommunication services. The popularity and affordability of SMS, however, have made it an ideal target for spamming. Spam can be defined as unwanted or unsolicited electronic messages sent in bulk to a group of recipients. SMS is a preferred medium of spammers due to the following factors: 1) the availability of low-cost bulk SMS plans; and 2) reliability of cellular network service (as opposed to, say, WiFi networks). While SMS spam can be benign unsolicited advertisements, it can also carry phishing or malicious con-

tent (e.g., scams, links to malicious websites). In the former case, spam is a major nuisance to mobile users and consumes network bandwidth, affecting service providers' profit margins. In the latter case, spam may lead to security breaches or criminal activities such as malware or ransomware downloads, and thefts of login credentials, financial information or identity. Therefore, the topic of SMS spam detection has been studied extensively [21] and still a focus of cybersecurity.

Early work in the area of SMS spam detection was content-based spam identification that relies on defining the rules that constitute spam and extracting them from the message content [22]. Most of the later work [23, 24, 25] was carried out using traditional machine learning classifiers, such as Support Vector Machine (SVM), Naïve Bayes, K-Nearest Neighbor, Random Forest, etc. These classifiers depend on handcrafted features extracted from the training data. For a comprehensive survey of SMS spam classifiers based on traditional machine learning techniques, refer to Abdulhamid et al. [21]. Almeida et al. [23] introduced a public and annotated SMS spam collection and quantitatively compared approximately 20 classifiers that use traditional machine learning approaches. Their comprehensive evaluation indicate that SVM outperforms all the other evaluated classifiers. It is worth noting that these traditional machine-learning-based classifiers require heavy feature engineering, which is a very time consuming and labor-intensive task.

Latest developments in the field of deep learning have led to better performance in SMS spam detection. Al Moubayed et al. [26] propose an unsupervised deep learning model named TM-SDA (Topic Modelling - Stacked Denoising Autoencoder) to detect spam and demonstrate its applicability concerning SMS spam. Jain et al. explore SMS spam detection using Long Short Term Memory (LSTM) [27] and Convolutional Neural Networks (CNN) [28], and introduce two deep neural models named SLSTM (Semantic LSTM) [29] and SSCL (Sequential Stacked CNN-LSTM) [30]. It is worth noting that, while deep learning models mentioned above offer fantastic results, they typically take a lot of time and memory to train. We believe that the explanation for this is that the computing diagrams comprise a considerable quantity of learnable parameters and operations.

In this dissertation, we propose a novel lightweight deep neural model named Lightweight Gated Recurrent Unit (LGRU). Instead of relying only on static knowledge implicitly present in the training data, enhancing semantics is retrieved from external knowledge sources (e.g., WordNet) to assist in understanding SMS text inputs. Our model significantly improves the training efficiency without compromising the performance. Experimental results show that our model outperforms more than 30 state-of-the-art SMS spam classifiers. The significance of this work lies in the fact that it reveals a way to minimize unnecessary complexity in training recurrent models without compromising the performance, which can be applied to many other complex recurrent models.

1.2.2 Detailed Contributions

Twitter Bot Detection Using Neural Networks and Linguistic Embeddings:

In this dissertation, we propose a recurrent neural network (RNN) model using BiLGRU and linguistic embeddings to distinguish Twitter bots from human accounts. We name the proposed model BOTLE (BOT detection using Linguistic Embeddings). We define the problem of Twitter bot detection as a text classification problem with a binary output: bot or non-bot. We use only the textual content of tweets as the input to our RNN model. To the best of our knowledge, our Twitter bot detection model is the first that does not require any handcrafted features; or prior knowledge or assumptions about account profiles, friendship networks or historical behavior.

Following are our contributions:

- We propose an RNN model to distinguish Twitter bots from human accounts. BiLGRU connects two hidden layers of opposite directions to the same output. With this form of generative deep learning, the output layer can get information from past (backward) and future (forward) states simultaneously. This method can effectively capture textual features across tweets and achieve competitive classification performance compared to

existing state-of-the-art Twitter bot detection models.

- We use several types of linguistic embeddings to encode tweets, namely, word embeddings, character embeddings, part-of-speech embeddings, and named-entity embeddings. We avoid using handcrafted features, which require time-consuming and labor-intensive feature engineering. This advantage allows for faster and easier implementation and deployment of the bot detection scheme.
- We conducted experiments on real-world datasets. Experiments on the *cresci-2017* dataset show that BOTLE performs better or comparably to state-of-the-art Twitter bot detection models [11,8,7,12,9], although our model does not require prior knowledge or assumptions about users' profiles, friendship networks, or historical behavior of the target accounts.
- We validate the effectiveness of various linguistic embeddings and recurrent variants for the task of Twitter bot detection. The results will potentially help researchers design high-performance deep-learning models for similar tasks.
- In addition to the classification performance, we also provide the running time (for training and inference) and memory usage of the proposed model, which show the model is feasible to deploy in real life.

An Attention-Based Neural Network Using Human Semantic Knowledge and Its Application to Clickbait Detection:

Some prior works have explored the correlations between headlines and their associated articles for the task of clickbait detection [15, 16, 17]. However, these models suffer from two main limitations. First, they overlook the use of human semantic knowledge, which plays a crucial role in assessing the semantic relationship between a headline and its linked article [18]. The example shown in Figure 1.1 demonstrates how semantic understanding can enhance clickbait detection. Second, the similarity matching approaches employed, such as

cosine similarity [15] or basic attention mechanisms [17], are inadequate for capturing deeper semantic similarities.

To overcome these challenges, we propose an attention-based neural network using human semantic knowledge, which can be applied to clickbait detection and several other tasks (e.g., word sense disambiguation and machine reading comprehension).

Our contributions are as follows.

- We propose a novel attention-based neural network model named Knowledge-Enhanced Clickbait Detector (KED) that uses linguistic knowledge graphs built from WordNet to guide the attention mechanisms. The proposed neural network can effectively capture discriminative features from local and global similarities. Global similarities are captured using the proposed knowledge-enhanced joint-attention mechanism. To minimize the impact of noise, the model selects the most useful similarity features for the final predictions using the proposed knowledge-enhanced self-attention mechanism.
- We incorporate human semantic knowledge into the neural network and its attention mechanisms to better capture semantic correlations of headline-article word pairs. Experimental results show that this novelty significantly enhances the performance of KED and that of several pre-trained models such as BERT, RoBERTa, and XLNet.
- We conducted extensive experiments to evaluate the proposed model on two real-world clickbait datasets, Clickbait Challenge [31] and FNC Challenge [32]. Experimental results show that our model significantly outperforms existing state-of-the-art models/systems [33, 34, 35, 36, 16, 37, 17, 38, 39], even when training data is limited. The proposed model also performs better or comparably to powerful pre-trained models, namely, BERT, RoBERTa, and XLNet, while being much more lightweight.
- Based on extensive comparative experiments, we carry out comprehensive analyses which will benefit future studies.

Illegitimate marketing has been used to defraud consumers via products and services

such as counterfeit goods, and pyramid, Ponzi, “make money quick” and pump-and-dump schemes. Fake news has had profound impacts on a country’s political and social stability [40, 41], democracy [42], financial markets [43], and public health [44]. Our work in this dissertation, clickbait detection, is a first step towards developing effective countermeasures to the threats of consumer frauds and fake news.

A Lightweight Deep Neural Model for SMS Spam Detection:

Latest developments in the field of deep learning have led to better performance in SMS spam detection [26, 27, 28, 29, 30]. While these models yield strong results, they are computationally demanding due to their large number of learnable parameters and while cell phones have lower computational power and storage capacity compared to laptops and desktops.

To address the challenges of efficient and effective SMS spam detection, we propose a lightweight recurrent neural network model enhanced with external semantic knowledge. The following are main contributions of this work:

- We propose a novel deep learning architecture named Lightweight Gated Recurrent Unit (LGRU) for SMS spam detection. The proposed model adopts a resource-efficient structure that significantly reduces the number of trainable parameters, leading to lower training time and memory consumption compared to existing deep learning approaches [29, 30].
- To enhance the model’s semantic understanding of SMS content, we incorporate external linguistic knowledge from WordNet [20]. This semantic enrichment enables the model to capture deeper contextual cues and improves its ability to distinguish between spam and legitimate messages (ham).
- We conduct extensive empirical evaluations by benchmarking LGRU against more than 30 SMS spam classifiers, including both traditional machine learning and deep learning. Experimental results demonstrate that our model consistently achieves superior

performance in terms of precision, recall, and accuracy.

- Beyond achieving state-of-the-art performance, the proposed LGRU model exemplifies how architectural simplification can reduce computational overhead without compromising classification performance. This finding offers valuable insights into the design of light weight yet effective recurrent models for other broader natural language processing tasks.

1.3 Organization of the Dissertation

The remainder of the dissertation is organized as follows. We provide a review of deep learning methods and their application to cybersecurity, especially detection of Twitter bots, clickbait, and SMS spam, in Chapter 2. In Chapter 3, we present a novel RNN model to distinguish Twitter bots from human accounts. In Chapter 4, we propose a novel attention-based neural network model named Knowledge-Enhanced Clickbait Detector (KED) that uses linguistic knowledge graphs built from WordNet to guide the attention mechanisms. In Chapter 5, we develop a novel lightweight deep neural model named Lightweight Gated Recurrent Unit (LGRU) for SMS spam detection. We conclude the dissertation, and outline future research directions in Chapter 6.

Chapter 2

Literature Review

This chapter begins by providing an overview of the use of deep learning (DL) in cybersecurity, focusing on areas such as clickbait detection, malware classification, intrusion detection, and SMS spam detection. Then, we present a detailed review of existing research on these topics.

2.1 Applications of Deep Learning in Cybersecurity

Deep learning (DL), a rapidly advancing branch of machine learning (ML), focuses on developing methods, theories, techniques, and application frameworks that simulate, extend, and enhance human intelligence. Recent breakthroughs in DL have been fueled by the rise of high-performance computing and the explosion of big data. These advancements have enabled individuals and organizations to harness larger datasets, achieve high predictive performance, and unlock new opportunities. Deep learning has several advantages over traditional machine learning methods. It can learn from raw data without time-consuming and labor-intensive feature engineering. DL model is able to capture complex and nonlinear relationships, scale well with large and diverse datasets. Like recent large language models such as ChatGPT, Gemini, LLaMA, Mistral and DeepSeek, it can be applied in multitask

domains.

In recent years, deep learning has contributed to the advancements of numerous fields such as cybersecurity, natural language processing, speech recognition, and computer vision. Since very little manual engineering is required, deep neural networks (DNNs) are able to benefit from the increase in data volume and computing power [45]. In other words, patterns of latent features from a large data set can be effectively and efficiently derived by deep learning techniques. Therefore, more advancements in many applications can be expected in the future thanks to deep learning [46].

Recurrent neural networks (RNN) and convolutional neural networks (CNN) are two popular DNN variants.

2.1.1 Applications of Recurrent Neural Network to Cybersecurity

Recurrent neural networks (RNNs) are neural networks that relate a sequence's current output to the previous output. The specific interpretation is that the network will recall and incorporate knowledge from the previous state into the current output computation; that is, the nodes between the hidden layers are interconnected, and the hidden layer data includes both the input layer output and the hidden layer output from the previous state. Technically, RNNs can handle any length of sequence data. However, to simplify complexity, the current state is always assumed to be only related to prior states [47].

There are two enhanced, widely used RNN-based models, namely, gated recurrent units (GRU) and long short-term memory (LSTM).

To process sequence data, recurring network structure are employed. Information from the input layer passes through the hidden layer eventually reaching the output layer, in the conventional neural network model; the layers are fully connected and there is no interaction between nodes across each layer. There are also numerous issues that cannot be addressed by this traditional neural network [47].

That said, RNNs update hidden layers utilizing the information received from input layers and the activation from the proceeding forward propagation. Compared to feed-forward neural networks (FFNNs), RNNs are capable of managing the temporal information encoded in the sequence of input data. We believe this explains why RNNs are suitable for tackling various cybersecurity problems, such as text-based tasks (e.g., Twitter bot detection, SMS spam detection, clickbait detection, etc.), temporal-based cybersecurity tasks (e.g., intrusion detection, etc.). These observations motivate researchers to leverage RNNs as an effective approach to solve cybersecurity dilemma [48]. Tables 2.1, 2.2, and 2.3 list RNN-based approaches applied to intrusion detection, clickbait detection, and SMS spam detection, individually, that are reviewed in this section.

Table 2.1: List of applications of RNN approaches on intrusion detection surveyed highlighting the method and the task.

Paper	Method	Task
[49]	LSTM-based	Intrusion detection
[50]	Standard RNN-based	Intrusion detection
[51]	LSTM-based	Intrusion detection
[52]	LSTM-based	Intrusion detection
[53]	Standard RNN-based	Intrusion detection
[54]	LSTM-based	Intrusion detection
[55]	GRU-based	Intrusion detection

Intrusion Detection

The LSTM recurrent neural network classifier for intrusion detection was first introduced by Staudemeyer [49]. To address the issue of high false-alarm rates, Kim et al. [51] proposed a system-call language modeling approach for developing LSTM-based host intrusion detection

Table 2.2: List of applications of RNN approaches on clickbait detection surveyed highlighting the method and the task.

Paper	Method	Task
[36]	LSTM-based	Clickbait detection
[37]	BiGRU-based	Clickbait detection
[16]	Standard RNN-based	Clickbait detection
[17]	Attention-based BiGRU	Clickbait detection

Table 2.3: List of applications of RNN approaches on SMS spam detection surveyed highlighting the method and the task.

Paper	Method	Task
[29]	LSTM-based	SMS spam detection
[30]	LSTM-based	SMS spam detection

systems. Furthermore, Kim et al. [52] introduced another LSTM-based neural model using the KDD Cup 99 dataset. In a subsequent study, Kim et al. [54] compared the effectiveness of six widely used optimizers on the LSTM intrusion detection model.

In addition, Krishnan and Raajan [50] employed recurrent neural networks (RNNs) for intrusion detection using the KDD Cup 99 dataset, which categorizes attacks into four main types: Denial of Service (DoS), Root to Local (R2L), User to Root (U2R), and Probe (Probing). Building on RNN architectures, Yin et al. [53] proposed a novel intrusion detection model named RNN-IDS.

Agarap et al. [55] proposed a modification to the standard GRU-based model by replacing the softmax output layer with a linear support vector machine (SVM), aiming to enhance classification performance.

Clickbait Detection

Clickbait detection based on deep learning was also recently developed in order to bypass the time-consuming and labor-intensive feature engineering [33, 56, 57, 37, 58].

Linguistically infused neural network (LiNN) models, based on CNNs and LSTM, are proposed by Glenski et al. [36] to learn the features of clickbait data.

An attentive BiGRU model is adopted by Zhou et al. [37] to learn the latent representations of clickbait headlines and bodies. Then, for the inference, the two acquired latent representations are concatenated and fed into fully connected layers [17].

A hybrid approach is proposed by Kumar et al. [16] for clickbait detection. To learn the inputs, they first use attentive bidirectional RNN-based techniques, and then concatenate the latent inputs with the relationship feature they learned for the final inference with Siamese Net [17].

A deep similarity-aware attentive model is proposed by Dong et al. [17] to capture and interpret those correlations with better expressiveness.

SMS Spam Detection

Jain et al. [29] introduce a model named semantic long short-term memory (SLSTM). It is an LSTM neural network with a semantics layer on top of it. That is, before using the LSTM for spam detection, the word vectors are enriched semantically.

Jain et al. [30] further explore SMS spam detection using a combination of long short-term memory (LSTM) and convolutional neural networks (CNN), and introduce a hybrid model called Sequential Stacked CNN-LSTM (SSCL).

2.1.2 Applications of Convolutional Neural Network to Cybersecurity

Recursive neural networks (RNNs) are widely used for processing sequence data. In typical neural network models, data from input layers, hidden layers, and output layers are all linked together. Although these layers are fully connected, no interaction occurs between nodes within each layer. Traditional neural networks cannot address many challenges. [47].

Since each type of neural network has its advantages and disadvantages, it is essential to select a suitable type for distinct applications [59]. Feedforward neural networks (FFNNs) have demonstrated promising results in addressing cybersecurity challenges, such as analyzing HTTPS network traffic to detect malicious activities. This approach effectively mitigates various cyber threats, including SQL injections and DOS attacks. Convolutional neural networks (CNNs) have considerably fewer parameters and connections than similarly-sized FFNNs. Thus, they are easier to train and likely only slightly less effective in achieving the theoretical optimal performance [60]. In other words, CNNs are also very promising for detecting network interference. CNNs show great promise for intrusion detection applications. Moreover, fortunately, current GPUs are powerful enough to facilitate the training of large CNNs [60]. In particular, recent datasets contain enough labeled samples to train such models without severe overfitting. Tables 2.4, 2.5, 2.6, 2.7, and 2.8 list CNN-based approaches applied to intrusion detection, malware classification, traffic classification, clickbait classification, and SMS spam detection, individually, that are reviewed in this section.

Table 2.4: List of applications of CNN approaches on intrusion detection surveyed highlighting the method and the task.

Paper	Method	Task
[61]	Convolutional autoencoder based	Intrusion detection
[62]	Standard CNN-based	Intrusion detection

Table 2.5: List of applications of CNN approaches on malware classification surveyed highlighting the method and the task.

Paper	Method	Task
[63]	Convolution- & feedforward-based	Malware classification

Table 2.6: List of applications of CNN approaches on traffic classification surveyed highlighting the method and the task.

Paper	Method	Task
[64]	Standard CNN-based	Traffic classification
[65]	LeNet-based	Traffic classification

Table 2.7: List of applications of CNN approaches on clickbait classification surveyed highlighting the method and the task.

Paper	Method	Task
[36]	Convolution- & LSTM-based	Clickbait detection
[35]	Standard CNN-based	Clickbait detection

Table 2.8: List of applications of CNN approaches on SMS spam detection surveyed highlighting the method and the task.

Paper	Method	Task
[30]	Convolution- & LSTM-based	SMS spam detection

Intrusion Detection

Yu et al. [61] introduce a deep learning approach, named dilated convolutional autoencoders (DCAEs), for the network intrusion detection task. Their model incorporates the benefits of stacked autoencoders and CNNs, to learn useful feature representations automatically and effectively from large volumes of unlabeled raw network traffic data by using deep learning techniques.

Saxe and Berlin [62] present an eXpose neural network that employs the technique of depth learning. The depth learning method has been built to take regular raw short strings as input, such as a common case for security inputs that involve artifacts like potentially malicious URLs, named pipes, file paths, registry keys, and named mutexes. It simultaneously learns to extract features and perform detection using character-level embedding and convolutional neural networks simultaneously.

Malware Classification

Kolosnjaji et al. [63] introduce a hybrid neural structure to model the execution sequence of disassembled binary malicious files. The neural model is composed of convolution and feedforward neural structures. This architecture incorporates a method of hierarchical extraction of features that integrates convolution of n-grams [66] of instruction with plain features vectorization.

Traffic Classification

Wang et al. [64] introduced an end-to-end encrypted traffic classification approach based on a one-dimensional convolutional neural network. This system integrates feature extraction, feature selection, and classification modules into a unified end-to-end framework, enabling it to automatically learn the nonlinear relationship between the input data and the ground truth.

Wang et al. [65] propose a malware traffic classification approach by exploiting LeNet-based convolutional neural networks, where the input data was traffic image. No hand-crafted features were needed for this approach, as raw traffic data was directly fed into the model.

Clickbait Detection

Glenski et al. [36] introduce a novel approach for detecting clickbait by combining CNNs and LSTMs into a linguistically-infused neural network model (LiNN). The authors highlight the importance of considering both textual and image data in their multi-modal approach, which allows the model to provide a more comprehensive analysis.

Zheng et al. [35] present a CNN-based architecture tailored for identifying clickbait, named clickbait convolutional neural network (CBCNN). The proposed CNN model automatically extracts relevant features from the raw input without the need for manual feature engineering, making it highly effective in identifying hidden patterns in clickbait content.

SMS Spam Detection

Jain et al. [30] employ a combination of convolutional neural networks (CNN) and long short-term memory (LSTM) to address the problem of SMS spam detection. The authors introduce a hybrid model named the sequential stacked CNN-LSTM (SSCL), where three layers (semantics, CNN, and LSTM) are arranged in sequence to progressively enhance feature extraction and detection.

2.2 Related Work

In this section, we review the relevant work in the literature on the deep learning methods and their applications to cybersecurity, divided into three main categories, i.e., Twitter bot detection, clickbait detection, and SMS spam detection.

2.2.1 Twitter Bot Detection

Existing methods of Twitter bot detection can be broadly categorized into two main approaches: supervised learning and unsupervised learning. Supervised learning approaches rely on labeled data, where a model is trained on a dataset containing known examples of bots and human accounts. The model learns to distinguish between the two based on features. In contrast, unsupervised learning methods do not require labeled data. Instead, they focus on identifying clusters within the data that may indicate bot-like behavior. Both approaches have their advantages and disadvantages. Supervised methods often provide higher accuracy but require extensive labeled data, while unsupervised methods offer greater flexibility and scalability in detecting previously unseen types of bots.

Supervised Learning

Lee et al. [67] implemented 30 classification algorithms and tested their performance. Tree-based supervised classifiers showed the highest accuracy results. In particular, random forest produced the highest accuracy. In order to improve the random forest classifier, standard boosting and bagging techniques have been applied additionally. The authors trained the content polluter classifier based on different feature combinations.

The system by Yang et al. [8] provides a supervised machine learning classifier that infers whether a Twitter account is a human account or a bot by relying on relationships among accounts, tweet timestamps and level of automation. In addition, they designed ten new behavior-based features. According to their evaluation, the detection rate using their new feature set is significantly higher than that of existing work.

Alsaleh et al. [68] presented a system that utilizes feedforward neural networks (FFNN) to dynamically detect Twitter bot accounts. The classification results show satisfying detection rates. But it still requires complicated handcrafted features.

Davis et al. [7] grouped features into six main classes: network, user, friends, temporal

features, content and sentiment, and employed random forest.

Varol et al. [10] proposed a supervised machine learning system that extracts more than a thousand features in six different classes: users, friends meta-data, tweet content, sentiment, network patterns and activity time series.

Guo et al. [69] introduced, BGSRD, a model that symmetrically combines BERT [70] and graph convolutional network for Twitter bot detection. The model constructs a heterogeneous graph over the dataset and represents Twitter as nodes using BERT representations.

More recently, Feng et al. [13] introduced a bot detection framework that leveraged the topological structure of user-formed heterogeneous graphs. To evade detection, new emerging Twitter bots steal genuine users' tweets and dilute malicious content with benign tweets. Lei et al. [14] proposed a model that makes the text and graph modalities interactive and detects tweet semantic inconsistency. Periasamy et al. [71] introduced a transformer-based model for Twitter bot detection. Mohanty et al. [72] applied three boosting-based ensemble learning methods (i.e., adaptive boost, gradient boosting, extreme gradient boosting) to detect Twitter bots. Tan et al. [73] presented, BotPercent, a multi-dataset multi-model community-oriented bot detection pipeline to overcome generalization issues in existing individual-level models. BotPercent consists of two parts: training data and model architecture. Yang et al. [74] introduced, FedACK, a federated adversarial contrastive knowledge distillation framework for Twitter bot detection. The authors devise a GAN-based federated knowledge distillation mechanism for efficiently transferring knowledge of data distribution among clients. Chawla et al. [75] presented a hybrid technique utilizing digital DNA as a base approach and augmenting it with the state-of-the-art BERT model pre-trained on the sentiment classification task. Wu et al. [76] introduced, BotTriNet, a unified embedding framework that utilizes textual content posted by accounts for Twitter bot detection. The authors designed a triplet network to tune the raw embeddings, which are produced by traditional natural language processing techniques, for improving classification performance. Di et al. [77] introduced an algorithm to transform sequences of digital DNA into images and applied pre-trained convolutional neural networks (CNNs) models over the generated images for Twitter bot

detection. DNA sequences are extracted from Twitter account profiles.

Unsupervised Learning

The method by Miller et al. [9] considers vectors made of 126 features, extracted from both accounts and tweets, as input to modified versions of the DenStream [78] and StreamKM++ [79] clustering algorithms, to cluster feature vectors of a set of unlabeled accounts. The classifier by [12] exploits a set of 14 generic statistical behavior features related to URLs, hashtags, mentions and retweets. Feature vectors generated in this way are then compared with one another via a Euclidean distance measure. Chavoshi et al. [80] developed an unsupervised method, named DeBot, which calculates cross-user activity correlations to detect bot accounts in Twitter. DeBot detects thousands of bots per day with a 94% precision and generates reports online every day. Cresci et al. [11] proposed an unsupervised method to detect bots, by comparing their behavior with the aim of finding similarities between automated accounts. They introduced a bio-inspired technique to model online user behaviors using so-called “digital DNA” sequences. Extracting digital DNA for an account means associating that account with a string that encodes its behavioral information. Although it achieves good detection performance, numerous handcrafted behavioral features are still required.

A recent research direction is to test the limits of current bot detection frameworks in an adversarial setting. The goal is to create bots that are difficult or impossible to detect. Cresci et al. [81] proposed the use of evolutionary algorithms to improve social bot skills. Grimme et al. [82] employed a hybrid approach involving automatic and manual actions to create bots that would be classified as human by a supervised bot detection system.

Datasets for Twitter Bot Detection Research

Cresci et al. [83] provided a widely used dataset for Twitter bot detection. It contains 3,474 human accounts and 10,894 bot accounts. The dataset is composed of both user information

and tweets.

Feng et al. [84] introduced TwiBot-20, the first publicly available bot detection dataset that includes user follow relationships. The dataset contains 229,573 Twitter users, 8,723,736 user property items, 33,488,192 tweets and 455,958 follow links.

Feng et al. [85] presented TwiBot-22, a comprehensive graph-based Twitter bot detection dataset. It provides diverse entities and relations on the Twitter network, and has considerably good annotation quality. The dataset contains 1,000,000 Twitter users and 88,217,457 tweets.

For a more comprehensive review of Twitter bot detection methods/techniques, readers are referred to [86, 87, 88, 89, 90].

To the best of our knowledge, our work is the first that does not require sophisticated or labor-intensive handcrafted feature engineering but performs comparably to or better than state-of-the-art bot detection models/systems.

An earlier version of our neural model and preliminary results were reported in [91]. The earlier model and results in [91] were based on GloVe [92] word embeddings and BiLSTM. Compared with [91].

2.2.2 Clickbait Detection

As a new topic of study, prior research works on clickbait detection extracted and applied hidden features from the data [93, 94, 95]. In such works, both non-textual information (e.g., images) and textual content were taken into account. In addition to the features of semantic and lexical levels extracted from the content, user behavior characteristics were also considered. A variety of machine learning methods were applied based on those obtained features, such as SVM and naïve Bayes.

Chen et al. [95] examined both contextual (e.g., semantic and lexical features) and non-contextual information (e.g., figures, user behavior). The authors applied traditional machine

learning methods such as support vector machines (SVM) and naïve Bayes to these features for final predictions. Potthast et al. [93] examined the features from both the headlines and linked articles/webpages. Additionally, the authors leveraged linguistic information (e.g., the average word length and sentiment preference) and lateral information (e.g., the authorship of a headline). They then fed a combination of these features into traditional machine learning models such as random forest, naïve Bayes, and logistic regression.

