



# Ein systematischer Vergleich von Named Entity Recognition Ansätzen für Cyber Threat Intelligence

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Data Science**

eingereicht von

**Maximilian Ranzinger, BSc**

Matrikelnummer 12023139

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Ing. Dr.techn. Peter Knees

Mitwirkung: Mag.rer.soc.oec. Dipl.-Ing. Dr.techn. Dr.rer.soc.oec. Florian Skopik  
Dipl.-Ing. Dženan Hamzić

Wien, 9. Februar 2026

---

Maximilian Ranzinger

---

Peter Knees



# A Systematic Comparison of Named Entity Recognition Approaches for Cyber Threat Intelligence

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

**Diplom-Ingenieur**

in

**Data Science**

by

**Maximilian Ranzinger, BSc**

Registration Number 12023139

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr.techn. Peter Knees

Assistance: Mag.rer.soc.oec. Dipl.-Ing. Dr.techn. Dr.rer.soc.oec. Florian Skopik  
Dipl.-Ing. Dženan Hamzić

Vienna, February 9, 2026

---

Maximilian Ranzinger

---

Peter Knees

# Erklärung zur Verfassung der Arbeit

Maximilian Ranzinger, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 9. Februar 2026

---

Maximilian Ranzinger

# Danksagung

Ich möchte dem Austrian Institute of Technology (AIT) meinen aufrichtigen Dank aussprechen, dass mir die Möglichkeit geboten wurde, diese Masterarbeit im Rahmen eines Praktikums zu verfassen. Besonders danke ich Florian Skopik, der mir nicht nur diese Gelegenheit eröffnet hat, sondern mir auch erlaubt hat, die Themenfelder der künstlichen Intelligenz und der Cybersicherheit miteinander zu verbinden. Seine konstruktiven Anmerkungen zur inhaltlichen Ausrichtung und Umsetzung der Arbeit waren dabei von großem Wert.

Darüber hinaus danke ich Dženan Hamzić für die engagierte Betreuung sowie die kontinuierliche Unterstützung bei der konkreten Durchführung dieser Arbeit. Ebenso gilt mein Dank Max Landauer und Markus Wurzenberger für ihre hilfreichen Ratschläge sowie für die sorgfältige Durchsicht meiner Arbeit.

Mein besonderer Dank gilt außerdem meinem universitären Betreuer Peter Knees für seine wertvollen Vorschläge, Anregungen und Ideen.

Nicht zuletzt möchte ich meiner Familie danken, die mich während meines gesamten Studiums stets ermutigt und unterstützt hat.

# Acknowledgements

I would like to express my sincere gratitude to the Austrian Institute of Technology (AIT) for giving me the opportunity to write this master's thesis as part of an internship. In particular, I would like to thank Florian Skopik for not only enabling this opportunity, but also for allowing me to combine the fields of artificial intelligence and cybersecurity. His constructive feedback on the direction and implementation of this work was of great value.

Furthermore, I would like to thank Dženan Hamzić for his dedicated supervision and continuous support regarding the concrete execution of this work. I would also like to thank Max Landauer and Markus Wurzenberger for their valuable advice and for carefully reviewing my thesis.

In addition, I would like to thank my academic supervisor Peter Knees for his suggestions, comments and ideas.

Finally, I would like to thank my family, who supported and encouraged me throughout my entire studies.

# Kurzfassung

Cyber Threat Intelligence (CTI) ist ein essenzieller Teil der Cybersecurity. Sie umfasst Wissen, das Technologien und Individuen zur Abwehr von Cyberangriffen benötigen. Angesichts der großen Menge an täglich generierten CTI-Informationen, benötigen Cybersicherheits-Analysten automatisierte Werkzeuge, um diese Daten effizient zu verarbeiten, um aussagekräftige Erkenntnisse zu erlangen und davon Handlungsweisen abzuleiten. Eines solcher Werkzeuge ist Named Entity Recognition (NER). NER identifiziert relevante Entitäten in unstrukturiertem Text, um Folgeaufgaben, wie Suche, Filterung und Analyse zu erleichtern.

In dieser Arbeit werden Ansätze für NER auf CTI-Daten im Rahmen einer Systematic Literature Review (SLR) untersucht, die mithilfe eines in dieser Studie vorgeschlagenen LLM-basierten Extraktors durchgeführt wird, und im Hinblick auf ihre Architekturen analysiert. Darüber hinaus werden sechs Publikationen, die ihre Ansätze öffentlich bereitstellen, hinsichtlich ihrer Tagging-Performance sowie ihrer Trainings- und Betriebskosten evaluiert und miteinander verglichen. Um aussagekräftige Einblicke in die Tagging-Performance zu ermöglichen, wird diese aus vier Perspektiven analysiert: der Gesamtpformance, der Performance auf Entitätstypen, der Performance in Bezug auf Entitätsattribute (z.B. Entitätshäufigkeit) sowie der Fähigkeit, vollständige Dokumente, anstatt einzelne Sätze, zu taggen. Dies ermöglicht eine fundierte Entscheidungsfindung bei der Auswahl des am besten geeigneten NER-Ansatzes für das Tagging von CTI-Nachrichten.

Die Evaluation zeigt, dass einfachere Architekturen über alle analysierten Perspektiven hinweg konsistent bessere Ergebnisse erzielen als komplexere Alternativen. Encoder-basierte Language Models (LMs) wie BERT, kombiniert mit einer einfachen Decoder-Ebene (CRF oder Softmax), erreichen die höchste Performance. Entgegen den Erwartungen führt die Einbeziehung von Kontext auf Dokumentenebene nicht zu einer Leistungsverbesserung, sondern verschlechtert die Ergebnisse. Die Analyse zeigt ferner, dass in den meisten betrachteten Veröffentlichungen NER-Performancewerte künstlich erhöht sind, verursacht durch Entitätsüberlappungen zwischen Trainings- und Testdaten. Prompt-basierte Extraktoren, die auf dem allgemeinen Wissen großer Large Language Models (LLMs) beruhen, schneiden nicht nur schlechter ab als domänenspezifische Ansätze, sondern verursachen auch deutlich höhere Kosten, mit bis zu 50-fach langsamerer Inferenz und über 100-fach höherem Energieverbrauch, was sie für CTI-Anwendungen mit hohem Durchsatz ungeeignet macht.

# Abstract

Cyber Threat Intelligence (CTI) is an essential element of the cybersecurity domain. It encompasses knowledge that supports technologies and individuals in mitigating cyber attacks. Given the large volume of cybersecurity information generated daily, cybersecurity analysts require automated tools to efficiently process this data and extract meaningful insights and actionable outcomes. One such concept is Named Entity Recognition (NER), which identifies and labels relevant entities in text to improve searching, filtering and analysis.

In this work, approaches for NER on CTI data are examined through a Systematic Literature Review (SLR) enabled by an LLM-based extractor proposed in this study and analyzed with respect to their architectures. Furthermore, six publications that publicly provide their approaches are evaluated and compared based on tagging performance as well as training and operational costs. To provide meaningful insights into tagging performance, we analyze it from four perspectives: overall performance, performance on specific entity types, performance on specific entity attributes (e.g. entity frequency) and the capability to tag entire documents. This enables informed decision-making in selecting the most suitable NER approach for tagging CTI news.

The evaluation shows that simpler architectures consistently outperform more complex alternatives across all analyzed perspectives. Encoder-based language models (LMs) such as BERT, combined with non-complex decoding layers (CRF or Softmax), achieve the highest tagging performance. Contrary to expectations, the incorporation of document-level context does not improve performance, but degrades results. The analysis further indicates that NER evaluation performance scores are inflated due to data leakage, caused by entity overlap between training and test data. Prompt-based extractors that rely on the general knowledge of large language models (LLMs) not only underperform compared to domain-specific approaches but also incur substantially higher computational costs, with inference up to 50 times slower and over 100 times greater energy consumption, which renders them unsuitable for high-throughput CTI applications.

# Contents

<b>Kurzfassung</b>	<b>vi</b>
<b>Abstract</b>	<b>vii</b>
<b>Contents</b>	<b>viii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem Statement . . . . .	1
1.2 Research Questions . . . . .	2
1.3 Aim of the Work . . . . .	2
1.4 Thesis Structure . . . . .	3
<b>2 Background and Related Work</b>	<b>4</b>
2.1 Cyber Threat Intelligence (CTI) . . . . .	4
2.2 Named Entity Recognition (NER) . . . . .	5
2.3 Components of CTI-NER Approaches . . . . .	7
<b>3 Methodology</b>	<b>11</b>
3.1 Systematic Literature Review (SLR) . . . . .	11
3.2 Evaluation Framework . . . . .	15
<b>4 Results</b>	<b>23</b>
4.1 Systematic Literature Review (SLR) . . . . .	23
4.2 Evaluation . . . . .	40
<b>5 Discussion</b>	<b>59</b>
<b>6 Future Work</b>	<b>62</b>
<b>7 Conclusion</b>	<b>64</b>
<b>A LM-based Feature Extraction</b>	<b>66</b>
A.1 Evaluation . . . . .	80

<b>B High-level Architecture of NER Approaches that do not provide implementation Code</b>	<b>82</b>
<b>C Baseline Entity Type Definitions (Prompts)</b>	<b>88</b>
<b>Overview of Generative AI Tools Used</b>	<b>91</b>
<b>Übersicht verwendeter Hilfsmittel</b>	<b>92</b>
<b>List of Figures</b>	<b>93</b>
<b>List of Tables</b>	<b>95</b>
<b>Acronyms</b>	<b>96</b>
<b>Bibliography</b>	<b>98</b>

# Introduction

Cyber Threat Intelligence (CTI) is a key area within cybersecurity [Ins24; ISO25]. It involves the collection, analysis, enrichment, aggregation, summarization and extraction of information in the cybersecurity domain. This includes data about threat actors, their Tactic, Technique, Procedures (TTPs), vulnerabilities, updates and new products like firewalls. These informations often come in an unstructured format, such as news articles, log files, white papers, blog or social media posts. The volume of information has reached a level where it is impractical for analysts to complete CTI tasks manually.

The field of Natural Language Processing (NLP) aims to enhance, aggregate and summarize unstructured data while also extracting structured information. One of its tasks is Named Entity Recognition (NER), which involves the automated identification and classification of entities in unstructured text into predefined categories. These entity types may include threat actors, vendors, or vulnerability identifiers [GS96].

Tagging entire articles of unstructured text enables security analysts to quickly locate relevant information. However, manual tagging is a time-consuming and labor-intensive task that requires significant expertise, making it costly [AI23]. To address this challenge, NER methods are applied to CTI data to automate the process, improve efficiency and reduce costs. Effective tagging forms the foundation for crucial follow-up processes, such as generating timely security recommendations, automated alert notifications, detailed threat reporting and clustering of articles [Woi+25], as structured data can be more easily processed by computers compared to unstructured text.

## 1.1 Problem Statement

A major challenge in applying NER techniques is the wide range of methods that can be used in various combinations, each with varying effectiveness across different text corpora, entity types and languages. To ensure sufficient performance for the use case

of CTI, it is essential to systematically evaluate available models, architectures and techniques. Current State-of-the-art (SOTA) NER methods rely on Language Models (LMs), but these models require significant computational resources [KMN24]. To address this challenge, it is important to also evaluate approaches that incorporate distilled or smaller LMs, more efficient fine-tuning techniques, Knowledge Graphs (KGs), rule-based systems and traditional statistical methods.

Moreover, current models require costly retraining when entity types change or when new information needs to be understood. Therefore, in this work, they are compared against a prompt- and Large Language Model (LLM)-based NER baseline that incorporates world knowledge, does not require training and is, due to its prompt-based nature, easy to adapt to new entity types.

Furthermore, a systematic evaluation involves numerous challenges, including ensuring that models available even run and designing evaluation procedures that accurately reveal the aspects in which they perform well and those in which they do not.

## 1.2 Research Questions

The research questions that arise from the problems described in Section 1.1 are the following:

- RQ1:** To what extent does domain-specific adaptation, such as fine-tuning on CTI data, enhance the performance ( $F_1$  score) of NER models in the cybersecurity domain?
- RQ2:** To what extent do different NER model architectures (e.g. traditional statistical methods, rule-based systems, reasoning-based LLMs, larger context windows) differ in their effectiveness ( $F_1$  score) at extracting CTI-relevant entity types?
- RQ3:** What is the trade-off between cost and performance when training and deploying NER models in real-world CTI scenarios?

## 1.3 Aim of the Work

The primary objective of this thesis is to provide an evaluation framework that supports informed decision-making in selecting the most suitable NER approach for tagging CTI news.

This is achieved by answering the research questions defined in Section 1.2. First, approaches available in the literature are collected and analyzed in the context of a Systematic Literature Review (SLR). Publications that provide implementation code for their models are then examined in more detail to answer the research questions.

RQ1 is answered by analyzing the overall performance results of the models and comparing them with the baseline results on the same dataset. RQ2 is addressed by categorizing the

different approaches according to their characteristics and comparing their performance holistically. To relate performance to resource usage, training and inference time as well as energy consumption are measured, compared and analyzed to answer RQ3.

## 1.4 Thesis Structure

The remainder of this thesis is organized as follows: Chapter 2 presents background and related work in the domain of NER for cybersecurity. First, the terms CTI as well as NER are defined. This is followed by an overview of the historical evolution of NER approaches, with a focus on how the underlying design philosophy has changed over time. In addition, the choice of entity types that are most suitable for cybersecurity is discussed. Finally, the chapter introduces the core components addressed in this thesis, including Transformers, Recurrent Neural Networks (RNNs) and statistical learners such as Conditional Random Fields (CRFs) and explains their application in NER approaches for CTI.

Chapter 3 introduces the methodology used in this work, while Chapter 4 presents the results of the analysis. Both chapters are divided into two main parts: SLR and evaluation. Chapter 3 outlines how the SLR is conducted and how it is supported by an LLM-based SLR feature extractor to facilitate the process. In addition, it describes how the models are evaluated with respect to tagging performance as well as training and operational cost and it details the experimental setup. Chapter 4 first presents the results of the SLR and then explains which approaches are selected for evaluation and reports their evaluation results.

In Chapter 5, the results of the SLR and the evaluation are jointly interpreted and situated within the context of the research questions (see Section 1.2). This interpretation highlights the implications of the findings and discusses the limitations of the evaluation. Chapter 6 outlines how the limitations identified in Chapter 5 can be addressed and proposes directions for future research. Chapter 7 summarizes the approach of this work and its main insights.

# Background and Related Work

## 2.1 Cyber Threat Intelligence (CTI)

Cyber Threat Intelligence (CTI) is a subfield of cybersecurity and is defined as

Cyber threat information that has been aggregated, transformed, analyzed, interpreted, or enriched to provide the necessary context for decision-making processes [Joh+16].

by the U.S. National Institute of Standards and Technology (NIST). A cyber threat is further defined as

Any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information and/or denial of service. Also, the potential for a threat-source to successfully exploit a particular information system vulnerability [Nat06].

In simplified terms, this refers to the collection and preparation of information about how an entity can be compromised or shielded in order to enable technologies and individuals to prevent such incidents.

The sources of CTI are highly diverse and range from publicly available information to data obtained through espionage and surveillance. The latter is primarily used in governmental and military contexts. Both public institutions and private companies also strongly rely on publicly available information, commonly referred to as Open Source

Intelligence (OSINT). This includes information from news organizations, social media posts and public Indicators of Compromise (IoCs) databases, among others. Additional sources include information obtained through digital forensics as well as data collected from the deep and dark web [Dal14; Sae+23].

## 2.2 Named Entity Recognition (NER)

Named Entity Recognition (NER) is a task that focuses on extracting structured information from unstructured text by identifying predefined categories such as organizations, persons and locations. It was first explicitly mentioned in the Message Understanding Conference-7 (MUC-7) [CR98], while the initial appearance of the “Named Entity” task was traced back to MUC-6 [95].

NER primarily consists of three subtasks. First, the entity types to be extracted are defined. Next, these entities are recognized within the text. This recognition step is further divided into two subtasks. The second subtask involves identifying the span of text that contains an entity and the third subtask assigns an entity type to the identified span. The second and third subtasks are usually performed jointly [95; CR98].

### 2.2.1 Evolution of NER Approaches

In a recent comparative study, Keraghel et al [KMN24]. categorize the field of NER (see Figure 2.1). Knowledge-based methods rely on static rules that enforce predefined patterns and conditions to extract entities from text. Feature engineering-based methods aim to reduce the manual effort of rule construction by employing statistical techniques such as Hidden Markov Models (HMMs) and CRF. Deep learning-based methods encompass models built on Transformer, Convolutional Neural Network (CNN) and RNN architectures.

All the building blocks in Figure 2.1 can be combined or integrated into one another to create new models that potentially improve performance. Keraghel et al [KMN24]. find that LLMs based on Transformers perform well on datasets with broad knowledge. In contrast, the statistical method CRFs demonstrates strong performance in domains with highly specific vocabularies, such as the biomedical field.

The best-performing method across all datasets in [KMN24] was the GLiNER architecture, based on DeBERTa [Zar+23]. GLiNER utilizes bidirectional transformers (e.g. DeBERTa) to extract embeddings for entities and sentences and matches them, achieving SOTA performance. Hamzić [Ham25] released AITSecNER. A NER model training GLiNER on the AnnoCTR dataset [Lan+24] for CTI use.

