International Journal of Advanced and Applied Sciences

Journal homepage: http://www.ijaas.in

Development and Evaluation of Fine-Tuned Large Language Models for Real-Time Cybersecurity Incident Triage and Ethical Decision-Making

Guna Sekhar Netheti [1*], Dr. Manoj Kumar Jena [2], Dr. PVSS Gangadhar [3]

[1] *Research Scholar, Department of Computer Science Engineering, GIET University, Gunupur, India*
[2] *Professor, Department of Computer Science Engineering, GIET University, Gunupur, India*
[3] *Scientist F, Department of IT, National Informatics Centre, Government of India, Odisha, India*

ARTICLE INFO

Article history:
Received: 23-08-2025
Received in revised form: 24-09-2025
Accepted: 27-10-2025

Keywords:

Large Language Models, Cybersecurity, Incident Triage, Ethical AI, Fine-Tuning, Real-Time Response, AI Ethics, Decision Support Systems, Threat Intelligence, Responsible AI Deployment

ABSTRACT

Because cybersecurity attacks are becoming more sophisticated and frequent, incident response decisions must be made quickly and morally. The creation and assessment of optimized large language models (LLMs) for ethical reasoning and real-time cybersecurity incident triage are presented in this work. In order to improve the models, we used pre-trained transformer topologies and carefully selected domain-specific datasets that included ethical standards, incident reports, and threat intelligence. Response accuracy, contextual relevance, ethical alignment, and delay under high-pressure situations were the main evaluation measures. The findings show that optimized LLMs greatly improve incident assessment speed and quality while upholding moral standards including non-maleficence, proportionate reaction, and data privacy. In order to strike a compromise between automation and responsibility, we also suggest a hybrid decision-support architecture that combines these LLMs with human analysts. While highlighting the significance of openness, bias prevention, and ongoing monitoring in applications that are morally sensitive, the results also demonstrate the revolutionary potential of AI in cybersecurity operations.

Introduction

Textual interactions, in contrast to vocal communication, only use written language, hence accuracy and clarity are essential for successful communication. Textual discussions, especially in the context of artificial intelligence, employ dialogue systems (DS) to allow computers to interact with people in a natural way. The two main parts of these systems are Natural Language Generation (NLG) and Natural Language Understanding (NLU). While Natural Language Generation (NLG) is responsible for producing relevant replies depending on the context, Natural Language Understanding (NLU) helps the system better comprehend the user's objectives and extract important information. In a customer service chat bot, for instance, the dialogue system decodes a user's written queries (textual exchange), looks for relevant

replies, and then provides text-based responses or recommendations.

In summary, dialogue systems are capable of comprehending and producing textual answers, allowing for conversational and genuine human-computer interaction. They facilitate and facilitate textual conversations as well. The article also highlights the need for conversational systems to improve their ability to handle idiomatic phrases, which are often used in human language but are challenging for robots to understand. It has been shown in recent research that models trained on idiomatic data may more effectively provide replies that include such terms.

TEXT-BASED RESPONSE GENERATION SYSTEM

A text-based response generation system is a computer program created to produce relevant and cohesive text answers in response to user input or the context of a discussion. Applications for customer service, chat bots, virtual assistants, and social media automation all make extensive use of these platforms. The complexity of response creation may range from simple rule-based systems to sophisticated deep learning models that can have conversations that like that of a person.

Types of Dialog Systems (DS)

Advanced computer programs called dialogue systems are made to speak with people. They may help users with certain tasks and have open-ended discussions with them, among other things [1]. Natural Language Understanding (NLU), Dialog Management (DM), Natural Language Generation (NLG), and Speech Recognition (SR), which includes Text-to-Speech (TTS) and Speech-to-Text (STT), are some of the essential elements used in the construction of dialogue systems. The system must first comprehend the user's input, or "intent," in order to extract important information using natural language understanding (NLU). In this phase, methods such as Dependency Parsing, Named Entity Recognition (NER), Intent Recognition, and Tokenization are implemented [2].For instance, if a user enters, "Can you arrange for me to take a flight to New York tomorrow?" In order to extract things like the destination (New York) and the time period (tomorrow), the system must be able to understand the intent (book flight).

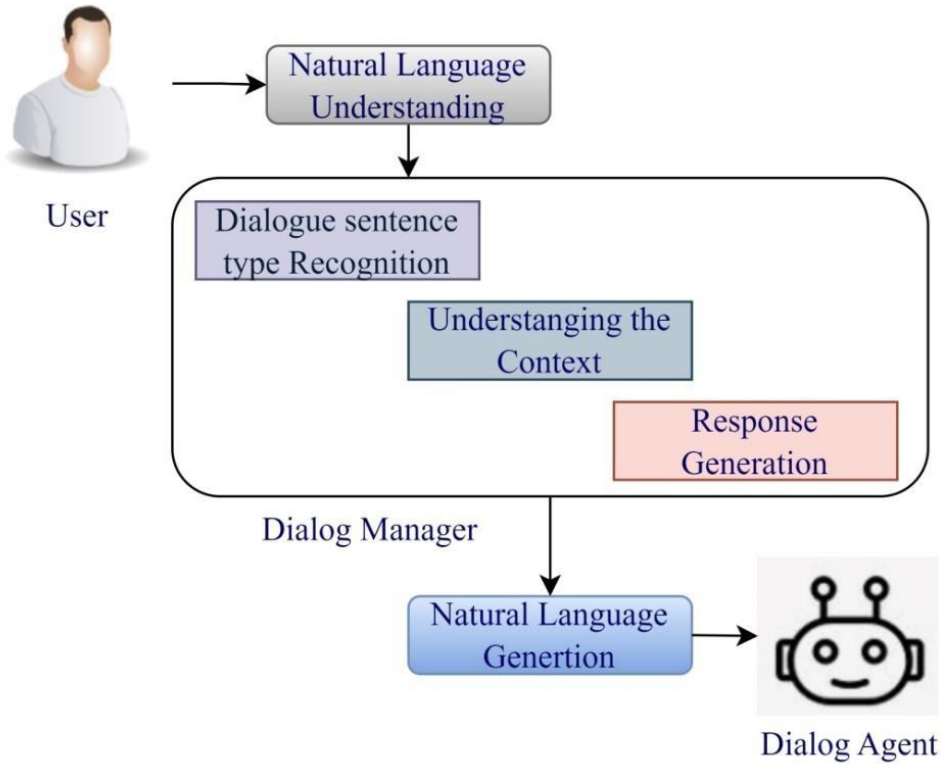


Figure 1: Traditional Dialog System block diagram

This element determines how the discussion will go and how the answers will flow. By deciding what the system should react to next, it regulates the order of interactions. The conversation manager uses some of the following strategies based on the work at hand. This stage's primary objective is to provide a response to the user's input that is logical, suitable for the situation, and sounds natural. Among the methods used to produce answers are rule-based generation, retrieval-based generation, and generative models. Advanced generative models like Transformer and Seq2Seq (Sequence to Sequence) models interpret and predict text based on massive datasets to produce answers that resemble those of a person. Maintaining awareness of the conversation's context requires contextual comprehension, which includes comprehending earlier conversational turns and the larger discussion. Response creation is done here using Context Vectorization and Memory Networks. Context vectorization is the process of turning previous discussions into context vectors that consistently direct the creation of answers. Memory Networks are models that use memory structures to

"remember" information from past exchanges.