Recent research on clickbait detection leverages supervised learning frameworks using a wide range of features, such as readability, forward references, term frequencies [15]), which are linguistic-based; webpage links [93], which is non-linguistic-based; headline stance [96] and user interests [97,98]. Other studies showed that clickbait could be identified by readability, cardinal numbers, particular adjectives, and nouns describing sentiment, which reflect information on authority and sensationalism [99]. More recently, in order to avoid the labour-intensive task of feature engineering, deep learning has been used more widely for clickbait detection [56,57,37].

Razaque et al. proposed a RNN model to determine if a link (URL) in a clickbait message is malicious or harmless [100]. Another RNN model by the same authors determines if the content pointed to by a link (e.g., an article or a social media post) is malicious or harmless [101]. Zhou et al. proposed a clickbait detection model based on graph convolutional networks, and evaluated the model using a dataset in the Chinese language [102]. Recent deep learning models for clickbait detection [38,39,103] adapted and fine-tuned transformer models such as BERT [104], RoBERTa [105], XLNet [106], ELECTRA [107] and ALBERT [108]. Mareddy et al. [103] evaluated their transformer-based model using a large dataset in the Telugu language.

To benefit from deep learning techniques, word vectors [109,110] have also been used to represent textual information in many studies [111]. In [35], the headlines were mapped into word embeddings matrices, which were then fed into convolutional neural networks (CNN). Additionally, recurrent neural networks (RNN), because of their effectiveness in handling sequential data, are commonly adopted for the detection of clickbait. In the Clickbait Chal-

lenge competition, all the top five teams applied RNN-based models such as attention-based BiGRU [37] and LSTM [36] to the analysis of textual input. The first ranked team [37] applied bi-directional gated recurrent units (BiGRU) with incorporated attention mechanisms [112] to model contextual information from the clickbait dataset. In addition, Glenski et al. [36] adopted long short term memory (LSTM) [27] and convolutional neural networks (CNN) to capture both textual and visual information from the dataset and feed them into the models.

However, the above ANN-based models [36, 37] mainly focus on integrating a variety of neural network structures. The importance of semantic correlations between a headline and its linked article was overlooked in the above works. In particular, every pair of words, one from the headline and the other from the linked article (headline-article word pair), can be examined to derive semantic correlations between them and, ultimately, between the headline and the article [18]. The semantic correlations between word pairs are then used to build linguistic knowledge graphs [113]; the graphs will enable neural models to identify similarity patterns in a latent feature space more effectively.

Only a few existing models have used the correlations between headlines and their linked articles for clickbait identification. Biyani et al. [15] used similarities between the headline and the first five sentences of the linked article for clickbait detection, integrated with conventional handcrafted textual features (e.g., the number of upper case words and the number of acronyms). Kumar et al. [16] used Siamese networks to assess similarities between the textual content and visual images. The authors fed the similarity features into several fully connected neural network layers. Dong et al. [17] used bi-directional gated recurrent units (GRU) to learn contextual features of headlines and their linked articles. Additionally, the authors applied a cosine similarity-based method to learn global matching similarities between headlines and their linked articles.

2.2.3 SMS Spam Detection

SMS spam is defined as unsolicited and unwanted messages received on mobile phones via the short message service (SMS). Existing work on SMS spam detection can be broadly divided into two categories: (i) classifiers using traditional machine learning techniques; and (ii) classifiers using deep learning techniques.

Classifiers Based on Traditional Machine Learning

Conventional SMS spam detection systems typically rely on the application of well-known machine learning algorithms such as Support Vector Machine (SVM), Naïve Bayes, K-Nearest Neighbor, Random Forest, etc.

Almeida et al. [23] introduce a public and annotated SMS spam collection and baseline classifiers for further comparison. The list of all classifiers evaluated by Almeida et al. [23] is presented in Table 2.9. Some of the implementations of the evaluated classifiers are provided by Waikato Environment for Knowledge Analysis (WEKA). The algorithms were implemented with their default parameters except when otherwise is specified. In addition, two different tokenizers are used. In the first type of tokenizer (tok1), a token begins with a printable letter, followed by any number of alphanumeric letters, excluding commas, colons, and dots from the middle of the pattern. In the second type of tokenizer (tok2), a token is formed by any sequence of letters separated by tabs, returns, blanks, commas, colons, dots, and dashes.

In addition to the above-mentioned baseline classifiers, Almeida et al. introduce a simple baseline, namely, trivial rejector (TR) and expectation-maximization (EM) clustering algorithm [114], over a vector representation based on the second type of tokenizer (tok2). Their comprehensive evaluation concluded that SVM outperformed all the other evaluated classifiers.

However, these traditional machine-learning-based classifiers require heavy feature engineering, which is a very time consuming and labor-intensive task.

Table 2.9: Classifiers evaluated by Almeida et al. [23].

Basic Naïve Bayes (NB) – Basic NB [115]
Multinomial term frequency NB – MN TF NB [115]
Multinomial Boolean NB – MN Bool NB [115]
Multivariate Bernoulli NB – Bern NB [115]
Boolean NB – Bool NB [115]
Multivariate Gauss NB – Gauss NB [115]
Flexible Bayes – Flex NB [115]
Boosted NB [116]
Logistic Regression [117, 118]
Multilayer Perceptron [119]
Linear Support Vector Machine – SVM [120, 121]
Sequential Minimal Optimization – SMO [122]
Minimum Description Length – MDL [123]
K-Nearest Neighbors – KNN [124, 125] (K = 1, 3 or 5)
C4.5 [126, 125]
Boosted C4.5 [125]
PART [127, 125]
Random Forest [128, 129]

Classifiers Based on Deep learning

Recently, deep learning models have demonstrated their powerful natural language processing (NLP) capabilities, including the tasks of information retrieval [130, 131, 132], machine reading comprehension, and text classification [91]. For sentence representation based on neural networks in text classification, external domain knowledge is quite helpful [30]. In the process of modeling convolutional neural networks (CNN) and recurrent neural networks (RNN), text representations are built by leveraging both the input word sequence and syntactic parse tree. Such a model tends to be reinforced by the pooling feature from CNN via gathering both local and global features from sequential context windows.

Al Moubayed et al. [26] presents a model named TM-SDA (Topic Modelling - Stacked Denoising Autoencoder) for SMS spam detection using recent advances in text mining and unsupervised outlier detection based on deep learning. The topics generated per SMS are then used by stacked denoising auto-encoders (SDA) [133] - an unsupervised deep learning technique - to build a data model. An onset detection approach based on the built SDA model is then used to increase the separation between spam and ham (legitimate messages). Finally, a Fisher linear discriminate analysis (FDA) [134] is used to classify data into spam and ham. Their classifier were compared against state-of-the-art spam detection algorithms: it achieved over 97% accuracy and compared favorably to the best-reported algorithms at the time.

Jain et al. [29] introduce Semantic Long Short Term Memory (SLSTM). It is an LSTM neural network with a semantics layer on the top of it. That is, before using the LSTM for spam classification, the word vectors are enriched semantically. Jain et al. [30] further explore SMS spam detection using a combination of Long Short Term Memory (LSTM) and Convolutional Neural Network (CNN), and introduce a hybrid model called Sequential Stacked CNN-LSTM (SSCL). As shown in Figure 2.1, the three layers (semantics, CNN and LSTM) are sequentially stacked one over another. This combination of two deep learning structures can also be found in some NLP tasks [135] and speech recognition [136].

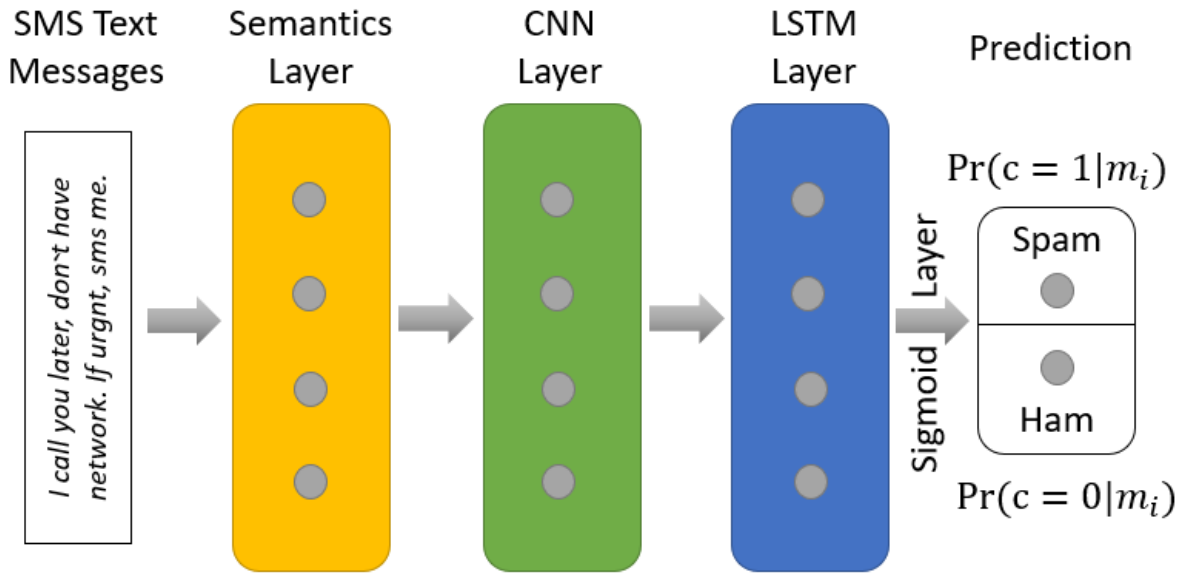


Figure 2.1: The architecture of SSCL.

The deep learning models mentioned above, while producing very good results, typically require a large amount of time and memory to train due to high numbers of learnable parameters and operations in their computational graphs.

Chapter 3

Twitter Bot Detection Using Neural Networks and Linguistic Embeddings

3.1 Motivations and Contributions

In this study, we introduce BOTLE (BOT detection using Linguistic Embeddings), a novel recurrent neural network (RNN) model based on Bidirectional Lightweight Gated Recurrent Units (BiLGRU) for distinguishing Twitter bots from human-operated accounts. The bot detection task is formulated as a binary text classification problem, where only the textual content of tweets is utilized as input. BOTLE integrates multiple linguistic embeddings—namely word-level, character-level, part-of-speech, and named-entity embeddings—to represent tweet content in a rich and diverse manner. Notably, our approach avoids the use of handcrafted features and does not rely on prior knowledge or assumptions regarding user profiles, network structures, or behavioral history, thereby offering a lightweight and generalizable solution.

We evaluate BOTLE on the widely used *cresci-2017* dataset, where it achieves classification performance that is competitive with, and in some cases superior to, existing state-of-the-art Twitter bot detection methods, despite not using prior knowledge or assumptions

about user profiles, friendship networks, or historical behavior of target accounts. The BiLGRU architecture effectively captures both past and future contextual dependencies in tweet sequences, contributing to its strong performance. Additionally, we conduct an extensive ablation study on the impact of different linguistic embeddings and recurrent network variants, providing valuable insights for the development of future deep learning models for social media analysis. Finally, we assess the model’s computational efficiency in terms of training time, inference latency, and memory usage, demonstrating that BOTLE not only achieves state-of-the-art detection results but also meets the requirements for real-time and resource-constrained deployment.

The remainder of the chapter is organized as follows. The proposed model is described in Section 3.2. In Section 3.3, we present experimental results, comparing the performance of our proposed model with that of existing state-of-the-art systems. Sections 3.4 and 3.5 validate the effectiveness of a variety of linguistic embeddings and recurrent variants, respectively, on the proposed neural model for Twitter bot detection. Section 3.6 concludes the chapter.

3.2 The Proposed Model

In this section, we introduce various linguistic embeddings, the BiLGRU architecture, and our proposed Twitter bot detection model, BOTLE. BOTLE employs a recurrent neural network (RNN) to distinguish bots from human-operated accounts. The BiLGRU combines two hidden layers that process input in opposite directions—forward and backward—enabling the model to capture contextual information from both past and future tokens. This bidirectional structure effectively models textual patterns across tweets and delivers competitive classification performance compared to state-of-the-art Twitter bot detection systems.

We utilize multiple types of linguistic embeddings to encode tweets, including word, character, part-of-speech (POS), and named-entity embeddings. By avoiding handcrafted features, which often require extensive domain expertise and manual effort, our approach sup-

ports faster development, easier deployment, and greater scalability of the detection system.

3.2.1 Linguistic Embeddings

We discuss different types of linguistic embeddings, namely, word embeddings, character embeddings, part-of-speech embeddings, and named-entity embeddings.

Word Embeddings

Word embeddings, also known as distributed word representation, is an important research topic in NLP. In recent years, it has been widely used in various NLP tasks, including information retrievals [130, 131, 132], text classification [137], machine translation [138] and machine comprehension [139]. The success of word embedding [110, 92] encourages researchers to focus on machine-learned representation instead of heavy feature engineering in NLP. By using word embeddings as the typical feature representation for words, neural networks become more competitive compared to traditional approaches in NLP.

An important advantage of word embeddings compared to conventional NLP techniques of representation (e.g., bag-of-words [140], part-of-speech tagging [141]) is that it achieves a significant dimensional reduction of the feature set needed to represent tweets, resulting in a reduction in training and inference time of an NLP model.

In this work, we adopt pre-trained GloVe [92] word vectors on Twitter. It is based on two billion tweets, including 27 billion tokens, with a vocabulary size of 1.2 million words. We define our vocabulary as the intersection between the words in all training samples and those in the pre-trained 200-dimensional GloVe. Given a word w , if it is in the vocabulary, we set its word-level embedding α_w to its GloVe word vector, which is fixed during the training; otherwise we have $\alpha_w = \alpha_o \in \mathbb{R}^{200}$, where α_o is a trainable parameter serving as the shared word vector of all out-of-vocabulary (OOV) words. Each tweet is sent to the Stanford CoreNLP [142] toolkit for sentence splitting and tokenization. All words containing Twitter special memes, i.e., hashtag (#), mention (@) and shortened URL (http://t.co), are

mapped to several pre-defined tokens individually, i.e., $\langle HASHTAG \rangle$, $\langle USER \rangle$ and $\langle URL \rangle$, using regular expression matches.

Character Embeddings

Kim et al. [143] introduced a character-level embedding to provide additional features in dealing with unknown or out-of-vocabulary (OOV) words in NLP tasks. Robust approaches for obtaining shape and morphological knowledge from words need to consider all characters of the word and choose the significant characteristics for the task to be carried out [144]. For instance, in the task of Twitter bot detection, abbreviations, spelling errors, and special characters ('#'), are very common. Moreover, important information can appear in different parts of a hashtag (e.g., "#ComingSoon", "#ILikePRODUCT"). Since character information can provide knowledge on structural differences between words, character embeddings are beneficial to a Twitter bot detection model from learning unknown and uncommon words [145]. Additionally, character embeddings can improve the detection performance using suffixes and prefixes such as '-able' and 'dis-', which carry semantic knowledge [146].

In this experiment, we incorporate convolutional-based character embeddings (Char-CNN) [143] to our Twitter bot detection model. Specifically, let V be the vocabulary size of words, C be the vocabulary of characters. Given a word $w \in V$ composed of sequential characters $\{c_1, c_2, \dots, c_n\}$, where n is the length of the word w , the character embedding of w is denoted as $C^w \in \mathbb{R}^{d \times n}$. A feature map $f^w \in \mathbb{R}^{n-h+1}$ is obtained after applying a convolution between C^w and a kernel $K \in \mathbb{R}^{d \times h}$ of width h . Then, max-pooling is performed over each feature map. Finally, the character-level representation of the word w is obtained as $E^w = [e_1^w, e_2^w, \dots, e_k^w]$, with a total of k kernels. Following [147], we set k to 100 and h to 5 in this experiment.

Learned Character Embedding

According to [143], every CharCNN filter learns to identify certain character n-grams. In order to obtain a better visualization of what the character embedding is learning, the learned character representations via principal components analysis (PCA) are plotted as shown in Fig. 3.1. Each character is fed into the CharCNN and the CharCNN’s output is used as the fixed dimensional representation for the corresponding character n-gram. From Figure 3.1, we can observe that the learned model is able to distinguish between suffixes (grey), prefixes (blue), and others (red). Note that the embeddings are particularly sensitive to character n-grams containing hyphens (green) in that this is a solid indication of a word’s part-of-speech property (e.g., hyphen used if the two words work together as an adjective before the noun they illustrate.) [143].

Part-of-Speech Embeddings

Words are typically ambiguous in their part of speech. In a sentence, the context of a word could address this vagueness [148]. For example, *work* can be either a noun or a verb. In the sentence “*There is plenty of work to be done in the garden.*”, *work* can only be a noun.

Part-of-speech (POS) tagging is a process that assigns the part of speech (e.g., adjective, verb) to words using their contextual information. In order to be effectively applied to neural networks, POS tags are converted into vectors of fixed length, and considered as trainable variables that are randomly initialized. Applications of POS embedding exist in many NLP tasks such as information extraction, and machine reading comprehension [149].

Following [149], we use 9-dimensional POS embedding for a total of 56 distinct types of POS tags.

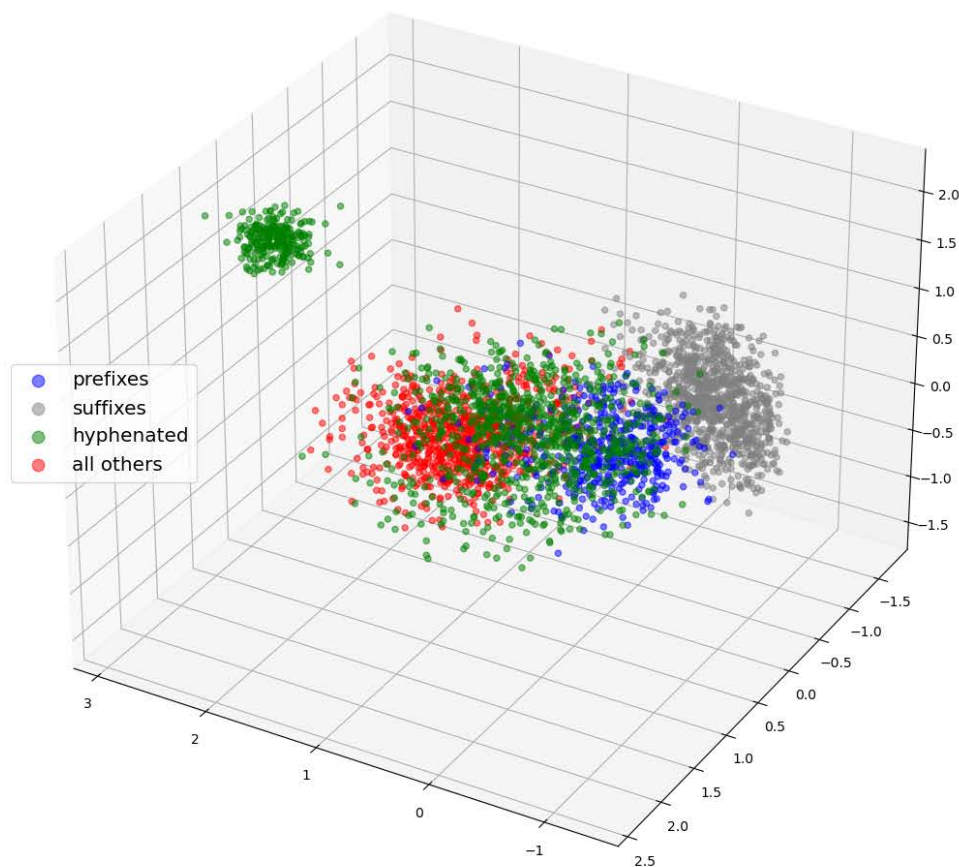


Figure 3.1: Plot of character n-gram representations via PCA. Colors correspond to prefixes (blue), suffixes (grey), hyphenated (green), and all others (red). Prefixes refer to character n-grams that start with the start-of-word character. Suffixes likewise refer to character n-grams which end with the end-of-word character.

Named-Entity Embeddings

‘Named entity’ (NE) is a term widely used in natural language processing (NLP). A named entity refers to an entity such as a location, organization, product or person that can be named using a proper noun. Table 3.1 illustrates the necessity of identifying named entities in a wide variety of NLP tasks. In the absence of contextual information, it would be difficult or impossible to know that “Bulls” is an organization. Likewise, the term “White Sox” mentioned in the example is an organization, not a type of clothing.

Table 3.1: Example illustrating named-entity identification.

PERSON [PER] GEO-POLITICAL ENTITY [GPE] ORGANIZATION [ORG] MISCELLANEOUS [MISC]									
Michael Jordan	[PER]	played	13 seasons	[MISC]	with	the	Bulls	[ORG]	with
a year and a half	[MISC]	hiatus to play baseball in the	Chicago	[GPE]	White Sox	[ORG]			
minor league system.									

Named-entity embedding of context words can improve the accuracy of machine reading comprehension [150]. The method of encoding named-entity tags is similar to that of POS tags. Named-entity tags are mapped into a fixed-size matrix, and considered as learnable variables in neural models.

Following [149], we use 8-dimensional named-entity embeddings for a total of 18 distinct types of named-entity tags.

3.2.2 BiLGRU Neural Networks

We briefly describe RNN, LGRU, BiLGRU, and then our proposed Twitter bot detection model.

RNN [151] is a class of artificial neural sequence models, as shown in Fig. 3.2, where connections between units form a directed cycle. It takes arbitrary embedding sequences $x = (x_1, \dots, x_T)$ as input, and uses its internal memory network to exhibit dynamic temporal behavior. It consists of a hidden unit h and an optional output y . T is the last time step and is also the length of the input sentence in this text sequence learning task. At each time step t , the hidden state h_t of the RNN is computed based on the previous hidden state h_{t-1} and the input at the current step x_t :

$$h_t = g(Ux_t + Wh_{t-1}) \quad (3.1)$$

where U and W are weight matrices of the network; $g(\cdot)$ is a non-linear activation function, such as an element-wise logistic sigmoid function. The output at time step t is computed as $y_t = \text{softmax}(Vh_t)$, where V is another weight parameter of the network; *softmax* is an activation function often implemented at the final layer of a network.

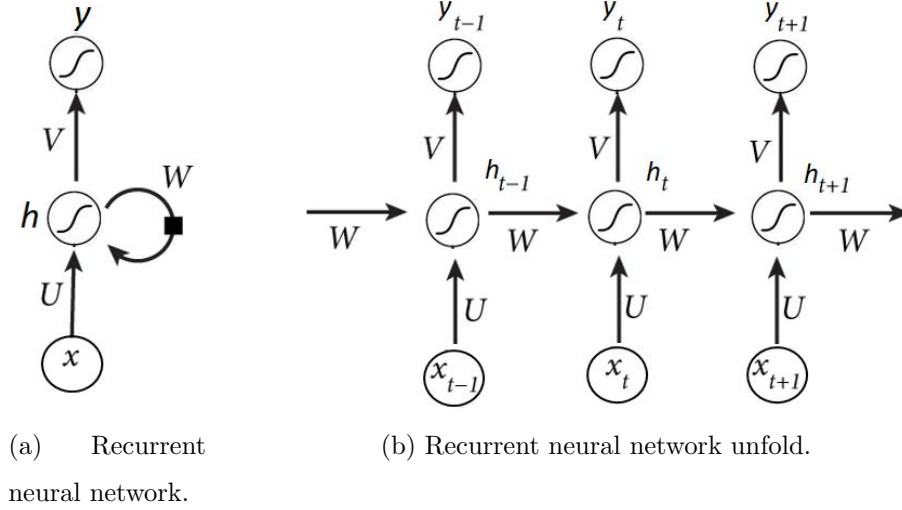


Figure 3.2: (a) Recurrent neural network and (b) Recurrent neural network unfold. Adapted from [45].

There are several types of RNNs, such as long short-term memory (LSTM), gated recurrent unit (GRU), and minimal gated unit (MGU).

LSTM is a variant of RNN designed to deal with the vanishing gradients problem. Hochreiter and Schmidhuber [27] found that the LSTM, which uses purpose-built memory cells to store information, is better at finding and utilizing long-range context.

GRU [152] is introduced to avoid the issues of vanishing or exploding gradients suffered by a vanilla recurrent neural network. Compared to LSTM, GRUs have fewer memory cells in their architecture, which results in lower memory requirements. In GRUs, recurrent units adaptively capture the dependencies of various time scales. There are two gating units, namely, the reset gate and the forget gate, that modulate the flow of information inside the GRU. Unlike LSTM, GRUs do not have separate memory cells; furthermore, GRUs have

no mechanism to control the degree to which its state or memory content is exposed, while LSTM can control how much memory content it wants to reveal [153].

Minimal Gated Unit (MGU) [154] is an alternative to GRU. MGU is composed of one gate, namely, the forget gate, and does not utilize the complex peephole connection [155]. This leads to fewer parameters compared to LSTM and GRUs. As a result, MGUs can obtain comparable classification performance but need less training time than GRUs for the tasks of sentiment analysis, image classification, and language modelling [154].

LGRU [156] is a recently introduced recurrent structure that is simpler than but as effective as LSTM. Like LSTM, LGRU relies on gating mechanisms to obtain long-term dependencies in sequential data:

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \quad (3.2)$$

$$\tilde{h}_t = \phi(W_h[f_t \cdot h_{t-1}, x_t] + b_h) \quad (3.3)$$

$$h_t = (1 - f_t) \cdot h_{t-1} + f_t \cdot \tilde{h}_t \quad (3.4)$$

where f_t denotes forget gate; w_f is weight vectors of the network; the symbol $\cdot$ stands for scalar multiplication.

In the above equations, the update gate and the reset gate in GRU are substituted by a forget gate f_t in LGRU. Inspired by [157], we further simplify LGRU by i) transforming the weight matrices of forget gates into vectors and ii) replacing element-wise multiplications with scalar multiplications. The motivation is that applying fewer learnable parameters in f_t and simplifying operations in $\tilde{h}_t$ and h_t can result in lower complexity in training recurrent models without compromising the performance. As a result, LGRU is expected to be more lightweight than all other variants of GRU (including LSTM) while offering comparable performance. Figure 3.3 illustrates the differences in data flows and operations between GRU and our simplified LGRU.