Another relevant work is PromptNER by Ashok and Lipton [AL23], who explicitly study prompting techniques for NER. They find that naïve few-shot prompting of LLMs are far from SOTA in fully supervised NER, but by providing entity type definitions and asking the LLM to justify each extracted span, their approach achieved SOTA results

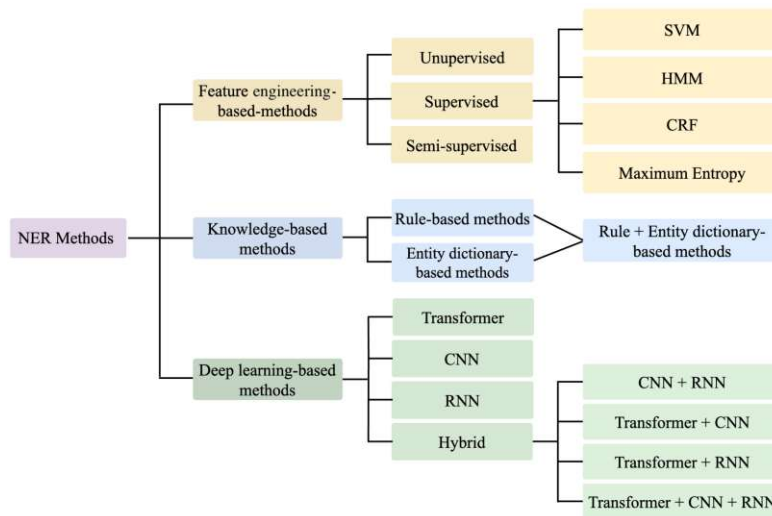


Figure 2.1: Main approaches to NER [KMN24]

in prompt-based few-shot NER. This indicates that with clever prompting strategies, LLMs can be extremely powerful for NER when large, labeled datasets are not available or entity types change frequently.

### 2.2.2 Entity Types in CTI

Cybersecurity is a highly diverse field with numerous application scenarios, including network event detection, the collection of IoCs and the identification of semantically relevant entities. As a result, the literature on CTI-focused NER (CTI-NER) exhibits variability in the definition and selection of entity types, based on the specific use case. Early work, such as Bridges et al. (2014) [Bri+14], concentrated on software- and vulnerability-centric entities, including vendor, product, version, programming language, vulnerability name and related terms. In contrast, Husari et al. (2017) [Hus+17] modeled hundreds of fine-grained threat actions mapped to tactics, techniques and kill-chain phases using the MITRE CAPEC ontology and further linked these actions to STIX 2.1 Domain Objects (SDOs) such as Malware, Vulnerability and Attack Pattern.

Other approaches derived entity schemas from other glossaries or reporting standards. For example, Dionisio et al. (2019) [Dio+19] relied on the ENISA risk management glossary, while Sarhan and Spruit (2021) [SS21] defined entities centered on IoCs and software metadata. Several publicly available datasets have also strongly influenced entity design. DNRTI [Wan+20b] introduces entities such as threat actor roles, tools, organizations, attack methods and exploits. APTNER [Wan+22b] expands coverage to authentication identities, even more fine-grained IoCs such as IP addresses, domains and URLs, cryptographic elements (MD5, SHA-1, SHA-2, etc.) and attack actions.

More recent work on CTI-NER assigns labels corresponding to Structured threat Infor-

mation Expression (version 2.1) (STIX 2.1) entity types, referred to as SDOs. Among others, this includes the identification of attack patterns, campaigns, intrusion sets, threat actors, malware, tools, indicators, vulnerabilities and courses of action, as demonstrated by Fujii et al. (2022) [Fuj+22; Fuj+23], Arora and Park (2023) [AP23], Marchiori et al. (2023) [MCV23] and Liu et al. (2025) [LLP25]. This trend toward standardization is primarily motivated by interoperability. The use of widely accepted entity types enables CTI-NER outputs to be directly integrated into information sharing and analysis platforms such as MISP<sup>1</sup>, thereby bridging the gap between unstructured threat reports and operational cyber defense workflows.

## 2.3 Components of CTI-NER Approaches

This section explains the most prominent technologies used in CTI-NER.

### 2.3.1 RNNs: LSTM and BiLSTM

A Long Short-Term Memory (LSTM) is a type of RNN designed to capture long-term dependencies in sequential data by maintaining a cell state (long-term memory) and a hidden state (short-term memory). Unlike standard RNNs, LSTMs use gating mechanisms (forget, input and output gates) to regulate the flow of information and mitigate the vanishing/exploding gradient problem during training. This allows the model to retain relevant context over long sequences, which is particularly useful in NER because understanding an entity often depends on context that can span many tokens [HS97].

An extension of this approach is to make the LSTM bidirectional. Here, two LSTM layers process the sequence in both the forward and backward directions and their outputs are combined, enabling the model to use both past and future context for each token, improving performance in tasks like NER, where context from both past and future tokens matters. These models are called Bidirectional Long Short-Term Memory (BiLSTM) [SP97].

### 2.3.2 Transformer

The Transformer architecture represents a paradigm shift in deep learning for NLP by relying on attention mechanisms rather than RNNs or CNNs. Traditional RNN process text sequentially, which limits their ability to capture long-range dependencies and makes parallelization difficult. In contrast, Transformers process all tokens in a sequence simultaneously, enabling the model to consider relationships between every pair of tokens across the entire input through self-attention. The core building blocks of a Transformer include Multi-Head Self-Attention (MHSA) layers and position-wise feed-forward networks, with positional encodings added to token embeddings to incorporate information about word order. These attention mechanisms allow the model to dynamically weight the

<sup>1</sup><https://www.misp-project.org/>

relevance of each token relative to others, capturing both local and distant contextual cues that are essential for language understanding [Vas+17].

A key innovation in the Transformer architecture is **MHSA**, in which the model performs multiple parallel attention operations, referred to as heads, on the same input. Each head projects the input into separate query, key and value spaces, allowing it to focus on different aspects of the sequence before these perspectives are combined into a richer representation. This increases the expressiveness of the model and its ability to better understand language. Beyond its use in Transformers, MHSA can for example also be applied independently for refining representations in NER. There they help the model assign greater weight to relevant contextual cues while suppressing irrelevant information, thereby learning which tokens are most informative for a given entity type [Vas+17].

### 2.3.3 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a language representation model designed to learn deep, context-aware text embeddings using only the Transformer encoder stack rather than a full encoder-decoder architecture. Original Transformer models consist of both encoder and decoder components, where the encoder maps input tokens into latent representations and the decoder uses these representations to generate output sequences. In contrast, BERT employs an encoder-only architecture focusing on learning representations of input sequences. As a result, BERT is not a generative model. It does not generate text like decoder-only models such as GPT or GPT-2 [Rad+18; Rad+19], but instead produces contextualized embeddings that can be used for downstream tasks [Dev+19].

During pre-training, BERT learns from large unlabeled text corpora using masked language modeling (MLM). In this objective, a subset of input tokens is randomly masked and the model is trained to predict these masked tokens based on their surrounding context. This process enables the encoder to integrate contextual information bidirectionally across the entire sequence [Dev+19].

Once pre-trained, BERT can be fine-tuned for specific downstream tasks by adding task-specific output layers or decoders. For example, a classification head can be applied to BERT's special sequence summary token for sentence-level classification and token-level classifiers with Softmax can be used for NER. In this two-stage fine-tuning paradigm, BERT is first pre-trained or further adapted using MLM and then fine-tuned on task-specific labeled data. The pre-trained encoder weights provide a strong foundation for language understanding and are subsequently optimized to improve performance on the target task [Dev+19].

### 2.3.4 Large Language Models (LLMs)

Large Language Models (LLMs) are a class of Artificial Intelligence (AI) models designed to process and generate natural language based on large-scale textual data. The concept

of LLMs is closely tied to the idea of pretraining on vast corpora, as reflected in reputable definitions:

A large language model (LLM) is a type of artificial intelligence (AI) model that has been trained through deep learning algorithms to recognize, generate, translate and/or summarize vast quantities of written human language and textual data [SAR23].

[Pretraining:] learning knowledge about language and the world from vast amounts of text. [We] call the resulting pretrained language models large language models [JM25].

The term LLM gained significant traction with the introduction of ChatGPT<sup>2</sup> (see Google Trends<sup>3</sup>). A defining characteristic of LLMs is their massive scale, both in terms of model parameters and the computational resources required for training and inference, as well as being based on the Transformer architecture [JM25].

### 2.3.5 Conditional Random Fields (CRFs)

Conditional Random Fields (CRFs) were developed to address limitations of earlier sequence labeling models, particularly Hidden Markov Models (HMMs) and Maximum Entropy Markov Models (MEMMs). HMMs are generative models that rely on strong assumptions. Each label depends only on the previous label (Markov assumption) and each observation is generated independently given the current label. These assumptions restrict the incorporation of rich and overlapping input features. Sequence labeling used for NER, however, is a discriminative problem in which the objective is to predict labels given an observed sequence. MEMMs move in this direction by modeling the conditional probability of the next label given the current label and the observations, which allows flexible feature representations. Their main drawback is local normalization [LMP01]. At each state, outgoing transition probabilities must sum to one, which forces probability mass to be distributed regardless of the evidence. This behavior leads to the label bias problem, in which states with fewer outgoing transitions are systematically preferred [LMP01; JM25].

CRFs address this issue by modeling the conditional probability of an entire label sequence given an input sequence and by normalizing globally over all possible label sequences. For example, in NER, the sentence “Barack Obama was born in Hawaii” is mapped to a label sequence such as [B-PER, I-PER, O, O, O, B-LOC]<sup>4</sup>. A CRF assigns a score to each complete labeling of the sentence and converts these scores into probabilities using a single normalization term over the whole sequence. This global normalization allows

<sup>2</sup><https://openai.com/de-DE/index/chatgpt/>

<sup>3</sup><https://trends.google.de/trends/explore?date=2016-01-01%202026-01-03&q=llm>

<sup>4</sup>PER and LOC are abbreviations for the entity types Person and Location. O means no entity type is applicable. The prefixes mark whether a token marks the start (B-) or is inside (-) an entity.

evidence from any position in the sequence to influence the final decision and eliminates the label bias problem. Training maximizes the conditional log-likelihood of the correct label sequences in a manner closely related to logistic regression but extended from individual labels to sequences. Inference is performed using the Viterbi algorithm, to efficiently identify the highest-scoring label sequence. Because training requires repeated sequence-level inference, CRFs are more computationally expensive than MEMMs, but they provide better performance on sequence labeling [LMP01; JM25].

### 2.3.6 word2vec

word2vec is an embedding generation model whose objective is to learn vector representations of words during training. Embeddings are numerical vectors that represent words or tokens, with words that appear in similar contexts mapped to nearby points in the vector space. Architecturally, word2vec uses a simple shallow Neural Network (NN) consisting of an input layer, a linear projection layer and an output layer, which makes training efficient. In the Continuous Bag-of-Words (CBOW) training variant, embeddings of surrounding context words are averaged or summed to predict a target word, whereas in the skip-gram variant, a single target word embedding is used to predict multiple surrounding context words independently. Training is self-supervised and relies solely on raw text. Observed word-context pairs serve as positive examples and randomly sampled word pairs serve as negative examples, with optimization typically performed using negative sampling or hierarchical softmax to improve efficiency [Mik+13; JM25].

### 2.3.7 GloVe

Like word2vec, GloVe (**Global Vectors for Word Representation**) is a model for generating word embeddings. Its main difference from word2vec lies in how contextual information is exploited during training. While word2vec learns embeddings by predicting words from local context windows, thereby capturing corpus statistics implicitly, GloVe constructs word representations by directly modeling global word-word co-occurrence statistics across the entire dataset. Specifically, GloVe builds a large co-occurrence matrix that counts how often pairs of words appear within a fixed context window and learns embeddings by minimizing a weighted least-squares objective that constrains the dot product of two word vectors, together with bias terms, to approximate the logarithm of their co-occurrence count. This formulation emphasizes ratios of co-occurrence probabilities, which allows the model to capture meaningful semantic relationships while down-weighting extremely frequent or rare word pairs. As a result, Glove produces an interpretable vector-space [PSM14].

# Methodology

## 3.1 Systematic Literature Review (SLR)

To identify relevant research in the field of NER in cybersecurity, a SLR, will be conducted in accordance with the guidelines presented in [Car+22; KC07]. The process will begin by determining which publications will be included in the review. Subsequently, a structured data extraction form will be developed to systematically collect relevant information from the selected papers.

### 3.1.1 Search Strategy

The search strategy defines the methods and sources that are used to identify relevant scientific publications. This includes the selection of databases, keywords and inclusion and exclusion criteria and a qualitative assessment.

The databases selected for the SLR include the ACM Digital Library and IEEE Xplore, due to their focus on engineering- and technology-related publications. In addition, ScienceDirect, Scopus, Google Scholar<sup>1</sup> and Web of Science are utilized to ensure broad coverage. Preprints from arXiv are also considered to capture the most recent advancements in this fast-moving field. As the use of multiple databases results in duplicate entries, the most recent version published in a journal or conference is selected.

These databases or libraries support advanced search queries, allowing users to concatenate keywords using Boolean operators and apply filters to specific fields. To capture all publications related to NER in the field of cybersecurity, the following keyword query is used:

---

<sup>1</sup>Due to the large number of results (over 40,000), only the Top 100 results were assessed.

(“Named Entity Recognition” OR “Entity Extraction” OR “NER”  
 OR “Information Extraction” OR “Entity Recognition”)  
 AND  
 (“Cybersecurity” OR “Cyber Threat Intelligence” OR “Threat Intelligence”  
 OR “CTI” OR “Cyber Security”)

(3.1)

The selection of keywords was an iterative process. It began with the query “Named Entity Recognition” AND “Cyber Threat Intelligence” and was progressively expanded to include synonyms and abbreviations, thereby capturing all relevant publications on CTI-NER. These additional terms were identified through an initial screening of papers retrieved using the original query, as well as a general search in Google Scholar.

In order to avoid selecting publications that just mention the topics associated with the keywords they are only searched for in the title, abstract and keywords of publications. To further scope the search in the right direction, the following inclusion and exclusion criteria (XC) are used:

- XC1 Period:** The SLR is not restricted to a specific timeframe to capture all NER approaches since its inception. The practical cutoff date for included publications is the 28<sup>th</sup> of April 2025.
- XC2 Language:** The SLR is restricted to publications that are written in the English language.
- XC3 Type of literature:** Primarily publications in conferences or journals. The SLR does not include book chapters, master theses, posters or surveys.
- XC4 Accessibility:** Only sources which are available in electronic form and accessible through the Vienna University of Technology library.

To the extent permitted by the advanced search functionalities of the publication databases, the inclusion and exclusion criteria are predefined within the search settings, thereby eliminating the need for manual filtering to an extent.

The next step in the selection process involves a qualitative assessment of the publications identified in the previous stages. Publications must meet the following quality criteria (QC):

- QC1** “There exists a more recent publication that presents the same or a similar study. The purpose of this criteria is to ensure that the most up-to-date versions of specific approaches are analyzed.” [Lan+23]

- QC2** Model architecture as well as benchmarking approach are clearly described.
- QC3** If models do not propose a new model or architecture, but only use existing approaches or use existing approaches in a bigger framework, these papers are not included in the SLR.
- QC4** The publication does not adopt approaches designed for other domains but focuses exclusively on CTI-specific NER (see Section 2.2 for definition) approaches. This also includes publications that focus on improving specific components of a NER approach, such as embedding construction.
- QC5** If the authors proposed an updated approach, the publication presenting the earlier version is excluded.
- QC6** The proposed approach is designed to support inputs in English language.

The criteria are initially assessed based on information contained in the title and abstract and the general approach is scanned. Publications that pass this initial screening are further evaluated based on their approach, including architectural and model-specific properties. Due to the volume of identified papers (see Section 4.1), the full analysis, using the data extraction form (see Section 3.1.2), is conducted only on publications that provide their implementation publicly. The remaining papers are assessed solely with respect to architecture-related features using an LM-based extraction pipeline (see Section 3.1.3).

### 3.1.2 Reviewed Features

For the later comparison of the models (see Section 4.1) it is important to extract the right information from the publications. The features of the publications are defined in Table A.1. The features used in this study were selected based on both architectural properties and evaluation criteria previously identified in a general NER study [KMN24].

The features are categorized into Approach/Architecture, Evaluation, Dataset, Code and Research Context. Features from Approach/Architecture are extracted for all publications. This category captures which architectural components are used, their intended purposes and how they are interconnected, thereby enabling the identification of currently employed architectures. Evaluation features encompass information on results, findings and evaluation methodologies presented in the publications. For the subsequent assessment of approaches (see Section 4.1) Code availability is critical, as it enables faster reproducibility and facilitates further evaluation without substantial manual effort. Finally, Research Context comprises insights derived from the publication.

### 3.1.3 LLM-based Feature Extraction

Due to the large number of papers to be evaluated, only the architecture-related features (see the green-highlighted rows in Table A.1) of approaches without publicly and readily

available implementations are extracted using an LLM-based retrieval pipeline. At a high level, the workflow begins with the normalization of the papers' PDFs using Mistral OCR<sup>2</sup>, which converts them into Markdown format. For retrieval, OpenAI's RAG cloud solution [Por25] is employed, which provides a vector store with automated vector and keyword search, chunk embedding, chunk reranking and free storage for this use case<sup>3</sup>. It also supports filtering, which makes it possible to restrict retrieval to chunks from a single paper at a time. Subsequently, the vector store is queried using validated prompts (see Section A.1) for each feature of each paper.