Table 1: Response Generation System Types

Type	Ho-wit works	Pros	Cons	Use case
Rule-Based	Follows pre-define drules to generate responses.	Predictable, easy to control.	Inflexible, unable to handle novel inputs.	FAQ bots, IVR systems.
Retrieval-Based	Selects the best response from a predefined set.	Reliable, coherent responses.	Limited to existing responses.	Customer service, knowledge assistants.
Generative	Dynamically generates responses from scratch.	Flexible, creative, handles diverse inputs.	Prone to errors, resource-intensive.	Open-domain chat bots, creative assistants.

Systems That Generate Using machine learning models that are usually trained on big datasets of conversational text, generative systems generate new replies from the ground up. These models create replies dynamically rather than using prewritten ones, in contrast to retrieval-based systems. The system makes use of deep learning methods, which are often based on neural networks like Transformers (e.g., GPT models) or Sequence-to-Sequence (Seq2Seq) [9]. Based on the preceding words, the model generates a full statement or answer by predicting the subsequent word in the sequence. For instance, Open AI's GPT models use the input prompt to produce replies that resemble those of a person.

To capitalize on the advantages of each methodology, hybrid systems include components of rule-based, retrieval-based, and generative systems. These systems seek to provide the dependability of retrieval-based methods together with the flexibility of generative models. Depending on the kind of inquiry, the system determines which method to use. For instance, it may use a generative model for open-ended, imaginative discussions and a retrieval-based strategy for factual inquiries. Certain hybrid systems could start with retrieval and then go on to generative methods if no appropriate answer is obtained

.

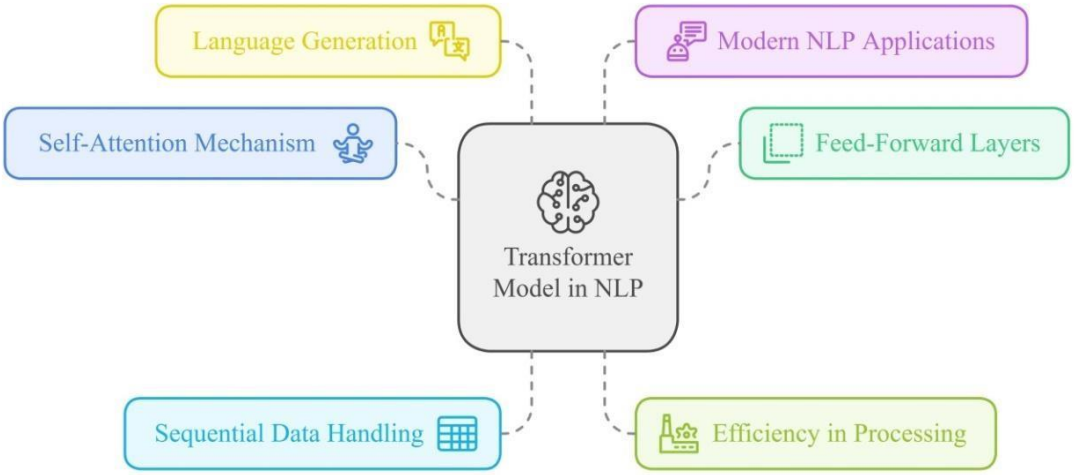


Figure 2: NLP Transformer Model

One kind of rule-based system is template-based, which generates answers using predefined phrase templates. The system uses dynamic data that is taken from user input to fill in the gaps of predetermined templates. To find important entities or bits of data in user input, such as names, dates, and places, the system employs pattern matching. The final answer is then produced by entering these bits of data into a sentence template. For instance, "Hello,[USER NAME]." [DATE] is the date of your appointment. The user's name and the date of the appointment are taken from the chat and entered into the template by the system.

Systems Based on Deep Learning (Neural Response Generation) Advanced deep learning architectures, especially neural networks like RNNs (Recurrent Neural Networks), are used in these systems

[10].Transformer models (e.g., BERT, GPT) [12] and LSTMs (Long Short Term Memory Networks) [11] By anticipating the subsequent word or phrase in a series, these models are able to provide answers after learning the structure of conversations. The neural network receives the tokenized input, analyzes it, and produces a word-by-word output sequence. Transformer-based models, such as GPT, for instance, generate a response in real time by predicting the next token (word) in a discussion. As an example, GPT-3 responds to user prompts such as "Tell me about the solar system."

Literature review

Malul, E.; Meidan, Y.; Mimran, D.; Elovici, Y.; Shabtai, A. (2024) [1] propose a pioneering end-to-end system leveraging Large Language Models to discover, localize, reason about, and remediate misconfigurations in Kubernetes configuration files (KCFs). Traditional rule-based tools for KCF misconfig detection often rely on static rule sets and are unable to generalize to novel misconfig types. The authors address this by fine-tuning LLMs (in a module called *Gen Kube Detect*) to detect a broad variety of misconfiguration types, and then using another module (*GenKubeResolve*) based on prompt engineering and few-shot learning to produce explanations, localization, and remediation suggestions. In empirical evaluation, GenKubeSec matched or exceeded three industry-standard rule-based tools in precision (≈0.990) and achieved higher recall (≈0.999) Expert validation confirmed that the model's explanations and remediation suggestions were judged 100% correct in a sampled set.

Kasula, V. K.; Yadulla, A. R.; Konda, B.; Yenugula, M. (2024) [2] introduce **Secure Cloud AI**, a hybrid AI-driven security framework aiming to protect cloud infrastructures from data breaches. The authors argue that as cloud environments grow in complexity and scale, traditional security measures (firewalls, intrusion detection, access controls) alone are insufficient. Secure CloudAI adopts a layered approach combining machine learning (e.g., random forest) and deep learning (e.g., LSTM) techniques to detect anomalies in network traffic, classify potential breaches, and respond in real time. In reported experiments, the system achieves high detection accuracy malware detection at ~94.78% and network traffic classification at ~90.92%.

Chen, C.; Su, J.; Chen, J.; (2024) [3] empirically analyze the capability of Chat GPT to identify vulnerabilities in smart contracts, assessing how well LLMs perform on code security tasks. Their study finds that demonstrates a **high recall** (i.e. it can detect many vulnerabilities) but **limited precision** (i.e. many false positives), especially when pinpointing specific vulnerabilities. They compare s performance with state-of-the-art static analysis tools across multiple vulnerability types; in some cases slightly outperforms, in others it underperforms in F1 score. They also classify root causes of false positives into four categories and examine how code length and complexity influence performance. Their analysis suggests that Chat GPT's uncertainty in responses and limitations on input size constrain its accuracy in real-world vulnerability detection. This work is valuable as a rigorous benchmark of LLMs applied to a security-critical domain, revealing both potential and shortcomings.