BiLGRU uses two LGRUs to learn each token of the sequence based on both the past and the future context of the token. As shown in Fig. 3.4, one LGRU processes the sequence from left to right; the other one, from right to left. At each time step t , a hidden forward

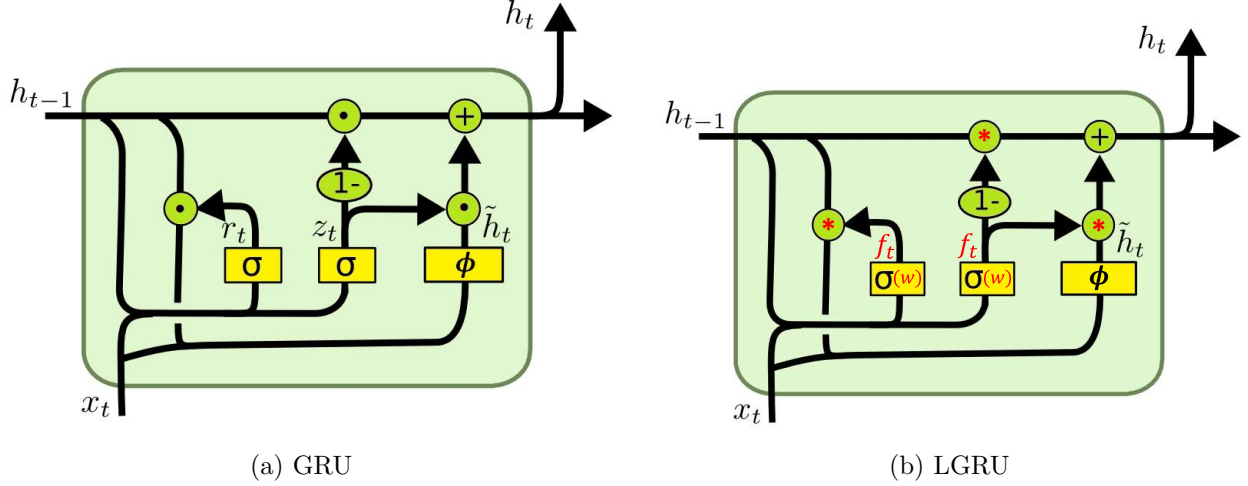


Figure 3.3: Comparison of data flow and operations between GRU and our LGRU. The differences are highlighted in red.

layer with the hidden unit function $\vec{h}$ is computed based on the previous hidden state $\vec{h}_{t-1}$ and the input at the current step x_t . Additionally, a hidden backward layer with the hidden unit function $\overleftarrow{h}$ is computed based on the future hidden state $\vec{h}_{t+1}$ and the input at the current step x_t . The forward and backward context representations, generated by $\vec{h}_t$ and $\overleftarrow{h}_t$, respectively, are concatenated into a long vector. The combined outputs are the predictions of target sequences.

The Proposed Model. As shown in Fig. 3.5, we make use of a fully connected softmax layer to output posterior probabilities over labels from two classes, standing for Twitter bots or humans. The input is a sequence of n tokens, $(x_1, \dots, x_n)$. The predictions in both directions are modeled by three-layer BiLGRUs with hidden states $\vec{h}_{i,\ell}$ and $\overleftarrow{h}_{i,\ell}$ for input token x_i at the layer level $\ell = 1, \dots, L$. The final layer's hidden state $h_{i,L} = [\vec{h}_{i,L}; \overleftarrow{h}_{i,L}]$ is used to output the probabilities over binary labels after softmax normalization. They share the word embedding layer and the softmax layer, parameterized by Θ_e and Θ_s , respectively. The model is trained to minimize the negative log-likelihood in both directions:

$$\mathcal{L} = -\left(\log p_{\rightarrow}(y \mid x_{1:n}; \Theta_e, \vec{\Theta}_{\text{LGRU}}, \Theta_s) + \log p_{\leftarrow}(y \mid x_{1:n}; \Theta_e, \overleftarrow{\Theta}_{\text{LGRU}}, \Theta_s)\right) \quad (3.5)$$

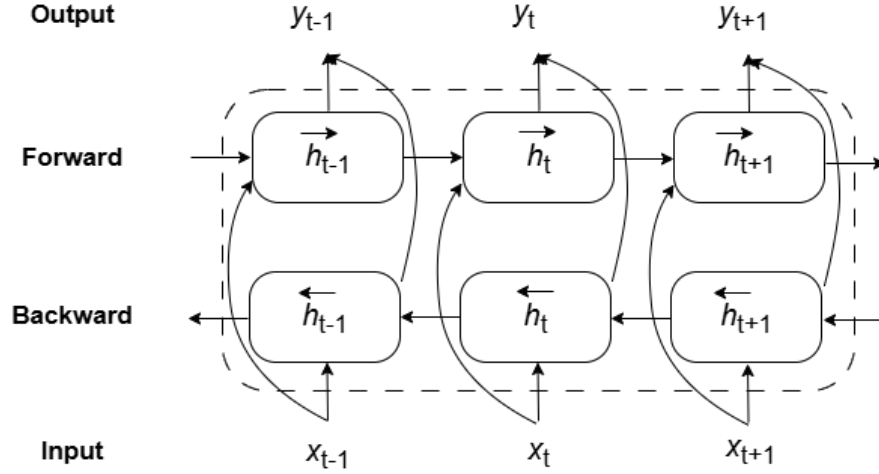


Figure 3.4: Bidirectional LGRU.

3.3 Experimental Results

We conducted experiments using real-world datasets. Results on the *cresci-2017* dataset demonstrate that BOTLE outperforms or matches the performance of state-of-the-art Twitter bot detection models [11, 8, 7, 12, 9], despite not relying on prior knowledge or assumptions about user profiles, friendship networks, or the historical behavior of target accounts.

We further evaluate the effectiveness of various linguistic embeddings and recurrent neural network variants for Twitter bot detection. These findings can guide researchers in developing high-performance deep learning models for similar tasks.

In addition to the classification performance, we also report the training and inference times, along with memory consumption of the proposed model, demonstrating its practicality for real-world deployment.

We first briefly describe the datasets in Section 3.3.1. In Section 3.3.2, 3.3.3, and 3.3.4, the existing systems/models used for performance comparisons, settings, and evaluation metrics are described, respectively. Section 3.3.5 presents experimental results, comparing the performance of the proposed model with that of existing state-of-the-art systems and

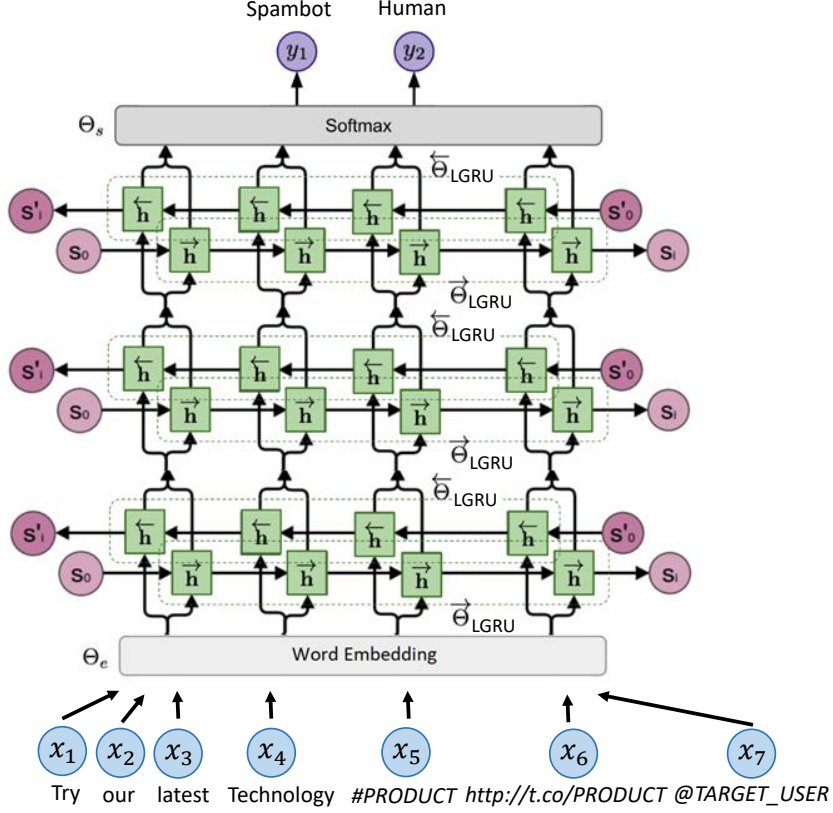


Figure 3.5: An illustration of the proposed BiLGRU neural model with word embeddings for distinguishing Twitter bots from human accounts.

analyzing the results in detail.

3.3.1 Datasets

We evaluate our proposed model using the public annotated dataset *cresci-2017* [158], consisting of 3,474 human accounts that generated 8.4 million tweets and 1,455 bots that generated 3 million tweets. We prepared two test sets following [83]. The test set #1 and test set #2 refer to groups where human accounts are mixed with accounts from dataset *social-bot-1* and dataset *social-bot-3*, respectively. *Social-bot-1* is about retweeters of an Italian political candidate, while *social-bot-3* is about spammers of products on sale on Amazon.com. The test set #1 is composed of 1,982 accounts and 4,061,598 tweets, while the test set #2 is

composed of 928 accounts and 2,628,181 tweets. The statistics of datasets are shown in Table 3.2 in detail.

Table 3.2: Statistics of the datasets used in the experiments.

Dataset	Description	Statistics	
		Accounts	Tweets
Human accounts	Verified accounts that are human-operated	3,474	2,839,361
Social-bot-1	Retweeters of an Italian political candidate	991	1,610,034
Social-bot-3	Spammers of products on sale at Amazon.com	464	1,418,557
Test set #1	Mixed set of 50% human accounts + 50% social-bot-1	1,982	4,061,598
Test set #2	Mixed set of 50% human accounts + 50% social-bot-3	928	2,628,181

In order to gain more insight into the datasets, and thus the effectiveness of our proposed model, we generated a *word-cloud* for comparison of the most frequent words in the two datasets, i.e., human accounts and social-bot-3, as shown in Fig. 3.6. Word-cloud is a visualisation method that displays how frequently words appear in a given body of text, by making the size of each word proportional to its frequency of appearing in the text. Note that Amazon social bots on Twitter usually use exaggerated words such as *Check awesome*, *Read Fascinating* and *Creative Writing*, to attract people’s attention in order to advertise their products or services. Furthermore, a manual analysis of 100 randomly selected tweets in social-bot-3 showed that a majority of their tweets contain links to external web pages. This is in contrast to the human accounts (in the random sample), which describe the accounts’ owners using words such as *love*, *happy birthday*, *haha*, *lol*, *thank*, and *friend*, and most of whom seldom tweet links to external web pages.

hashtags, mentions and retweets.

- Cresci et al. [11] create two groups of user behavior features: tweet type DNA and tweet content DNA.

To obtain the classification performance of the above methods/systems for a comparison, we extracted the results reported Cresci et al. [11], which are deemed the best results the authors could achieve when they evaluated their proposed methods/systems.

Note that most state-of-the-art algorithms/systems for Twitter bot detection require a large number of data-demanding features. The above systems require feature engineering based on six features [11] to more than 1,000 [7], as listed in Table 3.3. Feature engineering is very labor-consuming and expensive in terms of data collection and pre-processing. Our proposed RNN model does not rely on any feature engineering and uses only the textual content of the tweets.

Table 3.3: The numbers of handcrafted features used in Twitter bot detection methods/models.

Model	BotOrNot[7]	Miller et al. [9]	Yang et al. [8]	Ahmed et al. [12]	Cresci et al. [11]	BOTLE
Number of features	1000+	126	25	14	6	0

3.3.3 Neural Network Model Setup

We implement the proposed model using TensorFlow [159] with Python. Based on the design of the experiments, we tested several sets of parameters to select one that gave the experiments the best performance. We applied the grid search method to obtain the optimal combination of hyperparameters. These hyperparameters are as follows:

- *Learning rate*: the model is trained using the stochastic gradient descent [160] algorithm, while the learning rate is set to 0.01.

- *Network structure*: three stacked BiLGRU layers with 200 recurrent units and one fully connected softmax layer.
- *Dropout* [161] is adopted during training, initially set to 0.5, slowly decreasing during training until it reaches 0.1 at the end.
- *Number of epochs*: 30.
- *Momentum*: 0.9.
- *Mini-batch*: 64.

3.3.4 Evaluation Metrics

To evaluate the effectiveness of our proposed Twitter bot detection model, we use four standard indicators:

- *True Positives (TP)*: the number of bots correctly recognized;
- *True Negatives (TN)*: the number of human accounts correctly recognized;
- *False Positives (FP)*: the number of human accounts erroneously recognized as bots;
- *False Negatives (FN)*: the number of bots erroneously recognized as human accounts.

For each test set, we use the four standard evaluation metrics, namely, precision, recall, accuracy, and F1-score, to compare the performance of the classifiers.

Precision and recall are two widely applied metrics to evaluate a Twitter bot detection system. If precision is low, the system requires manual verification by humans. It is labour intensive, if the number of false positives is high. However, if recall is low, the system fails to catch many bot accounts. Since the F1-score takes both false positives (precision) and false negatives (recall) into account, it reflects the balanced overall performance of the system.

3.3.5 Results and Discussion

Table 3.4 shows the performance of our proposed neural model BOTLE along with that of the baseline models reported in [11]. Using the test set #1, BOTLE achieves a new recall score of 0.977, outperforming the best (Cresci et al. [11]) by 0.5 percentage points (0.977 vs. 0.972). Using the test set #2, BOTLE achieves a new F1-score of 0.936, surpassing the best (Cresci et al. [11] and Ahmed et al.[12]) by 1.3 percentage points (0.936 vs. 0.923); our accuracy score of 0.939 also outperforms the best (Cresci et al. [11]), by 1.0 percentage points (0.939 vs. 0.929). In addition, BOTLE achieves the second-best results in the other cases (highlighted in blue color in Table 3.4). In summary, BOTLE offers the best or second-best results in all cases.

BOTLE outperforms the current state-of-the-art system by Cresci et al. [11] on several metrics such as accuracy and F1-score (using the test set #2) and recall (using the test set #1). Although BOTLE performs slightly below Cresci et al. [11] in some cases, it offers many significant advantages over [11] as follows.

No handcrafted features required: BOTLE does not rely on any human-engineered features. Existing classifiers require feature engineering, ranging from six features (Cresci et al. [11]) to more than 1,000 (BotOrNot [7]). One of the best performers, Cresci et al. [11], still requires six features, namely, URLs, hashtags, media, entities, mentions, and plain text. Furthermore, the process of constructing bio-inspired “DNA” sequences in [11] is very time-consuming and labor-intensive.

No prior knowledge required: BOTLE does not require prior knowledge or assumptions about users’ profiles, friendship networks, or historical behavior of the target accounts. It uses only the textual contents of users’ tweets for classification. The method by Cresci et al. [11], nonetheless, requires prior knowledge such as replies or retweets to construct bio-inspired “DNA” sequences. It is expensive to collect, store and pre-process a large amount of data based on features. Without feature engineering, BOTLE can be implemented and deployed much faster and earlier than the other models.

Table 3.4: Performance comparison of various bot detection methods/models using the cresci-2017 dataset. The proposed model BOTLE uses LGRU and four linguistic embeddings (word, character, POS, and NE). The best and second best results are highlighted in green and blue, respectively. The results of human annotators and of the baseline models BotOrNot? [7], Yang *et al.* [8], Miller *et al.* [9], Ahmed *et al.* [12], and Cresci *et al.* [11] are taken from [11]; the results of Wei & Nguyen [91] are from [91].

			Detection Result			
			Precision	Recall	Accuracy	F1-score
Test Set #1	Human annotators	Manual	0.267	0.080	0.698	0.123
	BotOrNot?[7]	Supervised	0.471	0.208	0.734	0.288
	Yang <i>et al.</i> [8]	Supervised	0.563	0.170	0.506	0.261
	Wei & Nguyen [91]	Supervised	0.940	0.976	0.960	0.958
	BOTLE	Supervised	0.956	0.977	0.969	0.967
	Miller <i>et al.</i> [9]	Unsupervised	0.555	0.358	0.526	0.435
	Ahmed <i>et al.</i> [12]	Unsupervised	0.945	0.944	0.943	0.944
	Cresci <i>et al.</i> [11]	Unsupervised	0.982	0.972	0.976	0.977
Test Set #2	Human annotators	Manual	0.647	0.509	0.829	0.570
	BotOrNot?[7]	Supervised	0.635	0.950	0.922	0.761
	Yang <i>et al.</i> [8]	Supervised	0.727	0.409	0.629	0.524
	Wei & Nguyen [91]	Supervised	0.933	0.919	0.929	0.926
	BOTLE	Supervised	0.938	0.934	0.939	0.936
	Miller <i>et al.</i> [9]	Unsupervised	0.467	0.306	0.481	0.370
	Ahmed <i>et al.</i> [12]	Unsupervised	0.913	0.935	0.923	0.923
	Cresci <i>et al.</i> [11]	Unsupervised	1.000	0.858	0.929	0.923

3.4 Effectiveness of Linguistic Embeddings

For the results In Table 3.4, we use word embeddings, character embeddings, part-of-speech embeddings and named-entity embeddings. In this section, we validate the effectiveness of these linguistic embeddings by adding them one at a time to the bot detection model and measure the performance improvement as each type of embedding is added.

3.4.1 Linguistic Embeddings: Detection Performance

As shown in Table 3.5, we started with only word embeddings. We then added character, POS, and NE embeddings, one type at a time, in that order. Every time a type of embedding was added, we conducted the same experiments using test sets #1 and #2, and recorded the resulting precision, recall, accuracy and F1-score.

The results in Table 3.5 and and Figure 3.7 show that the detection performance is improved every time a new type of embedding was added. For example, using the test set #1, the accuracy is improved from 0.961 to 0.965, 0.966 and 0.969 as character, POS, and NE embeddings were added one by one, respectively. The same observation applies to the other performance metrics, and to test set #2 as well. This consistently demonstrates the effectiveness of the additional types of linguistic embeddings. They enable the proposed model to encode rich linguistic and syntactic knowledge such as affixes, grammar, and part-of-speech.

On the other hand, incorporating additional linguistic embeddings will increase the run time and memory usage of a model. In the next section we examine the overheads incurred by additional linguistic embeddings in the proposed model BOTLE.

3.4.2 Linguistic Embeddings: Running Time and Memory Usage

In this section, we report the runtime (for training and inference) and memory usage of BOTLE as we added character, POS, and NE embeddings, one type at a time, in that

Table 3.5: Empirical assessment of different linguistic embeddings: word, character, part-of-speech (POS), and named-entity (NE) embeddings. All models use LGRU. The best results are highlighted in green. The performance of the bot detector improves as more embeddings are added to the model.

	Linguistic Embeddings				Precision	Recall	Accuracy	F1-score
	Word	Char	POS	NE				
Test Set #1	✓				0.943	0.974	0.961	0.958
	✓	✓			0.949	0.975	0.965	0.962
	✓	✓	✓		0.953	0.974	0.966	0.963
	✓	✓	✓	✓	0.956	0.977	0.969	0.967
Test Set #2	✓				0.934	0.919	0.930	0.927
	✓	✓			0.935	0.926	0.933	0.931
	✓	✓	✓		0.936	0.929	0.936	0.933
	✓	✓	✓	✓	0.938	0.934	0.939	0.936

order, resulting in four linguistic embedding combinations: (1) word embeddings only; (2) word and character embeddings; (3) word, character and POS embeddings; and (4) word, character, POS and NE embeddings. For each combination (model), we conducted the same experiments using test set #2, and recorded the resulting run time and memory usage.

To obtain the training time, we trained a model with each of the above four combinations using 2,099,261 samples from the test set #2. To obtain the average inference time, we ran a model with each combination to classify 528,920 samples from the test set #2. The total inference time for classifying 528,920 samples was then divided by 528,920 to obtain the average inference time of one sample.

To obtain the GPU memory consumption of a model, we profiled the job using NVIDIA

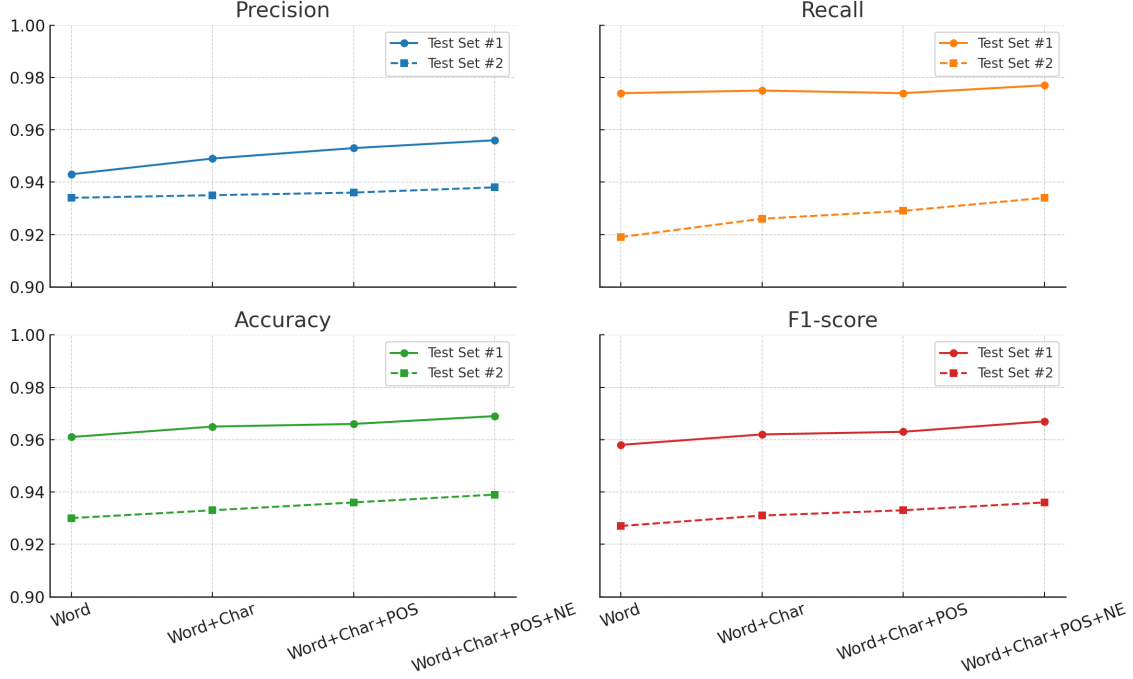


Figure 3.7: Performance trends across linguistic embeddings (LGRU).

NVML [162]. The NVIDIA library tracked and summarized the memory usage throughout the whole process of training and inference. All the experiments were run on a Linux workstation with the following configuration: NVIDIA GeForce GTX 1080 graphics card, Intel Core i7-6700 3.4GHz processor, 12GB DDR4 memory, and 1TB solid state drive.

Table 3.6 illustrates the total training time, average inference time, and memory usage of the four linguistic embedding combinations. We observe that the run time and memory usage increase as more linguistic embeddings are added to the detection model. These increases are expected because they are the price to pay for higher classification performance (shown in Table 3.5) as more linguistic embeddings are incorporated into the classifier. These increases are acceptable for the task of Twitter bot detection on laptops, desktops or cloud services to achieve higher detection performance via the use of all types of linguistic embeddings. For example, using all four types of linguistic embeddings, the average inference time per sample is only 25.8 msec, and the combined memory consumption for training and inference is 5.9

gigabytes. (For other applications on low power, low memory devices, a trade-off analysis should be made to select the type(s) of linguistic embeddings optimal for both classification performance and resource constraints.)

Table 3.6: Empirical assessment of running time and memory usage incurred by the linguistic embeddings: word, character, part-of-speech (POS), and named-entity (NE) embeddings. All models use LGRU.

Linguistic Embeddings				Training time (minutes)	Average inference time per sample (milliseconds)	Memory usage (gigabytes)
Word	Char	POS	NE			
✓				2,257	12.6	4.5
✓	✓			2,551	18.2	5.2
✓	✓	✓		2,782	22.5	5.6
✓	✓	✓	✓	2,997	25.8	5.9

3.5 Performance Comparison of Different Recurrent Variants

During the design process, we considered several recurrent variants, namely, long short-term memory (LSTM), gated recurrent unit (GRU), minimal gated unit (MGU) and lightweight gated recurrent unit (LGRU), which are described in Section 3.2.2. We implemented and tested these variants and selected the best performer for our task, which is LGRU. In this section, we report the results of this selection process to help other researchers decide on a variant for similar tasks. The selection criteria are detection performance, run time and memory usage.

3.5.1 RNN Variants: Detection Performance

Table 3.7 shows the classification performance of the recurrent variants LSTM, GRU, MGU, and LGRU with all four embeddings (word, character, POS, and NE) on the *cresci-2017* dataset. The other parameters are the same as those in Section 3.3.3. Overall, LGRU yields the best performance among the variants. Although the classification improvement of LGRU over the other variants seems marginal, LGRU really shines when it comes to run time and memory usage as discussed in the next section.

Table 3.7: Performance of four recurrent variants: LSTM, GRU, MGU, and LGRU. Model structures are all bidirectional RNNs, but each model uses a different recurrent variant. The models are enhanced with all four embeddings: word, character, POS and NE. The best results are highlighted in green.

	Recurrent Variants	Precision	Recall	Accuracy	F1-score
Test Set #1	LSTM	0.952	0.977	0.967	0.965
	GRU	0.948	0.977	0.965	0.962
	MGU	0.949	0.975	0.964	0.962
	LGRU	0.956	0.977	0.969	0.967
Test Set #2	LSTM	0.936	0.931	0.937	0.934
	GRU	0.935	0.934	0.938	0.934
	MGU	0.933	0.934	0.937	0.933
	LGRU	0.938	0.934	0.939	0.936

3.5.2 RNN Variants: Running Time and Memory Usage

In this section, we report the run time (training and inference) and memory usage of the four recurrent variants LSTM, GRU, MGU and LGRU obtained from the same experiments discussed above in Section 3.5.1. The total training time, average inference time and memory usage were measured and collected as described in Section 3.4.2 and are shown in Table 3.8.

Table 3.8: Empirical assessment of running time and memory usage incurred by four recurrent variants LSTM, GRU, MGU, and LGRU. Model structures are all bidirectional RNNs, but each model uses a different recurrent variant. The models are enhanced with all four embeddings: word, character, POS and NE. The best results are highlighted in green.

Linguistic Embeddings				Training time (minutes)	Average inference time per sample (milliseconds)	Memory usage (gigabytes)
LSTM	GRU	MGU	LGRU			
✓				4,168	31.3	6.6
	✓			3,705	29.0	6.4
		✓		3,380	27.7	6.2
			✓	2,997	25.8	5.9

The training time of LGRU is 39%, 24%, and 12.8% shorter than that of LSTM, GRU and MRU (2,997 minutes vs. 4,168, 3,705 and 3,380 minutes), respectively. The average inference time of LGRU is 21.3%, 12.4% and 7.4% shorter than that of LSTM, GRU and MRU (25.8 msec vs. 31.3, 29, and 27.7 msec), respectively. LGRU also consumes less memory than LSTM, GRU and MRU: 11.9%, 8.5% and 5.1% less memory (5.9 gigabytes vs. 6.6, 6.4 and 6.2), respectively.

The above results demonstrate that LGRU is the best performer among the RNN variants. It provides the best of both worlds: higher classification performance and lower overheads in terms of memory usage and run time.

3.6 Chapter Summary

This chapter presents an RNN model, specifically BiLGRU, with four linguistic embeddings (i.e., word, character, POS, and NE embeddings) to distinguish Twitter bots from human accounts. Our model requires no prior knowledge or assumption about users' profiles, friendship networks, or historical behavior on the target account. To the best of our knowledge, our work is the first that develops an RNN model with several types of linguistic embeddings to detect bots that relies only on tweets and does not require any feature engineering. Experiments on the public dataset *cresci-2017* show that our model performs similarly or better than existing work, but without handcrafted feature engineering, which is labor-intensive and time-consuming. This advantage allows for faster and easier implementation and deployment of the bot detection scheme. In addition, our proposed bidirectional recurrent neural architecture can be relatively easily adapted to a new problem, for example, using BiLGRU with linguistic embeddings to detect phishing emails, spam webpages or spam SMS.

Chapter 4

An Attention-Based Neural Network Using Human Semantic Knowledge and Its Application to Clickbait Detection

4.1 Motivations and Contributions

Clickbait refers to content crafted to entice users into clicking hyperlinks that lead to web pages, articles, images, or videos. It is commonly used for three purposes:

1. phishing attacks, which aim to steal sensitive information or install malicious software.
2. marketing strategies, both legitimate and illegitimate, which rely on attention-grabbing headlines to generate user clicks and advertising revenue.
3. dissemination of misinformation or disinformation through sensational headlines linked to false content.