**Document normalization.** Mistral OCR is chosen because it represents the state of the art<sup>4</sup> and produces stable Markdown with a consistent heading structure formatting as well as reliable image extraction and description, thereby facilitating subsequent processing steps. Markdown is selected as the normalization target because it is both human-readable and easily interpretable by LLMs, enabling straightforward validation and chunking as well as requiring less storage (including images) compared to PDFs (83% reduction in storage required).

**Chunking.** The OpenAI vector store is populated with Markdown-formatted chunks enriched with metadata, enabling traceability to the original chunk content and its corresponding paper. Chunks are generated in a content-aware manner by splitting at the deepest header level (section-aware), which kept their size manageable and reduced the risk of the lost-in-the-middle problem [Liu+23], as no fixed-size chunking is applied. Header levels are included at the beginning of each chunk to indicate the section of origin, allowing both the LLM and human evaluators to assess the relevance of the retrieved content. To preserve context across boundaries, each chunk is augmented with adjacent sections from preceding and following chunks. This approach minimizes the risk of misleading cutoffs, which is primarily addressed through the section-aware chunking process and the reliable output of Mistral OCR. Consequently, only a small overlap of 64 tokens is chosen. Sections related work and references are excluded, as they do not describe the authors' own contributions [Sch23; Mic25].

**Retrieval.** For retrieval, each feature is queried for every paper. Each query employs an identical developer prompt combined with a feature-specific user prompt<sup>5</sup>.

The developer prompt (see Figure A.1) instructs the LM on its role and constraints. It defines the model as an NLP expert specialized in NER and as a cybersecurity domain expert. The prompt also specifies the expected input format and explicitly prohibits fabricating information. Evidence must be provided in the form of phrases extracted directly from the papers. The task is formulated as a one-shot instruction, requiring

<sup>2</sup><https://mistral.ai/news/mistral-ocr>

<sup>3</sup><https://platform.openai.com/docs/assistants/tools/file-search>

<sup>4</sup><https://mistral.ai/news/mistral-ocr>

<sup>5</sup><https://platform.openai.com/docs/guides/text>

the model to provide an immediate answer without requesting further clarification. The output must be formatted in Markdown and if information is unavailable, the model should return “Not Reported”. In addition, templates from the OpenAI GPT-5 Prompting Guide [KLZ25] are incorporated to support chunk search instructions and thorough question answering.

The user prompt consists of an extraction question and an accompanying guideline, primarily defining the output format and providing examples. The guideline ensures that the answer contains the relevant properties and information. Based on the initial validation, some system prompts were refined through metaprompting. For this process, OpenAI’s prompt template was used with ChatGPT-5-Thinking to improve prompts by addressing the issues identified therein (see Section A.1) [KLZ25]<sup>6</sup>.

Due to cost considerations, GPT-5-nano is employed as the primary model with a reasoning effort set to medium (options: minimal, low, medium, high). For the architecture-related features “Description,” “High-level architecture description” and “Entity types” GPT-5-mini with reasoning effort “medium” is used, as these represent the most complex extraction tasks.

## 3.2 Evaluation Framework

In this section, the strategy for comparing and benchmarking the models with respect to performance and cost is outlined. First, the criteria for model selection, the baseline used for comparison and the datasets on which the models are trained are defined. The experimental setup is then described, including the tools employed and their configuration, which are used to obtain the results.

### 3.2.1 Model Selection

Model candidates for evaluation are drawn from publications identified in the SLR that provide implementation code (see Section 4.1.3). Making the source code available does not guarantee that it functions out of the box due to factors such as bugs, missing documentation or reliance on deprecated dependencies.

To avoid substantial modifications to the source code to ensure reproducibility, the following exclusion process is proposed based on four reproducibility criteria (RC), which are applied in the following order:

**RC1** Incomplete implementation: Essential components missing from the published code

**RC2** Insufficient Documentation: Absent or minimal code documentation.

<sup>6</sup><https://platform.openai.com/docs/guides/prompt-engineering>

**RC3** Deprecated Dependencies: Unmaintained packages causing unresolvable dependency issues without substantial code rewriting

**RC4** Implementation Bugs: Critical bugs occur when running the original code with the original dataset and cannot be resolved without substantial code rewriting.

If an approach is excluded due to a RC, the subsequent RCs are not evaluated.

### 3.2.2 Baseline

As a baseline for comparison, Google’s LangExtract<sup>7</sup> [GK25] is selected. It is a prompt-based NER extraction framework that can be applied to cloud-based LLMs as well as local LLMs using Ollama<sup>8</sup>.

It is selected to compare the performance of LLMs with general world knowledge against models specifically designed for the CTI-NER domain. Furthermore, no publication identified in the SLR that provides implementation code utilizes LLMs, including distilled or reasoning-based LLMs with large context windows. This choice enables comparison with such models and allows RQ2 to be addressed more comprehensively. The prompt inputs are provided in Appendix C and follow the principles for effective NER prompting described by Ashok and Lipton [AL23], i.e. the provision of entity definitions and examples.

Furthermore, LangExtract is configured such that no chunking of large documents is performed and each document is extracted in a single pass. For the STIXnet dataset and its STIX 2.1-compatible entity types, the official descriptions from the STIX 2.1 documentation are adopted (see Appendix C). More granular indicator definitions are taken from Wikipedia. The DNRTI entity type descriptions are generated by an LLM, as the authors do not provide descriptions for their custom entity types [Wan+20b].

As LangExtract relies on examples (few-shot approach) as the primary source of entity type definitions, therefore a greedy algorithm that selects samples from the training set with the highest number of entity mentions until all entity types are covered is applied. This ensures that each entity type is covered at least once using a minimal number of examples, which helps prevent smaller LLMs from being overwhelmed by the input prompt, an issue observed when using a naïve approach with one example per entity type.

Due to compute resource constraints, Gemma3 with 27 billion parameters [Tea+25] is selected as a representative distilled LLM and DeepSeek-R1 with 32 billion parameters [Dee+25] is selected as a representative reasoning-based LLM. Gemma3 is developed by Google, the authors of LangExtract, which supports compatibility and the 27B variant represents the largest model that can be executed on 40 GB of VRAM. DeepSeek-R1

<sup>7</sup><https://github.com/google/langextract>

<sup>8</sup><https://ollama.com/>

was the first open-source model to incorporate reasoning capabilities and, unlike GPT-OSS:20B [Ope+25], does not produce malformed JSON outputs<sup>9</sup>, which are required by LangExtract.

### 3.2.3 Datasets

The ideal dataset for this comparison satisfies the following dataset criteria (DC). See Table 3.1 for which criteria the datasets satisfy.

- DC1** Large sample size (> 5,000 sentences)
- DC2** Ability to recombine sentences with their source documents
- DC3** Manually annotated or reviewed entities
- DC4** Data sourced from cybersecurity news reports or technical write-ups
- DC5** Public availability
- DC6** Established use within the scientific community
- DC7** SDOs as entity types

Because no single dataset satisfies all criteria, two datasets are used. The first is the STIXnet dataset [MCV23], which supports sentence-to-document backtracking. The second is the DNRTI dataset [Wan+20b], which comprises a substantially larger corpus. Both datasets are in the BIO format, in which each token is assigned a tag indicating whether it marks the beginning of an entity, is inside an entity or is not part of any entity, followed by the entity type. The first token of an entity receives the prefix “B-” for “begin” and all subsequent tokens receive the prefix “I-” for “inside”. Non-entity tokens receive the tag “O”. Other formats include BIOES or IO. In BIOES, the end of an entity is marked with the prefix “E-” and single-token entities receive the special prefix “S-”. In IO, entities and non-entities are tagged without special prefixes.

Dataset	DC1	DC2	DC3	DC4	DC5	DC6	DC7
DNRTI	✓		✓	✓	✓	✓	
STIXnet		✓	✓		✓		✓

Table 3.1: Coverage of dataset criteria of the datasets employed

#### STIXnet Dataset

The STIXnet dataset contains 52 threat reports, which enables analysis of model performance when using either single sentences or entire documents as input, thereby allowing investigation of the impact of cross-sentence context. In addition, it provides annotations for SDOs.

<sup>9</sup><https://github.com/google/langextract/issues/116>

Table 3.3 presents the key characteristics of the STIXnet dataset when used as a single-sentence dataset and as a whole-document dataset. Stratified splitting is applied to divide the dataset into training, validation and test sets, ensuring an even entity distribution across these splits and providing a realistic basis for evaluation on the test set.

A major limitation of the STIXnet dataset is its small size and uneven entity type distribution. Table 3.2 presents the distribution of entity mentions across the training, validation and test splits and reveals that some entity types are barely represented, with the file paths entity type appearing only once. Therefore, this dataset is used solely to analyze performance and cost differences between tagging sentences with and without document-level context.

Table 3.2: Entity mention distribution across entity types for the STIXnet single sentence (SSS) dataset

Entity Type	Train	Test	Valid
Attack Pattern	134	28	32
Campaign	5	1	1
Domain Dame	1	1	1
File Paths	1	0	0
Identity	67	22	15
Indicator	1	1	1
Intrusion Set	458	99	98
Location	115	23	18
Malware	49	11	12
SHA-256	1	1	0
Tactic	1	1	2
Threat Actor	1	1	0
Tool	116	21	19
URL	18	3	1
Vulnerability	9	2	2

### DNRTI Dataset

The DNRTI dataset is widely used in this domain, with 55 citations on IEEE and contains a large number of sentences and entity mentions (see Table 3.3). I adopt the authors' original data split to ensure comparability with prior work. Entity mention distributions are balanced across the splits. Although the DNRTI dataset does not annotate SDOs, it includes comparable entity types that can be mapped to SDOs [MS24], namely hacker organization, attack, sample file, security team, tool, time, purpose, area, industry, organization, way, loophole/exploit and features. The entity type distribution can be seen in Figure 4.9b.

Table 3.3: Characterizing statistics of the STIXnet dataset (single sentences and whole documents) and the DNRTI dataset

Dataset	STIXnet Single Sentence	STIXnet Whole Document	DNRTI
Total number of sentences (/documents)	910	52	6583
Average number of words per sentence	15.8	276.4	26.69
Total number of entity mentions	1396	1396	23404
Distinct entity mentions	447	447	5677
Number of entity types	15	15	13
Train/test/validation split	70/15/15	70/15/15	80/10/10

### 3.2.4 Performance Evaluation

For performance evaluation, the nervaluate framework<sup>10</sup> is used to compute performance scores. In addition, experiments adapted from [FLN20] are employed to obtain further insights into entity type-specific performance characteristics of the models.

nervaluate is a module for evaluating NER models as defined in the SemEval 2013 Task 9.1 [Dia13]. I select it because it enables analysis of model performance across different levels of tagging correctness. The four types are defined as follows:

- **Strict:** The entity type and span must exactly match the ground truth.
- **Exact:** The entity span must exactly match the ground truth, while the entity type is not considered.
- **Partial:** The entity span must overlap with the ground truth, while the entity type is not considered.
- **Entity type:** The entity type must be correct and the span must overlap with the ground truth.

This enables separate analysis of span detection and entity classification performance, as well as assessment of how precisely each model operates. In addition, it provides insights into performance at the entity type level.

The reported metrics are precision, recall and F1 score. Overall model performance across correctness types is summarized using the micro-averaged F1 score.

<sup>10</sup><https://github.com/MantisAI/nervaluate>

To further analyze the conditions under which a model performs well or poorly, Fu et al. [FLN20] propose categorizing entities according to specific attributes. The following categories are defined:

- **eLen – Entity Length:** Number of words or tokens in the entity.
- **sLen – Sentence Length:** Number of words in the sentence.
- **eDen – Entity Density:**  $\frac{\text{number of entities in the sentence}}{\text{sentence length}}$
- **oDen – out-of-vocabulary (OOV) Density:**  $\frac{\text{number of OOV words in the sentence}}{\text{sentence length}}$ , where an OOV word is any word that does not appear in the training vocabulary.
- **eFre – Entity Frequency:**  $\frac{\text{count}(\text{entity in training})}{\text{total entities in training}}$ , representing how often the exact entity appears in the training data.
- **tFre – Word Frequency:**  $\frac{1}{n} \sum_{i=1}^n \text{freq}(\text{word}_i)$ , denoting the average frequency of the entity’s constituent words in the training data.
- **eCon – Entity Consistency:**  $\frac{\text{count}(\text{entity with gold entity type})}{\text{count}(\text{entity})}$ , indicating how consistently an entity text is assigned the same label in training.
- **tCon – Word Consistency:**  $\frac{1}{n} \sum_{i=1}^n \frac{\text{count}(\text{word}_i \text{ with expected BIO label})}{\text{count}(\text{word}_i)}$ , measuring the average consistency of the entity’s constituent tokens with their expected BIO labels in the training data.

To enable comparison of model behavior across different conditions, the attribute values are discretized into buckets, which allows identification of entity buckets or categories for which a model performs better or worse. The buckets are defined as follows, based on the design proposed by Fu et al. [FLN20]:

- **Bucketing for eCon, tCon (Label Consistency)**
  - **XS:** value = 0 (never observed with this label in training)
  - **S:**  $0 < \text{value} < 0.5$  (inconsistently labeled)
  - **L:**  $0.5 \leq \text{value} < 1$  (mostly consistent)
  - **XL:** value = 1 (always assigned the same label in training)
- **Bucketing for eFre, tFre, oDen (Frequency and OOV)**
  - **XS:** value = 0 (never seen in training or no OOV words)
  - **S, L, XL:** non-zero values are divided into three equal-sized buckets based on entity count
- **Bucketing for sLen, eDen (Sentence Length and Entity Density)**
  - **XS:** lowest 25% of entity values

- **S**: next 25% of entity values
- **L**: next 25% of entity values
- **XL**: highest 25% of entity values
- **Bucketing for eLen (Entity Length)**
  - **XS**: length = 1 word (single-token entities)
  - **S**: length = 2 words
  - **L**: length = 3 words
  - **XL**: length  $\geq 4$  words (long entities)

InterpretEval’s bucketing approach consists of three phases: boundary determination, entity assignment and matching. First, bucket boundaries are established using only the gold standard entities.

For entity-level attributes (entity length, entity frequency, entity consistency) and token-level attributes (token frequency, token consistency), each entity is assigned to a bucket based on its own intrinsic value. A gold entity and its predicted counterpart may have different values and thus be assigned to different buckets. For example, if a gold entity “custom malware” has length 2 tokens (bucket 1) but the model incorrectly predicts two separate 1-token entities “custom” and “malware” (both in bucket 0), this reveals an entity-splitting. This failure pattern becomes visible in the bucket-level metrics: the gold entity appears as a false negative in bucket 1 (long entities), while the two fragments appear as false positives in bucket 0 (short entities), indicating systematic difficulties with multi-token entity recognition.

For sentence-level attributes (entity density, sentence length, OOV density), all entities in the same sentence must share the same attribute value, calculated using the gold standard as reference. This consistency is essential to ensure that correctly predicted entities are counted as true positives rather than being artificially separated into different buckets. For example, consider a sentence with 2 gold entities and 3 predicted entities (2 correct predictions and 1 false positive), with sentence length 20 tokens. If entity density were calculated separately – using 2 gold entities for gold ( $eDen = 2/20 = 0.10$ , bucket 0) but 3 predicted entities for predictions ( $eDen = 3/20 = 0.15$ , bucket 1) – the two correctly predicted entities would have different eDen values (0.10 vs 0.15) than their gold counterparts and be assigned to different buckets. Set intersection in bucket 0 would find no matches (gold present, predictions absent), counting the correct predictions as false negatives, while bucket 1 would find no matches in the opposite direction (predictions present, gold absent), counting them as false positives. By consistently using the gold entity count ( $eDen = 2/20 = 0.10$  for all entities in that sentence), both gold and predicted entities are assigned to bucket 0, where set intersection correctly identifies them as true positives.

Within each bucket, entities are matched using set intersection based on (sentence ID, start index, end index, label): TPs are entities appearing in both gold and predicted

sets, FPs are in predictions only and FNs are in gold only. For example, if bucket 1 contains gold = (sent 5, 10, 12, “malware”), (sent 7, 5, 7, “tool”) and predicted = (sent 5, 10, 12, “malware”), (sent 9, 3, 5, “tool”), then  $TP = 1$ ,  $FP = 1$ ,  $FN = 1$ , yielding  $F1 = 2 \times 1 / (2 \times 1 + 1 + 1) = 50\%$ . There are no TN in NER calculation, as they make semantically no sense.

### 3.2.5 Cost Evaluation

To quantify the cost of training and using the models, I rely on two metrics: time and energy consumption. Time is measured in seconds (s) and energy consumption is measured in watt-hours (Wh). To obtain these measurements, I employ the Python package `codecarbon`<sup>11</sup>, which allows functions to be wrapped for automatic measurement of execution time and power consumption.

The measurements are then separated into training and inference costs, enabling comparison of the cost of model development and model usage. If, due to implementation constraints or model design, separate measurements for training and inference cannot be obtained, the cost metrics are partitioned according to the number of sentences. This is feasible because the datasets are evenly distributed across splits (see Section 3.2.3), allowing this proportional cost allocation.