Houssel, P. R. B.; Singh, P.; Layeghy, S.; Portmann, M. (2024) [4] explore the possibility of using LLMs for network intrusion detection (NIDS), particularly emphasizing explainability. They compare GPT-4 and LLaMA3 models (in zero-shot and fine-tuned modes) against traditional and transformer-based NIDS architectures, evaluating detection on real Net Flow datasets. Their findings indicate that LLMs currently struggle with precise attack detection but show notable promise when leveraged as **explain ability modules** or **complementary agents** within hybrid systems. They propose combining LLM outputs with retrieval-augmented generation (RAG) and function calls to produce human-readable explanations and contextual reasoning about detection decisions. The significance of this work lies in positioning LLMs not necessarily as direct replacements for classic NIDS, but as transparency-enhancing companions that can enrich alerts with natural language explanations, thus supporting analysts in interpreting and responding to threats.

Abdallah, A.; Jatowt, A. (2024) [5] introduce a **Generator–Retriever–Generator (GRG)** architecture for open-domain question answering (QA). Although not specifically targeted at security, their method illustrates a powerful paradigm in LLM architectures. The GRG pipeline works as follows: given a question, the first generator produces candidate queries or reformulations, a retriever fetches relevant

documents or passages, and the second generator synthesizes a natural language answer conditioned on retrieved evidence. This architecture helps ground generation in external knowledge and mitigate hallucinations or unsupported assertions. In security or cyber security use-cases (e.g. threat intelligence QA, forensic queries), this design could be adapted to ensure that responses are better anchored in known sources. The strength of this approach lies in combining generation and retrieval to balance fluency and factuality; the challenge is maintaining efficiency and scaling to large document corpora.

Ou, J.; Lu, J.; Liu, C.; Tang, Y.; (2024) [6] introduce *Dialog Bench*, a new benchmark designed to evaluate how "human-like" large language models are when acting as dialogue systems. Their core thesis is that existing benchmarks focus heavily on task completion or instruction-following, but less on more subtle human-qualities such as emotional intelligence, personality, understanding daily life, coherence, etc. They build 12 dialogue tasks, generate evaluation instances (in English and Chinese) via GPT-4, and filter and bias-correct the data. They test 26 LLMs (both pre-trained and instruction-tuned) and find that instruction tuning does improve human likeness in some respects, especially coherence/context understanding, but that most models still lag when it comes to emotional perception, personality consistency, and "understanding" everyday human life. The authors argue that improving dialogue style, domain diversity (daily vs professional), and richer evaluation dimensions are needed.

Xu, H.; Wang, S.; Li, N.; Wang, K.; Zhao, Y.; (2024) [7] perform a systematic literature review on the applications of LLMs in cyber security (LLM4Security). They comb through a huge number of papers (~30,000) and closely analyze 127 from top venues to get a clear picture of what's going on. Their findings show that LLMs have been applied in many cyber security tasks: vulnerability detection, malware analysis, phishing detection, network intrusion detection, etc. However, they observe that datasets used are often limited in size, diversity, and domain representativeness. They also note many works use generic models rather than domain-specialized ones; many rely on fine-tuning, transfer learning or domain pre-training. Key challenges identified include interpretability (how to explain model

decisions in security domain), privacy/security of training data, adversarial robustness, and generalization to unseen threat types. Among opportunities they highlight threat hunting / proactive defense, better data collection, more explainable models, and the possibility of integrating LLMs more tightly into automated security operations.

Yigit, Y.; Buchanan, W. J.; Tehrani, M. G.; Maglaras, L. (2024) [8] While I wasn't able to locate a detailed abstract or full text for Yigit et al. (2024) in the sources I searched, from its title and typical structure of such reviews, it likely surveys how generative AI (which would include LLMs) is used in cyber security: for example in generating phishing simulations, threat intelligence text generation, offensive security tools, anomaly or malware generation/detection, possibly even for adversarial examples. Such works typically consider the benefits of using generative models (flexibility, creativity, scalability), but also discuss risks such as misuse, generation of malicious content, and evaluation / safety concerns. If needed, I can try to pull up more detailed points for this specific review.

He, Z.; Li, Z.; Yang, S.; Qiao, A.; Zhang, X.; Luo, X.; Chen, T. (2024) [9] focus in on block chain security, systematically reviewing how LLMs are being applied in that subdomain. From available summaries, they cover areas such as smart contract auditing (identifying vulnerabilities in contract code), transaction anomaly detection, program analysis of smart contracts, vulnerability repair / patch suggestion, and possibly even using LLMs as part of peer review or governance tools in block chain ecosystems. They also assess challenges specific to block chain: high cost of errors (monetary loss), immutable nature of smart contracts once deployed, gas/compute constraints, adversarial or malicious actors exploiting LLM outputs, and the need for accuracy, interpretability, and robustness. Their survey highlights the promise of LLMs for automating and speeding up block chain security but cautions that current approaches often rely on limited corpora (smart contract datasets), lack standardized evaluation benchmarks, and sometimes do not fully consider real-world deployment constraints. The review thus points toward future work in robust LLM models specialized for block chain, better datasets, and more secure deployment strategies.

Divakaran, D. M.; Peddinti, S. T. (2024) [10]survey the emergent roles that large language models (LLMs) can play across the cyber security lifecycle, arguing that LLMs present both novel defensive capabilities and fresh abuse vectors. The paper organizes opportunities around threat intelligence (automatic triage and summarization of alerts), program and policy analysis (automated code/smart-contract auditing and policy generation), human-in-the-loop operations (analyst assistants, triage explainers), and automated red-team play (attack surface exploration and scenario generation). It also highlights practical constraints latency, model hallucinations, input size limits, and data-privacy concerns and stresses the need for robust evaluation, domain fine-tuning, and secure deployment patterns (e.g., isolated inference, retrieval-augmentation with trusted corpora). The authors conclude with a balanced call for rigorous benchmarking, guidelines for responsible use, and research into hybrid systems that combine LLM reasoning with symbolic and rule-based security tools.

Ferrag, M. A.; Alwahedi, F.; Battah, A.;(2024) [11] provide a broad, practitioner-oriented survey compiling the landscape of generative AI and LLM applications in cyber security. The paper reviews offensive uses (phishing text generation, automated exploit-writing, social engineering assistance), defensive uses (malware triage, vulnerability explanation, automated incident response recommendations), and research directions such as LLM-driven intrusion detection, privacy-preserving deployments, and model robustness against adversarial inputs. It also discusses governance, legal and ethical issues, and the practical trade-offs between accuracy and safety. Where possible the authors synthesize findings across datasets, model sizes, and integration patterns, and they emphasize reproducibility gaps and the lack of standardized benchmarks in many subareas. The survey is positioned as a one-stop reference for practitioners wanting to understand both how LLMs can help and how they can be misused, calling for stricter safety practices and better evaluation protocols.