This study focuses on the latter two cases. Specifically, it aims to determine whether a headline constitutes clickbait by evaluating the semantic similarity between the headline and the textual content of the linked article. A headline that closely reflects the article content is considered non-clickbait, while a significant mismatch indicates clickbait. An example is given in Table 1.1.

It should be noted that not all articles with clickbait headlines are fake news. However, clickbait headlines are one of the hallmarks of fake news. Therefore, the classification of clickbait is useful as the first step toward a more fine-grained classification of fake news (e.g., by examining the content of the article for authenticity).

The remainder of this chapter is organized as follows. Section 4.2 describes in detail the proposed attention-based neural network that uses human semantic knowledge. Section 4.3 discusses the application of the proposed neural network to clickbait detection. In Section 4.4, we present and analyze experimental results to validate the effectiveness of pre-trained word embeddings, knowledge enhancement to the attention mechanisms, and human semantic knowledge in the proposed model KED. We also compare the performance of the proposed model with that of existing state-of-the-art models/systems. Section 4.5 presents concluding remarks and future work.

4.2 The Proposed Attention-based Neural Network

We propose a novel attention-based neural network model, Knowledge-Enhanced Clickbait Detector (KED), designed to improve clickbait detection by incorporating structured linguistic knowledge. KED consists of two main components: (a) human semantic knowledge extraction, and (b) attention mechanisms guided by knowledge graphs.

At the core of KED is the integration of semantic knowledge from WordNet-based linguistic knowledge graphs. These graphs are used to guide both joint-attention and self-attention mechanisms, allowing the model to better capture semantic relationships between headline

and article word pairs.

The knowledge-enhanced joint-attention mechanism captures global semantic similarities, while the knowledge-enhanced self-attention mechanism filters out noise and emphasizes the most relevant similarity features for final predictions. This dual attention strategy enables KED to effectively learn both local and global discriminative features.

By effectively embedding external semantic knowledge into the attention mechanisms, KED significantly improves the model’s ability to detect nuanced semantic correlations, which is an essential capability in distinguishing clickbait from legitimate headlines.

Experimental results demonstrate that this approach not only improves the performance of KED itself, but also enhances several powerful pre-trained language models, including BERT, RoBERTa, and XLNet.

4.2.1 Human Semantic Knowledge Extraction Using WordNet

In this section, we describe in detail our human semantic knowledge extraction scheme, which uses WordNet [20] to extract human semantic knowledge from headline-article word pairs in the dataset. Semantic relations extracted from WordNet are used to build linguistic knowledge graphs, which are then input into the model.

Intuitively, knowledge about antonyms, co-hyponyms, synonyms, hyponyms, hypernyms, and synonyms between given words can potentially help to align word pairs between a given headline and its linked article. For instance, knowledge about synonyms, hypernyms and hyponyms is helpful to identify entailment relations; knowledge about antonyms and co-hyponyms helps to recognize contradiction relations [113].

We extend [113] to extract relations of lexical pairs in a configurable manner by using a hyperparameter $\tau \in \mathbb{N}$ as follows.

- (a) Synonym relation: If the words in the pair are synonyms in WordNet, the pair yields 1 (true). Otherwise, it yields 0 (false).

- (b) Antonym relation: If the words in the pair are antonyms in WordNet, the pair yields 1 (true). Otherwise, it yields 0 (false).
- (c) Hypernym relation: If one word is a hypernym of another word in WordNet, the pair yields value $1 - n/\tau$, where n denotes the number of edges between the two words in a hierarchy, and the hyper-parameter τ denotes the exact path length. Otherwise, it yields 0 (false). Consider two word pairs: “fisherman, skilled worker” and “fisherman, worker”, and $\tau = 5$. Since ‘skilled worker’ is the direct hypernym of ‘fisherman’, and ‘worker’ is the direct hypernym of ‘skilled worker’ (i.e., $\text{worker} \xleftarrow[\text{hypernym}]{\text{direct}} \text{skilled worker} \xleftarrow[\text{hypernym}]{\text{direct}} \text{fisherman}$), they yield $\{\text{fisherman, skilled worker}\} = 1 - \frac{1}{5} = 0.8$ and $\{\text{fisherman, worker}\} = 1 - \frac{2}{5} = 0.6$, respectively. It is worth noting that a large τ increases the chances of trivial hypernym relations retrieved from the knowledge base, eventually leading to lower performance. On the other hand, a small τ would limit the number of hypernym relations obtained from the knowledge base, also resulting in lower performance.
- (d) Hyponym relation: It is the inverse of the hypernym feature.
- (e) Co-hyponym relation: If the two words have the same hypernym, but they do not belong to the same synset, the pair yields 1 (true). Otherwise, it yields 0 (false).

Note that hyperparameter τ enables effective extractions of hypernym relations (and, consequently, hyponym and co-hyponym relations) from the knowledge base by avoiding extracting too few or too many hypernym relations.

The above extracted relations from the knowledge base WordNet applied to the joint-attention mechanism facilitates the matching between headlines and their linked articles.

4.2.2 The Proposed Knowledge Enhanced Attention Mechanisms

Attention mechanisms are essential components of our model. In this section, we describe in detail the proposed knowledge enhanced attention mechanisms.

Attention mechanisms aim to fuse the associated representations of headline-article word pairs. There are two attention mechanisms: joint attention [147] and self attention [163]. Given a headline and its linked article, the purpose of *joint attention* is to fuse the headline representation into the article representation in order to obtain a headline-aware representation of the article [17]. The purpose of *self attention* is to fuse the headline-aware representation of the article into itself in order to obtain the final article representation [37].

Existing works [37, 17] apply either standard dot-scale attention or cosine similarity-aided attention to align a headline and its linked article, without using any external knowledge. In contrast, our proposed neural network leverages WordNet to build knowledge graphs which are then used to guide the focus of the two attention mechanisms. We name the attention mechanisms in the proposed model *knowledge-enhanced joint-attention* and *knowledge-enhanced self-attention*, respectively.

Knowledge-Enhanced Joint-Attention

Joint attention aims to interconnect the representations of a headline and its linked article, and produces a set of headline-aware feature vectors for the words in the linked article [147]. The input to the knowledge-enhanced joint-attention layer includes the representations of a headline and its linked article from the previous layer of a model, and the output is headline-aware vector representations of the words in the linked articles.

Formally, given a set of linked articles $B \in \mathbb{R}^n$ with n words and a set of headlines $H \in \mathbb{R}^m$ with m words, let $E_B \in \mathbb{R}^{d \times n}$ and $E_H \in \mathbb{R}^{d \times m}$ denote the embeddings of the linked articles and headlines, respectively. Following [164], we compute the similarity of each headline-article word pair, rendering a similarity matrix $J \in \mathbb{R}^{n \times m}$. Then, each row of J is normalized by applying the softmax function, and a matrix $\bar{J}$ is obtained. The article-to-headline attention is then computed as $\bar{J} \cdot H^T \in \mathbb{R}^{d \times n}$. The headline-aware vector $p_i \in P$ of article $b_i \in B$ is computed for the words in b_i as follows [165, 147]:

$$p_i = \mathbf{w}[h_i; b_i; h_i \odot b_i; h_i \odot \hat{b}_i] \quad (4.1)$$

where $\mathbf{w} \in \mathbb{R}^{4d}$ is a trainable parameter; $h_i \in H$ is the headline representation; $\hat{b}_i$ represents the headline-to-article vector; $[\cdot]$ stands for a vector concatenation across rows; and $\odot$ represents element-wise multiplications.

Since contextual embeddings contain only high-level information, we believe that introducing pre-extracted human semantic knowledge into the computation of similarities will enhance the ability of the model to identify the boundaries of the clusters and is helpful to the final predictions. Therefore, we apply the pre-extracted human semantic knowledge in order to enhance context embeddings.

Next, the headline-aware body representation P is encoded by the neural network and a matrix $U \in \mathbb{R}^{d \times n}$ is obtained as coarse-grained memories for words of the linked articles. The matrix U captures the interaction between the words of an articles with respect to its linked headline and is further processed by the knowledge-enhanced self-attention mechanism.

Knowledge-Enhanced Self-Attention

Self-attention is a part of the fine-grained memory layer. Its purpose is to fuse the coarse-grained memories U into themselves. Through the knowledge-enhanced joint-attention layer, the headline-aware representation u_i^B of an article $b_i \in B$ is generated to locate important parts of the linked article. However, such a representation has very limited knowledge of the context. To overcome this problem, we directly match the headline-aware representation of article b_i against itself [166, 167]. This process can effectively pinpoint important word pairs within the representation and update the representation with this useful information. The fine-grained memories $v_i^B \in V$ of article b_i is computed as follows, where $V \in \mathbb{R}^{d \times n}$ is a matrix that represents fine-grained memories of B and H .

$$v_i^B = \text{BiGRU}(v_{i-1}^B, [u_i^B, a_i]) \quad (4.2)$$

where $a_i = \text{att}(u^B, u_i^B)$ is an attention-pooling vector of the whole article (u^{b_i}) and computed as follows:

$$a_i = \sum_{k=1}^n c_k^i u_k^B, \quad a_i \in A, \quad (4.3)$$

where

$$c_k^i = \frac{\exp(r_k^i)}{\sum_{j=1}^n \exp(r_j^i)} \quad (4.4)$$

$$r_j^i = p^\top \tanh(W_u^B u_j^B + W_u^{\hat{B}} u_i^B) \quad (4.5)$$

We apply an additional gate to $[u_i^B, a_i]$ to adaptively control the input to the neural network [166]. Thus, a matrix $V \in \mathbb{R}^{d \times n}$ is obtained as fine-grained memories.

Based on the relationships between words in the headline and its linked article, knowledge-enhanced self-attention extracts semantic information from the whole linked article.

4.3 Applying the Proposed Attention-based Neural Network to Clickbait Detection

As depicted in Fig. 4.1, our model contains four different layers to capture different linguistic representations. Following is a detailed description of the four layers.

- The first layer extracts information from headlines and their linked articles at the word level. We compute representations of words in the headlines and linked articles as follows [168]. For word embeddings, we use pre-trained word vectors GloVe [92] for both headlines and their linked articles. In addition, we use the following two types of linguistic features for each token in the linked articles: (i) named-entity recognizer (NER) encoding for 18 different types of the NER tags; (ii) part-of-speech (POS) tagging encoding for 56 different types of POS tags.
- The purpose of the second layer, a *coarse-grained memory layer*, is to apply both article-to-headline and headline-to-article attentions, based on representations of headlines and their linked articles. We generate preliminary memories over headline-article word pairs. Specifically, we apply *knowledge-enhanced joint-attention* (discussed in section 4.2.2) to fuse the body representation into the headline representation. Then

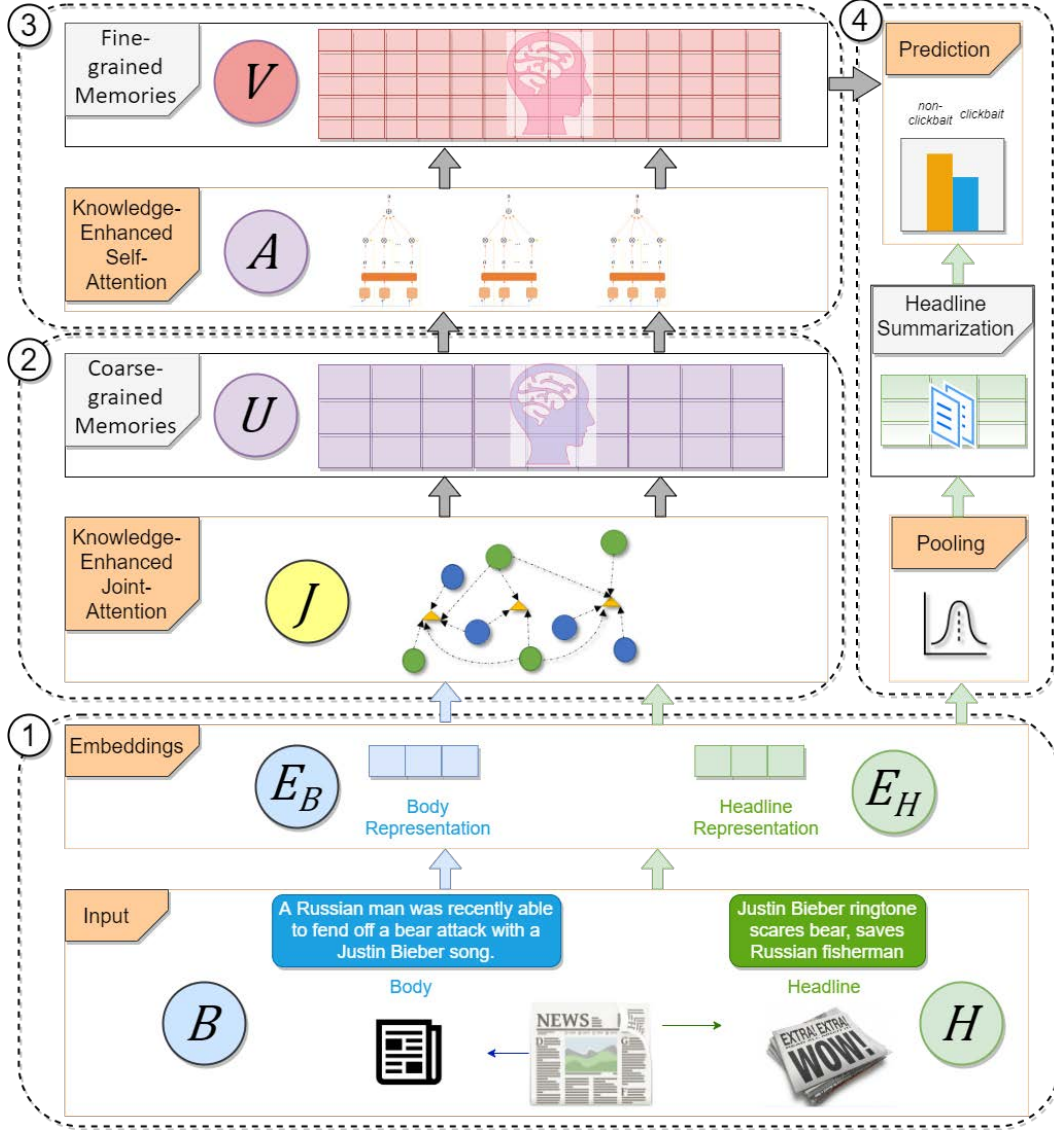


Figure 4.1: Our proposed end-to-end neural model.

we process this output with a BiGRU layer. Finally, the coarse-grained memories are obtained by concatenating the forward GRU outputs and the backward GRU outputs, which are the headline-aware body representation.

- In the third layer, a *fine-grained memory layer*, we construct the refined memories over headline-article word pairs using the coarse-grained memories. First, we apply *knowledge-enhanced self-attention* (discussed in section 4.2.2) to fuse coarse-grained

memories into themselves. Then we process this output with a BiGRU layer. Finally, the fine-grained memories are obtained by concatenating the forward GRU outputs and the backward GRU outputs, which are the final body representation.

- The last layer, a *label prediction layer*, yields the label prediction based on the fine-grained memories and the headline context embeddings. Specifically, we apply attention pooling to the headline representation to obtain a summary of the headline. Finally, we compute a posterior distribution of the two candidate labels to determine if the headline is clickbait or not.

Example and Visualization

We present an example to qualitatively demonstrate the effectiveness of the proposed knowledge-enhanced attention mechanisms. We first implemented a base model by applying full bipartite attention (for joint attention) and standard dot-scale attention (for self attention) to the model, similar to the work in [37, 17]. These attention mechanisms in the base model do not utilize human semantic knowledge. As a result, we expected that it would not perform as well as the KED model that has knowledge-enhanced attention mechanisms. We name this base model NKED (No Knowledge Enhancement). The structure of NKED is depicted in Fig. 4.2. We then replaced full bipartite attention and standard dot-scale attention with the proposed knowledge-enhanced joint-attention and knowledge-enhanced self-attention mechanisms, respectively, to obtain the proposed model KED.

Table 4.1 shows an example of a headline and its linked article, and the visualization of the output from the knowledge-enhanced attention schemes. Qualitatively, we display the contribution of the phrases to the classification in terms of knowledge-enhanced self-attention. The most important phrases are highlighted in red with the intensity of the color indicating the degree of contribution. Meanwhile, we use waves to indicate the key phrases with high attention scores in terms of knowledge-enhanced joint-attention. (Detailed visualization techniques can be found in [169].)

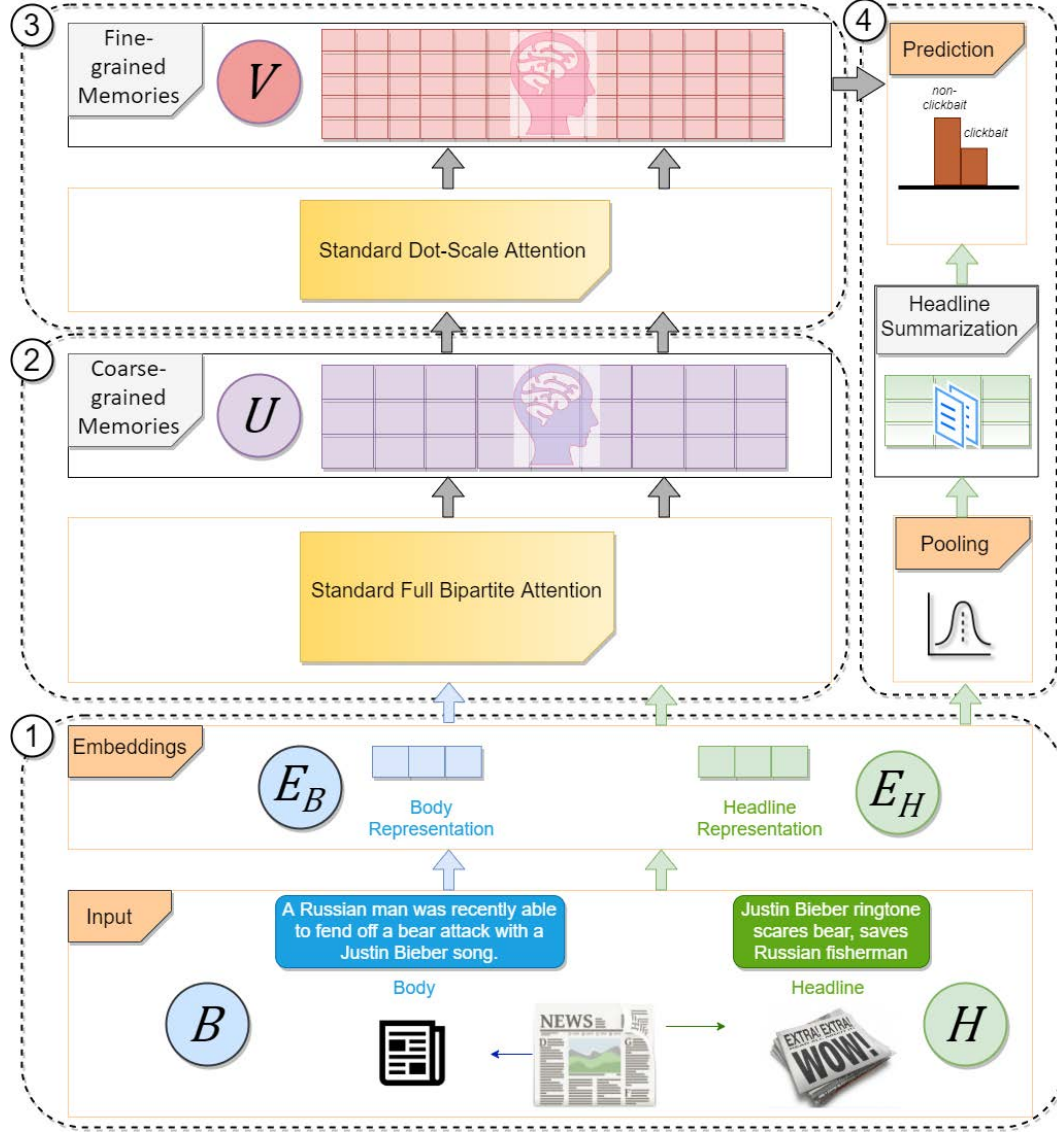


Figure 4.2: The diagram of NKED model.

As shown in Table 4.1, NKED (with no knowledge enhancement to the attention mechanisms) captures key phrases *saves* and *able to fend*, which have very low similarity. This mismatch contributes to NKED misclassifying the example as *clickbait*. In contrast, KED is able to correctly classify it as *non-clickbait*. The knowledge-enhanced joint-attention scheme first captures the global information described by the words *ringtone*, *song*, *scares*, *fend off*, *man* and *fisherman*. Informed by this global information, the knowledge-enhanced self-

Table 4.1: Visualization of a sample from the FNC Challenge dataset. The proposed KED model is implemented with knowledge enhancement to the attention mechanisms and NKED is without knowledge enhancement. The most important phrases in self-attention are highlighted in red where the intensity of the color indicates the degree of contribution. The waves indicate the key phrases with high attention scores in knowledge-enhanced joint-attention.

Model	Sentence Samples		Prediction
NKED	Headline	Justin Bieber ringtone scares bear, saves Russian fisherman	clickbait ✗
	Body	A Russian man was recently able to fend off a bear attack with a Justin Bieber song.	
KED	Headline	Justin Bieber ringtone scares bear, saves Russian fisherman	non-clickbait ✓
	Body	A Russian man was recently able to fend off a bear attack with a Justin Bieber song.	

attention scheme reduces its attention to *saves*, while capturing key phrases $\{Justin\ Bieber\ ringtone, Justin\ Bieber\ song\}$, $\{a\ Russian\ man, Russian\ fisherman\}$, and $\{scares\ bear, fend\ off\ a\ bear\}$. In this case, the model infers that the headline matches its linked article. Accordingly, the model makes a correct prediction labeled as *non-clickbait*.

In short, the global representation learned by the knowledge-enhanced joint-attention scheme provides an overall grasp of the whole text, which includes both semantic and structural information. It effectively helps the knowledge-enhanced self-attention scheme capture better instance-specific local features and improves classification performance.

In summary, the proposed neural network model is different from existing clickbait detection models in that it uses i) human semantic knowledge, which is pre-extracted using WordNet; ii) knowledge-enhanced attention mechanisms that enable the model to extract the

most meaningful headline-article word pairs. On the one hand, the coarse-grained memory layer uses human semantic knowledge to assist both article-to-headline and headline-to-article attention. On the other hand, the fine-grained memory layer uses human semantic knowledge to assist the self-attention mechanism. As a result, the proposed model performs significantly better than existing state-of-the-art models, as will be shown next.

4.4 Evaluations

We conducted extensive experiments on two real-world clickbait datasets, namely, Clickbait Challenge [31] and FNC Challenge [32], to evaluate the effectiveness of the proposed model. Our results demonstrate that the model significantly outperforms existing state-of-the-art approaches [33,34,35,36,16,37,17,38,39], even under limited training data conditions. Moreover, the proposed model achieves better or comparable performance to powerful pre-trained language models such as BERT, RoBERTa, and XLNet, while maintaining a much lighter architecture. To further support these findings, we conducted comprehensive comparative analyses, providing insights that can potentially guide future research in the field.

This section is organized as follows. Section 4.4.1 describes experiments settings. In Section 4.4.2, we validate the effectiveness of pre-trained word embeddings, knowledge enhancement to the attention mechanisms, and human semantic knowledge when applied to KED. Section 4.4.3 presents experimental results, comparing the performance of the proposed model with that of existing state-of-the-art systems and analyzing the results in detail. Section 4.4.4 verifies that when only a limited amount of training data is available (i.e., data scarcity), KED performs significantly better than the state-of-the-art clickbait detection models. Finally, Section 4.4.5 demonstrates the application of the proposed human semantic knowledge extraction scheme to several pre-trained models in the semi-supervised domain, and their significantly improved performance thanks to human semantic knowledge.

4.4.1 Experiment Settings

This section describes the datasets used in the experiments, the parameters of the proposed model, and the baseline models for comparison.

Datasets

We use two public annotated datasets for the experiments in this article.

Clickbait Challenge. This benchmark dataset [31] was released in 2017. It contains a total of 21,997 annotated samples, among which 17,598 are used for training and 4,399 for testing. Each sample is a pair of a headline and a linked article from Twitter posts. Five human annotators assigned a score to each sample, which ranges from 0 to 1, where ‘1’ denotes a post “heavily click-baiting” and ‘0’ indicates “not click-baiting” [170]. The samples with a mean score above 0.5 are considered as clickbait in this work [17].

FNC Challenge. This benchmark dataset was released by the Fake News Challenge competition in 2017 [32]. It contains 49,972 pairs of headline and linked article for training and 25,413 pairs for testing. Each pair of headline and linked article is classified into one of the four groups: ‘agree’, ‘discuss’, ‘disagree’, and ‘unrelated’. The samples with the label ‘unrelated’ are regarded as clickbait in this work [17].

Table 4.2 summarizes the statistics of the above datasets.

Table 4.2: Statistics of the datasets.

Dataset	Clickbait	Non-Clickbait	Clickbait
Clickbait Challenge	5,523	16,474	33.5%
FNC Challenge	54,894	20,491	72.8%

Model Settings

To implement the proposed neural network model, we use the Stanford CoreNLP [142] to preprocess the datasets. We apply the WordNet interface provided by NLTK [171] to extract human semantic knowledge. Additionally, we implement the proposed model using TensorFlow [159] with Python. In the stage of extraction of human semantic knowledge, we set the hyper-parameter τ to 4 and 5 for the Clickbait Challenge and FNC Challenge dataset, respectively. We limit the vocabulary size to 150,000 and 200,000, and set the maximum sequence length to 256 and 300 for the Clickbait Challenge and FNC Challenge dataset, respectively. Table 4.3 summarizes the above parameters.

Table 4.3: Model settings: vocabulary size and maximum sequence length.

Corpus	Vocabulary	Length	τ
Clickbait Challenge	150,000	256	4
FNC Challenge	200,000	300	5

Following are the parameters common to both datasets. For the dense layers and the BiGRUs, we set the dimension d to 256. With respect to the training, we use Adam [172] as our optimizer. The learning rate was set to 0.001 and the mini-batch size, to 32. To avoid overfitting, we apply dropout [173] with a value of 0.1, and apply early stopping with a patience of 3. To avoid the exploding gradient problem, we apply gradient clipping [174] with a cutoff threshold of 2. We apply exponential moving averages with a decay rate of 0.999 to optimize the performance.

The above parameters were chosen to find the optimal settings for our experiments. Each data point in the reported results was averaged over three runs.

Baseline Models

We compare the proposed KED model with the following baseline models, which can be divided into the five following categories:

- **Deep feedforward networks:** DSSM [33]
- **CNN:** CLSM [34] and CBCNN [35]
- **RNN:** LiNN [36]
- **Attention-based networks:** MSA [16], BiGRU-ATT [37] and LSDA [17]
- **Transformer-based models:** LSACD [38], Transfer Learning [39]

The above five categories are listed in the first five blocks of Table 4.4. Following are brief descriptions of the baseline models.