### 3.2.6 Experimental Setup

To train and evaluate the approaches, Google Colaboratory<sup>12</sup> (Colab) is used. Within this environment, two runtime configurations are employed:

- **Runtime environment 1: CPU-only**
  - Hardware accelerator: CPU
  - High-RAM: Yes
  - Runtime type: Python
  - Runtime version: 2025.10
- **Runtime environment 2: GPU**
  - Hardware accelerator: NVIDIA A100 GPU (40 GB GPU RAM)
  - High-RAM: No
  - Runtime type: Python
  - Runtime version: 2025.10

The models are executed within the software environments specified by the original authors, which results in different Python and package versions across models in order to ensure reproducibility.

<sup>11</sup><https://codecarbon.io/>

<sup>12</sup><https://colab.research.google.com>

# Results

## 4.1 Systematic Literature Review (SLR)

This chapter presents the results of the SLR. The discussion begins with the selection and exclusion of papers according to the criteria in Section 3.1.1, following a two-step qualitative analysis. The first step evaluates all papers based on their approach and provides a holistic categorization and overview of the identified approaches, while the second provides a detailed assessment of those that include implementation code.

### 4.1.1 Selection and Exclusion

Figure 4.1 illustrates the exclusion process of the papers. Initially, all papers were collected from academic databases. Due to overlapping search results, deduplication was performed as the first step to reduce the number of papers without omitting relevant information.

The largest number of exclusions was due to QC4, as shown by the “Not Relevant” category in Figure 4.1. This was followed by the exclusion of publications that employed only existing approaches (QC3, “No New Model”), such as their integration into holistic frameworks. Additional deduplication was performed by removing preprints of papers that had already been published (“Published Preprint Removal”).

Several items were excluded based on XC3, including theses (“Theses removed”), a poster (“Excluded Literature Type”) and study or survey papers (“Survey or Study”). It is important to note that the academic database filters were configured to exclude book chapters, so these do not appear as a separate category in Figure 4.1. The same applies to the language filter, which limited the selection to papers written in English language (XC2). Furthermore, five papers were inaccessible (XC4, “Not Accessible”) and one paper was retracted (XC4, “Paper retracted”).

Eleven papers were excluded because the authors proposed updated approaches (QC5, “Deprecated”) or the publications did not meet quality standards (QC1, “Badly Written”). The latter category primarily included papers with poorly designed and unreadable graphics and tables, such as screenshots of command-line tables.

This resulted in 101 papers to be assessed for their architectures and 12 papers with publicly available implementation code to be examined in greater detail.

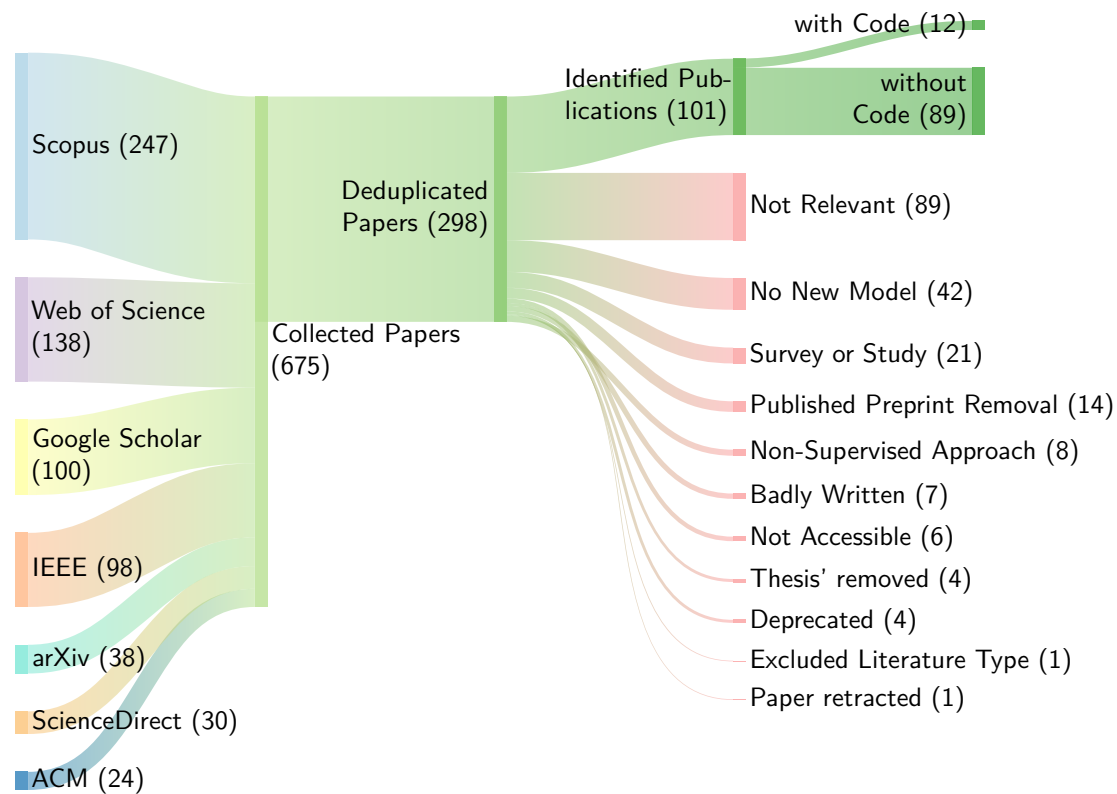


Figure 4.1: Sankey diagram of the paper selection process for the SLR. Starting from multiple academic databases, duplicate entries were removed and papers were filtered out according to the criteria defined in Section 3.1.1, resulting in a final set of publications classified by code availability.

#### 4.1.2 Architectural Overview

Figure 4.2 shows the temporal distribution of published publications. The number of publications has increased exponentially over time, with the first instance of CTI-NER reported in 2013. The apparent decline in 2025 relative to 2024 is due to the inclusion of only those publications available up to April. Over time, a clear shift is observed from approaches based on static word embeddings and statistical learners to those employing LM-based embeddings refined through Deep Learning (DL) components (see Figure 4.3).

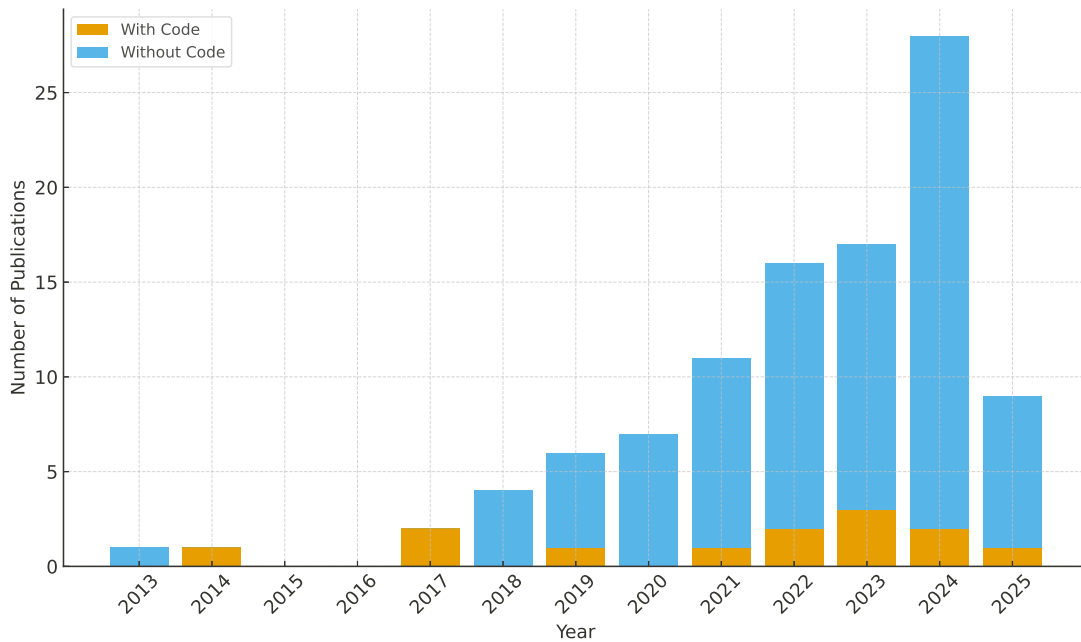


Figure 4.2: Distribution of publications per year

While the introduction of BERT in 2018 [Dev+19] and the release of ChatGPT in late 2022 mark significant milestones in NLP, they do not directly correspond to an immediate surge in NER publications, despite the steady increase in the use of LMs for NER since 2019 (see Figure 4.3).

All identified approaches support the English language, with nine also supporting Chinese. The approach by Sarhan and Spruit (2021) [SS21] additionally supports Spanish and Portuguese. However, it is important to note that language support depends on the languages covered by the pre-trained LMs used, as well as the languages in which the NER datasets are provided.

All approaches are classified into the Embedding → Encode → Decode pattern. Input sequences are encoded into numerical representations (embedding vectors) which get contextualized or merged in the encoding stage. In the Decode phase entities get extracted through some kind of classifier (e.g. CRF) sometimes accompanied by a rule-based extraction component that utilizes lists or regular expressions [Li+24b]. The boundaries between these categories are sometimes ambiguous and may not be applicable to certain architectures, such as when prompt engineering is used with a LLM.

**Method Families.** The NER architectures consist of components or elements that can be assigned into five categories: (i) Language Models (LMs), (ii) Deep Learning (DL), (iii) Statistical Learners, (iv) Rules and (v) Reinforcement Learning (RL). The architectures can include components from multiple categories. LM includes language

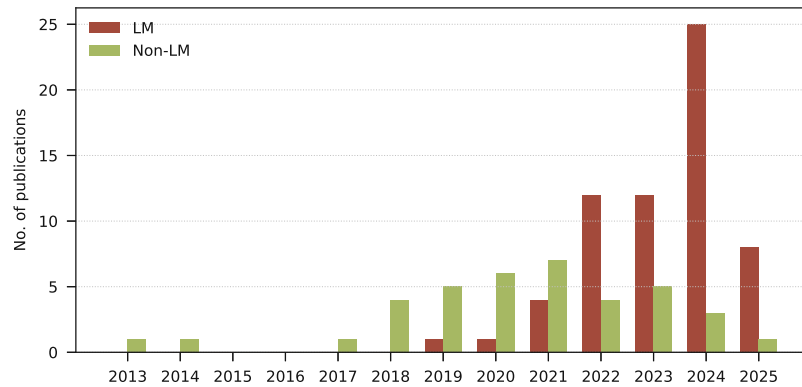


Figure 4.3: Temporal distribution of publications discriminated by their usage of LMs.

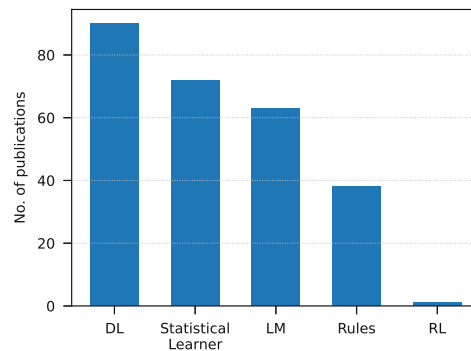


Figure 4.4: Number of publications that use each method family. A single publication may contribute to multiple categories if it employed more than one method family.

models from BERT onwards, excluding pre-transformer models and static word embeddings. DL includes NNs (e.g. Feed Forward Neural Networks (FFNNs)), CNNs and RNNs (e.g. BiLSTM). Statistical Learners incorporate traditional statistical models like CRF. Rules cover mechanisms such as regular expression matching, pattern matching and dictionary-based identification. Figure 4.4 shows that nearly all approaches include some form of DL, followed by Statistical Learners, which is mainly due to the use of CRF-based entity decoding. Fewer than half of the approaches use Rules for entity extraction. Some entity types, such as IoCs, can be effectively identified using Rules, but not all approaches extract the same entity types, which limits the applicability of this method. The only approach using RL was proposed by Xie et al. (2024) [Xie+24], employing deep RL to continuously correct and update entity labels.

**Embeddings, Encode, Decode pattern.** The NER extraction pipeline consists of the following steps: Embeddings → Encode → Decode. Input sequences are encoded into numerical representations (embedding vectors) which get contextualized or merged

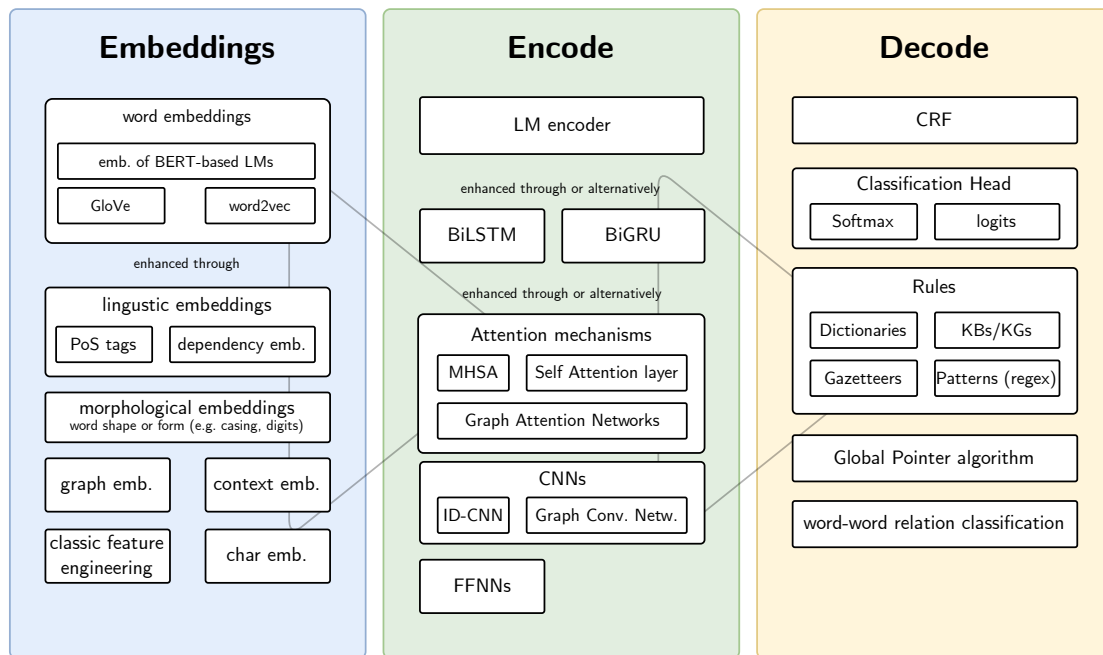


Figure 4.5: Overview of components used in NER architectures, categorized by their roles in embedding, encoding and decoding.

in the encoding stage. In the Decode phase entities get extracted through some kind of classifier (e.g. CRF) sometimes accompanied by a rule-based extraction component that utilizes lists or regular expressions [Li+24b]. The boundaries between these categories are sometimes ambiguous and may not be applicable to certain architectures, such as when prompt engineering is used with a LLM. An overview of all architectures can be found in the Tables 4.2 and B.1. Figure 4.5 gives an overview of the components used in the construction of NER architectures.

**Embeddings.** The transformation of text into embeddings can be performed in various ways depending on the desired granularity. The most straightforward and commonly used type is word embeddings<sup>1</sup>. The most frequently used sources of word embeddings were BERT-based models, followed by the static embedding algorithms word2vec (see Section 2.3.6) and GloVe (see Section 2.3.7). In most cases, pre-trained embeddings from these static algorithms were not used. Instead, the authors trained the algorithms on their own text corpora to address the OOV problem and obtain more representative embeddings.

Although sub-word tokenization inherently addresses the OOV problem during inference [Li+24b], approximately one fifth of approaches additionally employed character

<sup>1</sup>Due to ambiguity in the publications regarding whether tokenization was performed at the word or sub-word level, these approaches were grouped under the category of word embeddings.

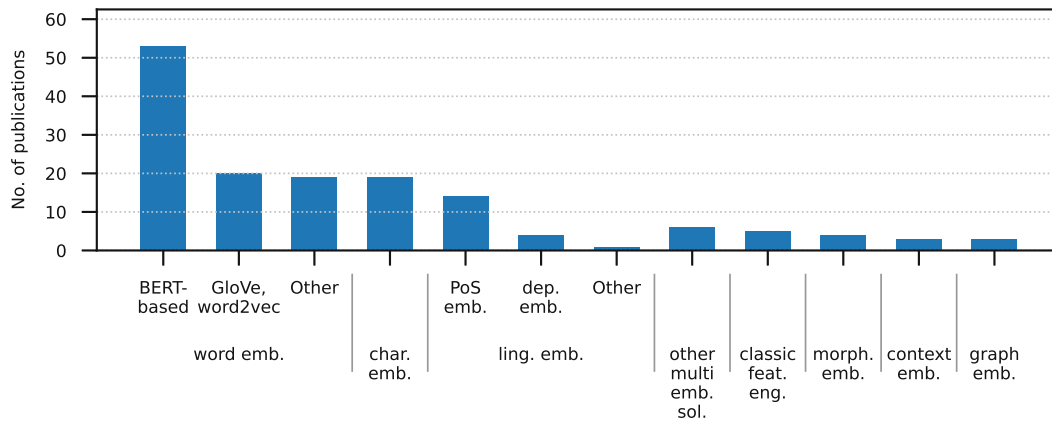
embeddings to better capture domain-specific patterns, such as file paths or version numbers. Because individual embeddings for sparse or rare words are often not sufficiently expressive, they are sometimes combined with embeddings of semantically or contextually similar words (referred to as contextual embeddings). For example, Ma et al. (2018) [Ma+18] applied Brown hierarchical clustering, while Chen et al. (2025) [Che+25] proposed a self-attention-based algorithm to identify similar words.