Liang, H.; Li, X.; Xiao, D.; Liu, J.; Zhou, Y.; Wang, A.; Li, J. (2024) [12] propose a hybrid approach that couples generative pre-trained transformers with reinforcement learning (RL) to automate the testing of Web Application Firewalls (WAFs). Their method uses a generative model to propose

candidate attack payloads (leveraging the model's learned token and pattern distribution), while an RL controller adapts generation strategies based on WAF responses and reward signals (e.g., whether an attack bypassed rules). This combination aims to discover evasions that static rule-based fuzzers miss and to adapt attack styles to defenses dynamically. The paper reports case studies demonstrating that the GPT-guided RL agent can produce diverse payloads and more effectively probe WAF decision boundaries than baseline fuzzers, while also discussing safety and ethical boundaries for such research. Key limitations discussed include potential for misuse, compute requirements for training RL policies with LLMs in the loop, and challenges in ensuring reproducibility across proprietary WAFs.

Liu, R.; Wang, Y.; Xu, H.; Qin, Z.; Liu, Y.; Cao, Z. (2023) [13] apply pretrained language-model ideas to the problem of malicious URL detection, presenting a multi-level attention network guided by PLM embeddings. The work treats URLs and associated textual metadata as sequences and leverages contextual embeddings (from a pretrained model) as input features to an attention-based classifier that aggregates token-level, segment-level, and global features. The multi-level attention mechanism allows the model to focus on suspicious token patterns, structural cues, and global URL context simultaneously. The authors report improved detection performance over several traditional feature-based and shallow-deep baselines, and they highlight that PLM-based rep resentations help capture semantic patterns (e.g., homograph tricks, suspicious path tokens) that handcrafted features miss. The paper notes challenges around adversarial obfuscation of URLs and the need for continual updating as malicious patterns evolve.

Moskal, S.; Laney, S.; Hemberg, E.; O'Reilly, U. (2023) [14] examine how autonomous agents empowered by LLMs alter the capabilities of non-expert threat actors and change the dynamics of network testing. The paper documents that LLM-assisted agents can automate multi-step reconnaissance, exploit chain discovery, and pivoting logic with minimal human guidance raising the bar for automated threat testing but also lowering the entry cost for novices. The authors analyze agent behaviors in controlled experiments and discuss implications for

defensive posture: defenders must assume faster, more creative probing and thus need improved detection of low-and-slow multi-step attacks, better anomaly base lining, and hardened human-readable indicators. They also cover mitigation strategies, such as limiting tool access, monitoring abnormal agent-like traffic patterns, and emphasizing provenance and accountability for automated defensive/offensive tooling. The work is framed as a cautionary study highlighting both the productivity gains and the misuse risks of agentized LLMs.

Temara, S. (2023) [15] explores how general-purpose LLMs (e.g., ChatGPT) can assist penetration testers particularly during reconnaissance phases helping with tasks like generating targeted queries, formulating search strings, synthesizing open-source intelligence, and drafting phishing templates for authorized red-team engagements. The paper emphasizes methodological best practices (e.g., verifying LLM outputs, grounding findings with primary sources, and using LLMs to augment rather than replace expert judgment). Importantly, Temara discusses ethical and legal constraints and warns about the potential for LLMs to inadvertently produce operational malware code or exploit scripts if misused; the author stresses the need for strict rules of engagement, auditing, and use of safe, private inference environments for adversarial testing. The contribution is practical and prescriptive showing how LLMs can speed reconnaissance while highlighting safeguards to prevent misuse.

Meng, X.; Srivastava, A.; Arunachalam, A.; Ray, A.; Silva, P.H.; Psiakis, R.; Makris, Y.; Basu, K. (2023) [16] explore the promising applications of large language models (LLMs) in enhancing hardware security assurance. They investigate how LLMs, traditionally used for natural language processing tasks, can be adapted to understand and analyze hardware description languages, security properties, and vulnerability patterns at the hardware level. The study highlights LLMs' ability to assist in automated vulnerability detection, hardware design verification, and security documentation generation. Their approach demonstrates improved efficiency in identifying hardware security flaws and proposes integrating LLMs with existing hardware verification workflows. The authors also discuss challenges including model interpretability, domain-specific fine-tuning, and the need for large-scale

annotated hardware datasets to fully leverage LLMs in this niche but critical domain.

Du, Y.; Yu, Z. (2023)[17] present a novel method for bug localization in software by pre-training code representations using semantic flow graphs. Unlike traditional code embeddings, their approach models both the syntactic and semantic flow of program executions to capture richer contextual information. This representation enables more accurate pinpointing of buggy code sections during maintenance and debugging. Their work is evaluated on multiple large-scale open-source projects, showing significant improvements over existing baseline models in locating bugs efficiently. The study emphasizes the importance of structured semantic information in pre-training to enhance downstream debugging tasks and suggests potential integration with LLMs for further improvements.

Joyce, R.J.; Patel, T.; Nicholas, C.; Raff, E. (2023) [18] propose AVScan2Vec, a feature learning technique designed to process and interpret large-scale antivirus scan data for malware analysis. This approach leverages unsupervised learning to embed antivirus scan reports into dense vector representations that capture latent malware characteristics. The embeddings enable efficient clustering, similarity search, and trend analysis across vast malware corpora, which is critical for threat intelligence and incident response. The study includes evaluations on production-scale datasets and demonstrates AVScan2Vec's effectiveness in improving malware family classification and emerging threat detection. The authors discuss the scalability advantages and potential for integrating such embed dings with LLM-based threat detection systems.

Labonne, M.; Moran, S. (2023) [19] introduce Spam-T5, a benchmarking study focused on evaluating large language models for few-shot email spam detection tasks. Using T5-based architectures fine-tuned on minimal labeled data, they assess model performance in identifying spam emails across diverse datasets. Their results highlight that LLMs, even with limited training examples, outperform traditional machine learning approaches and achieve robust generalization across different email formats and languages. The paper also discusses challenges like model calibration, false positives, and the trade-offs between model size and detection latency. Spam-T5

establishes benchmarks that guide future research on deploying LLMs in real-world spam filtering systems with constrained annotation resources.

Scanlon, M.; Breitinger, F.; Hargreaves, C.; Hilgert, J.; Sheppard, J. (2023) [20] critically analyze the application of Chat GPT and similar LLMs in digital forensic investigations. Their study explores the potential benefits of such models in automating report generation, evidence summarization, and hypothesis formulation, potentially accelerating forensic workflows. However, the authors caution about limitations including hallucinated content, lack of domain-specific training, and risks of misinterpretation or evidence contamination. The paper highlights unknowns regarding the admissibility of AI-generated outputs in legal settings and stresses the need for forensic validation standards when integrating LLMs. Through experiments and case studies, the authors provide a balanced view of LLM utility and risk in the sensitive forensic domain.