Deep Semantic Similarity Model (DSSM): Huang et al. [33] used deep neural networks to obtain hidden features of inputs and quantify the similarity in the space of latent representations. The authors preprocessed textual characteristics with n-gram schemes, and used the estimated similarities for prediction.

Convolutional Latent Semantic Model (CLSM): Similar to DSSM, Shen et al. [34] leveraged convolutional neural networks for extracting inherent features.

Linguistically-infused Neural Network (LiNN): Glenski et al. [36] introduced a model based on CNN and LSTM, which uses knowledge from both headlines and their linked articles. LSTM and CNN networks benefit the model by learning vectorized textual and visual knowledge individually.

BiGRU-ATT: Zhou [37] proposed an attention-based BiGRU model. The model first discovers inherent features of headlines and their linked articles. The discovered latent features are then consolidated to be fed into attention-based recurrent networks for the classification task.

Table 4.4: Performance comparison using the Clickbait Challenge and FNC Challenge datasets. The best result in each column is highlighted in green; our previously published best results in [18], from KED_R , are shown in blue; other researchers’ previously published best results are shown in red. The results in the first to fourth block (from DSSM to LSDA) come from [17]. The results in the fifth block are taken from [38, 39]. NKED: no knowledge enhancement to attention mechanisms. KED_R : with knowledge enhancement and randomly initialized word embeddings. KED: with knowledge enhancement and pre-trained GloVe word embeddings.

Model	Clickbait Challenge				FNC Challenge			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
DSSM [33]	0.817	0.655	0.661	0.658	0.747	0.894	0.740	0.811
CLSM [34]	0.833	0.683	0.643	0.662	0.756	0.959	0.762	0.853
CBCNN [35]	0.844	0.654	0.653	0.653	0.789	0.852	0.845	0.857
LiNN [36]	0.827	0.642	0.621	0.631	0.868	0.925	0.884	0.913
MSA [16]	0.826	0.699	0.474	0.565	0.859	0.920	0.877	0.907
BiGRU-ATT [37]	0.856	0.719	0.650	0.683	0.879	0.924	0.897	0.919
LSDA [17]	0.860	0.722	0.699	0.710	0.894	0.933	0.912	0.928
LSACD [38]	0.880	-	-	0.740	-	-	-	-
Transfer Learning [39]	0.858	0.713	0.668	0.690	-	-	-	-
NKED	0.862	0.724	0.704	0.714	0.892	0.930	0.911	0.920
KED_R [18]	0.880	0.754	0.756	0.755	0.913	0.948	0.935	0.941
KED	0.892	0.761	0.763	0.762	0.928	0.952	0.943	0.947

Multi-Strategy Approach (MSA): Kumar et al. [16] introduced a composite form of clickbait detection models. First, the authors applied an attention-based bidirectional RNN-based method to learn the input data. Then they incorporated the latent information with the relationship knowledge into Siamese networks for the final predictions.

Clickbait Convolutional Neural Network (CBCNN): Zheng et al. [35] proposed a convolutional model that leverages only features from the headlines to detect clickbait.

The authors first converted headlines into text vectors and then made predictions with textCNN [175].

Similarity-Aware Deep Attentive Model (LSDA): Dong et al. [17] introduced a deep learning model that is similarity-aware and attention-based to acquire and represent similarities.

Lure and Similarity for Adaptive Clickbait Detection (LSACD): Zheng et al. [38] proposed a RNN model that examines the similarity between a headline and its linked content, and the headline’s degree of enticement (“lure”) to infer whether the headline is clickbait or not.

Transfer Learning: Rajapaksha et al. [39] adapted transformer models BERT, RoBERTa and XLNet for the clickbait detection task by applying several fine-tuning methods and model configuration changes.

The last three blocks of Table 4.4 list the following models that we implemented:

- NKED: randomly initialized word embeddings; without knowledge enhancement to the attention mechanisms.
- KED_R [18]: randomly initialized word embeddings; with knowledge enhancement to the attention mechanisms.
- KED: pre-trained word embeddings; with knowledge enhancement to the attention mechanisms.

4.4.2 Validations

In this section, we validate the effectiveness of pre-trained word embeddings, knowledge enhancement to the attention mechanisms, and human semantic knowledge when applied to KED.

Effectiveness of Pre-trained Word Embeddings

We investigate the effectiveness of pre-trained word embeddings on KED. In our previous work [18], the model was implemented using randomly initialized word embeddings, which we name KED_R . For this article, we replace randomly initialized word embeddings with pre-trained GloVe vectors [92], expecting the latter will improve the classification performance of the model. We name the enhanced version KED. It is expected that GloVe can enhance the performance of KED over KED_R .

In this experiment, we use GloVe vectors with 300 dimensions to initialize the word embeddings. The results in terms of F1-score, accuracy, precision, and recall on the two datasets Clickbait Challenge and FNC Challenge are listed in Table 4.4.

Overall, the performance of the proposed architecture can be noticeably improved by using pre-trained GloVe word embeddings. For example, KED achieves a higher F1 score of 0.762, 0.7 percentage points (0.762 vs. 0.755) higher than KED_R on the Clickbait Challenge dataset, and of 0.947, 0.6 percentage points (0.947 vs. 0.941) higher on the FNC Challenge dataset. KED also performs much better than the state-of-the-art model LSDA. KED outperforms LSDA by 5.2 percentage points (0.762 vs. 0.710) and 1.9 percentage points (0.947 vs. 0.928) on the two datasets, respectively. More importantly, the performance gain is particularly significant on the relatively difficult Clickbait Challenge dataset. The experimental results confirm that pre-trained word embeddings can improve the performance of our KED model.

Effectiveness of Knowledge Enhancement to Attention Mechanisms (Ablation Study)

We conducted an ablation study by comparing the performance of KED_R (with knowledge enhancement to the attention mechanisms) to that of NKED (no knowledge enhancement). As depicted in Fig. 4.2, full bipartite attention and standard dot-scale attention are implemented in NKED. They are replaced by the proposed knowledge-enhanced joint-attention and knowledge-enhanced self-attention, respectively, in KED_R . Both KED_R and NKED use

randomly initialized word embeddings.

The results are given in Table 4.4. We observe that KED_R performs significantly better than NKED in all cases, thanks to the knowledge enhancement. In particular, on the relatively difficult task Clickbait Challenge, KED_R outperforms NKED by 5.2 percentage points (0.756 vs. 0.704) in terms of recall, and by 4.1 percentage points (0.755 vs. 0.714) in terms of F1-score. The results demonstrate that human semantic knowledge is beneficial to the clickbait detection task, and plays an important role in our neural model.

Effectiveness of Human Semantic Knowledge

To verify the effectiveness of applying human semantic knowledge to clickbait detection, we obtain ten augmented datasets by varying τ from 0 to 9, and train a different KED system on each augmented clickbait detection dataset. Table 4.5 and Figure 4.3 show that when the hyper-parameter τ is increased from 0 to 9, the amount of human semantic knowledge (represented by the average edge count per word pair) rises monotonically. The performance in terms of F1-score of KED rises until τ reaches 4 and 5 for Clickbait Challenge and FNC Challenge, respectively. The reason is that a larger τ value allows more hypernym relations to be extracted from the knowledge base as explained earlier in Section 4.2.1. As τ increases beyond a peak value (4 or 5), the performance of the model degrades. The reason is that very large values of τ increase the chances for trivial hypernym relations to be extracted from the knowledge base, leading to lower performance. Thus it can be seen that human semantic knowledge provided by the WordNet-based human semantic knowledge extraction scheme plays an essential role in the training of the proposed KED system.

4.4.3 Performance Comparison with Existing Works

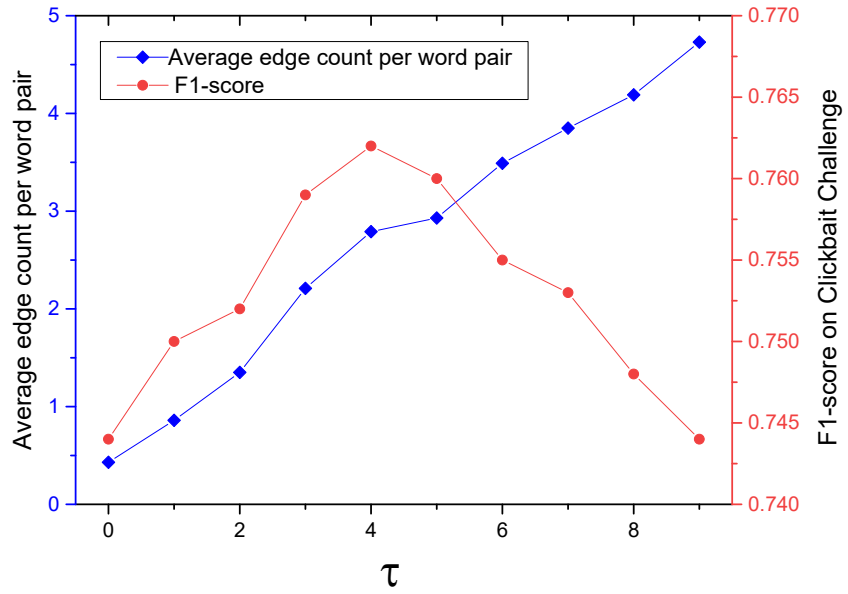
We compare the performance of the proposed KED model with that of the baseline models listed in Section 4.4.1 using the two benchmark datasets of clickbait detection. Table 4.4 summarizes the experimental results with four commonly used metrics: accuracy, recall,

Table 4.5: Effectiveness of human semantic knowledge. The hyper-parameter τ selected for the best result is marked in **bold**.

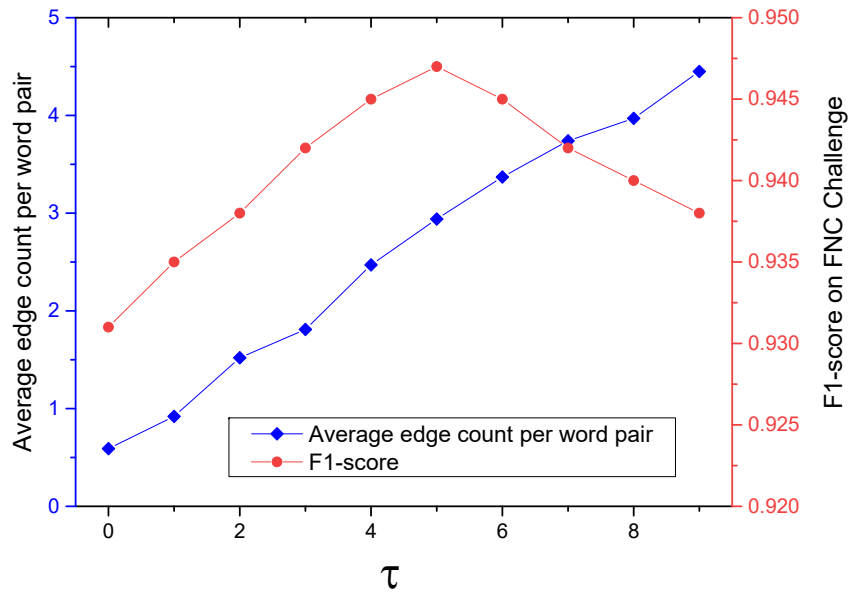
τ	Clickbait Challenge		FNC Challenge	
	Average edge count per word pair	F1-score	Average edge count per word pair	F1-score
0	0.43	0.744	0.59	0.931
1	0.86	0.75	0.92	0.935
2	1.35	0.752	1.52	0.938
3	2.21	0.759	1.81	0.942
4	2.79	0.762	2.47	0.945
5	2.93	0.760	2.94	0.947
6	3.49	0.755	3.37	0.945
7	3.85	0.753	3.74	0.942
8	4.19	0.748	3.97	0.940
9	4.73	0.744	4.45	0.938

precision, and F1-score. The experimental results show that the proposed model KED, thanks to the use of human semantic knowledge and knowledge-enhanced attention mechanisms, significantly outperforms the above state-of-the-art models for both clickbait detection benchmark datasets. For example, on the FNC Challenge dataset, KED outperforms LSDA by 3.4 percentage points in terms of accuracy (0.928 vs. 0.894) and by 3.1 percentage points in terms of recall (0.943 vs. 0.912), with LSDA having the previously published best results on this dataset.

It is worth noting that even the very basic version NKED (with randomly initialized word embeddings and without knowledge enhancement to attention mechanisms) performs better than most of the above state-of-the-art models. For example, on the relatively difficult task Clickbait Challenge, NKED outperforms the best performer of the fourth block (LSDA) by



(a) Clickbait Challenge



(b) FNC Challenge

Figure 4.3: Illustrations of the results given in Table 4.5.

0.5 percentage points (0.704 vs. 0.699) in terms of recall, and 0.04 percentage points (0.714 vs. 0.710) in terms of F1-score. The reason for the higher performance of NKED is its use of human semantic knowledge for the task of classification, while the previous models do not use human semantic knowledge.

We observe that the CNN and RNN based models in the second and third block perform better than the deep feed-forward networks in the first block. We believe the reason for this is the capability of CNN and RNN in capturing location information.

The models shown in the fourth block, MSA, BiGRU-ATT, and LSDA, perform better than those listed in the first three blocks thanks to the attention mechanisms, which can capture key phrases more effectively. The transformer-based models shown in the fifth block, LSACD and Transfer Learning, perform better than those listed in the first four blocks thanks to the transformer framework that capture key phrases even better. However, all the models underperform KED because they do not use human semantic knowledge or knowledge-enhanced attention mechanisms.

Human semantic knowledge combined with the proposed knowledge-enhanced attention mechanisms enables KED to outperform existing clickbait detection models. The following example illustrates the above advantages of KED over existing models. Consider the following two text samples:

1. “Little John was looking for his toy box. Finally, he found it. The box was in the **pen**.”
2. “The first **ballpoint** model was patented by the American leather tanner and inventor John J. Loud.”

There are no semantic correlations between ‘pen’ and ‘ballpoint’ in this context. The knowledge-enhanced attention mechanisms allow KED to assign a very low similarity score to the word pair {‘pen’, ‘ballpoint’} in this example, while previous models may (incorrectly) give it high similarity scores, leading to lower classification performance.

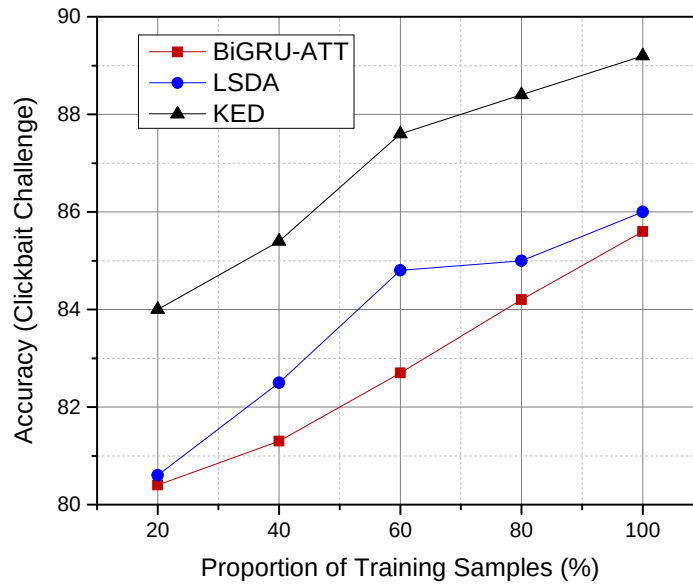
4.4.4 Quantitative Analysis of Data Scarcity

While deep learning does not require labour-intensive, time-consuming feature engineering compared to traditional machine learning techniques such as SVM, it usually requires a large amount of training data to work effectively. One of the challenges researchers in deep learning often face is the lack of training data. Therefore, it is beneficial to know how a model performs when training data is scarce. In this experiment, we compare the performance of KED with that of state-of-the-art models in the case of data scarcity.

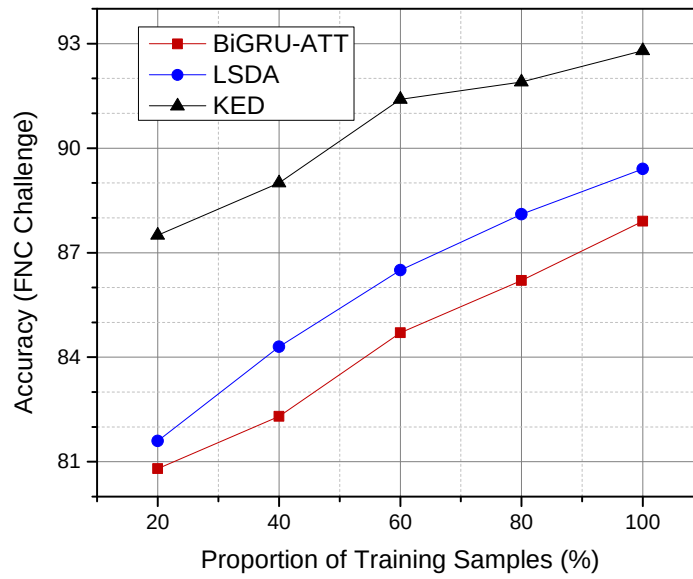
Instead of using all the training examples, we produced several subsets of training data of different sizes so as to study the relationship between the amount of available training data and the model performance. Specifically, from each of the two training datasets (Clickbait Challenge and FNC Challenge) we created four training subsets, which contain 20%, 40%, 60%, and 80% of the original training data set, respectively. We randomly selected headlines (and their linked articles) until the number of headlines reached 20% (40%, 60%, or 80%) of the size of an original data set (listed in Table 4.2). The random selections of headlines maintain the same clickbait and non-clickbait ratio of each original data set as listed in Table 4.2.

We compare KED with BiGRU-ATT and LSDA, and evaluate their performance in terms of F1-score, accuracy, precision, and recall using the subsets of training data mentioned above. We choose these two models to compare with KED because LSDA is among the previous top two performers and ranked first in most metrics; BiGRU-ATT is the first ranked model in the Clickbait Challenge competition [31]. Furthermore, BiGRU-ATT, which uses self-attention, and LSDA, which uses global similarity, are the most similar to our model.

As shown in Figures 4.4(a) through (h), as the percentage of training samples increases, the F1-score, accuracy, precision, and recall of all three models increase as expected. In all cases, KED performs much better than the other two. For instance, when 80% training samples are available, in terms of F1-score on the Clickbait Challenge dataset, KED achieves 0.738, 6.8 percentage points higher than LSDA (0.738 vs. 0.670) and 7.6 percentage points

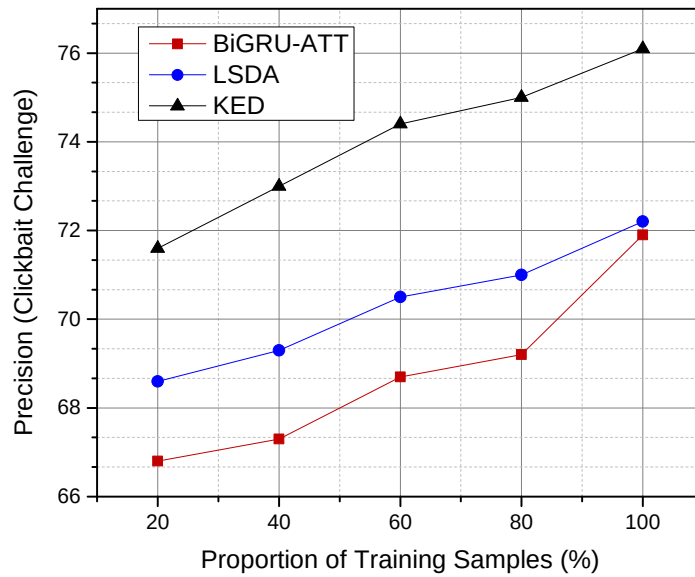


(a) Clickbait Challenge, accuracy

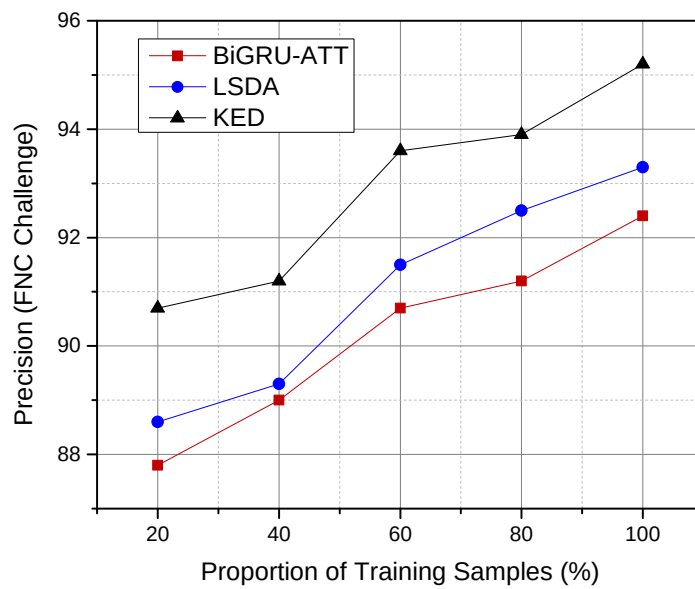


(b) FNC Challenge, accuracy

Figure 4.4: Illustrations of the quantitative analysis of data scarcity

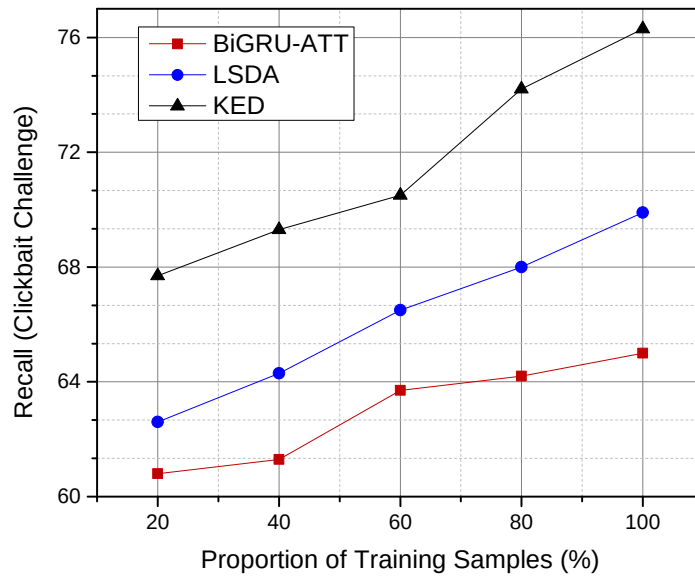


(c) Clickbait Challenge, precision

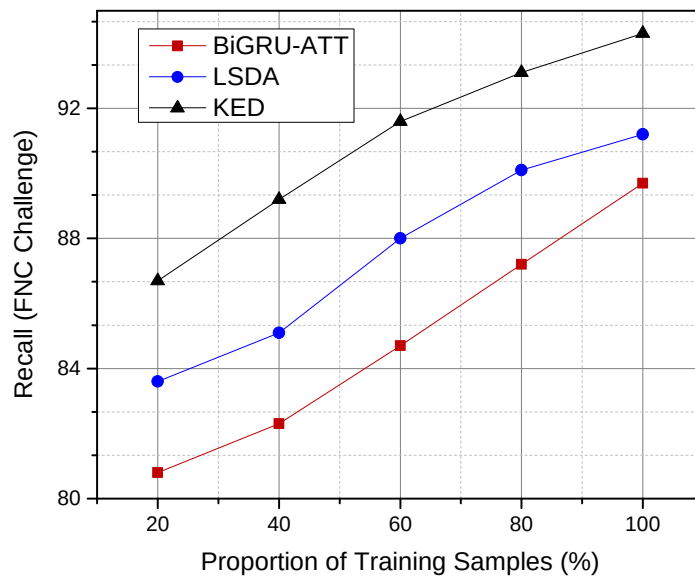


(d) FNC Challenge, precision

Figure 4.4: Continued

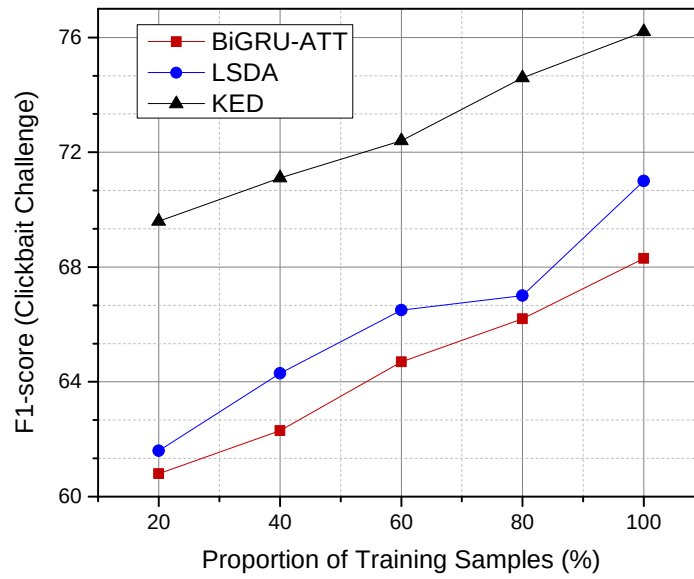


(e) Clickbait Challenge, recall

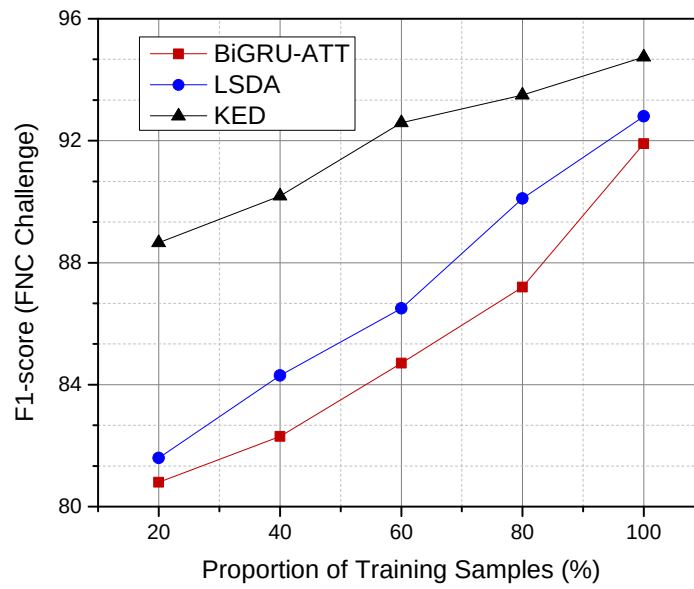


(f) FNC Challenge, recall

Figure 4.4: Continued



(g) Clickbait Challenge, F1-score



(h) FNC Challenge, F1-score

Figure 4.4: Continued

higher than BiGRU-ATT (0.738 vs. 0.662). Similarly, when 60% training samples are available, in terms of F1-score on the FNC Challenge dataset, KED achieves 0.920, 5.5 percentage points higher than LSDA (0.920 vs. 0.865) and 7.3 percentage points higher than BiGRU-ATT (0.920 vs. 0.847). In addition, when 80% training samples are available, in terms of recall on the Clickbait Challenge dataset, KED achieves 0.733, 5.3 percentage points higher than LSDA (0.733 vs. 0.680) and 9.1 percentage points higher than BiGRU-ATT (0.733 vs. 0.642). Overall, from the graphs, we can observe that our KED performs significantly better than the state-of-the-art clickbait detection models even when only a limited amount of training data is available.