To facilitate the identification of embeddings and improve entity recognition, linguistic features such as Part-of-Speech (PoS) tags (e.g., noun, verb) and dependency roles (e.g., subject, object) were often incorporated using feature templates. These features were either used to identify potential entities or included as additional embedding inputs to help models learn contextual patterns – such as malware names typically being nouns. Some approaches extended this further by integrating morphological features such as casing and numerical structure (morphological embeddings). The distribution of embedding types across all publications is shown in Figure 4.6a.

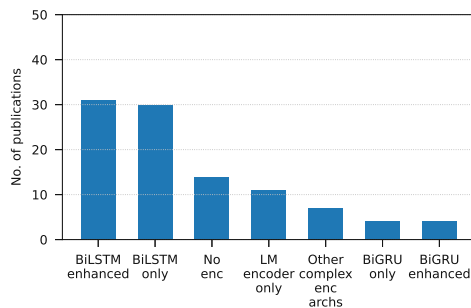
**Encode.** The embeddings are then fed into the encoder, enabling the model to capture intra-sequence context. More than half of the approaches used a BiLSTM (see Section 2.3.1). In approximately half of these cases, the input or output of the BiLSTM was further refined using additional components such as MHSA, CNNs, or FFNNs (see Figure 4.6b).

Encoding context does not always require a separate encoder. In LM-based NER approaches, the built-in encoder component of Transformer architectures (see Section 2.3.2) is sometimes used exclusively, with a classification head trained on top to extract entities. Other architectures rely on plain FFNNs or variants of CNNs and in some cases, multiple encoders are applied to encode embeddings in different ways. Chen et al. (2021) [Che+21] employed Iterated Dilated CNNs (ID-CNNs) instead of RNNs to increase processing speed. Fang et al. (2021) [FZH21] used Graph Convolutional Networks (GCNs) to incorporate cross-sentence context. Wang et al. (2024) [Wan+24a] were the only ones to integrate visual information, aligning image regions with entity mentions in text to enrich the representations.

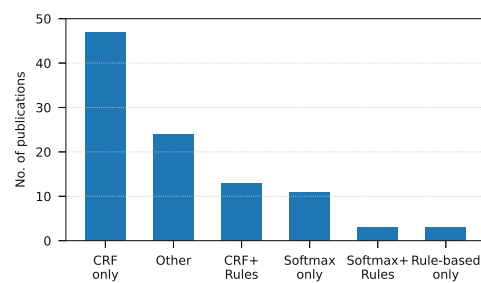
**Decode.** The most commonly used decoder is the CRF (see Figure 4.6c; more on CRF see Section 2.3.5) closely followed by Softmax-based decoders. Both decoder types are sometimes enhanced with rule-based components to improve performance. Several publications illustrate the diversity of decoding strategies. Li et al. (2021) [Wan+21] applied adversarial training to improve NER performance on small datasets and used an LSTM followed by a Softmax layer for decoding, arguing that this configuration is faster than using a CRF. Chaleshtori and Ray (2023) [HR23] combined the outputs of three Transformer-based NER taggers and determined the final label through majority voting. Other notable alternatives to CRF and Softmax-based decoders include the GlobalPointer algorithm [Su+22a] and W2NER [Li+21a].



(a) Embeddings



(b) Encoder



(c) Decoder

Figure 4.6: Types of architectures for each NER pipeline step. A single publication may employ multiple embedding types. For encoding and decoding, each publication was assigned to a single type.

**Type of Task.** While NER is most commonly approached as a sequence labeling task – where each token in a sequence is assigned an entity tag using classifiers such as CRF or Softmax – alternative formulations have also been proposed. One notable direction involves span classification, in which entity spans are first detected and then classified by type. Beyond this, researchers have explored reformulating the task to better align with pretraining objectives or to leverage different model capabilities. You et al. (2024) [You+24] modeled NER as a masked word prediction task, following the training paradigm of BERT. Arora and Park (2023) [AP23] and Faruk (2024) [Far24] framed it as an extractive question answering problem, while others [FAR24; Hu+24b] used generative LLMs to produce entity mentions directly from text.

**Enhancements.** The most common type of enhancement occurs at the embedding step. Approximately one third of the publications use more than one embedding method

to represent the input (see Figure 4.7). This is closely followed by rule-based methods, which are used to either tag or correct entities and by attention mechanisms designed to help the model focus on the most informative parts of a sequence.

The most frequently observed architecture combines a BiLSTM encoder with a BiLSTM decoder, fed by BERT-based, GloVe and/or word2vec embeddings. Another notable enhancement involves replacing the standard cross-entropy loss with focal loss to address the class imbalance inherent in NER tasks. This imbalance arises because the majority of tokens belong to the non-entity O class, leading models trained with cross-entropy to favor predicting O in order to maximize overall accuracy. This increases false negatives for actual entities [CLH23]. Wang et al. (2020) [Wan+20a] proposed Triplet Sorted Focal Loss (TSFL) as an improved focal loss function for NER.

**Language Models.** As shown in Figure 4.8, the majority of approaches used BERT [Dev+19] and its derivatives to embed and encode sequences. One key factor contributing to the widespread adoption of BERT is its non-generative nature. Unlike generative LLMs, BERT does not hallucinate, as it operates as a masked-token encoder that predicts or selects from existing input tokens rather than generating new content [Pal+24]. Another reason for its popularity is the availability of numerous BERT-based models pre-trained on cybersecurity-specific corpora, which offer improved representations of domain-relevant terms.

The identified approaches employed models such as SecureBERT [Agh+22], SecureBERT+<sup>2</sup>, SecBERT [Lib22], CyBERT [Ran+21], CySecBERT [Bay+22], DarkBERT [Jin+23] and SecRoBERTa<sup>3</sup>. These models were further fine-tuned on the NER task. In some cases, only the embeddings were used while the encoder layers remained frozen. Fine-tuning typically involved updating all layers of the LM. However, Chan et al. (2022) [Cha+22] reported fine-tuning only the final layer of BERT, aiming to evaluate the inherent capabilities of the underlying model while also reducing computational cost.

Fieblinger et al. (2024) [FAR24] were the only ones to apply a more advanced partial fine-tuning technique, using QLoRA [Det+23] to fine-tune LLaMA-2-7B-Chat. However, this model did not outperform the non fine-tuned and larger LLaMA-2-70B-chat. Across all analyzed approaches, none employed LLMs with reasoning capabilities and only two publications reported their best results using a LLM [Hu+24b; FAR24].

The predominant use of BERT-based LM imposes a hard sequence length limit of 512 tokens, restricting their ability to tag longer sequences. While models such as GPT-4 (8,192 tokens) and SciBERT (1,536 tokens) support longer contexts, these lengths are still insufficient for recognizing entities across multi-page documents. As a result, all identified approaches process either single sentences or very short reports or chunks at a time.

<sup>2</sup><https://github.com/ehsanaghaei/SecureBERT-plus>

<sup>3</sup><https://huggingface.co/jackaduma/SecRoBERTa>

Initially, this constraint was likely due to limited computational resources. Currently, the main limiting factors appear to be the sequence length restrictions of BERT-based models and the structure of widely used datasets (DNRTI [Wan+20b], APTNER [Wan+22b], MalwareTextDB [Lim+17]) which consist only of isolated sentences without a way to trace them back to full articles.

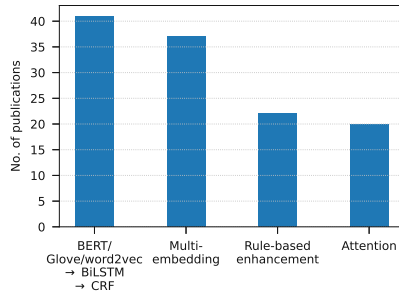


Figure 4.7: Most common enhancement types in NER architectures. Publications use more than one enhancement type.

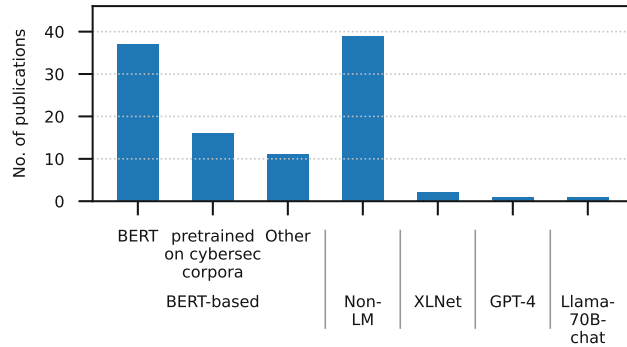


Figure 4.8: Usage of different LMs across the identified publications

**Knowledge Base and Graph.** Out of the 101 publications, 41 also performed relation extraction between entities, making them capable of constructing **KGs**. Among these, 23 explicitly generated a KG. Five publications employed Graph Attention Networks or GCNs, thus using graphs as input embeddings.

In contrast, the pure NER approaches focused on constructing **Knowledge Bases (KBs)**, including ontologies, dictionaries and gazetteers. Fourteen papers used such resources to support entity extraction and four incorporated them directly as embeddings.

### 4.1.3 Publications with implementation code

The 12 papers that provided implementation code for their NER approaches are analyzed with respect to architecture and evaluation insights. For a detailed evaluation of NER extraction performance of a subset of these publications, see Section 4.2.

One of the first CTI-NER publications is Bridges et al. [Bri+14], who applied “classic” feature engineering, by creating embeddings for individual words using PoS tags, BIO tags and morphological or surface features (e.g., word in full upper or lower case) for the current and surrounding words in unigram and bigram fashion. These features were fed into two history-based Maximum Entropy Models (MEMs) with averaged perceptrons. The first handled BIO tag classification and the second entity classification. Extraction was performed using a greedy algorithm rather than the Viterbi algorithm or CRF, prioritizing speed while maintaining a good F1 score, with 750,000 labeled words trained in under 17 seconds. To enhance extraction, a gazetteer for software products and vendors

was incorporated. Their analysis showed that in unconstrained settings gazetteers achieve poor performance, which is why they updated their gazetteer during training.

Another two-staged approach is Split-NER by Arora and Park (2023) [AP23], which separates the NER task into span detection and span classification. For each classification task, BERT was fine-tuned and then applied in a Question–Answer approach, where the embeddings of the answer were used. This doubly supervised end-to-end fine-tuning approach with a classification head was shown to outperform sequence tagging approaches that rely only on BERT’s word embeddings, single QA approaches performing span detection and classification jointly and even sequence tagging for span detection combined with a QA approach for span classification. Single QA models were found to bias toward frequent entity types, leading to misclassification of rare mentions. The Split-NER (QA-QA) approach achieved faster training but slower inference compared to sequence tagging. To improve performance on entity types with few samples, dice loss was applied during training. The performance of this approach was noted to be upper-bounded by the span detection task, which is currently not optimal. Incorporating character embeddings for OOV handling was further observed to improve the identification of domain-specific terminology.

TTPDrill developed by Husari et al. [Hus+17] is an approach that identifies 392 TTPs, maps them into STIX 2.1 objects for sharing and organizes them into kill chain phases using a hierarchical ontology. It leverages the NLP library Stanza (formerly Stanford NLP) [Qi+20] for PoS extraction, which is then used by the Stanza Dependency Parser to extract a predefined set of dependencies. These dependencies are subject-verb-object (SVO) triples that serve as candidate entities. The candidates are enriched through synonym expansion using WordNet, a thesaurus and the IBM Watson Synonym Service. The authors note that in some cases this expansion is insufficient for cybersecurity terms (e.g., “Internet Explorer” and “IE” are not recognized as synonyms). Using BOW-based TF-IDF with BM25, candidate actions are used as queries to retrieve TTPs encoded with their MITRE descriptions. This approach requires essentially no training, since Stanza provides pretrained models and TF-IDF only requires matrix construction. It is also highly efficient, labeling a report with 163 words and 47 candidate actions in under 700 ms. The authors also tested combining multiple sentences for candidate action extraction when the similarity score increased, but recall decreased with this method.

Dionisio et al. [Dio+19] identified X (formerly Twitter) as a valuable source of CTI information. They trained word2vec embeddings and concatenated them with character embeddings generated through a BiLSTM to embed OOV words and handle informal language more effectively. In a comparison of word2vec, GloVe and randomly initialized embeddings, the first two outperformed the latter, but only by a non-significant margin.

To create their KG Open-CyKG, Sarhan and Spruit [SS21] employed a NER model based on a Bidirectional Gated Recurrent Unit (BiGRU) RNN initialized with random embeddings for contextualization. This approach outperformed that of Bridges et al. [Bri+14] on the Microsoft Bulletin dataset [Bri+14] and achieved better performance than a BiLSTM under otherwise identical conditions. The authors argue that the BiGRU

performs better than the BiLSTM because Gated Recurrent Units (GRUs) expose the complete memory state, whereas LSTMs use gates that restrict information flow. To evaluate performance, stratified cross-validation was used to address class imbalances.

The CyNER Python library developed by Fujii et al. [Fuj+22; Fuj+23] uses pre-trained (among some pre-trained by the authors) and fine-tuned embeddings of LMs available through Huggingface, with a softmax classifier head to extract STIX 2.1 entity types. In their experiments, BERT was used as a baseline, RoBERTa as a larger LM and ALBERT as a smaller and faster LM. The results showed that RoBERTa outperformed the other models in terms of F1 score and that CTI-specific pre-training and fine-tuning resulted in better performance than models pre-trained on generic corpora. The LM-based extraction was extended with a regular expression-based IoCs extractor, which also applied regex for refanging defanged IoCs (e.g., transforming “hxxp” back to “http”). The authors note that refanging only helps to a limited extent, since it is not feasible to write comprehensive rules for all defanging methods. Their analysis further showed that using hashes for blocking is not suitable, as they are typically reported for only 1–2 days, whereas IPs and URLs may remain valid for months or even years.

Liu et al. (NER4CTI) [Liu+22] implemented a sophisticated approach utilizing six different types of embeddings, four of which were contextualized using a BiLSTM and refined with MHSA (see Section 2.3.2) to filter the most relevant features. These four embeddings included static GloVe word embeddings, PoS embeddings, character embeddings to handle OOV words and component embeddings to capture surface features (e.g., digits and casing). The resulting mixed feature embedding was further augmented through a gate fusion mechanism with semantically similar tokens from a CTI-specific corpus (Internal Semantic Augmentation (ISA) – Hard Semantic Augmentation (HSA)), which helps enrich rare entities by linking them to semantically similar domain words and fine-tuned BERT embeddings (External Semantic Augmentation (ESA)) leveraging transfer learning from large-scale corpora to provide global knowledge context. For HSA, similar words were identified using word2vec CBOV embeddings trained by the authors, with the k-nearest neighbor (KNN) algorithm and cosine similarity applied to select the most similar words, which were then weighted by similarity. The authors also proposed Soft Semantic Augmentation (SSA), which replaces cosine similarity-based weighting with attention-based relevance weighting. This approach achieved slightly better performance but was not statistically significant and required 11 times longer training, which is why it was not included in the final model. The model performed better on DNRTI [Wan+20b] compared to MalwareTextDB [Lim+17], which the authors attributed to the sparse nature of MalwareTextDB.

Vul-PAG by Li et al. [Li+24b] also follows the multiple embedding-BiLSTM-CRF approach and employs BERT embeddings in three ways. Both wordpiece (subword-level) and full-word tokenizers with BERT were used to create writing, semantic and syntactic features. The semantic feature corresponds to the last layer of coarse-grained BERT. The last layer is commonly referred to as BERT embeddings in the literature. These were fused with the other two embeddings through a linear combination. The writing

feature based on wordpiece BERT was introduced to handle OOV words. Although less granular than character embeddings, this was presented as an advantage, since similar OOV words produce similar embeddings, which is not necessarily the case for character embeddings. To merge subword tokens with full words, the authors introduced “subword reorganization”, giving greater weight to later subword embeddings, as these contain the most distinctive differences. Their analysis showed that OOV words in cybersecurity texts predominantly consist of file names and version identifiers, where suffixes (e.g., “.exe”) carry crucial information. Since the data extraction was performed on NVD/CVE descriptions, two templates were used to identify structural regularities in these texts. Logistic regression experiments indicated that BERT layer 8 embedded syntactic information most effectively. A grid search for the optimal linear combination of embeddings showed that the writing embedding received a weight of 0.9, while the others were weighted at 0.05, indicating that subword-level embeddings were the primary driver of extraction performance. The syntactic feature was found to be helpful mainly in settings where the text followed specific templates, as entities identifiable through templating achieved a performance boost.