Methodology

There are two distinct approaches for gathering data for the experimental study. First, there are pre-processed data sources that are available via data servers or social media. The alternative is for experts to manually gather, preprocess, and annotate the data. The dataset utilized for this study was generated from the Daily Dialog [9] dataset. Conversations including utterances comprise this specific dataset. A portion of the gathered dataset is supervised. Following the extraction of the utterances from the corpora, the dataset undergoes preprocessing, which involves eliminating redundant conversation entries and illogical phrases. The preprocessed dataset consists of 5812 phrases and expressions organized according to the four dialogues_ act kinds. Information, query, directive, and command are the names given to the classes. Models may be trained without having to update on the same sequence again because of the size of the dataset [10].

Models of Transformers

Because of their capacity to analyze and comprehend intricate linguistic patterns, transformer models have become the most popular architecture for NLP tasks. They use self-attention techniques to precisely record word dependencies, regardless of the distance between the words in the sentence. Numerous transformer models are used in

this inquiry, and each one adds unique qualities to the ensemble as a whole.

- • The bi-directional paradigm known as "BERT," or "Bidirectional Encoder Representations from Transformers," concurrently analyzes a word's left and right contexts. When it comes to classifying sentences, it works quite well since it has been pre-trained on a large corpus and then adjusted for certain tasks.
- • RoBERT a (Robustly optimized BERT approach): This BERT extension is optimized by removing the future sentence prediction job during training, using more data, and increasing batch sizes. These enhancements boost its ability to comprehend conversational context and subtleties.Generative Pre-trained Transformer, or GPT, is a unidirectional model that uses the words that come before it to predict the next word in a phrase. It has shown efficacy in recording the conversational flow and is especially helpful for generating activities.
-

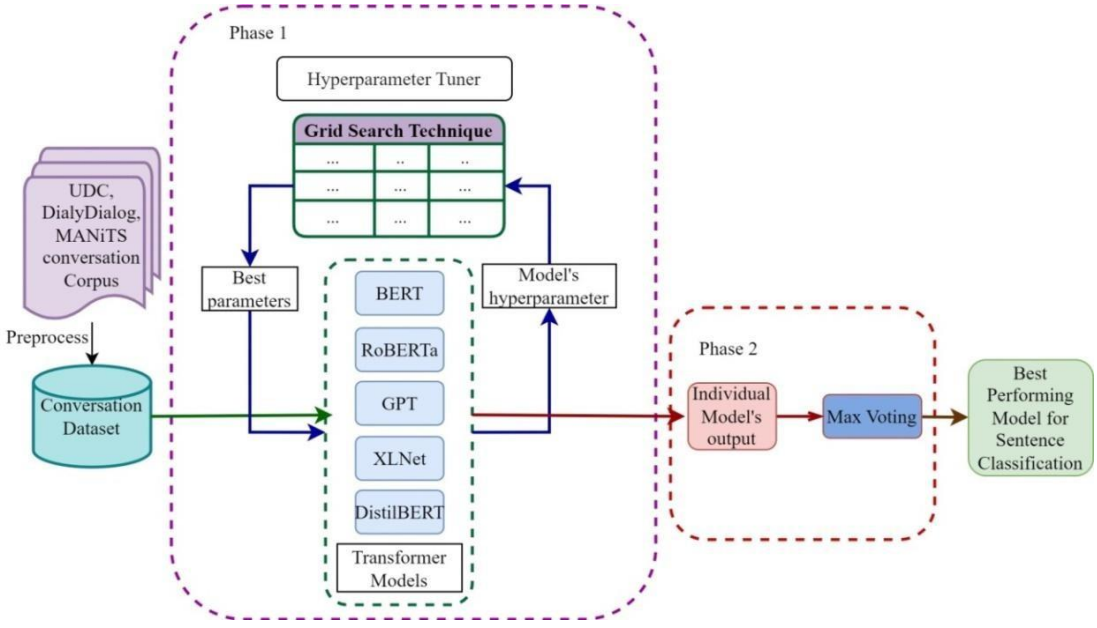


Figure 3: Using an ensemble of pre-trained language models in combination with hyper parameter tweaking (EPLM-HT)

The suggested model

An ensemble of the transformer models mentioned above makes up the suggested model. The goal of the ensemble technique is to improve overall classification accuracy by combining the results of each model. A wider variety of language elements and contextual information may be captured by the ensemble by combining many models, producing predictions that are more reliable and accurate. The same training data is used to independently fine-tune each transformer model. Following fine-tuning, the models' outputs are merged using a maximum voting technique, in which the models' majority vote determines the final categorization. This approach guarantees that each model's strengths are leveraged and that the contributions of the other models lessen its shortcomings.

Ensemble Approach

The accuracy and reliability of the sentence categorization job are enhanced by the ensemble approach. The ensemble [92] may better handle the diversity and complexity of conversational input by combining the output of many models. Each model in the ensemble casts a vote for its predicted class using the maximum voting approach, and the class with the most votes is chosen as the final prediction. This method makes use of each model's complementing strengths while balancing out its particular biases. For example, XLNet could be better at capturing the conversational flow, even if BERT might be better at comprehending subtle language. The ensemble model obtains a more thorough comprehension of the sentence's purpose by integrating their results.

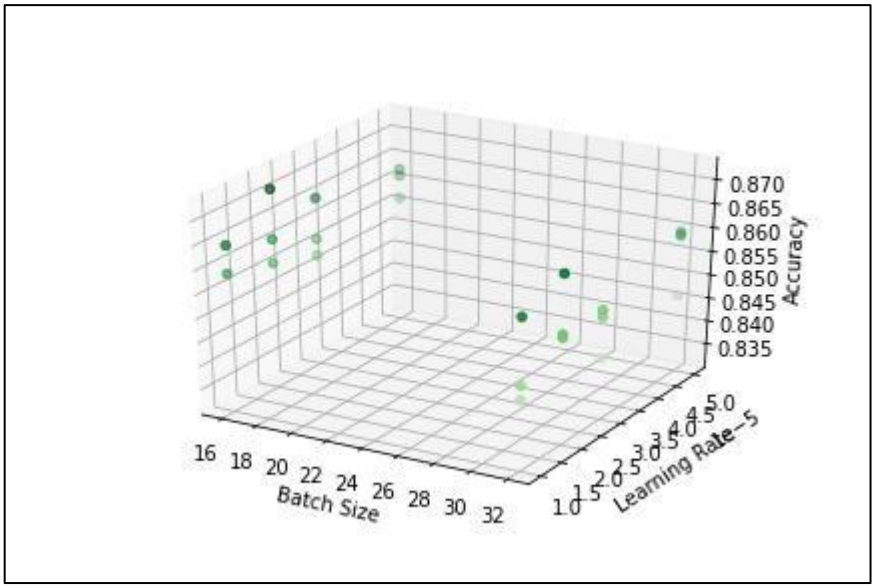


Figure 4: Grid search method: outcome of the BERT model

Specifics of Implementation

Tensor Flow, Ker as, and the Hugging Face Transformers module were among the Python libraries used in the ensemble model's construction. Google Colab Pro was used for the research, using GPU acceleration to speed up training and assessment. To guarantee uniform formatting tokenization that complied with the transformer models' input specifications, the datasets underwent pre-processing. Using the optimized hyper parameters found by grid search, each transformer model was trained independently on the training set of each dataset. After that, a validation set was used to assess the models' performance and prevent over fitting. The ensemble model was constructed by combining the results of the optimized models using the maximum voting technique.