4.4.5 Incorporating Human Semantic Knowledge into Pre-trained Models in the Semi-supervised Domain

Our model KED is a fully supervised model, in which all model parameters are trained from scratch. On the other hand, several transformer-based pre-trained NLP models in the semi-supervised domain such as BERT [104], RoBERTa [105], XLNet [106], ELECTRA [107] and ALBERT [108] have shown to be effective in learning common language representations and performing many NLP tasks [176]. Advantages of pre-trained NLP models over fully supervised models include quick and simple implementation of a classification model with acceptably good performance; much less labelled data required; and diverse use cases and applications. Therefore, they have been widely used in many NLP tasks [177]. We believe that human semantic knowledge can enhance the performance of pre-trained models in the semi-supervised domain. Therefore, we incorporated human semantic knowledge into BERT, RoBERTa, and XLNet, the most well-known pre-trained NLP models, and conducted experiments to verify the effectiveness of human semantic knowledge on these models.

In order to incorporate human semantic knowledge, we use the proposed WordNet-based human semantic knowledge extraction scheme to generate additional annotated data to expand the original datasets Clickbait Challenge and FNC Challenge, and then fine-tune these

pre-trained models using the newly expanded datasets. We perform grid searches over the learning rate, epochs, and batch size in $[2 \cdot 10^{-5}, 3 \cdot 10^{-5}, 5 \cdot 10^{-5}]$, $[2, 3, 4]$ and $[16, 32]$, respectively. Additionally, the AdamW optimizer [178] is applied, and the weight decay and warm-up step are set to 0.01 and 0.05, respectively. Other than the above parameters, we re-use the hyper-parameters of the original models for both the original and enhanced (with human semantic knowledge) versions [104, 105, 106]. The results from the original pre-trained models and the enhanced versions in terms of F1-score, accuracy, precision, and recall on the Clickbait Challenge and FNC Challenge datasets are listed in Table 4.6.

We note that after incorporating the proposed human semantic knowledge extraction scheme into the pre-trained models BERT, RoBERTa, and XLNet, their performance is improved noticeably. On the Clickbait Challenge dataset, the F1 scores of BERT, RoBERTa, and XLNet are improved by 2.9 percentage points (0.726 vs. 0.697), 3.1 percentage points (0.734 vs. 0.703), and 4.0 percentage points (0.763 vs. 0.723), respectively. On the FNC Challenge dataset, the F1 scores of BERT, RoBERTa, and XLNet are improved by 2.4 percentage points (0.931 vs. 0.907), 2.6 percentage points (0.937 vs. 0.911), and 3.4 percentage points (0.950 vs. 0.916), respectively.

Additionally, after being enhanced with human semantic knowledge, all the pre-trained models outperform the state-of-the-art model LSDA with both datasets and all metrics. For example, in terms of precision, BERT, RoBERTa, and XLNet outperform LSDA by 1.8 percentage points (0.740 vs. 0.722), 2.5 percentage points (0.747 vs. 0.722), and 4.0 percentage points (0.762 vs. 0.722), respectively, on the Clickbait Challenge dataset. Similarly, on the FNC Challenge dataset, BERT, RoBERTa, and XLNet outperform LSDA by 1.0 percentage points (0.943 vs. 0.933), 1.6 percentage points (0.949 vs. 0.933), and 2.2 percentage points (0.955 vs. 0.933), respectively. (LSDA uses fewer parameters than the pre-trained models, however, tens of millions vs. hundreds of millions).

We also note that the proposed model KED with pre-trained GloVe vectors outperforms the enhanced versions of BERT and RoBERTa, as shown in Table 4.6. The pre-trained model XLNet with human semantic knowledge performs slightly better than KED, e.g., 0.3

Table 4.6: Incorporation of human semantic knowledge into pre-trained models BERT, RoBERTa, and XLNet. The symbols “✓” and “✗” denote whether the proposed human semantic knowledge extraction scheme is incorporated into a model or not, respectively. The best result in each column is highlighted in orange. The second best result in each column is highlighted in green.

(a) Accuracy		
Model	Clickbait Challenge	FNC Challenge
KED	0.892	0.928
BERT (✗ ✓)	0.865 0.878	0.896 0.906
RoBERTa (✗ ✓)	0.871 0.882	0.902 0.913
XLNet (✗ ✓)	0.877 0.895	0.906 0.929
(b) Precision		
Model	Clickbait Challenge	FNC Challenge
KED	0.761	0.952
BERT (✗ ✓)	0.711 0.740	0.918 0.943
RoBERTa (✗ ✓)	0.716 0.747	0.922 0.949
XLNet (✗ ✓)	0.721 0.762	0.925 0.955

percentage points higher (0.950 vs. 0.947), in terms of F1-score on the FNC Challenge dataset. However, XLNet is much more resource consuming than KED (and so are BERT and RoBERTa). For example, the pre-trained models have more than 300 million parameters (335 million for BERT, 356 million for RoBERTa, 360 million for XLNet), whereas KED has much fewer parameters, in the range of tens of millions. In the next sub-section, we show that the training time and inference time of KED is significantly shorter than those incurred by BERT, RoBERTa and XLNet.

Table 4.6: Continued

(c) Recall		
Model	Clickbait Challenge	FNC Challenge
KED	0.763	0.943
BERT ($\times \checkmark$)	0.683 0.713	0.896 0.919
RoBERTa ($\times \checkmark$)	0.690 0.722	0.900 0.925
XLNet ($\times \checkmark$)	0.725 0.765	0.907 0.945
(d) F1-score		
Model	Clickbait Challenge	FNC Challenge
KED	0.762	0.947
BERT ($\times \checkmark$)	0.697 0.726	0.907 0.931
RoBERTa ($\times \checkmark$)	0.703 0.734	0.911 0.937
XLNet ($\times \checkmark$)	0.723 0.763	0.916 0.950

Overall, the experimental results in Table 4.6 show that the proposed human semantic knowledge extraction scheme significantly improves the performance of pre-trained models in the semi-supervised domain. Moreover, the proposed scheme is compatible with and easily applied to the pre-trained models.

4.4.6 Execution Time

As deep learning models become more complex, training and inference time is increased as more layers and more neurons per layer are added. Therefore, execution time should be considered to ensure that it is feasible to deploy a deep learning model for real life applications. In this sub-section, we report the training and inference time of the proposed KED model and the pre-trained models BERT, RoBERTa and XLNet that are adapted for

the clickbait detection task.

This set of experiments is the same as the set presented above in Section V-E. The execution time metrics are training time and inference time. All the experiments were run on a Windows workstation with the following configuration: NVIDIA GeForce 840M graphics card, Intel Core i7-4710 2.5GHz processor, 12GB DDR3 memory, and 1TB solid state drive.

Table 4.7 shows the training time of KED, BERT, RoBERTa and XLNet on the Clickbait Challenge and FNC Challenge datasets. Overall, we note that KED outperforms BERT, RoBERTa, and XLNet on both datasets. For example, on the Clickbait Challenge dataset, KED trained faster than BERT, RoBERTa, and XLNet by 47% (15m vs. 22m), 67% (15m vs. 25m), and 93% (15m vs. 29m), respectively.

Table 4.7: Training time (in minutes) on the two training datasets

	KED	BERT	RoBERTa	XLNet
Clickbait Challenge	15 min	22 min	25 min	29 min
FNC Challenge	28 min	36 min	38 min	41 min

To measure the inference time, we ran each model to classify 4,400 samples from the Clickbait Challenge dataset and 15,077 samples from the FNC Challenge dataset. The total inference time for classifying 4,400 samples from the Clickbait Challenge dataset was then divided by 4,400 to obtain the average inference time of one sample. The same calculation was done for the 15,077 samples from the FNC Challenge dataset. Table 4.8 lists the average inference time of one sample incurred by KED, BERT, RoBERTa and XLNet. We observe that the average inference time of KED is significantly shorter than that of BERT, RoBERTa, and XLNet on both datasets. For instance, on the Clickbait Challenge dataset, KED outperforms BERT, RoBERTa, and XLNet by 59% (32 vs. 51), 128% (32 vs. 73), and 175% (32 vs. 88), respectively.

It is worth noting that the classification performance of KED is only slightly lower than that of XLNet (accuracy of 0.928 vs. 0.929, and F1-score of 0.947 vs. 0.950 from 4.6),

Table 4.8: Average inference time (in milliseconds) of one sample

	KED	BERT	RoBERTa	XLNet
Clickbait Challenge	32 ms	51 ms	73 ms	88 ms
FNC Challenge	24 ms	40 ms	62 ms	75 ms

while the average inference time of KED is much shorter than that of XLNet, 2.8 to 3.1 times shorter (32 vs. 88, and 24 vs. 75, respectively, from 4.8). A similar comparison can also be said for inference time. The much shorter execution time of KED is expected because it is designed and optimized for a specific task, while the pre-trained models are intended to be adapted and used for a wide range of applications. It is also expected that transformer-based clickbait detection models such as LSACD [38], Transfer Learning [39] and LGBM [103], which adapted and fine tuned pre-trained models, would incur longer execution time than KED.

The above results demonstrate that KED is feasible to be deployed in real world scenarios for the clickbait detection task thanks to reasonably short execution time. It provides the best of both worlds: classification performance (e.g., accuracy and F1-scores) and execution time.

4.5 Chapter Summary

We propose a neural network that extracts human semantic knowledge to build linguistic knowledge graphs which guide the attention mechanisms. The model can be used for many applications, one of which being clickbait detection for the purpose of combating fake news and illegitimate marketing. To the best of our knowledge, our work is the first that incorporates human semantic knowledge into the task of clickbait detection. We carried out extensive comparative experiments to evaluate the effectiveness and performance of the proposed attention-based neural network model. Experimental results show that

- the proposed KED model significantly outperforms state-of-the-art models/systems such as BiGRU-ATT, LSDA and LSACD;
- KED performs better than the state-of-the-art models even when training data is limited;
- pre-trained word embeddings such as GloVe significantly improve the performance of KED compared to randomly initialized word embeddings;
- KED performs better or comparably to powerful pre-trained models, namely, BERT, RoBERTa, and XLNet, while being much more lightweight and incurring significantly less training time and inference time.
- the use of human semantic knowledge significantly enhances not only the performance of KED in a fully supervised domain but also that of pre-trained models in a semi-supervised domain, namely, BERT, RoBERTa, and XLNet.

The proposed neural attention-based architecture can be relatively easily adapted to a variety of other NLP tasks such as word sense disambiguation and machine reading comprehension.

Chapter 5

A Lightweight Deep Neural Model for SMS Spam Detection

5.1 Motivations and Contributions

In this chapter, we propose a novel lightweight deep neural model called Lightweight Gated Recurrent Unit (LGRU) for SMS spam detection. SMS spam remains a critical challenge in the broader context of spam detection due to its direct impact on mobile security, user privacy, and trust in communication systems. Unlike email spam, which can often be filtered at the server level, SMS spam reaches users instantly and can facilitate phishing, financial fraud, and the spread of malicious software. The problem is particularly severe in regions such as Africa, where mobile phones are the primary means of internet access and mobile money services are widely used, making individuals especially vulnerable to fraudulent SMS campaigns. Regulatory bodies worldwide have also recognized SMS spam as a pressing concern, highlighting the need for detection systems that are both accurate and efficient.

To address this challenge, we develop an LGRU model that incorporates enhanced semantic knowledge retrieved from external resources (e.g., WordNet [20]) to improve the model’s understanding of SMS text. We evaluate our model against more than 30 existing SMS

spam classifiers based on both traditional machine learning and deep learning techniques. Experimental results demonstrate that our model outperforms these approaches in terms of precision, recall, and accuracy. Moreover, it requires fewer training parameters and significantly less training time compared to state-of-the-art deep learning classifiers [29, 30], underscoring its suitability for real-world deployment in resource-constrained environments.

The contribution of our work is two-fold. First, we propose and evaluate a high-performance and resource-efficient SMS spam detection model based on LGRU. Second, the lightweight model shows a way to minimize unnecessary complexity in training recurrent models without compromising the performance, which we believe is applicable to many other complex recurrent models for other applications.

The subsequent sections of the chapter are organized as follows: Section 5.2 describes our proposed lightweight neural model based on Lightweight Gated Recurrent Unit (LGRU) for the SMS spam detection task. Section 5.3 discusses experimental results and compares the performance of our proposed models with that of existing state-of-the-art systems. Finally, Section 5.4 sums up the chapter with concluding remarks.

5.2 The Proposed Model with Lightweight gated recurrent unit (LGRU)

In this section, we describe in detail our model that is based on Lightweight Gated Recurrent Unit (LGRU).

5.2.1 Task Definition

The task of SMS spam detection, considered as an NLP classification problem using machine learning algorithms, can be defined as follows. Given a collection of M annotated text messages $m_i = \{x_1, \dots, x_n\}$, where $1 \leq i \leq M$, the task is to use sigmoid functions to

Table 5.1: Sample spam (T) and ham (F) pairs from the two datasets used in this work.

(a) SMS Spam dataset	
T	Congrats! 1 year special cinema pass for 2 is yours. call 09061209465 now! C Suprman V, Matrix3, StarWars3, etc all 4 FREE! bx420-ip4-5we. 150pm. Dont miss out!
F	I call you later, don't have network. If urgnt, sms me.
(b) ExAIS SMS Spam dataset	
T	Your recharge of NGN100.00 was successful. You have received NGN 10.00 Bonus for calls to all networks. This is valid for 7 days. To view your bonus please Dial *559*4
F	Pls snd me all the adresses of the websites you've done.

compute a posterior distribution of two candidates, namely, spam where $\Pr(c = 1|m_i)$ or ham (non-spam) where $\Pr(c = 0|m_i)$. Table 5.1 lists some examples from the dataset.

5.2.2 Lightweight Gated Recurrent Unit

GRU [179] is a recurrent structure that is simpler than but as effective as LSTM. Like LSTM, GRU relies on gating mechanisms to obtain long-term dependencies in sequential data:

$$z_t = \sigma(W_z[h_{t-1}, x_t] + b_z) \quad (5.1)$$

$$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r) \quad (5.2)$$

$$\tilde{h}_t = \phi(W_h[r_t \odot h_{t-1}, x_t] + b_h) \quad (5.3)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \quad (5.4)$$

where σ and ϕ are the sigmoid function and hyperbolic tangent, respectively; r_t and z_t denote reset and update gates, individually; W is weight matrices of the network; b is bias vector;

$\tilde{h}_t$ is the candidate output state and h_t is the actual output state at time t ; the symbol $\odot$ stands for element-wise multiplication. Different from the LSTM unit, having a memory cell and four gating units that adaptively control the information flow inside the unit, the GRU unit has only two gating units.

According to [180, 154], the forget gate is critical (and its biases b_f must be initialized to large values); the input gate is important, but the output gate is unimportant. Hence, we propose a Lightweight Gated Recurrent Unit (LGRU), which has the smallest possible number of gates in any gated unit:

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \quad (5.5)$$

$$\tilde{h}_t = \phi(W_h[f_t \cdot h_{t-1}, x_t] + b_h) \quad (5.6)$$

$$h_t = (1 - f_t) \cdot h_{t-1} + f_t \cdot \tilde{h}_t \quad (5.7)$$

where f_t denotes forget gate; w_f is weight vectors of the network; the symbol $\cdot$ stands for scalar multiplication.

In the above equations, the update gate z_t and the reset gate r_t in GRU (Eqs. 5.1-5.4) are substituted by a forget gate f_t in LGRU (Eqs. 5.5-5.7). We further simplify the LGRU by i) transforming the weight matrices of forget gates into vectors; and ii) replacing element-wise multiplications by scalar multiplications. This is inspired by [157]. Intuitively, we believe that applying fewer learnable parameters in f_t and simplifying operations in $\tilde{h}_t$ and h_t can result in lower complexity in training recurrent models without compromising the performance. As a result, LGRU is expected to be more lightweight than all other variants of GRU (including LSTM) while offering comparable performance. In fact, LGRU simplifies GRU by reducing operations from two gates to one, which lowers the GPU usage by approximately 30% in our experiments. Figure 3.3 illustrates the differences in data flows and operations between GRU and our LGRU.

5.2.3 Enhancing Semantics with External Knowledge

In the semantics layer, the word vectors are enriched semantically through external knowledge, i.e., WordNet. In this work, we exploit GloVe [92] word vectors, which is trained on the six-billion token corpora Wikipedia 2014 and Gigaword 5. The SMS text messages are mapped onto multi-dimensional space and the vectors having similar meanings are mapped close to each other. In the scope of the vocabulary, we use WordNet to obtain the synonyms of each word. Given a word w having k synonyms, whose word-level embeddings are represented as $U_\mu \in \mathbb{R}^{300 \times k}$, we perform self-attention [181] on U_μ to obtain the synonymy embedding of w :

$$\mu_w = U_\mu \text{softmax}(\tanh(W_\mu U_\mu)^\top v_\mu) \in \mathbb{R}^{300} \quad (5.8)$$

where W_μ and v_μ are learnable parameters. By applying synonymy embedding, we improve the vector-space similarity between synonyms and thus promote the effects of the semantic level inter-word connections.

Moreover, to better deal with unknown or out-of-vocabulary (OOV) words in SMS text messages, we obtain the character-level embedding of each word using CNN. Then, we concatenate four representations: word-level representation W^E , character-level representation C^E , synonym representation S^E , and feed them into a highway network [182] with SELU [183] activation (denoted as σ) on the output to get a hybrid semantic representation:

$$S^E = \sigma(\text{Highway}([W^E; C^E; S^E])) \quad (5.9)$$

5.2.4 The Proposed Lightweight Model

Based on the above design of LGRU, we can say that it is “lighter” than all other variants of GRU, including LSTM. Therefore we can substitute certain LSTMs in SSCL [30] by our LGRUs. In particular, the semantics layer and the CNN layer are maintained, while the LSTMs at the LSTM layer are replaced by LGRUs. The outcome of the above substitutions is a model “lighter” than SSCL.

5.3 Experimental Results

To evaluate the effectiveness and efficiency of our proposed deep neural model, we conduct experiments on two real-world benchmark datasets [23, 184], and compare its performance with that of state-of-the-art SMS spam classifiers [23, 26, 29, 30].

5.3.1 Datasets

We use two public annotated datasets for the experiments in this article.

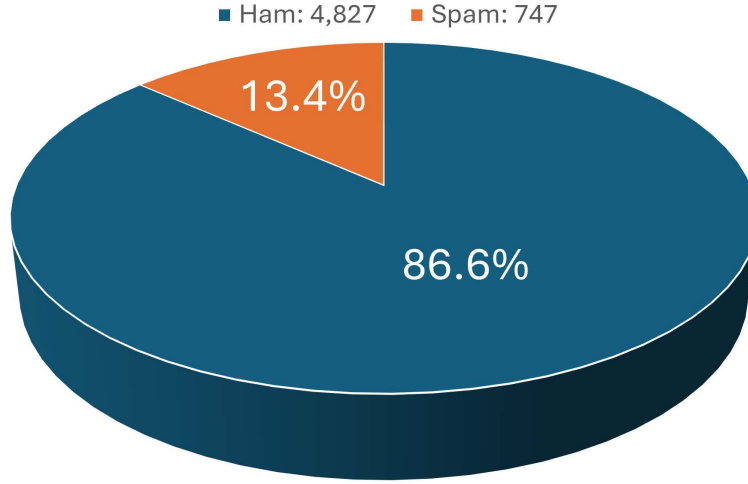
SMS Spam dataset. The SMS Spam dataset [23] is a public annotated benchmark dataset [185], and consists of a total of 5,574 English, and non-encoded SMS text messages, which is a mix of 4,827 ham (legitimate) and 747 spam messages. The class-wise SMS distribution is shown in Figure 5.1(a).

Table 5.2(a) presents the statistics related to the tokens extracted from the corpus. Note that, the dataset has a total of 81,175 tokens and SMS spam has, on average, ten tokens more than legitimate messages.

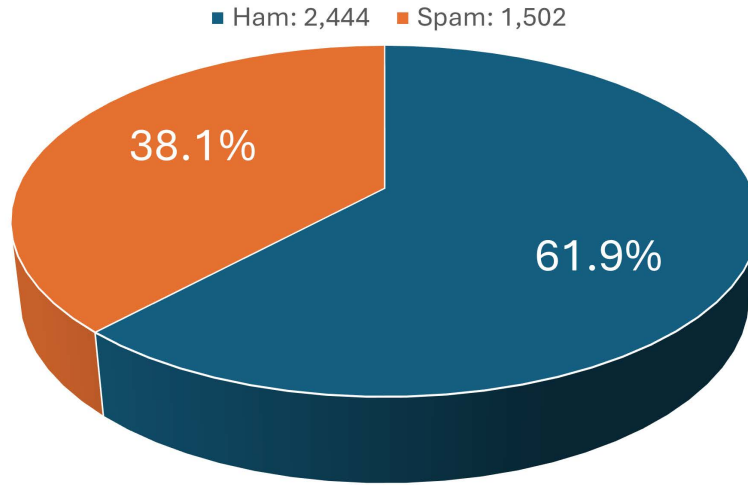
ExAIS SMS Spam dataset. The ExAIS SMS Spam dataset [184] was developed as part of a research project at the Federal University of Agriculture, Abeokuta, Nigeria, with the objective of constructing an indigenous SMS spam corpus that reflects the African-English linguistic context. Data collection involved 20 users from diverse demographics within the university, including students, faculty members, and non-teaching staff, aged between 20 and 50 years.

The authors [184] report that the original published dataset contains 5,240 SMS messages, including 2,350 spam messages and 2,890 ham (non-spam) messages. We further processed the dataset by removing low-quality SMS messages, such as duplicates and those without labels. The final retained dataset after processing consists of 3,946 samples, including 2,444 hams and 1,502 spams. The class-wise SMS distribution is shown in Figure 5.1(b).

Table 5.2(b) summarizes the token statistics for the ExAIS SMS Spam dataset. The



(a) The SMS Spam dataset [23]



(b) The ExAIS SMS Spam dataset [184]

Figure 5.1: Class-wise distribution of the two SMS spam datasets used.

dataset comprises a total of 78,397 tokens, with spam messages exhibiting a higher average token count than legitimate messages. On average, each message contains 19.9 tokens, with spam messages containing 3.6 more tokens than hams.

Token distribution differences, such as message length, play a crucial role in spam detection. Empirical studies consistently show that spam messages tend to be longer than legiti-

Table 5.2: Token statistics of the dataset used.

(a) SMS Spam dataset	
Number of tokens in ham samples	63,632
Number of tokens in spam samples	17,543
Total number of tokens	81,175
Average number of tokens per message	14.6
Average number of tokens per ham message	13.2
Average number of tokens per spam message	23.5
(b) ExAIS SMS Spam dataset	
Number of tokens in ham samples	45,218
Number of tokens in spam samples	33,179
Total number of tokens	78,397
Average number of tokens per message	19.9
Average number of tokens per ham message	18.5
Average number of tokens per spam message	22.1

mate ones, as spammers often include multiple keywords, promotional phrases, or malicious links to increase the chance of user engagement. In contrast, ham (legitimate) SMS messages are usually short and concise, reflecting natural conversational patterns. These structural differences affect not only the statistical distribution of tokens but also the semantic load of each message. Models that are sensitive to such distributional cues can therefore better distinguish between spam and ham. Ignoring token distribution risks overlooking an informative signal, since message length and token density are strongly correlated with the intent

and purpose of the message. Thus, analyzing token distribution is essential for improving classification performance in SMS spam detection.

5.3.2 Baseline Models

We compare the proposed lightweight model with the following baseline models, namely, SLSTM [29], SSCL [30], Naïve Bayes, Multinomial NB, Decision Tree, SVM, Random Forest, XGBoost, and BiLSTM.

SLSTM. Jain et al. [29] introduced an enhancement of the traditional LSTM network, specifically designed to improve spam detection by incorporating semantic information into word representations before feeding them into the LSTM. The key innovation lies in enriching word vectors semantically through a preprocessing layer, which aims to capture deeper contextual meaning.

SSCL. Jain et al. [30] extend their work [29] on SMS spam detection by proposing a hybrid deep learning model that combines convolutional neural networks (CNNs) and long short-term memory (LSTM) networks. This model leverages the complementary strengths of both architectures: CNNs are employed to extract local and global features from the input text, while LSTMs are used to capture long-range dependencies and sequential patterns, enabling more effective spam detection.

Naïve Bayes. A probabilistic classifier based on Bayes’ theorem with a strong assumption of feature independence. It is a simple and efficient method for text classification tasks.

Multinomial NB. Multinomial NB is a variant of Naïve Bayes algorithm. It is specifically designed for discrete count data, such as word frequencies in text, making it well-suited for document classification problems.

Decision Tree. A non-parametric model that makes predictions by recursively partitioning the feature space into regions based on feature values. It is interpretable and can capture non-linear relationships.

Table 5.3: Confusion matrix used for this work.

		Predicted Class	
		Spam	Ham
Ground Truth	Spam	True Positive (TP)	False Negative (FN)
	Ham	False Positive (FP)	True Negative (TN)

SVM. SVM [186] is a powerful classifier that finds the optimal hyperplane to separate data into classes with the largest margin. It is effective in high-dimensional spaces and robust to overfitting.

Random Forest. Random Forest [128] is an ensemble of decision trees that aggregates the predictions of multiple trees to improve generalization. It reduces overfitting and increases accuracy compared to individual trees.

XGBoost. XGBoost [187] is an advanced gradient boosting algorithm that builds an ensemble of trees in a sequential manner, optimizing performance through regularization and efficient computation. It is known for its high accuracy and scalability.

BiLSTM. BiLSTM [188] is a type of recurrent neural network that captures dependencies in sequential data from both past and future contexts. It is effective for capturing contextual dependencies in sequential data from both backward and forward directions.

5.3.3 Evaluation Metrics

The confusion matrix used for this work is as shown in Table 5.3. We further use the following standard evaluation metrics to compare the performance of different models:

- Accuracy: the number of correct predictions made divided by the total number of predictions made, i.e., $\frac{TP+TN}{TP+FP+FN+TN}$.
- Precision: the number of correct positive predictions divided by the total number of

positive class values predicted, i.e., $\frac{TP}{TP+FP}$.

- Recall (aka Spam Caught (SC%)): the number of correct positive predictions divided by the number of positive class values in the test data, i.e., $\frac{TP}{TP+FN}$.
- Specificity: the number of correct negative predictions divided by the number of negative class values in the test data, i.e., $\frac{TN}{FP+TN}$.
- Blocked Ham Percentage (BH%): the number of incorrect positive predictions divided by the number of negative class values in the test data, i.e., $\frac{FP}{FP+TN}$ or $1 - \text{Specificity}$.
- F1-score: defined as $2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$, which conveys the balance between the precision and the recall.

In addition to the above metrics, we also compare the efficiency of our LGRU model against two state-of-the-art deep neural models, SLSTM [29] and SSCL [30], using the SMS Spam dataset. To compare the efficiency, we use the following metrics: number of hidden recurrent units, number of trained parameters, and training time (expressed as seconds/epoch).

5.3.4 Implementation Details

We implement several baseline models, including both traditional machine learning (i.e., Naïve Bayes, Multinomial NB, Decision Tree, SVM, Random Forest, and XGBoost [187]) and deep learning (BiLSTM) approaches, using the scikit-learn toolkit [189]. Our lightweight model, as well as the baseline SLSTM [29] and SSCL [30] models, are implemented using PyTorch [190] with Python. To ensure a fair comparison of computational efficiency, all the models are trained on the same hardware platform, consisting of an Intel Core i7-6800K CPU and an NVIDIA GeForce GTX 1080 GPU. For data preprocessing, we tokenize the SMS Spam dataset and ExAIS SMS Spam dataset with spaCy, and utilize WordNet 3.0 via the NLTK toolkit [171] for lexical manipulation.