Contrary to the sophisticated approaches described above ([Liu+22; Li+24b]), Chen et al. [Che+23] proposed a simplified method that only leveraged BERT embeddings, processed through an fully connected (FC) layer for a Softmax classifier, intentionally omitting sequence contextualization with a BiLSTM. Compared to a BERT-BiLSTM-CRF approach, BERT-CRF showed slightly better performance in terms of F1 score, accuracy and recall. Similar to Fujii et al. [Fuj+22] they found that fine-tuning improves performance and fine-tuning the LM on multiple datasets further enhances it. Surprisingly, BERT-CRF outperformed SecBERT-CRF on the DNRTI dataset but not on the MalwareTextDB dataset, indicating that the fine-tuning corpora and the pre-trained cybersecurity-specific embeddings of secBERT interacted less effectively than the general-purpose BERT embeddings. The same observation held for the architecture with an additional contextual BiLSTM encoder. Chen et al. [Che+23] further evaluated their approach on real-world OSINT data, where BERT-CRF outperformed GPT-3.5 and DistilBERT, demonstrating that models trained on publicly available datasets can generalize well to real-world settings. From these results, it can be inferred that BiLSTM are not essential for achieving strong NER performance, although they appear to improve recall on the MalwareTextDB dataset.

With STIXnet, Marchiori et al. [MCV23] proposed an approach capable of extracting 14 out of the 18 STIX 2.1 objects. The excluded objects were grouping, note, observed data and opinion. No explicit explanation was provided for this exclusion, but it is likely because these represent meta fields that cannot be sensibly mapped to entities within text. Despite being introduced in 2023, the approach does not employ LMs for entity extraction, as they were regarded by the study’s creators as slow to adapt and effective only for a limited set of entity types, while they also acknowledged that their own novel entity extractor was not sufficiently sophisticated for more complex types such as tools and campaigns, where low recall was observed. Their approach is carried out through four

distinct modules, whose outputs are subsequently merged and conflicts resolved. IOC Finder [Flo25] was used for regular expression-based IoCs extraction, rcATT [Leg+20] for TTP extraction via a binary Support Vector classifier (SVC) for each TTP with TF-IDF-weighted Bag-of-words (BOW) features and a manually curated KB for entity extraction validated through handcrafted PoS entity type mappings (spans assigned to an entity type that did not fit the mapping were discarded). This was paired with a novel entity extractor that added entities by constructing a dependency graph and a limited set of handcrafted patterns designed to identify new entities. It was found that when the novel entities were batch-added to the KB with manual validation, the performance of STIXnet remained stable. Without this step, performance declined as the number of reports containing entities not represented in the KB increased.

Paladini et al. [Pal+24] analyzed the time gap between when vulnerabilities were discussed in hacker forums and when they were published in official vulnerability reports. To address the challenges posed by the specialized terminology and informal language used in hacker forums, they applied extensive preprocessing to improve the performance of their NER approach. These steps included misspelling correction using Jampell [Ozi25], passive-to-active conversion (from SpaCy), stopword removal (from NLTK [BKL10]), internet slang removal, synonym and alias homogenization using dictionaries and lemmatization. Their ablation study showed that misspelling correction and synonym homogenization decreased performance on hacker forum posts, while only misspelling correction negatively affected performance on vulnerability reports.

Liu et al. (2025) [LLP25] propose a model called CtiErRe for NER and relation extraction. First, they construct a domain KG by compiling domain-specific vocabularies and calculating word co-occurrence statistics using the Pointwise Mutual Information (PMI) algorithm. This produces an adjacency matrix of semantic relations, which is then processed with a GCN to embed domain knowledge. These GCN embeddings are fused with BERT embeddings using Layer Normalization (LayerNorm). LayerNorm rescales and shifts feature distributions so the different embeddings share a consistent statistical scale. Experiments show that integrating GCN-based domain knowledge embeddings consistently improves performance by about 2%. For decoding the Global Pointer algorithm [Su+22a] is used. Global Pointer is a span-based NER framework that encodes text with BERT and then scores all possible token spans using a start-end attention mechanism. To capture word order and relative distances, it introduces RoPE (Rotary Position Embeddings), so that the span scorer is sensitive to the relative positions of tokens rather than just their absolute positions. It also employs imbalance-aware loss, which better handles the extreme imbalance between the huge number of negative spans (non-entities) and the relatively few positive spans (entities).

While others [Li+24b; AP23; Che+23; Fuj+22] employed either BERT-base-based or -uncased variants without providing specific justification, Paladini et al. [Pal+24] demonstrated that the cased variant outperformed the uncased variant on vulnerability reports, with performance differences of up to five percentage points. In contrast, Liu et al. [Liu+22] found that the uncased variant performed better on the DNRTI and MalwareTextDB

datasets. These findings suggest that the optimal variant depends on the specific dataset. Paladini et al. [Pal+24] prefer BERT over popular generative LLMs as BERT variants are not susceptible to hallucinations, due to their architecture (see Section 2.3.3).

All architectural details of the described approaches are summarized in Table 4.2, while Table 4.1 provides an overview of the method families employed by each approach. None of the approaches utilized RL, more than half used LMs and nearly all relied on DL. This distribution closely mirrors the overall usage of method families across all identified publications. Additionally, Table 4.3 outlines how the approaches formulated the NER task, their use of KBs and KGs, the maximum input sequence lengths supported and their multilingual capabilities. Only Sarhan and Spruit [SS21] trained their model on a dataset that supports English, Spanish and Portuguese. Moreover, half of the publications included some form of cost evaluation, ranging from brief qualitative statements to Big O notation analyses and quantitative tracking of training and inference times.

Table 4.4 summarizes how LMs are used in the approaches that incorporate them. Only Fujii et al. (2022) [Fuj+22] pre-trained BERT on their own cybersecurity corpus, while Paladini et al. (2024) [Pal+24] employed DarkBERT<sup>4</sup> [Jin+23], which was then further fine-tuned for the NER task. No approach employed partial fine-tuning. Instead, models were either fully fine-tuned or used exclusively for frozen embeddings.

Table 4.1: Method Families used in NER approaches

Name & Publication	Lang. Model	Deep Learn.	Stat. Learn.	Rule-based
STUCCO (Bridges et al., 2014) [Bri+14]			✓	✓
TTPDrill (Husari et al., 2017) [Hus+17]		✓	✓	✓
Twitter (Dionisio et al., 2019) [Dio+19]		✓	✓	
Open-CyKG (Sarhan & Spruit, 2021) [SS21]		✓	✓	
CyNER (Fujii et al., 2022) [Fuj+22; Fuj+23]	✓	✓		✓
NER4CTI (Liu et al., 2022) [Liu+22]	✓	✓		
BERT-CRF (Chen et al., 2023) [Che+23]	✓	✓	✓	
Split-NER (Arora & Park, 2023) [AP23]	✓	✓		
STIXnet (Marchiori et al., 2023) [MCV23]		✓	✓	✓
Vul-PAG (Li et al., 2024) [Li+24b]	✓	✓	✓	✓
Paladini et al. (2024) [Pal+24]	✓	✓		✓
CtiErRe (Liu et al., 2025) [LLP25]	✓	✓		✓

<sup>4</sup>BERT fine-tuned on dark web forum posts

Table 4.2: High-level Architecture of NER approaches that provide implementation code

Name & Publication	Embeddings/Features	Encode	Decode
STUCCO (Bridges et al., 2014) [Bri+14]	context (unigram, bigram, nearby/current words) in the form of <ul style="list-style-type: none"> <li>PoS (NLTK),</li> <li>BIO tags,</li> <li>surface features</li> </ul>	IOB tagging (history-based MEM, Averaged Perceptron) → Domain tagging (history-based MEM, Averaged Perceptron)	Greedy decoder + gazetteer
TTPDrill (Husari et al., 2017) [Hus+17]	BOW, PoS (Stanza (BiLSTM))	Dependency Parser (Stanza (BiLSTM))	TF-IDF+BM25
Twitter (Dionisio et al., 2019) [Dio+19]	word embeddings (word2vec, author-trained) + char embeddings (random-BiLSTM)	BiLSTM → FFNN	CRF
Open-CyKG (Sarhan & Spruit, 2021) [SS21]	random embeddings (Keras default)	BiGRU → FC layer	CRF
CyNER (Fujii et al., 2022) [Fuj+22; Fuj+23]	RoBERTa-large embeddings	FC layer	Softmax + regex
NER4CTI (Liu et al., 2022) [Liu+22]	Input to Contextual Encoder (concat emb. w/ linear conversion) <ul style="list-style-type: none"> <li>word emb. (GloVe + rand. emb. (for OOV))</li> <li>character emb. (CNN)</li> <li>PoS (CBOW w/ Stanza)</li> <li>component embeddings (i.e. token form)</li> </ul> Embedding augmentation: ISA-HSA, ESA (fine-tuned BERT)	<ul style="list-style-type: none"> <li>Contextual encoder: BiLSTM→MHSA→FFNNs</li> <li>ISA-HSA: word2vec (CBOW, pretrained)→KNN (cosine) → weighted emb.</li> <li>GateFusion (contextualized emb. + ISA-HSA + ESA)</li> </ul>	Softmax
BERT-CRF (Chen et al., 2023) [Che+23]	BERT-base-uncased embeddings	FC layer	CRF
Split-NER (Arora & Park, 2023) [AP23]	(i) word emb. <ul style="list-style-type: none"> <li>BERT-base-uncased+QA</li> <li>char emb.(CNN)</li> <li>pattern emb.(CNN-BiLSTM)</li> <li>PoS emb. (spaCy)</li> </ul>	(ii) Span Detection(FC layer) (iv) Span Classification(BERT-base-uncased+QA→FC layer)	(iii) Span Detection (Softmax) (v) Span Classification (Softmax)

*(continues on next page)*

Name & Publication	Embeddings/Features	Encode	Decode
STIXnet (Marchiori et al., 2023) [MCV23]	<ul style="list-style-type: none"> <li>PoS (spaCy)</li> <li>TTP Extractor (rcATT: BOW)</li> </ul>	<ul style="list-style-type: none"> <li>Novel Entity Extraction (spaCy: dependency graph)</li> <li>TTP Extractor (rcATT: Linear SVC)</li> </ul>	<ul style="list-style-type: none"> <li>IOC Finder (regex)</li> <li>KB Entity Extraction (Aho-Corasick + PoS validation)</li> <li>Novel Entity Extraction (linguistic pattern matching)</li> <li>TTP Extractor (rcATT: MITRE KB-based validation)</li> </ul>
Vul-PAG (Li et al., 2024) [Li+24b]	<ul style="list-style-type: none"> <li>Writing feat. (BERT-base-cased wordpiece + subword reorg.)</li> <li>Semantic feat. (BERT-base-cased word emb.)</li> <li>Syntactic feat. (MidConst, log. reg., BERT-base-cased mid-layer)</li> </ul>	Embedding fusion (linear comb.) → BiLSTM	CRF
Paladini et al. (2024) [Pal+24]	DarkBERT embeddings	FC layer (not reported)	Softmax (not reported)
CtiErRe (Liu et al., 2025) [LLP25]	<ul style="list-style-type: none"> <li>BERT embeddings</li> <li>domain knowledge (Domain-Knowledge Adjacency Matrix (predefined rules, PMI) → GCN embeddings)</li> </ul>	Layer normalization	Global Pointer

Table 4.3: Knowledge and Miscellaneous Properties

Name & Publication	KB	KG	Type of Task	Context Window	Multi Ling.	RE
STUCCO (Bridges et al., 2014) [Bri+14]	gazetter updated during training		Sequence Labeling	theor. unlim. (comp. bounded)		
TTPDrill (Husari et al., 2017) [Hus+17]	fills hierarchical ontology		Span Classification	theor. unlim. (comp. bounded)		✓
Twitter (Dionisio et al., 2019) [Dio+19]			Sequence Labeling	theor. unlim. (comp. bounded)		

*(continues on next page)*

Name & Publication	KB	KG	Type of Task	Context Window	Multi Ling.	RE
Open-CyKG (Sarhan & Spruit, 2021) [SS21]		create KG	Sequence Labeling	theor. unlim. (comp. bounded)	✓	✓
CyNER (Fujii et al., 2022) [Fuj+22; Fuj+23]			Sequence Labeling	512 tokens (BERT-based model)		✓
NER4CTI (Liu et al., 2022) [Liu+22]			Sequence Labeling	512 tokens (BERT-based model)		
BERT-CRF (Chen et al., 2023) [Che+23]			Sequence Labeling	256 tokens (reduced from 512 due to resource constraints)		
Split-NER (Arora & Park, 2023) [AP23]			Extractive QA	256 tokens (reduced from 512 due to resource constraints)		
STIXnet (Marchiori et al., 2023) [MCV23]	manually maintained + interactive extension during training; used for tagging entities	creates KG in STIX 2.1 format	Span Classification	theor. unlim. (comp. bounded)		✓
Vul-PAG (Li et al., 2024) [Li+24b]			Sequence Labeling	256 tokens (reduced from 512 due to content length)		
Paladini et al. (2024) [Pal+24]	manually curated KB used for synonym homogenization		Sequence Labeling	512 tokens (BERT-based model)		
CtiErRe (Liu et al., 2025) [LLP25]		uses KG as embedding for GCN	Span Classification	512 tokens (BERT-based model)		✓

Table 4.4: LMs used in NER approaches

Name & Publication	LM Name	Reasoning LM	Integration
CyNER (Fujii et al., 2022) [Fuj+22; Fuj+23]	RoBERTa-large	No	author pre-trained LM (prior NER), full fine-tune (NER)
NER4CTI (Liu et al., 2022) [Liu+22]	BERT-base-uncased	No	full fine-tune (NER)
BERT-CRF (Chen et al., 2023) [Che+23]	BERT-base-uncased	No	full fine-tune (NER)
Split-NER (Arora & Park, 2023) [AP23]	BERT-base-uncased	No	extrative QA, full fine-tune (NER)
Vul-PAG (Li et al., 2024) [Li+24b]	BERT-base-cased	No	embeddings (in multiple variants, NER)
Paladini et al. (2024) [Pal+24]	DarkBERT	No	full fine-tune (NER)
CtiErRe (Liu et al., 2025) [LLP25]	BERT	No	embeddings (NER)

## 4.2 Evaluation

This section presents the results of the evaluation of the finally selected approaches. The evaluation is divided into two parts. The first part focuses on performance evaluation, in which tagging performance is measured. The second part examines the cost of training and inference in terms of time and power consumption.

### 4.2.1 Model Selection and Adaption

Based on the criteria defined in Section 3.2.1, six of the twelve publications with available implementation code identified in the SLR are excluded. The excluded approaches are listed below:

- **STIXnet** [MCV23] is excluded due to RC1 (incomplete implementation), as the authors do not provide their NER module publicly.
- **Vul-PAG** [Li+24b] is excluded due to RC2 (insufficient documentation).
- **Paladini et al.** [Pal+24] is excluded due to RC2 (insufficient documentation).
- **TTPDrill** [Hus+17] is excluded due to RC3, as it depends on AllenNLP<sup>5</sup>, an unmaintained NLP framework that relies on outdated PyTorch versions.

<sup>5</sup><https://github.com/allenai/allennlp>

- **OpenCY-KG** [SS21] is excluded due to RC3, as it depends on outdated Keras and TensorFlow versions. Addressing these issues would require substantial code rewriting, which conflicts with the goal of executing the models as originally implemented by the authors.
- **CtiErRe** [LLP25] is excluded due to RC4. The execution environment is not documented and the required modifications are unclear. Running the code results in immediate tensor size mismatches on the provided dataset, along with additional unresolved errors.

The remaining models are listed below, along with the adaptations required to execute them. File path adjustments and dataset format conversions are not explicitly described, as all approaches are executed using their required tagging schemes, including IO, BIO, or BIOES. Dependency issues had to be resolved for all models, either to enable execution on modern hardware or because dependency specifications were missing. Overall, insufficient documentation, particularly the lack of information about the intended execution environment such as Python versions and dependencies and unclear guidance on adapting models to new datasets, made the process of running the models cumbersome.

All models use the hyperparameters reported in their respective publications. If hyperparameters are not specified, the default values defined in the codebase are applied. For BERT-based models, the maximum input length is increased to 512 tokens, which is the practical limit of BERT, to avoid truncation issues when using the STIXnet whole document (SWD) dataset. In addition, all approaches are instrumented with the codecarbon Python package to track computational cost.

- **STUCCO** [Bri+14]
  - The gazetteer component is removed because it is incompatible with the entity types used in the datasets.
  - PoS tagging is applied to the datasets prior to model execution using `nltk:taggers/averaged_perceptron_tagger_eng`.
- **Dionisio et al.** [Dio+19]
  - The batch size is reduced from 256 to 32 due to memory constraints and the limited size of the dataset.
  - Minor code modifications are applied to resolve issues with the `gensim` dependency.
  - The pretrained `word2vec-google-news-300d` model downloaded from `gensim` is used, as it is implicitly referenced in the code but not explicitly documented.
  - TensorFlow is not upgraded to a newer version to support modern GPUs, as this would require a substantial rewrite of the codebase. As a result, the model is executed on CPU only.

- **CyNER** [Fuj+22; Fuj+23]
  - Custom code is implemented to extract prediction outputs from the model.
- **NER4CTI** [Liu+22]
  - GloVe.6B.50d embeddings<sup>6</sup> are manually downloaded from Kaggle and BERT is obtained from Hugging Face to enable model execution.
  - Entity types are hard-coded in the original implementation and are modified to be dynamically extracted from the training dataset.
- **BERT-CRF** [Che+23]
  - A wrapper script is implemented to generate predictions for the full test set, as the original code outputs predictions for only one sentence at a time.
  - PyTorch is upgraded to a recent version to enable execution on modern GPUs. The required code change is limited to changing a class name that was renamed in newer PyTorch releases.
- **Split-NER** [AP23]
  - The PoS tagging component is manually integrated into the original processing pipeline to enable compatibility with additional datasets.