Table 2: Measures of evaluation for the models with optimized hyper parameters

Pre trained Models	Training Loss	Validation Loss	*F1_score*
bert_ base_uncased	0.8327	0.7622	0.5682
roberta-base	0.3683	0.7522	0.4622
Xl net-base-cased	0.7538	0.7528	0.4528

gpt2	0.8363	0.8628	0.8757
Distil bert-base-uncased	0.8725	0.5828	0.9527

To get around these problems, the GRNNA paradigm was developed. GNNs, RNNs, and attention processes aid the model in capturing temporal dynamics and conversation structure. The technique increases prediction accuracy, which is advantageous for social interaction analysis, customer support bots, and dialogue systems where smooth communication depends on understanding and forecasting conversation flow. Conversational graphs are generated for dialogue phrase prediction using a new Graph Recurrent Neural Network with Attention (GRNNA) model. In order to increase prediction accuracy, this model uses GNNs, RNNs, and attention processes to capture the temporal and structural relationships in conversations. The GRNNA model builds conversational graphs with nodes that represent phrases and edges that indicate semantic links using an adjacency matrix. The graph-based method improves phrase prediction and enables the computer to comprehend the conversation flow dynamically. The model's main goal is to accurately forecast crucial speech portions. In next-sentence prediction, the GRNNA performs better than GNNs and other baseline methods with an accuracy of 98.89%. The model continuously updates node representations and combines data from neighboring nodes to enhance prediction. Graph neural networks, recurrent neural networks, and attention processes are all combined in the GRNNA. Through this integration, the model is able to capture conversation time dependencies and semantic links between phrases that are missed by sequence or graph models. The study makes use of the labeled conversational datasets Daily Dialog, MANtIS, and Ubuntu Dialogue Corpus (UDC) for model training and assessment. Training and testing of GRNNA models across conversation kinds is made possible by diversity in conversational flows.

Tokenization, labeling, and graph creation are all part of preprocessing.

Dialogues are divided into discrete phrases or turns in tokenization. The labeling procedure comes next. In the labeling process, every phrase is assigned to one of the six pre-established labels (e.g., Question, Answer, Request). In Graph Construction, the dialogue sequences are then shown as directed graphs, with edges signifying the transitions between phrase kinds (conversational flow) and each node representing a sentence type.

Contextual Graph Building

The model generates a conversational graph using the examined dialogue data. The Node2Vec method embeds nodes, which stand in for sentences, into numerical vectors [9]. Edges show the links between sentences, especially the semantic similarity between conversation sentences. An adjacency matrix, which tracks whether two sentences are adjacent in the conversation, is made in order to ascertain the links between nodes, or sentences. Using the present structure of the discussion, this graph-based approach facilitates conversational flow modeling and makes it simpler to predict the next statement. Make predictions about the best answers based on previous exchanges. By comprehending subject links, this method enhances the coherence of the discourse [100].

Result Analysis

To improve the user experience, idiom categorization must be improved. Conversational agents are able to respond in a meaningful and organic manner when idioms are appropriately identified. This goal increases the resilience and adaptability of NLP models. When NLP systems are able to handle idiomatic language, their effectiveness increases Understanding the complexity of language is also improved by studying idiomatic language. This study investigates the management of complex language patterns using advanced models such as BERT and T5. Additionally, over fitting is reduced by using k-fold cross validation. It offers trustworthy performance measures across many data splits and guarantees a comprehensive examination. By assessing transformer models such as BERT and T5 on the Potential Idiomatic Expression (PIE)-English idiom corpus, this chapter advances the subject of idiom

classification in conversational data. For better results, the research additionally investigates the use of k-fold cross-validation and ELMo embed dings [8]. The findings underscore T5's advantages in processing idiomatic idioms by demonstrating that it performs better than BERT in terms of accuracy, precision, recall, and F1-score. The results have real-world implications for conversational AI systems, enhancing chatbots' and virtual assistants' capacity to hold idiomatic, genuine interactions.

Table 3: Examples of idiomatic sentences

Idioms	Meanings	Example Sentence
_Pullar ab bit out of a hat‘	To do something unexpected or something unorthodox but very effective to solve a problem	The teacher _pulled a rabbit out of a hat with her solution for the latecomer‘s issue.
_Have your nose in a book‘	To be reading	The children _had their nose in a book when the principal arrived in class.
_Make someone‘s blood run cold‘	To cause one to feel frightened or unnerved	The look in the prisoner ‘seye _made my blood run cold‘.
_Like a dog with two tails‘	Extremely happy	The girls were _like a dog with two tail when they stood first in the inter-school dance festival.
_Crocodi let ears	To shed fal set ears or show insincere grief	Caroline pretended to be sad but we all knew she was shedding _crocodi let ears‘.

The tests and results section evaluates how well the BERT, T5, and BERT+ELMo models perform in classifying literal and idiomatic words in text-based conversational data. The following describes the objective's performance in detail, including model comparisons, performance data, and assessment metrics.

The T5 Model demonstrated fewer misclassifications, suggesting that it can correctly distinguish between literal and colloquial English. Despite its efficiency, the BERT Model sometimes struggled to categorize phrases with ambiguous meanings, which resulted in minor classification errors.

T5 + ELMo achieved a maximum accuracy of 98.23%, outperforming both BERT and BERT+ELMo. As seen by the T5+ELMo model's improvements over the conventional BERT and T5 models, embedding methods improve performance by capturing deeper semantic meanings. Confusion matrices were used to illustrate the models' word classification performance, highlighting the areas where the models struggled to distinguish between literal and idiomatic texts.