For the following experiments, we split the dataset into a training set (80%) and a test set (20%). Then we further randomly sample a validation set from the training set with 10% size of it, in order to optimize the hyper-parameters and choose the best epoch. Based on the design of the experiments, we tested several sets of parameters to select one that gives the experiments the best performance. We employ a learning rate of 0.001, batch size of 64, *Stochastic Gradient Descent* and the *Adam* optimizer. *Dropout* is adopted during training, initially set to 0.5, slowly decreasing during training until it reaches 0.1 at the end. For both word and character embeddings, we initialize them with the 300-dimensional pre-trained GloVe vectors and update them during the training. In addition to the above hyper-parameters, we follow all the other experiment settings of the model with the best performance from [29, 30].

5.3.5 Results and Discussions

We conducted two experiments to evaluate the effectiveness of the proposed lightweight model:

- Experiment 1: We evaluate the performance and efficiency of the proposed LGRU model against state-of-the-art SMS spam detection classifiers on the SMS Spam dataset to assess its effectiveness in comparison to existing approaches.
- Experiment 2: We evaluate the performance of the proposed LGRU model against popular traditional machine-learning and deep-learning-based state-of-the-art SMS spam classifiers on the ExAIS SMS Spam dataset to assess its effectiveness in comparison to existing methods.

All results are averaged over 5-fold cross-validation.

Experiment 1: Performance and Efficiency Analysis on the SMS Spam Dataset

We compare the performance of the proposed LGRU model with state-of-the-art SMS spam detection classifiers based on traditional machine learning and deep learning approaches. The SMS spam detection classifiers based on traditional machine learning are listed in the first and second blocks of Table 5.4, while those based on deep learning are presented in the third and fourth blocks.

Table 5.4 lists the performance of our model in terms of precision, recall, accuracy, percentage of blocked ham and F1-score. The performance of existing classifiers is reported in and imported from [23], [26] and [30]. The performance of our LGRU model was obtained from the experiment/implementation described above. All results in this paper, [23], [26] and [30] are obtained using the same dataset described earlier.

The results in Table 5.4 demonstrate that our LGRU model outperforms over 30 traditional and deep learning classifiers across key metrics, including precision, recall, and accuracy. It achieves an F1-score of 99.29%, matching the best performance reported by SSCL [30]. Notably, our model attains the highest accuracy of 99.28% and highest precision of 99.30% among all compared methods and is one of the few that maintain a BH% below one percent (at 0.62%), confirming its robustness and reliability in spam detection.

Table 5.4: Performance comparison with state-of-the-art algorithms on the SMS Spam dataset: traditional machine-learning-based (*traditional*) and deep-learning-based (*deep*). **Bold** font indicates best systems. Results in the first and third blocks come from [23], [26] respectively, and results in the second and last blocks come from [30].

	Classifier	Precision $\uparrow$	Recall (SC%) $\uparrow$	Accuracy $\uparrow$	BH% $\downarrow$	F1-score $\uparrow$
traditional	[23] TR	-	0	86.95	0	-
	[23] EM + tok2	-	17.09	85.54	4.18	-
	[23] 3NN + tok2	-	23.77	90.1	0	-
	[23] Boolean NB + tok1	-	98.04	77.13	26.01	-
	[23] Flex NB + tok1	-	47.35	90.72	2.77	-
	[23] Gauss NB + tok1	-	47.54	91.95	1.39	-
	[23] Basic NB + tok1	-	48.53	92.05	1.42	-
	[23] 1NN + tok2	-	43.81	92.7	0	-
	[23] MN Bool NB + tok1	-	51.87	93.72	0	-
	[23] MN TF NB + tok1	-	52.06	93.74	0	-
	[23] Bern NB + tok1	-	54.03	94	0	-
	[23] C4.5 + tok2	-	75.25	95	2.08	-
	[23] Random Forest + tok2	-	65.23	95.36	0.12	-
	[23] PART + tok2	-	78	95.87	1.45	-
	[23] MDL + tok1	-	75.44	96.26	0.35	-
	[23] Boosted C4.5 + tok2	-	81.53	97.05	0.62	-
	[23] Boosted NB + tok2	-	84.48	97.5	0.53	-
	[23] SMO + tok2	-	82.91	97.5	0.29	-
	[23] Logistic Reg. + tok2	-	95.48	97.59	2.09	-
	[23] SVM + tok1	-	83.1	97.64	0.18	-
	[30] KNN	91.37	90.4	90.4	-	88.13
	[30] NB	97.64	97.67	97.67	-	97.65
	[30] Random Forest	97.88	97.85	97.85	-	97.77
	[30] ANN	97.41	97.4	97.4	-	97.4
	[30] SVM	97.45	97.49	97.49	-	97.44

Table 5.4: Continued

	Classifier	Precision $\uparrow$	Recall (SC%) $\uparrow$	Accuracy $\uparrow$	BH% $\downarrow$	F1-score $\uparrow$
deep	[26] TM+SDA	-	85.59	97.51	0.62	-
	[30] CNN	97.67	99.79	97.93	-	98.82
	[30] LSTM	98.56	99.48	98.29	-	99.02
	[30] SCNN	98.78	99.39	98.65	-	99.07
	[29] SLSTM	98.74	99.35	99.01	-	99.24
	[30] SSCL	98.86	99.77	99.01	-	99.29
	Ours	99.30	99.28	99.28	0.62	99.29

Table 5.5 reports the class-wise and overall classification performance of our model on the SMS Spam dataset, evaluated using three standard metrics: precision, recall (also referred to here as the spam capture rate), and F1-score. For each class, “Ham” (non-spam) and “Spam,” we present the individual scores, while the overall performance is computed as a weighted average, accounting for class imbalance by giving proportionally more weight to the majority class. Precision reflects the proportion of predicted positives that are correct (for example, how many of the messages flagged as spam are truly spam), while recall captures the model’s ability to correctly identify all true spam messages. The F1-score provides a balanced measure by combining precision and recall into a single harmonic mean. Our model achieves an overall weighted F1-score of 99.29%, with near perfect performance across both classes. In particular, the spam class, despite being the minority, attains a high recall of 98.66%, demonstrating the model’s strong ability to identify rare but critical spam instances without significant compromise in precision.

In Table 5.6, we compare the efficiency of the proposed LGRU model with the state-of-the-art classifiers SLSTM [29] and SSCL [30]. The original models were reproduced by following their experimental setup, using the same number of recurrent hidden units as reported in the table. The results show that our model requires 40%-45% fewer learnable

Table 5.5: Class-wise and overall performance, in terms of precision, recall (spam capture rate), and F1-score on the SMS Spam dataset. **Bold** font highlights the best values across each category. The overall metrics are reported as weighted averages across the two classes: “Ham” (non-spam) and “Spam”.

Classifier	Precision $\uparrow$	Recall (SC%) $\uparrow$	F1-score $\uparrow$
Overall Weighted Average	99.30	99.28	99.29
Majority (“Ham”)	99.79	99.38	99.58
Minority (“Spam”)	96.08	98.66	97.35

parameters and incurs approximately 50% less training time. At the same time, it achieves higher accuracy and precision than the other two classifiers, and matches the best F1-score of 99.29% reported by SSCL [30], as shown in Table 5.4. In short, the proposed model significantly improve the training efficiency without compromising the performance. Additionally, we evaluate the efficiency of our proposed model by measuring its inference time. On average, the model requires only 39 ms to process a single SMS message. This low latency demonstrates that the system is capable of operating in real-time settings, such as live filtering of incoming text messages or large-scale spam monitoring. The fast inference speed further reinforces the practicality of our approach, ensuring that accuracy is achieved without sacrificing responsiveness.

Experiment 2: Performance Analysis on the ExAIS SMS Spam Dataset

In the second experiment, we evaluate the performance of the proposed LGRU model against widely used traditional machine learning and state-of-the-art deep learning SMS spam classifiers, namely, Naïve Bayes, Multinomial NB, Decision Tree, SVM, Random Forest, XGBoost, and BiLSTM, on the ExAIS SMS Spam dataset. These baseline classifiers were selected due to their widespread use in SMS spam detection research, as evidenced in prior studies [30, 23].

We use accuracy and weighted average F1-score as the main evaluation metrics. However,

Table 5.6: Overheads of existing SMS spam detection models and the proposed model evaluated on the SMS Spam dataset. Number of recurrent hidden units (Column 2) and performance accuracy (Column 5) are reported from [29, 30], while the number of parameters and training time per epoch are obtained by reproducing the original models based on their experimental setup.

Classifier	Recurrent Hidden Units	Parameters↓	Training Time (sec / epoch)↓	Performance (accuracy)↑
SLSTM [29]	200	5,115,415	1635	99.01
SSCL [30]	100	4,723,902	1427	99.01
Ours	100	2,857,483	872	99.04

in the presence of class imbalance, a model may achieve high accuracy by primarily predicting the majority class. Weighted average F1-score is also influenced by class distribution and may favor the majority class. To address this, we report precision, recall, and F1-score for the majority and minority classes separately.

The results presented in Table 5.7 highlight the effectiveness of the proposed model compared to several widely used traditional machine learning and state-of-the-art deep learning algorithms, evaluated on the ExAIS SMS Spam dataset. The key observations are as follows:

- **The proposed model significantly outperforms all baseline classifiers in terms of both overall accuracy and weighted F1-score, as shown in Table 5.7(a).** Traditional machine learning methods, including Naïve Bayes, Multinomial NB, Decision Tree, SVM, Random Forest, and XGBoost, achieved accuracies between 0.763 and 0.848, with corresponding F1-scores ranging from 0.766 to 0.847. The deep learning model BiLSTM performed notably well, achieving an accuracy of 0.905 and a weighted F1-score of 0.904. However, our model surpasses all competitors, achieving the highest accuracy and F1-score of 0.943. These results highlight the strength of our lightweight LGRU-based architecture, which integrates external seman-

Table 5.7: Performance comparison with popular traditional machine-learning and deep-learning-based state-of-the-art algorithms on the ExAIS SMS Spam dataset. **Bold** font indicates the best-performing systems.

(a) Overall classification accuracy and weighted F1-scores

Classifier	Accuracy	Weighted F1
Naïve Bayes	0.763	0.766
Multinomial NB	0.839	0.837
Decision Tree	0.832	0.832
SVM	0.846	0.843
Random Forest	0.848	0.847
XGBoost	0.847	0.845
BiLSTM	0.905	0.904
Ours	0.943	0.943

(b) Precision, recall and F1-scores for the minority class (“Spam”)

Classifier	Precision	Recall	F1
Naïve Bayes	0.661	0.835	0.738
Multinomial NB	0.838	0.740	0.786
Decision Tree	0.790	0.787	0.789
SVM	0.867	0.724	0.789
Random Forest	0.835	0.771	0.802
XGBoost	0.854	0.743	0.795
BiLSTM	0.929	0.825	0.874
Ours	0.956	0.898	0.926

Table 5.7: Continued

(c) Precision, recall and F1-scores for the majority class (“Ham”)

Classifier	Precision	Recall	F1
Naïve Bayes	0.867	0.716	0.784
Multinomial NB	0.840	0.905	0.871
Decision Tree	0.859	0.861	0.860
SVM	0.835	0.926	0.878
Random Forest	0.856	0.899	0.877
XGBoost	0.843	0.916	0.878
BiLSTM	0.892	0.958	0.924
Ours	0.935	0.973	0.954

tic knowledge to improve SMS spam classification. Importantly, this model achieves superior performance while maintaining lower computational complexity, indicating its potential for deployment in resource-constrained environments and its applicability as an efficient alternative to more complex recurrent architectures.

- **The performance of the minority class (“Spam”) improved significantly with our proposed model, as shown in Table 5.7(b).** Among the baseline models, Naïve Bayes achieved a precision of 0.661, a recall of 0.835, and an F1-score of 0.738, while Multinomial NB demonstrated higher precision at 0.838 but lower recall at 0.740, resulting in an F1-score of 0.786. Traditional models such as Decision Tree, Random Forest, and XGBoost attained F1-scores ranging from 0.789 to 0.802. The SVM model exhibited the highest precision among traditional approaches at 0.867 but was limited by a relatively low recall of 0.724, yielding an F1-score of 0.789. BiLSTM stood out among the baselines with an F1-score of 0.874, driven by a high precision of 0.929 and a recall of 0.825. Notably, our model achieved even stronger results, maintaining both

high precision at 0.956 and the highest recall at 0.898, leading to the best overall F1-score of 0.926. These results highlight the effectiveness of our approach in enhancing the detection of the minority “Spam” class while preserving a strong balance between precision and recall.

- **The performance of the majority class “Ham” also exhibited consistent improvements with the proposed model, as shown in Table 5.7(c).** Among the baseline classifiers, Naïve Bayes yielded the lowest recall at 0.716, which led to an F1-score of 0.784 despite a reasonable precision of 0.867. Multinomial NB and Decision Tree performed comparably, achieving F1-scores of 0.871 and 0.860, respectively. SVM, Random Forest, and XGBoost all showed strong recall values above 0.89 and F1-scores around 0.878. BiLSTM outperformed these traditional models with a notably higher F1-score of 0.924, driven by its high recall of 0.958 and a precision of 0.892. The proposed model achieved the best overall results, with the highest precision of 0.935, the highest recall of 0.973, and the highest F1-score of 0.954. These results demonstrate that the proposed model not only excels at correctly identifying the majority class “Ham”, but also delivers significantly more balanced and robust performance than existing methods.

Table 5.8 presents a performance comparison between the proposed SMS spam detection model and a state-of-the-art BiLSTM-based model by Abayomi et al. [191] on the same ExAIS SMS dataset. The results for Abayomi et al. [191] are taken from their paper [191]. Our experimental results show that, compared with [191], the proposed model achieves higher accuracy (0.943 vs. 0.934) and F1-score (0.943 vs. 0.942) and a more balanced performance between precision and recall. These results highlight the effectiveness of the proposed model for SMS spam detection.

Table 5.8: Performance comparison with the existing state-of-the-art SMS spam detection model evaluated on the ExAIS SMS dataset. **Bold** font indicates the best-performing systems.

Classifier	Accuracy	Precision	Recall	F1-score
Abayomi et al. [191]	0.934	0.969	0.917	0.942
Ours	0.943	0.944	0.943	0.943

5.4 Chapter Summary

In this chapter, we propose a novel lightweight deep neural model named Lightweight Gated Recurrent Unit (LGRU). Instead of relying only on static knowledge implicitly present in the training data, enhancing semantics is retrieved from external knowledge sources (e.g., WordNet) to assist in understanding SMS text inputs. Our model significantly improves the training efficiency without compromising the performance. Experimental results show that our model outperforms more than 30 state-of-the-art SMS spam classifiers. The significance of this work lies in the fact that it reveals a way to minimize unnecessary complexity in training recurrent models without compromising the performance, which we believe is applicable to many other complex recurrent models.

Chapter 6

Conclusion and Future Research Directions

6.1 Conclusion

Deep learning (DL) has emerged as a transformative technology with significant applications in cybersecurity and online content integrity. This dissertation studies the effectiveness and efficiency of deep learning methods for detecting Twitter bots, SMS spam, and clickbait content. By demonstrating how DL can enhance automated detection of such malicious or deceptive behaviors, this work contributes to the development of robust and intelligent security solutions that protect users and digital platforms from evolving threats. Following are summaries of our contributions.

6.1.1 Twitter Bot Detection Using Neural Networks and Linguistic Embeddings

Our first contribution is a novel RNN-based model, BiLGRU, for distinguishing Twitter bots from human accounts. By connecting two hidden layers in opposite directions, BiL-

GRU captures contextual information from both past and future states, enabling effective feature extraction across tweets. Unlike approaches relying on handcrafted features, our method uses multiple linguistic embeddings including word, character, part-of-speech, and named-entity, allowing for faster and less labor-intensive deployment. Experiments on real-world datasets show that our model, BOTLE, achieves competitive or superior detection performance compared to state-of-the-art methods, without requiring prior user profile or network information. We also validate the impact of different embeddings and recurrent variants, providing insights for future research. Additionally, we report the model’s training time, inference speed, and memory usage, demonstrating its practicality for real-world deployment.

6.1.2 An Attention-Based Neural Network Using Human Semantic Knowledge and Its Application to Clickbait Detection

Our second contribution is an attention-based neural network named Knowledge-Enhanced Clickbait Detector (KED), which leverages linguistic knowledge graphs from WordNet to guide its attention mechanisms. KED captures both local and global discriminative features through a knowledge-enhanced joint-attention mechanism for global similarities and a knowledge-enhanced self-attention mechanism to filter out noise and select relevant features. By incorporating human semantic knowledge, KED better models semantic correlations between headline and article word pairs. Experiments on two real-world clickbait datasets show that KED significantly outperforms state-of-the-art models and achieves competitive or better performance than large pre-trained models like BERT, RoBERTa, and XLNet, while remaining lightweight. We also provide detailed analyses to support future research in this area.

6.1.3 A Lightweight Deep Neural Model for SMS Spam Detection

Our third contribution is a lightweight deep neural model named Lightweight Gated Recurrent Unit (LGRU) for SMS spam detection. To improve understanding of SMS text, we also incorporate semantic information retrieved from external knowledge sources. We evaluated LGRU against more than 30 existing SMS spam classifiers based on traditional machine learning and deep learning approaches. Results show that LGRU achieves higher precision, recall, and accuracy while requiring fewer training parameters and less training time than state-of-the-art deep learning models. Our work contributes by proposing a high-performance, resource-efficient SMS spam detection model based on LGRU and demonstrating how to reduce training complexity in recurrent models without sacrificing accuracy, with potential applications beyond spam detection.

6.2 Future Research Directions

In this dissertation, we tackled several significant cybersecurity challenges through the application of deep learning methods. In recent years, the swift progress in large language models (LLMs) has unveiled new opportunities across diverse domains, including cybersecurity, which continually contends with a dynamic threat environment and the necessity for cutting-edge technologies. There are still numerous aspects and research topics that require further exploration. We intend to investigate the following open research problems:

1. How to detect AI-generated content: The release of ChatGPT, a generative AI chatbot developed by OpenAI, in November 2022, had a profound impact on both cyberspace and the real world. The increase of generative AI chatbots has drawn the attention of malicious actors, who may misuse these sophisticated tools for disreputable activities. This has led to the creation of AI content generation tools capable of producing fake news, phishing attempts, spam, and bot-generated content. Distinguishing between AI-generated and human-written texts has thus become a significant challenge. Therefore,

the consideration of developing AI-generated content detector can be subjects of future work.

2. How to detect voice impersonator and deepfakes: The widespread availability of audio-visual content on social media, along with open-source LLM models, advanced tools like PyTorch, cost-effective computing infrastructure, and the rapid advancement of deep learning (DL) techniques, has enabled the creation of deepfakes for purposes such as financial fraud, disinformation, hoaxes, and the disruption of government operations. Current audio and visual deepfake datasets exhibit biases and imbalances across different races and genders, which extends to the detection methods employed. Therefore, it is imperative to develop approaches that enhance data quality and ensure fairness in detection algorithms.
3. How to detect AI-generated malware and malicious code: The powerful and versatile ChatGPT can be used for a wide range of tasks, such as developing original software programs, including malware. This capability introduces the potential for a new and dangerous breed of cyber threats. An AI-generated malware dubbed BlackMamba was able to bypass cybersecurity technologies such as industry-leading endpoint detection and response (EDR) in an experimental project. Thus, there is an urgent need to investigate effective strategies for detecting and mitigating such sophisticated LLM-based polymorphic threats.
4. How to extend bot detection across languages and informal speech: Social media conversations are increasingly multilingual and often feature informal expressions, such as slang, sarcasm, and code-switching. Existing detection models trained primarily on English datasets may not generalize well in these contexts, leading to performance degradation. Future work could involve expanding training corpora to cover multiple natural languages and designing models capable of capturing the nuanced linguistic patterns found in casual, mixed-language online discourse.
5. How to evaluate interpretability with end-users: While attention-based mechanisms

have demonstrated promising results in highlighting salient features for clickbait detection, it remains unclear whether these align with human reasoning. Journalists, fact-checkers, and other end-users are critical stakeholders in the detection pipeline, as they rely on trustworthy explanations for decision-making. Future research should therefore focus on testing interpretability with domain experts and lay users, assessing whether model explanations correspond to human judgments, and refining attention mechanisms to improve transparency and usability.

Bibliography

- [1] Daniel S Berman, Anna L Buczak, Jeffrey S Chavis, and Cherita L Corbett, “A survey of deep learning methods for cyber security,” *Information*, vol. 10, no. 4, pp. 122, 2019.
- [2] Steven Aftergood, “Cybersecurity: The cold war online,” *Nature*, vol. 547, no. 7661, pp. 30–31, 2017.
- [3] Jian-hua Li, “Cyber security meets artificial intelligence: A survey,” *Frontiers of Information Technology & Electronic Engineering*, vol. 19, no. 12, pp. 1462–1474, 2018.
- [4] VA Golovko, “Deep learning: an overview and main paradigms,” *Optical Memory and Neural Networks*, vol. 26, no. 1, pp. 1–17, 2017.
- [5] Li Deng and Dong Yu, “Deep learning: methods and applications,” *Foundations and Trends in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.
- [6] Xiaoyong Yuan, “PhD forum: deep learning-based real-time malware detection with multi-stage analysis,” in *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2017, pp. 1–2.
- [7] Clayton Allen Davis, Onur Varol, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer, “BotOrNot: A system to evaluate social bots,” in *Proceedings of the 25th international conference companion on world wide web*, 2016, pp. 273–274.

- [8] Chao Yang, Robert Harkreader, and Guofei Gu, “Empirical evaluation and new design for fighting evolving Twitter spammers,” *IEEE Trans. Information Forensics Security*, vol. 8, no. 8, pp. 1280–1293, 2013.
- [9] Zachary Miller, Brian Dickinson, William Deitrick, Wei Hu, and Alex Hai Wang, “Twitter spammer detection using data stream clustering,” *Information Sciences*, vol. 260, pp. 64–73, 2014.
- [10] Onur Varol, Emilio Ferrara, Clayton Davis, Filippo Menczer, and Alessandro Flammini, “Online human-bot interactions: Detection, estimation, and characterization,” in *Proceedings of the International AAAI Conference on Web and Social Media*, 2017, vol. 11, pp. 280–289.
- [11] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi, “DNA-inspired online behavioral modeling and its application to spambot detection,” *IEEE Intelligent Systems*, vol. 31, no. 5, pp. 58–64, 2016.
- [12] Faraz Ahmed and Muhammad Abulaish, “A generic statistical approach for spam detection in online social networks,” *Computer Communications*, vol. 36, no. 10-11, pp. 1120–1129, 2013.
- [13] Shangbin Feng, Zhaoxuan Tan, Rui Li, and Minnan Luo, “Heterogeneity-aware Twitter bot detection with relational graph transformers,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022, vol. 36, pp. 3977–3985.
- [14] Zhenyu Lei, Herun Wan, Wenqian Zhang, Shangbin Feng, Zilong Chen, Qinghua Zheng, and Minnan Luo, “BIC: Twitter bot detection with text-graph interaction and semantic consistency,” *arXiv preprint arXiv:2208.08320*, 2022.
- [15] Prakhar Biyani, Kostas Tsioutsoulis, and John Blackmer, “‘8 amazing secrets for getting more clicks’: detecting clickbaits in news streams using article informality,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016, vol. 30.

- [16] Vaibhav Kumar, Dhruv Khattar, Siddhartha Gairola, Yash Kumar Lal, and Vasudeva Varma, “Identifying clickbait: A multi-strategy approach using neural networks,” in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 1225–1228.
- [17] Manqing Dong, Lina Yao, Xianzhi Wang, Boualem Benatallah, and Chaoran Huang, “Similarity-aware deep attentive model for clickbait detection,” in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2019, pp. 56–69.
- [18] Feng Wei and Uyen Trang Nguyen, “A neural attentive model using human semantic knowledge for clickbait detection,” in *2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCLOUD/SocialCom/SustainCom)*. IEEE Computer Society, 2020, pp. 770–776.
- [19] Lang Chen, Matthew A Lambon Ralph, and Timothy T Rogers, “A unified model of human semantic knowledge and its disorders,” *Nature Human Behaviour*, vol. 1, no. 3, pp. 1–10, 2017.
- [20] George A Miller, “WordNet: a lexical database for English,” *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.
- [21] S. M. Abdulhamid, M. S. Abd Latiff, H. Chiroma, O. Osho, G. Abdul-Salaam, A. I. Abubakar, and T. Herawan, “A review on mobile SMS spam filtering techniques,” *IEEE Access*, vol. 5, pp. 15650–15666, 2017.
- [22] Shabbir Ahmed and Farzana Mithun, “Word stemming to enhance spam filtering,” in *Proceedings of Conference on Email and Anti-Spam (CEAS)*. Citeseer, 2004.
- [23] Tiago Almeida, José María Gómez Hidalgo, and Tiago Pasqualini Silva, “Towards SMS spam filtering: Results under a new dataset,” *International Journal of Information Security Science*, vol. 2, no. 1, pp. 1–18, 2013.

- [24] Akshay Narayan and Prateek Saxena, “The curse of 140 characters: evaluating the efficacy of SMS spam detection on Android,” in *Proceedings of the Third ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*. ACM, 2013, pp. 33–42.
- [25] Kuldeep Yadav, Swetank K Saha, Ponnurangam Kumaraguru, and Rohit Kumra, “Take control of your SMSes: Designing an usable spam SMS filtering system,” in *2012 IEEE 13th International Conference on Mobile Data Management*. IEEE, 2012, pp. 352–355.
- [26] Noura Al Moubayed, Toby Breckon, Peter Matthews, and A Stephen McGough, “SMS spam filtering using probabilistic topic modelling and stacked denoising autoencoder,” in *International Conference on Artificial Neural Networks*. Springer, 2016, pp. 423–430.
- [27] Sepp Hochreiter and Jürgen Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [28] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel, “Handwritten digit recognition with a back-propagation network,” in *Advances in Neural Information Processing Systems*, 1990, pp. 396–404.
- [29] Gauri Jain, Manisha Sharma, and Basant Agarwal, “Optimizing semantic LSTM for spam detection,” *International Journal of Information Technology*, vol. 11, no. 2, pp. 239–250, 2019.
- [30] Gauri Jain, Manisha Sharma, and Basant Agarwal, “Spam detection in social media using convolutional and long short term memory neural network,” *Annals of Mathematics and Artificial Intelligence*, vol. 85, no. 1, pp. 21–44, 2019.
- [31] “Clickbait challenge,” <https://www.clickbait-challenge.org/>, 2017.
- [32] “FNC challenge,” <http://www.fakenewschallenge.org/>, 2017.