#### 4.2.2 Performance Evaluation

To gain insight into model performance under varying conditions, performance scores are interpreted from four complementary perspectives using the metrics defined in Section 3.2.4. First, an overall perspective is adopted to assess how well the models detect entities and their boundaries at a high level. Second, model performance is analyzed across different entity types. Third, performance differences are examined with respect to multiple entity attributes. Finally, the analysis compares model behavior when processing smaller versus larger inputs to evaluate how effectively additional contextual information is incorporated.

##### Overall Performance

Table 4.5 shows the models overall performance on the DNRTI test set. The overall performance across models is tightly clustered, with F1 scores ranging from 0.80 to 0.87.

CyNER achieves the strongest overall results, ranking first by strict micro F1 (0.8689) and also leading in strict macro F1 (0.8839). BERT-CRF follows very closely (strict micro F1 0.8659) and CyNER’s small advantage may plausibly be attributed to architectural choices, in particular the use of RoBERTa compared to BERT in BERT-CRF.

<sup>6</sup><https://www.kaggle.com/api/v1/datasets/download/watts2/glove6b50d.txt>

Table 4.5: Overall model performance on DNRTI test dataset across all models ordered by strict micro F1 score

Model	Strict Micro F1	Strict Macro F1	Exact Micro F1	Partial Micro F1	Entity Type Micro F1	Entity Type Macro F1
Fujii et al. (2022) - CyNER	<b>0.8689</b>	<b>0.8839</b>	0.8962	<b>0.9169</b>	0.9004	0.9119
Chen et al. (2023) - BERT-CRF	0.8659	0.8821	0.8873	0.9100	<b>0.9024</b>	<b>0.9140</b>
Bridges et al. (2014) - STUCCO	0.8620	0.8744	<b>0.8999</b>	<b>0.9169</b>	0.8750	0.8872
Arora and Park (2023) - SplitNER	0.8590	0.8758	0.8940	0.9157	0.8894	0.9033
Liu et al. (2022) - NER4CTI	0.8347	0.8536	0.8707	0.8974	0.8711	0.8861
Dionisio et al. (2019) - Twitter	0.8040	0.8190	0.8233	0.8617	0.8661	0.8772
LangExtract w/ Gemma3:27b	0.4849	0.4673	0.6323	0.7021	0.5645	0.5557
LangExtract w/ DeepSeek-R1:32b	0.4514	0.4244	0.5914	0.6733	0.5345	0.5153

A clear pattern across the results is that simpler architectures perform best. Models with relatively lightweight designs such as CyNER with a simple classifier on top of RoBERTa and BERT-CRF consistently outperform more complex approaches. In contrast, the most feature-engineered model, NER4CTI, ranks only fifth across evaluation metrics, suggesting that additional mechanisms targeting issues such as OOV or limited samples for certain entity types do not translate into improved performance on DNRTI.

Despite being the oldest approach, STUCCO remains highly competitive with more recent models and excels particularly in span detection, achieving the best exact micro F1 (0.8999). When the evaluation is relaxed from exact boundary matching to correctness of the entity type, BERT-CRF emerges as the strongest model, achieving the highest entity-type F1 scores, which indicates good labeling performance even when span boundaries are imperfect.

Across all models, a consistent pattern can be observed in the relationship between micro and macro F1 scores. For both strict and entity-type evaluation, macro F1 scores are higher than micro F1 scores, indicating that performance on less frequent entity types is comparatively strong and that models do not disproportionately favor majority classes.

Finally, Twitter<sup>7</sup> shows relatively weaker strict scores but noticeably stronger entity-type results, implying that it often predicts the correct entity label while struggling more with precise span boundaries compared to the top-performing systems. In contrast, the LangExtract LLM baselines perform substantially worse across all metrics, highlighting that, for DNRTI, specialized NER architectures generalize far more effectively than prompting-based extraction approaches. The reasoning model DeepSeek-R1 performed even worse than the non-reasoning model Gemma3, indicating that reasoning models do not provide better performance.

<sup>7</sup>This name is given to the approach from Dionisio et al. [Dio+19], to reference it easier. It is chosen, because the approach was developed and tested on a dataset consisting of Tweets.

### Entity Type-dependent Performance

Figures 4.9a and 4.9b present the strict micro-averaged F1 scores per entity type for the SSS and DNRTI test datasets. A key structural difference between the two corpora is their label distribution: The SSS dataset is heavily dominated by a small number of entity types, whereas the DNRTI dataset exhibits a more balanced spread across categories.

For SSS (see Figure 4.9a), the imbalance is directly reflected in model behavior. Specialized NER models achieve very high scores on the dominant entity types, while performance deteriorates sharply for rare categories. In this setting, the Baseline stands out by outperforming all specialized models on certain low-frequency entity types, most notably Campaign and SHA-256. This suggests that, for sparsely represented entities, leveraging pretrained world knowledge can partially offset the lack of supervised training data.

The DNRTI heatmap (see Figure 4.9b) largely reinforces the conclusions from the overall performance analysis in Section 4.2.2. The Baselines consistently underperform the specialized NER models across most entity types, confirming their weak overall results. Nevertheless, the per-entity breakdown shows that the baseline remains competitive on a small subset of categories, such as Area, Exp and HackOrg, even if it does not surpass the CTI-NER models. Moreover, the heatmap highlights that no single model dominates all entity types. Different specialized approaches achieve the highest scores for different categories, although the absolute differences between the best-performing models are generally very minor.

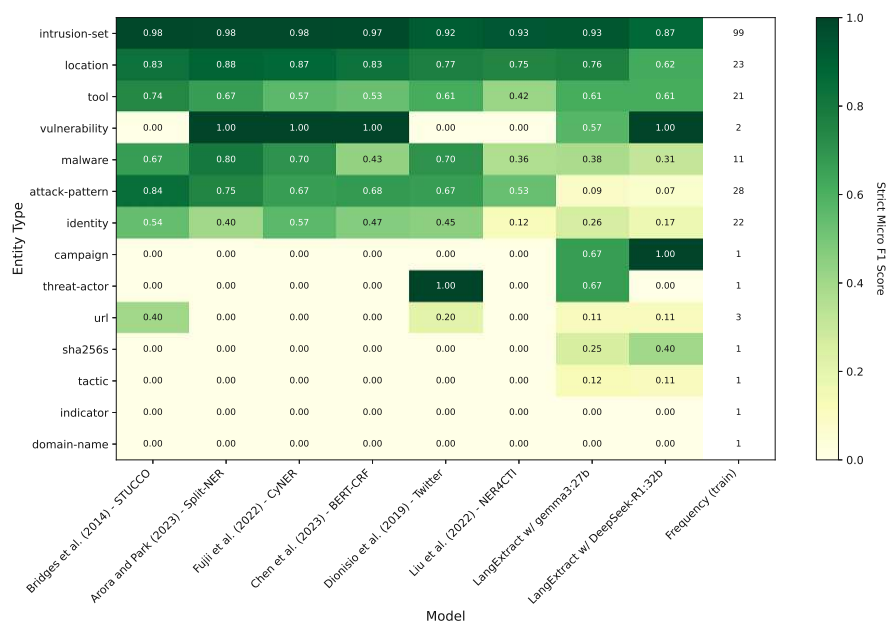
### Entity Attribute-dependent Performance

The entity attribute-dependent evaluation analyses performance on DNRTI by partitioning entities into four buckets (XS, S, L, XL) for each attribute, as defined in Section 3.2.4. For every model and attribute, strict micro F1 is computed separately for each bucket. If a model achieves similar F1 scores across buckets, its performance is largely invariant to the corresponding entity attribute. If bucket scores differ substantially, the model is sensitive to that attribute and performs better under some conditions than under others.

Figure 4.10 shows the variability in performance across entity attributes and buckets for each model. The BERT-based models (CyNER, BERT-CRF, SplitNER, NER4CTI) exhibit comparatively low variability, indicating robust behaviour across changes in entity attributes. In contrast, STUCCO and Twitter show higher variability and are therefore more sensitive to attribute changes, although both still achieve substantially higher overall performance than the baseline.

The detailed bucket trends are shown in two complementary views. Figure 4.11 groups results by attribute and compares how all models behave across the corresponding buckets. Figure 4.12 groups results by model and highlights how different attributes affect the same model. In the following, each attribute is interpreted in turn.

**Entity length (eLen)** captures the number of words in an entity. The CTI-NER models remain comparatively stable across buckets, while the baselines degrade sharply as entity



(a) STIXnet single sentence test dataset

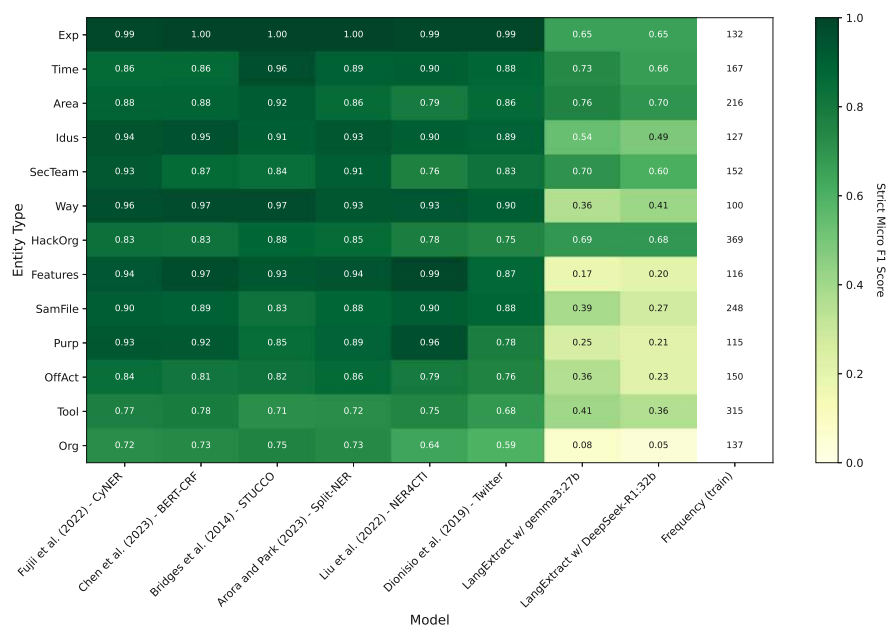
Figure 4.9: Heatmap of strict micro-averaged F1 scores across entity types. “Frequency (train)” denotes the number of samples in the training dataset. Models are ordered in descending order according to their overall strict micro F1 score. Entity types are ordered top to bottom (best to worst) by their strict micro F1 score averaged across all models.

length increases, dropping to near-random performance for entities of four or more tokens (see Figure 4.10). At this stage, it is not clear why this behaviour occurs and is left for future work. Importantly, it is unlikely to be explained purely by the strictness of exact boundary matching, since Table 4.5 shows that the baseline improves in a similar manner to the CTI-NER models when boundary requirements are relaxed. Among the supervised approaches (CTI-NER), STUCCO performs best for long entities, consistent with its overall strength in span detection.

**Sentence length (sLen)** measures the number of words in a sentence, bucketed by quartiles. The curves are largely flat across buckets, indicating that sentence length has little influence on strict micro F1 for any of the evaluated models.

**Entity density (eDen)** measures how many entities occur relative to sentence length. For the baselines, performance increases markedly with higher density, whereas CTI-NER models show a smaller increase or remain comparatively stable. One explanation could be that sentences containing multiple entities provide richer contextual cues and repeated structural patterns that can support extraction, even for a prompted model, such as the Baselines, operating with fewer than ten in-context examples.

**OOV density (oDen)** represents the proportion of words in a sentence that are not



(b) DNRTI test dataset

Figure 4.9: Heatmap of strict micro-averaged F1 scores across entity types. “Frequency (train)” denotes the number of samples in the training dataset. Models are ordered in descending order according to their overall strict micro F1 score. Entity types are ordered top to bottom (best to worst) by their strict micro F1 score averaged across all models.

present in the training vocabulary. All supervised CTI-NER models degrade as OOV density increases, reflecting their reliance on lexical and contextual regularities learned from the training distribution. NERACTI shows one of the steepest drops, suggesting that it is particularly sensitive to vocabulary mismatch. The baseline models display flatter curves across OOV density. This divergence in sensitivity points to a more fundamental difference between the two model categories. For CTI-NER models, OOV density and token frequency are strongly correlated in their effect on performance: both attributes ultimately measure how familiar the model is with the vocabulary encountered during training. The baselines, by contrast, show no such relationship, as these dataset-relative metrics hold little relevance for models that were never trained on DNRTI and instead rely on representations acquired during pretraining.

**Entity frequency (eFre)** and **token frequency (tFre)** quantify how often an entity, or its constituent words on average, occurs in the training data. For CTI-NER models, performance drops sharply in the lowest-frequency bucket and stabilizes once entities or words have been observed at least once, consistent with supervised models requiring exposure to domain-specific surface forms. For the baselines, these metrics are less informative, as they measure frequency relative to a training set the LLMs have never observed.

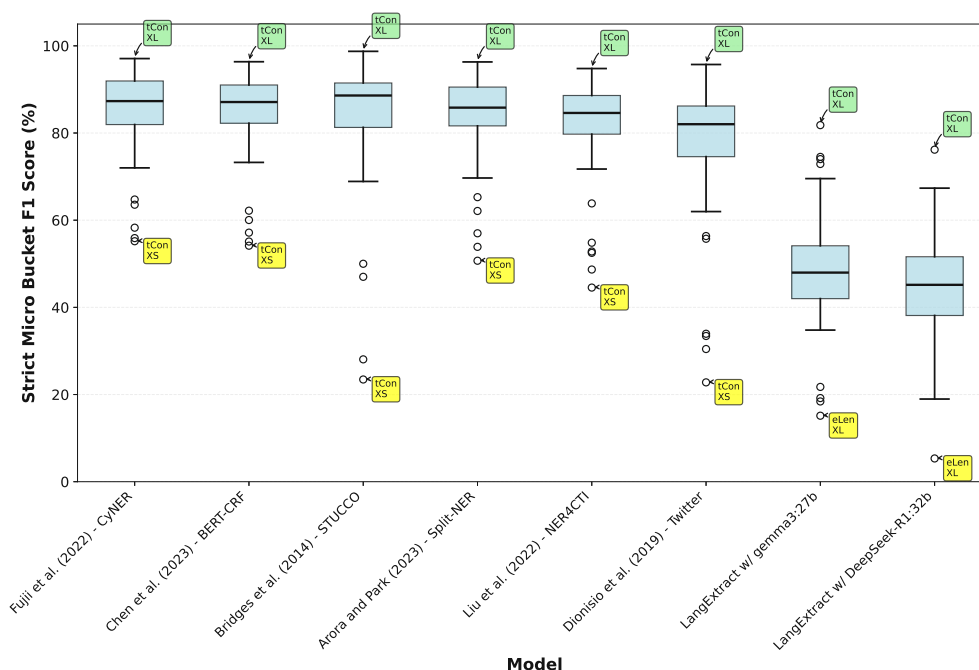


Figure 4.10: This boxplot shows the strict micro F1 score of each bucket over all attributes per model. The best and worst performing attribute-bucket combinations are labeled.

**Entity consistency (eCon)** and **token consistency (tCon)** measure how consistently an entity string, or its words, are associated with the same label in training. These attributes produce the largest performance gaps across buckets, exceeding 50 percentage points for most models. Notably, the impact is not limited to extreme cases. Even moderately inconsistent labelling leads to substantial degradation and well-performing models such as CyNER drop to approximately half their typical F1 on inconsistently labelled entities. This pattern shows that label instability introduces ambiguity that is difficult to resolve from context alone. Overall, consistency emerges as one of the strongest predictors of strict micro F1 on DNRTI, as Figure 4.10 shows.