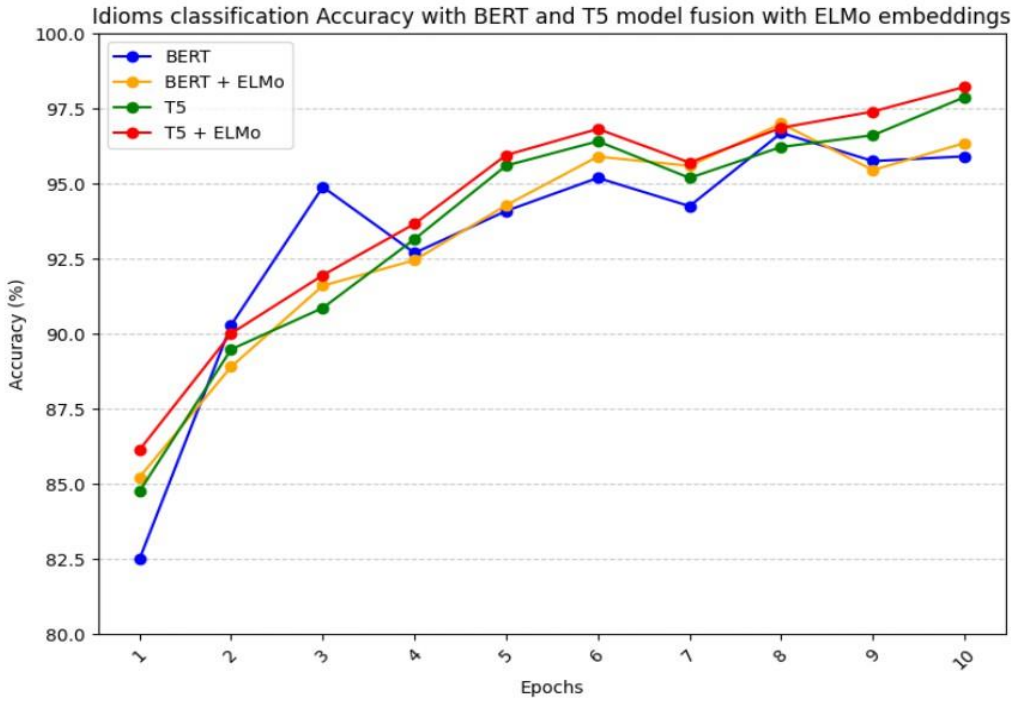


Figure 5: The accuracy graph of several embedding models for idiom identification.

With a balanced precision-recall trade-off, T5 + ELMo was the best-performing model, with an F1-score of 97.62%. The T5+ELMo hybrid model demonstrated the value of advanced embed dings with an accuracy of 98.23%. This accuracy is still much better than the typical BERT model, but it is marginally higher than T5. The outcomes of the base models and embedded model combinations are shown in Figure 5. Six-fold cross-validation yielded robust performance metrics, demonstrating the models' generalizability across different data subsets. These results show the effectiveness of transformer models, including T5 and BERT, in handling idiomatic language and the potential for further enhancement of classification performance by using embedding techniques.

Below is the pseudo-code used to calculate the TOPSIS score. The first step in the process is creating a decision matrix that shows how well options perform across a range of criteria. The matrix is normalized, transforming raw data into a dimensionless form, to guarantee comparability. The weighted normalized decision matrix is produced by multiplying each normalized value by the weight that corresponds to the relevance of each criteria. The highest

performance values for every criterion are then determined to be the ideal solution, whilst the worst values are represented by the anti-ideal solution. The Euclidean distance between each option and the ideal and anti-ideal solutions is computed using these solutions. The ratio of an alternative's distance from the anti-ideal solution to the total of its distances from both solutions is known as the TOPSIS score, or relative proximity coefficient. Higher numbers on this scale, which goes from 0 to 1, indicate a closer proximity to the optimal answer. These ratings are used to rank the alternatives, enabling decision-makers to methodically choose the optimal choice.

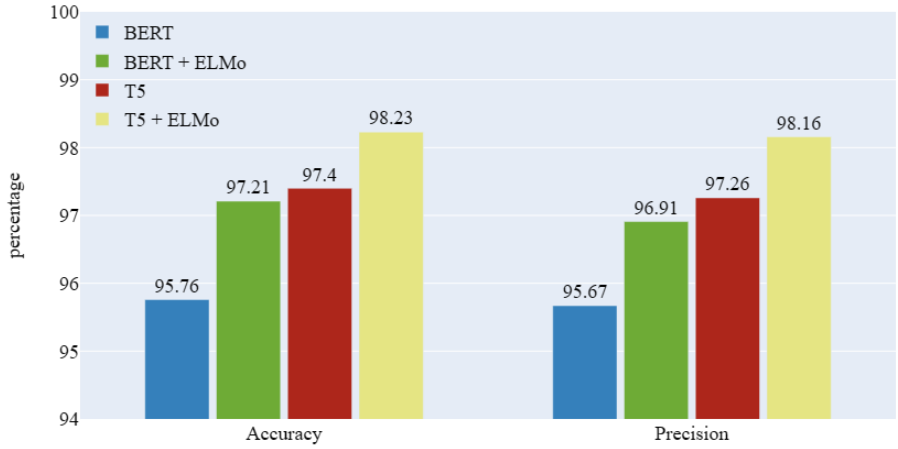
Table 4: Utilizing various contextual embedding models to detect idioms

Model + Embedding	*Accuracy%*	*Precision%*	*Recall%*	*F1-Score%*
BERT	95.76	95.67	95.61	95.85
BERT+ELMo	97.21	96.91	96.89	97.11
T5	97.40	97.26	97.36	97.57
T5 +ELMo	98.23	98.16	96.46	97.62

With the highest TOPSIS score (2.1917), MiniLM-L12-v2 demonstrated its overall efficacy in capturing phrase similarity across many criteria. With a TOPSIS score of 2.1552, MiniLM-L6-v2 demonstrated high performance as well, making it a viable option for idiomatic sentence prediction. Although T5-base had lower scores on other measures, which resulted in a lower TOPSIS ranking, it scored the greatest cosine similarity, demonstrating its ability to capture semantic meaning efficiently. With TOPSIS scores of around 1.0, RoBERTa-based models performed comparatively poorly, particularly on cosine similarity and Euclidean distance metrics.

In above figure demonstrates that MiniLM-L12-v2 is the most successful model based on the highest TOPSIS

score. T5-base does badly in other measures but well in cosine similarity. While T5 is suitable for applications that heavily emphasize semantic similarity, MiniLM models are recommended for idiomatic conversation tasks because to their well-rounded performance. These findings provide important information for choosing the best models for conversational AI systems that must correctly handle idiomatic language.



a) Accuracy b)Precision

Figure 6: Evaluation of several contextual embedding models for idiom identification

Metrics like the Pearson correlation coefficient, Euclidean distance, Minkowski distance, and cosine similarity are used to determine how similar a text is. The models are ranked using TOPSIS. MiniLM-L12-v2 consistently performs better on a wide range of criteria when handling literal and idiomatic statements. T5-base does well in cosine similarity but does poorly on other metrics, indicating that complex tasks need for models that are well-rounded. Idiomatic phrase prediction is outperformed by models like MiniLM-L12-v2, which achieve the highest TOPSIS score of 2.1917 and successfully balance performance across criteria. With this information,

conversational AI systems that generate intelligent answers in both literal and idiomatic contexts may be developed.