- [33] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck, “Learning deep structured semantic models for web search using clickthrough data,” in *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, 2013, pp. 2333–2338.
- [34] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil, “A latent semantic model with convolutional-pooling structure for information retrieval,” in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, 2014, pp. 101–110.
- [35] Hai-Tao Zheng, Jin-Yuan Chen, Xin Yao, Arun Kumar Sangaiah, Yong Jiang, and Cong-Zhi Zhao, “Clickbait convolutional neural network,” *Symmetry*, vol. 10, no. 5, pp. 138, 2018.
- [36] Maria Glenski, Ellyn Ayton, Dustin Arendt, and Svitlana Volkova, “Fishing for clickbaits in social images and texts with linguistically-infused neural network models,” *arXiv preprint arXiv:1710.06390*, 2017.
- [37] Yiwei Zhou, “Clickbait detection in tweets using self-attentive network,” *arXiv preprint arXiv:1710.05364*, 2017.
- [38] Jiaming Zheng, Ke Yu, and Xiaofei Wu, “A deep model based on lure and similarity for adaptive clickbait detection,” *Knowledge-Based Systems*, vol. 214, pp. 106714, 2021.
- [39] Praboda Rajapaksha, Reza Farahbakhsh, and Noel Crespi, “BERT, XLNet or RoBERTa: The best transfer learning model to detect clickbaits,” *IEEE Access*, vol. 9, pp. 154704–154716, 2021.
- [40] Hunt Allcott and Matthew Gentzkow, “Social media and fake news in the 2016 election,” *Journal of Economic Perspectives*, vol. 31, no. 2, pp. 211–36, May 2017.
- [41] Jessikka Aro, “The cyberspace war: propaganda and trolling as warfare tools,” *European View*, vol. 15, no. 1, pp. 121–132, 2016.

- [42] Duncan J Watts, David M Rothschild, and Markus Mobius, “Measuring the news and its impact on democracy,” *Proceedings of the National Academy of Sciences*, vol. 118, no. 15, 2021.
- [43] Carlos Carvalho, Nicholas Klagge, and Emanuel Moench, “The persistent effects of a false news shock,” *Journal of Empirical Finance*, vol. 18, no. 4, pp. 597–615, 2011.
- [44] Wen-Ying Sylvia Chou, Anna Gaysynsky, and Joseph N Cappella, “Where we go from here: health misinformation on social media,” 2020, vol. 110, pp. S273–S275, American Public Health Association.
- [45] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [46] Hengyue Pan, *A Study on Deep Learning: Training, Models and Applications*, Ph.D. thesis, York University, Toronto, 2017.
- [47] Yang Xin, Lingshuang Kong, Zhi Liu, Yuling Chen, Yanmiao Li, Hongliang Zhu, Mingcheng Gao, Haixia Hou, and Chunhua Wang, “Machine learning and deep learning methods for cybersecurity,” *IEEE Access*, vol. 6, pp. 35365–35381, 2018.
- [48] Xing Fang, Maochao Xu, Shouhuai Xu, and Peng Zhao, “A deep learning framework for predicting cyber attacks rates,” *EURASIP Journal on Information Security*, vol. 2019, no. 1, pp. 5, 2019.
- [49] Ralf C Staudemeyer, “Applying long short-term memory recurrent neural networks to intrusion detection,” *South African Computer Journal*, vol. 56, no. 1, pp. 136–154, 2015.
- [50] R Bala Krishnan and NR Raajan, “An intellectual intrusion detection system model for attacks classification using RNN,” *International Journal of Pharmacy & Technology*, vol. 8, no. 4, pp. 23157–23164, 2016.

- [51] Gyuwan Kim, Hayoon Yi, Jangho Lee, Yunheung Paek, and Sungroh Yoon, “LSTM-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems,” *arXiv preprint arXiv:1611.01726*, 2016.
- [52] Jihyun Kim, Jaehyun Kim, Huong Le Thi Thu, and Howon Kim, “Long short term memory recurrent neural network classifier for intrusion detection,” in *2016 International Conference on Platform Technology and Service (PlatCon)*. IEEE, 2016, pp. 1–5.
- [53] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He, “A deep learning approach for intrusion detection using recurrent neural networks,” *IEEE Access*, vol. 5, pp. 21954–21961, 2017.
- [54] Jihyun Kim, Howon Kim, et al., “An effective intrusion detection classifier using long short-term memory with gradient descent optimization,” in *2017 International Conference on Platform Technology and Service (PlatCon)*. IEEE, 2017, pp. 1–6.
- [55] Abien Fred M Agarap, “A neural network architecture combining gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection in network traffic data,” in *Proceedings of the 2018 10th International Conference on Machine Learning and Computing*, 2018, pp. 26–30.
- [56] Ankesh Anand, Tanmoy Chakraborty, and Noseong Park, “We used neural networks to detect clickbaits: You won’t believe what happened next!,” in *European Conference on Information Retrieval*. Springer, 2017, pp. 541–547.
- [57] Md Main Uddin Rony, Naeemul Hassan, and Mohammad Yousuf, “Diving deep into clickbaits: Who use them to what extents in which topics with what effects?,” in *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, 2017, pp. 232–239.
- [58] Xinyi Zhou and Reza Zafarani, “Fake news: A survey of research, detection methods, and opportunities,” *arXiv preprint arXiv:1812.00315*, vol. 2, 2018.

- [59] Halenar Igor, Juhasova Bohuslava, Juhas Martin, and Nesticky Martin, “Application of neural networks in computer security,” *Procedia Engineering*, vol. 69, pp. 1209–1215, 2014.
- [60] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, “ImageNet classification with deep convolutional neural networks,” *Advances in Neural Information Processing Systems*, vol. 25, 2012.
- [61] Yang Yu, Jun Long, and Zhiping Cai, “Network intrusion detection through stacking dilated convolutional autoencoders,” *Security and Communication Networks*, vol. 2017, pp. 1–10, 2017.
- [62] Joshua Saxe and Konstantin Berlin, “eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys,” *arXiv preprint arXiv:1702.08568*, 2017.
- [63] Bojan Kolosnjaji, Ghadir Eraisha, George Webster, Apostolis Zarras, and Claudia Eckert, “Empowering convolutional networks for malware classification and analysis,” in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 3838–3845.
- [64] Wei Wang, Ming Zhu, Jinlin Wang, Xuewen Zeng, and Zhongzhen Yang, “End-to-end encrypted traffic classification with one-dimensional convolution neural networks,” in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2017, pp. 43–48.
- [65] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng, “Malware traffic classification using convolutional neural network for representation learning,” in *2017 International Conference on Information Networking (ICOIN)*. IEEE, 2017, pp. 712–717.

- [66] Bojan Kolosnjaji, Apostolis Zarras, George Webster, and Claudia Eckert, “Deep learning for classification of malware system call sequences,” in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2016, pp. 137–149.
- [67] Kyumin Lee, Brian Eoff, and James Caverlee, “Seven months with the devils: A long-term study of content polluters on Twitter,” in *Proceedings of the International AAAI Conference on Web and Social Media*, 2011, vol. 5, pp. 185–192.
- [68] Mansour Alsaleh, Abdulrahman Alarifi, Abdul Malik Al-Salman, Mohammed Alfayez, and Abdulmajeed Almuhsin, “TSD: Detecting Sybil accounts in Twitter,” in *2014 13th International Conference on Machine Learning and Applications*. IEEE, 2014, pp. 463–469.
- [69] Qinglang Guo, Haiyong Xie, Yangyang Li, Wen Ma, and Chao Zhang, “Social bots detection via fusing bert and graph convolutional networks,” *Symmetry*, vol. 14, no. 1, pp. 30, 2021.
- [70] 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.
- [71] JK Periasamy, R Srinidhi, and S Srividhya, “A deep learning approach for Twitter bot detection,” in *2022 1st International Conference on Computational Science and Technology (ICCST)*. IEEE, 2022, pp. 374–377.
- [72] Sanjukta Mohanty, Satya Prakash Dwivedy, Arup Abhinna Acharya, Suvakanta Mohapatra, Shivam Swastik Sahoo, Sibadatta Samal, and Smrutisrita Samal, “Enhancing the detection of social bots on twitter using ensemble machine learning technique,” in *2022 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC)*. IEEE, 2022, pp. 1–6.

- [73] Zhaoxuan Tan, Shangbin Feng, Melanie Sclar, Herun Wan, Minnan Luo, Yejin Choi, and Yulia Tsvetkov, “Botpercent: Estimating twitter bot populations from groups to crowds,” *arXiv preprint arXiv:2302.00381*, 2023.
- [74] Yingguang Yang, Renyu Yang, Hao Peng, Yangyang Li, Tong Li, Yong Liao, and Pengyuan Zhou, “Fedack: Federated adversarial contrastive knowledge distillation for cross-lingual and cross-model social bot detection,” *arXiv preprint arXiv:2303.07113*, 2023.
- [75] Vaishali Chawla and Yatin Kapoor, “A hybrid framework for bot detection on Twitter: Fusing digital DNA with BERT,” *Multimedia Tools and Applications*, pp. 1–24, 2023.
- [76] Jun Wu, Xuesong Ye, and Man Yan Yuet, “Bottrinet: A unified and efficient embedding for social bots detection via metric learning,” *arXiv preprint arXiv:2304.03144*, 2023.
- [77] Edoardo Di Paolo, Marinella Petrocchi, and Angelo Spognardi, “From online behaviours to images: A novel approach to social bot detection,” *arXiv preprint arXiv:2304.07535*, 2023.
- [78] Feng Cao, Martin Estert, Weining Qian, and Aoying Zhou, “Density-based clustering over an evolving data stream with noise,” in *Proceedings of the 2006 SIAM International Conference on Data Mining*. SIAM, 2006, pp. 328–339.
- [79] Marcel R Ackermann, Marcus Mörtens, Christoph Raupach, Kamil Swierkot, Christiane Lammersen, and Christian Sohler, “StreamKM++: A clustering algorithm for data streams,” *J. Experimental Algorithmics, (JEA)*, vol. 17, pp. 2–4, 2012.
- [80] Nikan Chavoshi, Hossein Hamooni, and Abdullah Mueen, “DeBot: Twitter Bot Detection via Warped Correlation,” in *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE Computer Society, 2016, pp. 817–822.

- [81] Stefano Cresci, Marinella Petrocchi, Angelo Spognardi, and Stefano Tognazzi, “On the capability of evolved spambots to evade detection via genetic engineering,” *Online Social Networks and Media*, vol. 9, pp. 1–16, 2019.
- [82] Christian Grimme, Dennis Assenmacher, and Lena Adam, “Changing perspectives: Is it sufficient to detect social bots?,” in *International Conference on Social Computing and Social Media*. Springer, 2018, pp. 445–461.
- [83] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi, “The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race,” in *Proceedings of the 26th International Conference on World Wide Web Companion*, 2017, pp. 963–972.
- [84] Shangbin Feng, Herun Wan, Ningnan Wang, Jundong Li, and Minnan Luo, “TwiBot-20: A comprehensive Twitter bot detection benchmark,” in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 4485–4494.
- [85] Shangbin Feng, Zhaoxuan Tan, Herun Wan, Ningnan Wang, Zilong Chen, Binchi Zhang, Qinghua Zheng, Wenqian Zhang, Zhenyu Lei, Shujie Yang, et al., “TwiBot-22: Towards graph-based Twitter bot detection,” *arXiv preprint arXiv:2206.04564*, 2022.
- [86] Abdelouahid Derhab, Rahaf Alawwad, Khawlah Dehwah, Noshina Tariq, Farukh Aslam Khan, and Jalal Al-Muhtadi, “Tweet-based bot detection using big data analytics,” *IEEE Access*, vol. 9, pp. 65988–66005, 2021.
- [87] Kadhim Hayawi, Susmita Saha, Mohammad Mehedy Masud, Sujith Samuel Mathew, and Mohammed Kaosar, “Social media bot detection with deep learning methods: a systematic review,” *Neural Computing and Applications*, pp. 1–16, 2023.
- [88] Zineb Ellaky, Faouzia Benabbou, and Sara Ouahabi, “Systematic literature review of social media bots detection systems,” *Journal of King Saud University-Computer and Information Sciences*, vol. 35, no. 5, pp. 101551, 2023.

- [89] Raad Al-azawi and O Safaa, “Feature extractions and selection of bot detection on twitter a systematic literature review,” *Inteligencia Artificial*, vol. 25, no. 69, pp. 57–86, 2022.
- [90] Emilio Ferrara, “Twitter spam and false accounts prevalence, detection and characterization: A survey,” *arXiv preprint arXiv:2211.05913*, 2022.
- [91] Feng Wei and Uyen Trang Nguyen, “Twitter bot detection using bidirectional long short-term memory neural networks and word embeddings,” in *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. IEEE, 2019, pp. 101–109.
- [92] Jeffrey Pennington, Richard Socher, and Christopher D Manning, “GloVe: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.
- [93] Martin Potthast, Sebastian Köpsel, Benno Stein, and Matthias Hagen, “Clickbait detection,” in *European Conference on Information Retrieval*. Springer, 2016, pp. 810–817.
- [94] Xianzhi Wang, Quan Z Sheng, Lina Yao, Xue Li, Xiu Susie Fang, Xiaofei Xu, and Boualem Benatallah, “Truth discovery via exploiting implications from multi-source data,” in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2016, pp. 861–870.
- [95] Yimin Chen, Niall J Conroy, and Victoria L Rubin, “Misleading online content: recognizing clickbait as” false news”,” in *Proceedings of the 2015 ACM on workshop on multimodal deception detection*, 2015, pp. 15–19.
- [96] Peter Bourgonje, Julian Moreno Schneider, and Georg Rehm, “From clickbait to fake news detection: an approach based on detecting the stance of headlines to articles,” in *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism*, 2017, pp. 84–89.

- [97] Abhijnan Chakraborty, Bhargavi Paranjape, Sourya Kakarla, and Niloy Ganguly, “Stop clickbait: Detecting and preventing clickbaits in online news media,” in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2016, pp. 9–16.
- [98] Hai-Tao Zheng, Xin Yao, Yong Jiang, Shu-Tao Xia, and Xi Xiao, “Boost clickbait detection based on user behavior analysis,” in *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data*. Springer, 2017, pp. 73–80.
- [99] Bram Vijnen et al., “The listicle: An exploring research on an interesting shareable new media phenomenon,” *Studia Universitatis Babes-Bolyai-Ephemerides*, vol. 59, no. 1, pp. 103–122, 2014.
- [100] Abdul Razaque, Bandar Alotaibi, Munif Alotaibi, Shujaat Hussain, Aziz Alotaibi, and Vladimir Jotsov, “Clickbait detection using deep recurrent neural network,” *Applied Sciences*, vol. 12, no. 1, pp. 504, 2022.
- [101] Abdul Razaque, Bandar Alotaibi, Munif Alotaibi, Fathi Amsaad, Ansagan Manasov, Salim Hariri, Banu B Yergaliyeva, and Aziz Alotaibi, “Blockchain-enabled deep recurrent neural network model for clickbait detection,” *IEEE Access*, vol. 10, pp. 3144–3163, 2021.
- [102] Mengxi Zhou, Wei Xu, Wenping Zhang, and Qiqi Jiang, “Leverage knowledge graph and GCN for fine-grained-level clickbait detection,” *World Wide Web*, vol. 25, no. 3, pp. 1243–1258, 2022.
- [103] Mounika Marreddy, Subba Reddy Oota, Lakshmi Sireesha Vakada, Venkata Charan Chinni, and Radhika Mamidi, “Clickbait detection in Telugu: Overcoming NLP challenges in resource-poor languages using benchmarked techniques,” in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.

- [104] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.
- [105] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019.
- [106] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le, “Xlnet: Generalized autoregressive pretraining for language understanding,” in *Advances in neural information processing systems*, 2019, pp. 5754–5764.
- [107] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning, “ELECTRA: Pre-training text encoders as discriminators rather than generators,” *arXiv preprint arXiv:2003.10555*, 2020.
- [108] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut, “ALBERT: A lite BERT for self-supervised learning of language representations,” *arXiv preprint arXiv:1909.11942*, 2019.
- [109] Radim Rehurek and Petr Sojka, “Software framework for topic modelling with large corpora,” in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 2010, p. 45.
- [110] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.
- [111] Manqing Dong, Lina Yao, Xianzhi Wang, Boualem Benatallah, Quan Z Sheng, and Hao Huang, “Dual: a deep unified attention model with latent relation representations

- for fake news detection,” in *International Conference on Web Information Systems Engineering*. Springer, 2018, pp. 199–209.
- [112] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy, “Hierarchical attention networks for document classification,” in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 2016, pp. 1480–1489.
- [113] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei, “Neural natural language inference models enhanced with external knowledge,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 2406–2417.
- [114] Arthur P Dempster, Nan M Laird, and Donald B Rubin, “Maximum likelihood from incomplete data via the em algorithm,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.
- [115] Tiago A Almeida, Jurandy Almeida, and Akebo Yamakami, “Spam filtering: how the dimensionality reduction affects the accuracy of naive bayes classifiers,” *Journal of Internet Services and Applications*, vol. 1, no. 3, pp. 183–200, 2011.
- [116] Yoav Freund, Robert E Schapire, et al., “Experiments with a new boosting algorithm,” in *International Conference on Machine Learning*. Citeseer, 1996, vol. 96, pp. 148–156.
- [117] S James Press and Sandra Wilson, “Choosing between logistic regression and discriminant analysis,” *Journal of the American Statistical Association*, vol. 73, no. 364, pp. 699–705, 1978.
- [118] Andrew Y Ng and Michael I Jordan, “On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes,” in *Advances in Neural Information Processing Systems*, 2002, pp. 841–848.
- [119] Simon Haykin, *Neural Networks and Learning Machines, 3/E*, Pearson Education India, 2010.

- [120] George Forman, Martin Scholz, and Shyamsundar Rajaram, “Feature shaping for linear svm classifiers,” in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2009, pp. 299–308.
- [121] José María Gómez Hidalgo, “Evaluating cost-sensitive unsolicited bulk email categorization,” in *Proceedings of the 2002 ACM Symposium on Applied Computing*. ACM, 2002, pp. 615–620.
- [122] John Platt, “Sequential minimal optimization: a fast algorithm for training support vector machines,” *MSRTR: Microsoft Research*, vol. 3, no. 1, pp. 88–95, 1998.
- [123] Tiago A Almeida, Akebo Yamakami, and Jurandy Almeida, “Filtering spams using the minimum description length principle,” in *Proceedings of the 2010 ACM Symposium on Applied Computing*. ACM, 2010, pp. 1854–1858.
- [124] David W Aha, Dennis Kibler, and Marc K Albert, “Instance-based learning algorithms,” *Machine Learning*, vol. 6, no. 1, pp. 37–66, 1991.
- [125] José María Gómez Hidalgo, Guillermo Cajigas Bringas, Enrique Puertas Sáenz, and Francisco Carrero García, “Content based SMS spam filtering,” in *Proceedings of the 2006 ACM Symposium on Document Engineering*. ACM, 2006, pp. 107–114.
- [126] JR QUINLAN, “C4.5: Programs for machine learning,” in *Proceedings of 10th International Conference on Machine Learning*, 1993, pp. 252–259.
- [127] Eibe Frank and Ian H Witten, “Generating accurate rule sets without global optimization,” in *Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 1998, pp. 144–151.
- [128] Leo Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [129] Lior Rokach, “Ensemble-based classifiers,” *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 1–39, 2010.

- [130] Mingbin Xu, Feng Wei, Sedtawut Watcharawittayakul, Yuchen Kang, and Hui Jiang, “The YorkNRM systems for trilingual EDL tasks at TAC KBP 2016,” in *Proc. Text Analysis Conference, (TAC)*, 2016.
- [131] Mingbin Xu, Nargiza Nosirova, Kelvin Jiang, Feng Wei, and Hui Jiang, “FOFE-based deep neural networks for entity discovery and linking,” in *Proc. Text Analysis Conference, (TAC)*, 2017.
- [132] Feng Wei, Uyen Trang Nguyen, and Hui Jiang, “Dual-FOFE-net neural models for entity linking with PageRank,” in *International Conference on Artificial Neural Networks*. Springer, 2019, pp. 635–645.
- [133] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.
- [134] Sebastian Mika, Gunnar Ratsch, Jason Weston, Bernhard Scholkopf, and Klaus-Robert Mullers, “Fisher discriminant analysis with kernels,” in *Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop*. IEEE, 1999, pp. 41–48.
- [135] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao, “Recurrent convolutional neural networks for text classification,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015, vol. 29.
- [136] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak, “Convolutional, long short-term memory, fully connected deep neural networks,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2015, pp. 4580–4584.
- [137] Jin Wang, Liang-Chih Yu, K Robert Lai, and Xuejie Zhang, “Investigating dynamic routing in tree-structured lstm for sentiment analysis,” in *Proceedings of the 2019*

- Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 3432–3437.
- [138] Stéphane Clinchant, Kweon Woo Jung, and Vassilina Nikoulina, “On the use of BERT for neural machine translation,” *arXiv preprint arXiv:1909.12744*, 2019.
 - [139] Yanfu Xu, Zheng Lin, Yuanxin Liu, Rui Liu, Weiping Wang, and Dan Meng, “Ranking and sampling in open-domain question answering,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 2412–2421.
 - [140] Zellig S Harris, “Distributional structure,” *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.
 - [141] Dan Jurafsky, *Speech & Language Processing*, Pearson Education India, 2000.
 - [142] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky, “The Stanford CoreNLP natural language processing toolkit,” in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2014, pp. 55–60.
 - [143] Yoon Kim, Yacine Jernite, David Sontag, and Alexander Rush, “Character-aware neural language models,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016, vol. 30.
 - [144] Cicero Dos Santos and Bianca Zadrozny, “Learning character-level representations for part-of-speech tagging,” in *International Conference on Machine Learning*. PMLR, 2014, pp. 1818–1826.
 - [145] Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William W Cohen, “Tweet2Vec: Character-based distributed representations for social media,” *arXiv preprint arXiv:1605.03481*, 2016.

- [146] Lyan Verwimp, Joris Pelemans, Patrick Wambacq, et al., “Character-word lstm language models,” *arXiv preprint arXiv:1704.02813*, 2017.
- [147] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi, “Bidirectional attention flow for machine comprehension,” *arXiv preprint arXiv:1611.01603*, 2016.
- [148] Helmut Schmid, “Part-of-speech tagging with neural networks,” *arXiv preprint cmp-lg/9410018*, 1994.
- [149] Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao, “Stochastic answer networks for machine reading comprehension,” *arXiv preprint arXiv:1712.03556*, 2017.
- [150] Shanshan Liu, Xin Zhang, Sheng Zhang, Hui Wang, and Weiming Zhang, “Neural machine reading comprehension: Methods and trends,” *Applied Sciences*, vol. 9, no. 18, pp. 3698, 2019.
- [151] Jeffrey L Elman, “Finding structure in time,” *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.
- [152] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.
- [153] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee, “Recent advances in recurrent neural networks,” *arXiv preprint arXiv:1801.01078*, 2017.
- [154] Guo-Bing Zhou, Jianxin Wu, Chen-Lin Zhang, and Zhi-Hua Zhou, “Minimal gated unit for recurrent neural networks,” *International Journal of Automation and Computing*, vol. 13, no. 3, pp. 226–234, 2016.

- [155] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber, “Learning precise timing with LSTM recurrent networks,” *Journal of Machine Learning Research*, vol. 3, no. Aug, pp. 115–143, 2002.
- [156] Feng Wei and Uyen Trang Nguyen, “A lightweight deep neural model for SMS spam detection,” in *2020 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, 2020, pp. 1–6.
- [157] Ronald J Williams and David Zipser, “Gradient-based learning algorithms for recurrent,” *Backpropagation: Theory, Architectures, and Applications*, vol. 433, pp. 17, 1995.
- [158] Stefano Cresci, “MIB datasets,” <http://mib.projects.iit.cnr.it/dataset.html>, 2017.
- [159] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al., “TensorFlow: A system for large-scale machine learning,” in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.
- [160] Shunichi Amari, “A theory of adaptive pattern classifiers,” *IEEE Transactions on Electronic Computers*, , no. 3, pp. 299–307, 1967.
- [161] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv preprint arXiv:1207.0580*, 2012.
- [162] NVIDIA, “NVML API Reference Guide,” <https://docs.nvidia.com/deploy/nvml-api/index.html>, 2022.
- [163] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

- [164] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le, “QANet: Combining local convolution with global self-attention for reading comprehension,” in *International Conference on Learning Representations*, 2018.
- [165] Wuya Chen, Xiaojun Quan, Chunyu Kit, Zhengcheng Min, and Jiahai Wang, “Multi-choice relational reasoning for machine reading comprehension,” in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 6448–6458.
- [166] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou, “Gated self-matching networks for reading comprehension and question answering,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 189–198.
- [167] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom, “Reasoning about entailment with neural attention,” *arXiv preprint arXiv:1509.06664*, 2015.
- [168] Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao, “Stochastic answer networks for machine reading comprehension,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 1694–1704.
- [169] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky, “Visualizing and understanding neural models in nlp,” in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 681–691.
- [170] Martin Potthast, Tim Gollub, Matthias Hagen, and Benno Stein, “The clickbait challenge 2017: Towards a regression model for clickbait strength,” *arXiv preprint arXiv:1812.10847*, 2018.

- [171] Steven Bird, “NLTK: the natural language toolkit,” in *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, 2006, pp. 69–72.
- [172] Diederik P Kingma and Jimmy Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [173] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [174] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, “On the difficulty of training recurrent neural networks,” in *International Conference on Machine Learning*, 2013, pp. 1310–1318.
- [175] Yoon Kim, “Convolutional neural networks for sentence classification,” in *2014 Empirical Methods in Natural Language Processing (EMNLP)*. ACL, 2014, pp. 1746–1751.
- [176] Feng Wei and Uyen Trang Nguyen, “Stock trend prediction using financial market news and BERT,” *12th International Conference on Knowledge Discovery and Information Retrieval*, 2018.
- [177] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al., “Pre-trained models: Past, present and future,” *AI Open*, vol. 2, pp. 225–250, 2021.
- [178] Ilya Loshchilov and Frank Hutter, “Decoupled weight decay regularization,” in *International Conference on Learning Representations*, 2019.
- [179] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

- [180] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever, “An empirical exploration of recurrent network architectures,” in *International Conference on Machine Learning*, 2015, pp. 2342–2350.
- [181] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio, “A structured self-attentive sentence embedding,” *arXiv preprint arXiv:1703.03130*, 2017.
- [182] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber, “Highway networks,” *arXiv preprint arXiv:1505.00387*, 2015.
- [183] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter, “Self-normalizing neural networks,” in *Advances in Neural Information Processing Systems*, 2017, pp. 971–980.
- [184] Adebukola S Onashoga, Olusola O Abayomi-Alli, Adesina S Sodiya, and David A Ojo, “An adaptive and collaborative server-side SMS spam filtering scheme using artificial immune system,” *Information Security Journal: A Global Perspective*, vol. 24, no. 4-6, pp. 133–145, 2015.
- [185] “SMS Spam Collection,” <https://archive.ics.uci.edu/dataset/228/sms+spam+collection>, 2012.
- [186] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik, “A training algorithm for optimal margin classifiers,” in *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, 1992, pp. 144–152.
- [187] Tianqi Chen and Carlos Guestrin, “XGBoost: A scalable tree boosting system,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.
- [188] Alex Graves and Jürgen Schmidhuber, “Framewise phoneme classification with bidirectional LSTM and other neural network architectures,” *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

- [189] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al., “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [190] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al., “PyTorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.
- [191] Olusola Abayomi-Alli, Sanjay Misra, and Adebayo Abayomi-Alli, “A deep learning method for automatic SMS spam classification: Performance of learning algorithms on indigenous dataset,” *Concurrency and Computation: Practice and Experience*, vol. 34, no. 17, pp. e6989, 2022.