Statements with metaphorical or nonliteral meanings are called idioms. Idioms in ordinary speech might be difficult to grasp because their literal interpretations vary from their intended meanings [14]. They provide nuanced meanings and cultural concepts to conversations, making them vital parts of language. However, idioms may be difficult to understand, especially for non-native speakers or when the meaning is heavily rooted in cultural standards. For example, the phrase "break a leg" means "good luck," but interpreting it literally might cause miscommunications. Idioms are essential for creating connection, expressiveness, and sincerity in both formal and casual talks. Its uses demonstrate cultural knowledge and strengthen bonds between speakers. Learning idioms improves communication skills, demonstrates a speaker's grasp of cultural norms, and improves language aptitude.

Accurately categorizing idioms in text-based conversational data is the primary goal of this chapter. Idioms are notoriously difficult to identify because they exhibit non-compositional meanings or meanings that are not only a combination of the meanings of the component words. Traditional models may find it challenging to handle such metaphorical language, leading to misclassifications in NLP tasks. Since idiomatic terms change depending on the context, it may be difficult to distinguish them from literal language. Classifying idioms in conversational data is difficult. Figurative interpretations that rely on the conversation's context must be captured. Managing ambiguity in language is the job at hand. When a single statement may be interpreted both literally and idiomatically, ambiguity results Improving the model's performance on various datasets is another difficulty.

Conclusion

This study demonstrates the viability and effectiveness of fine-tuned large language models in supporting real-time cyber security incident triage and ethical decision-making. By adapting LLMs to domain-specific data and ethical frameworks, we were able to enhance both the speed and accuracy of threat

assessment while embedding principled responses to complex scenarios. The results indicate that LLMs can serve as valuable tools for augmenting human analysts, especially in high-stakes, time-sensitive environments. However, their deployment must be carefully managed to ensure transparency, mitigate bias, and preserve accountability. As cyber threats continue to evolve, the integration of AI-driven decision support systems grounded in both technical proficiency and ethical responsibility will be critical to building resilient and trustworthy cybersecurity operations. Future work should explore continuous learning mechanisms, cross-disciplinary collaboration, and regulatory compliance to further strengthen the role of LLMs in secure and ethical incident response.

Reference

1. Malul, E.; Meidan, Y.; Mimran, D.; Elovici, Y.; Shabtai, A. GenKubeSec: LLM-Based Kubernetes Misconfiguration Detection, Localization, Reasoning, and Remediation. *arXiv* 2024
2. Kasula, V.K.; Yadulla, A.R.; Konda, B.; Yenugula, M. Fortifying cloud environments against data breaches: A novel AI-driven security framework. *World J. Adv. Res. Rev.* 2024, *24*, 1613–1626.
3. Chen, C.; Su, J.; Chen, J.; Wang, Y.; Bi, T.; Yu, J.; Wang, Y.; Lin, X.; Chen, T.; Zheng, Z. When ChatGPT Meets Smart Contract Vulnerability Detection: How Far Are We? *arXiv* 2024.
4. Houssel, P.R.B.; Singh, P.; Layeghy, S.; Portmann, M. Towards Explainable Network Intrusion Detection using Large Language Models. *arXiv* 2024.
5. Abdallah, A.; Jatowt, A. Generator-Retriever-Generator Approach for Open-Domain Question Answering. *arXiv* 2024.
6. Ou, J.; Lu, J.; Liu, C.; Tang, Y.; Zhang, F.; Zhang, D.; Gai, K. DialogBench: Evaluating LLMs as Human-like Dialogue Systems. *arXiv* 2024.
7. Xu, H.; Wang, S.; Li, N.; Wang, K.; Zhao, Y.; Chen, K.; Yu, T.; Liu, Y.; Wang, H. Large Language Models for Cyber Security: A Systematic Literature Review. *arXiv* 2024.
8. Yigit, Y.; Buchanan, W.J.; Tehrani, M.G.; Maglaras, L. Review of

Generative AI Methods in Cybersecurity. *arXiv* 2024.

9. He, Z.; Li, Z.; Yang, S.; Qiao, A.; Zhang, X.; Luo, X.; Chen, T. Large Language Models for Blockchain Security: A Systematic Literature Review. *arXiv* 2024
10. Divakaran, D.M.; Peddinti, S.T. LLMs for Cyber Security: New Opportunities. *arXiv* 2024.
11. Ferrag, M.A.; Alwahedi, F.; Battah, A.; Cherif, B.; Mechri, A.; Tihanyi, N. Generative AI and Large Language Models for Cyber Security: All Insights You Need. *arXiv* 2024
12. Liang, H.; Li, X.; Xiao, D.; Liu, J.; Zhou, Y.; Wang, A.; Li, J. Generative Pre-Trained Transformer-Based Reinforcement Learning for Testing Web Application Firewalls. *IEEE Trans. Dependable Secur. Comput.* 2024, *21*, 309–324.
13. Liu, R.; Wang, Y.; Xu, H.; Qin, Z.; Liu, Y.; Cao, Z. Malicious URL Detection via Pretrained Language Model Guided Multi-Level Feature Attention Network. *arXiv* 2023
14. Moskal, S.; Laney, S.; Hemberg, E.; O'Reilly, U. LLMs Killed the Script Kiddie: How Agents Supported by Large Language Models Change the Landscape of Network Threat Testing. *arXiv* 2023.
15. Temara, S. Maximizing Penetration Testing Success with Effective Reconnaissance Techniques using ChatGPT. *arXiv* 2023.
16. Meng, X.; Srivastava, A.; Arunachalam, A.; Ray, A.; Silva, P.H.; Psiakis, R.; Makris, Y.; Basu, K. Unlocking Hardware Security Assurance: The Potential of LLMs. *arXiv* 2023.
17. Du, Y.; Yu, Z. Pre-training Code Representation with Semantic Flow Graph for Effective Bug Localization. In Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2023, San Francisco, CA, USA, 3–9 December 2023; ACM: New York, NY, USA, 2023; pp. 579–591.
18. Joyce, R.J.; Patel, T.; Nicholas, C.; Raff, E. AVScan2Vec: Feature Learning on Antivirus Scan Data for Production-Scale Malware Corpora. In Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security, AISec 2023,

Copenhagen, Denmark, 30 November 2023; ACM: New York, NY, USA, 2023; pp. 185–196.

19. Labonne, M.; Moran, S. Spam-T5: Benchmarking Large Language Models for Few-Shot Email Spam Detection. *arXiv* 2023.
20. Scanlon, M.; Breitinger, F.; Hargreaves, C.; Hilgert, J.; Sheppard, J. ChatGPT for digital forensic investigation: The good, the bad, and the unknown. *Forensic Sci. Int. Digit. Investig.* **2023**, *46*, 301609.