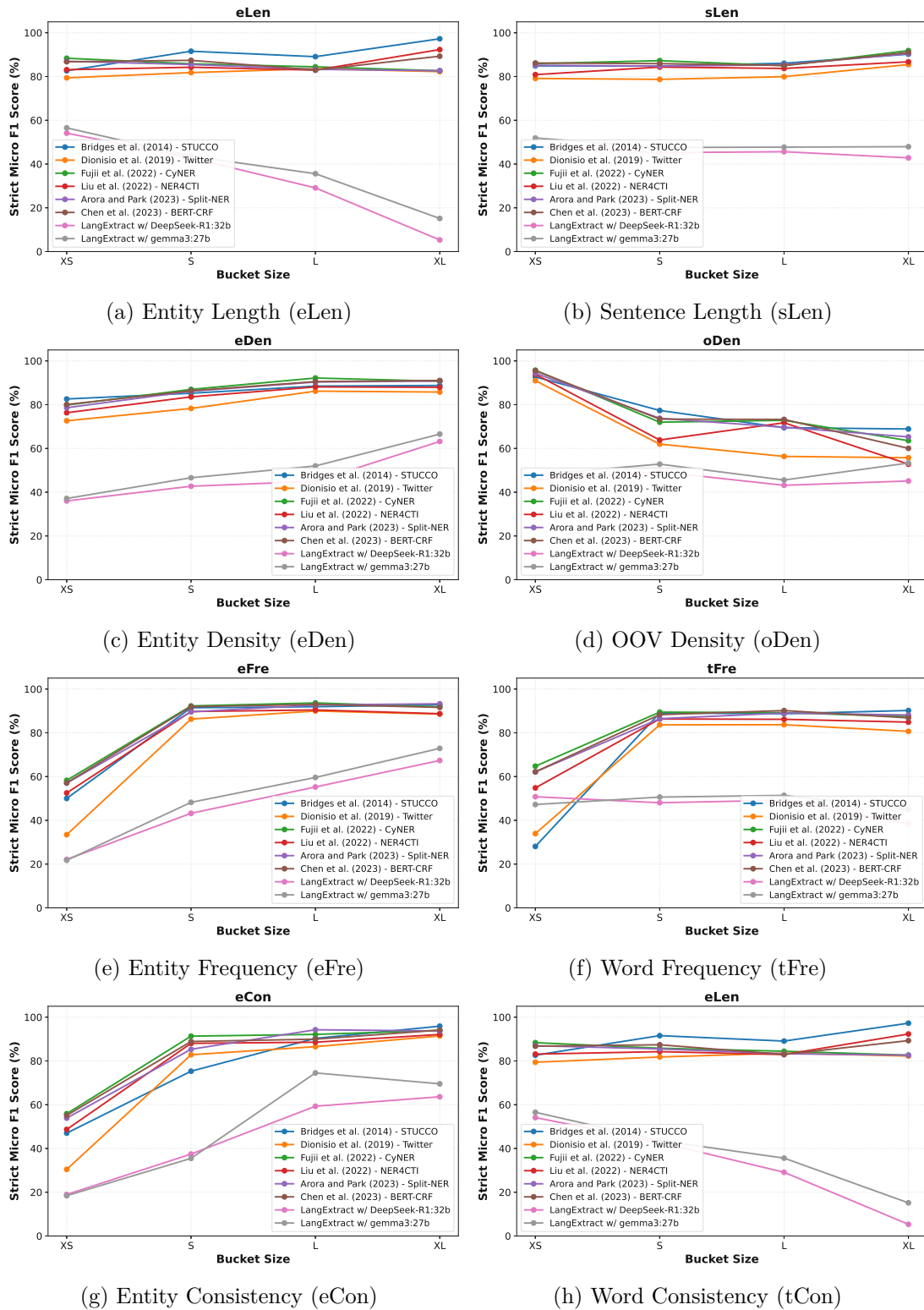
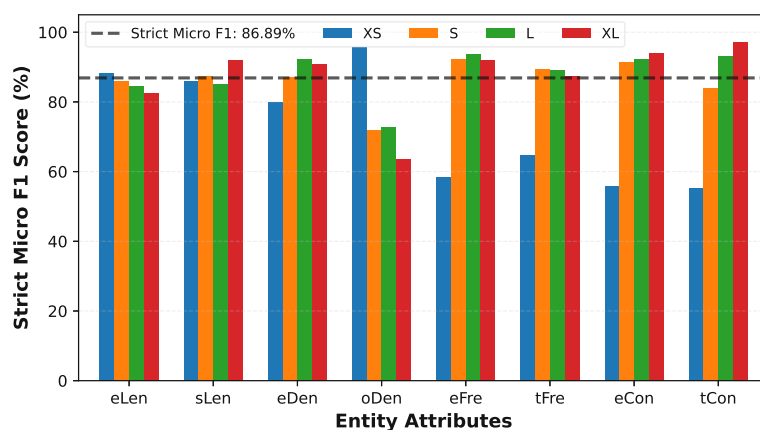
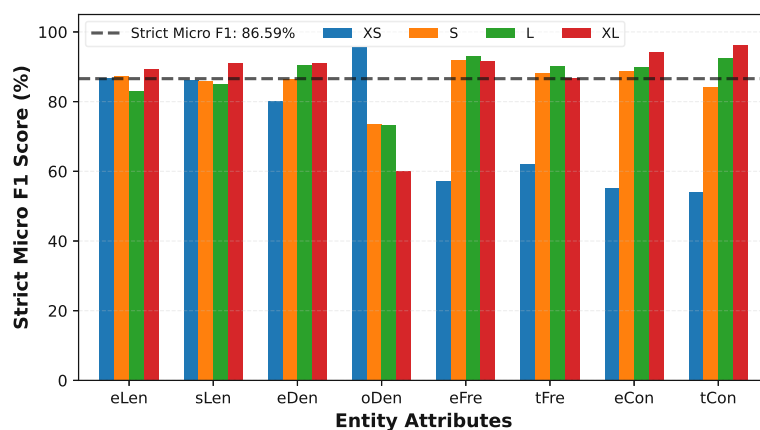


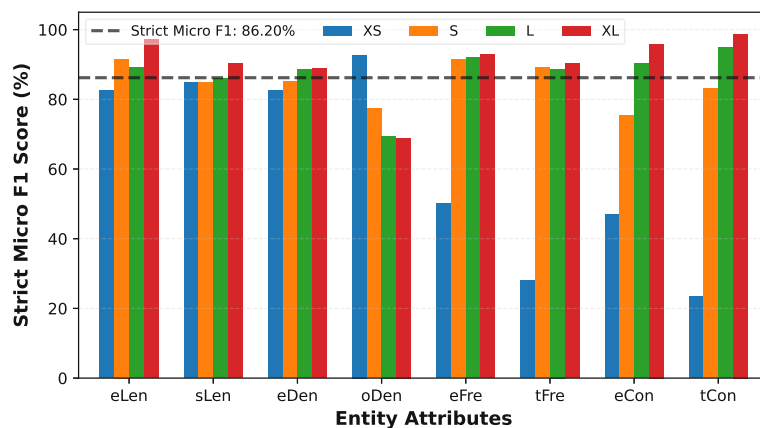
Figure 4.11: Each subfigure presents, for a single attribute, the change in performance across buckets for all models. Each attribute is subdivided into predefined buckets, as defined in Section 3.2.4.



(a) Fujii et al. (2022) - CyNER

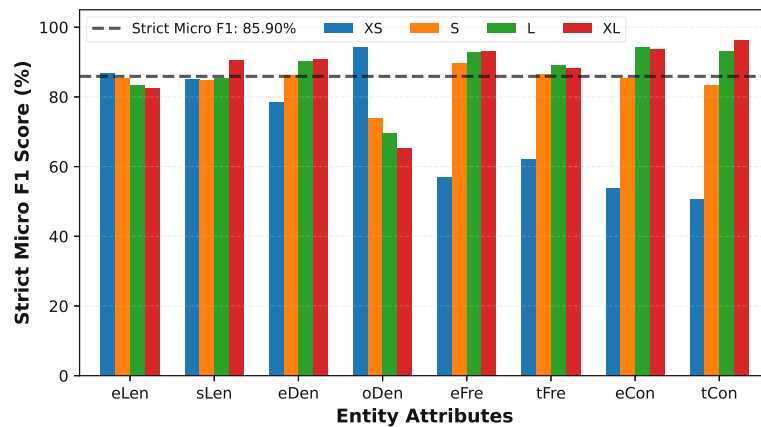


(b) Chen et al. (2023) - BERT-CRF

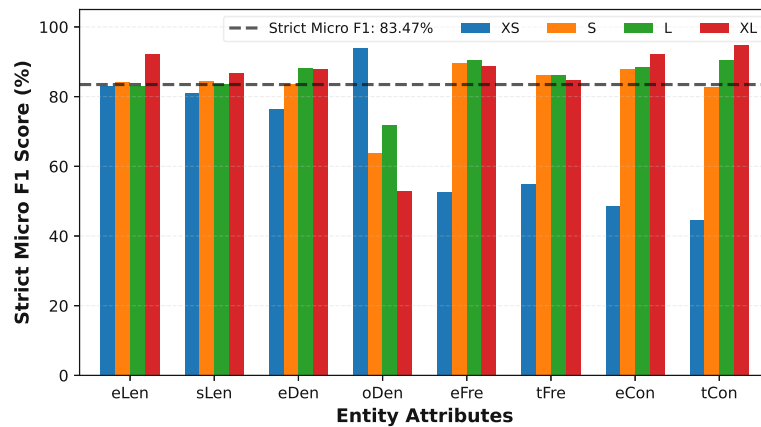


(c) Bridges et al. (2014) - STUCCO

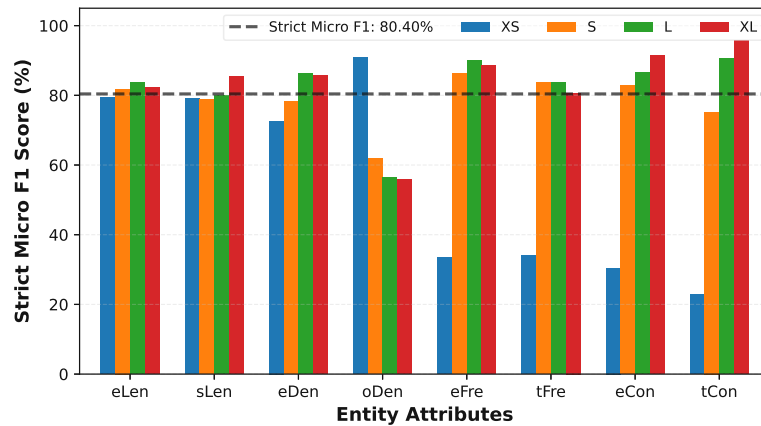
Figure 4.12: Each subfigure presents, for a single model, the change in performance across buckets for all entity attributes, with each property subdivided into predefined buckets as defined in Section 3.2.4.



(d) Arora and Park (2022) - Split-NER

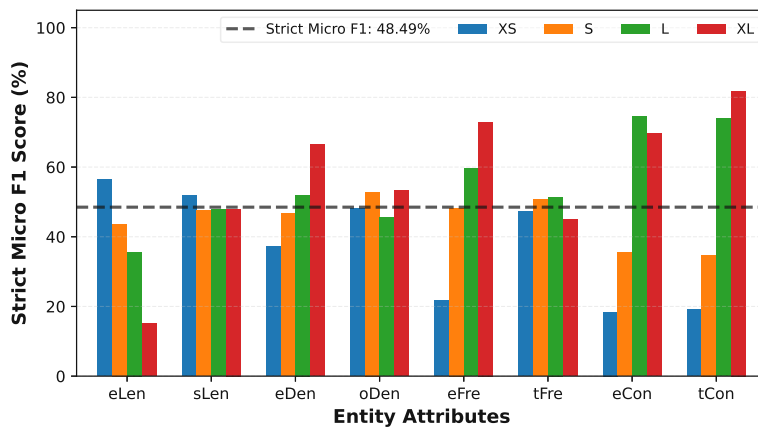


(e) Liu et al. (2023) - NER4CTI

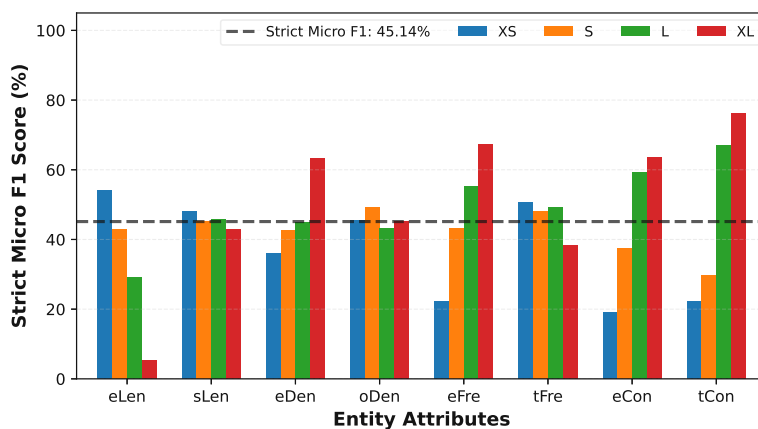


(f) Dionisio et al. (2019) - Twitter

Figure 4.12: Each subfigure presents, for a single model, the change in performance across buckets for all entity attributes, with each property subdivided into predefined buckets as defined in Section 3.2.4.



(g) LangExtract w/ Gemma3:27b



(h) LangExtract w/ DeepSeek-R1:32b

Figure 4.12: Each subfigure presents, for a single model, the change in performance across buckets for all entity attributes, with each property subdivided into predefined buckets as defined in Section 3.2.4.

### Impact of Context Length on Model Performance

In this experiment, model performance is evaluated when tagging an entire document at once (SWD) compared to tagging individual sentences without contextual information (SSS). The results shown in Figure 4.13 indicate that the expectation of improved tagging performance through additional document-level context is not supported. Only the DL model Twitter achieves a notable improvement, with an increase of six percentage points in strict micro F1 score, while CyNER shows a marginal improvement of one percentage point. All other approaches exhibit performance degradation when processing full documents. The LangExtract baseline shows the largest decline, with decreases of 24 and 35 percentage points in strict micro F1 score for Gemma3 and DeepSeek, respectively. This is followed by STUCCO and NER4CTI, both of which experience a reduction of 22 percentage points. BERT-CRF and Split-NER also show substantial declines, with performance drops exceeding 10 percentage points.

Several factors may contribute to the observed degradation. Although the models are trained on document-level inputs, the number of such training examples is limited, which may be insufficient for learning robust cross-sentence patterns. Longer input sequences also introduce additional noise that can dilute attention signals. For LLMs, extended context may provide more opportunity for hallucination and over-generation of entities. The improvements observed for Twitter and CyNER suggest that relatively simple DL architectures, such as a BiLSTM encoder as used in Twitter or encoder-only LMs such as BERT in CyNER, combined with CRF or softmax decoders, are better suited to handling longer input sequences.

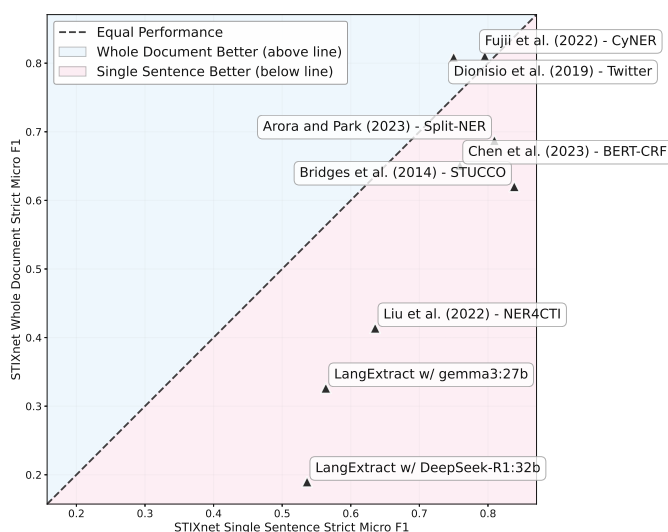


Figure 4.13: This scatterplot illustrates the performance of the models on the STIXnet dataset, measured by strict micro-averaged F1 score. Performance is shown for two settings: tagging each sentence individually (SSS test dataset) and tagging all sentences of a document jointly (SWD test dataset).

### 4.2.3 Cost Evaluation

For cost evaluation, runtime measured in seconds and energy consumption measured in kilowatt-hours are considered, as defined in Section 3.2.5. First, overall training time and energy consumption are examined and related to the respective model performance. Second, the evaluation compares the cost of tagging many short sentences with the cost of processing fewer, larger inputs.

#### Cost-Performance Trade-off

**Inference** The inference cost-performance trade-off on the DNRTI test dataset is illustrated in Figure 4.14, which jointly considers strict micro F1 score, inference time per sentence and energy consumption. Models on the Pareto frontier represent optimal trade-offs, as no other model achieves higher performance without incurring greater inference cost.

The statistical learner STUCCO achieves the fastest inference, processing a sentence in under one millisecond while delivering performance comparable to the BERT-based models. This makes STUCCO the most efficient option in terms of runtime. CyNER is the second-fastest model at approximately 13 ms per sentence, which is about  $24\times$  slower than STUCCO, but it achieves the highest strict micro F1 score overall. Among the models for which energy measurements are available, CyNER also exhibits the lowest inference energy consumption, making it the sole Pareto-optimal model when energy efficiency is considered explicitly.

The DL model Twitter, despite employing a CRF layer, remains relatively efficient at around 23 ms per sentence. In contrast, BERT-CRF is the slowest CTI-NER model at approximately 103 ms per sentence, making it roughly 180 times slower than STUCCO. A comparison between CyNER and BERT-CRF suggests that a softmax classification head is generally faster than a CRF layer, although Twitter’s efficiency (CRF) indicates that the CRF layer alone does not fully explain BERT-CRF’s higher inference cost.

The Baselines are substantially more expensive at inference time. Both variants exceed five seconds per sentence, making them roughly 50 times slower than BERT-CRF (slowest CTI-NER model) and the only approaches to exceed the one-second barrier, while their energy consumption increases disproportionately, by a factor of 88 to 133. This demonstrates that LLM-based extraction not only underperforms, but also incurs extreme operational costs, making it unsuitable for high-throughput deployment. Overall, Figure 4.14a shows that STUCCO and CyNER dominate all other models for inference, with STUCCO minimizing runtime and CyNER maximizing performance.

**Training** Training efficiency is analysed in Figure 4.15, which relates strict micro F1 to training time and energy consumption, excluding the cost of pre-training publicly available models. The resulting Pareto frontier highlights models that achieve favourable performance without excessive training cost.

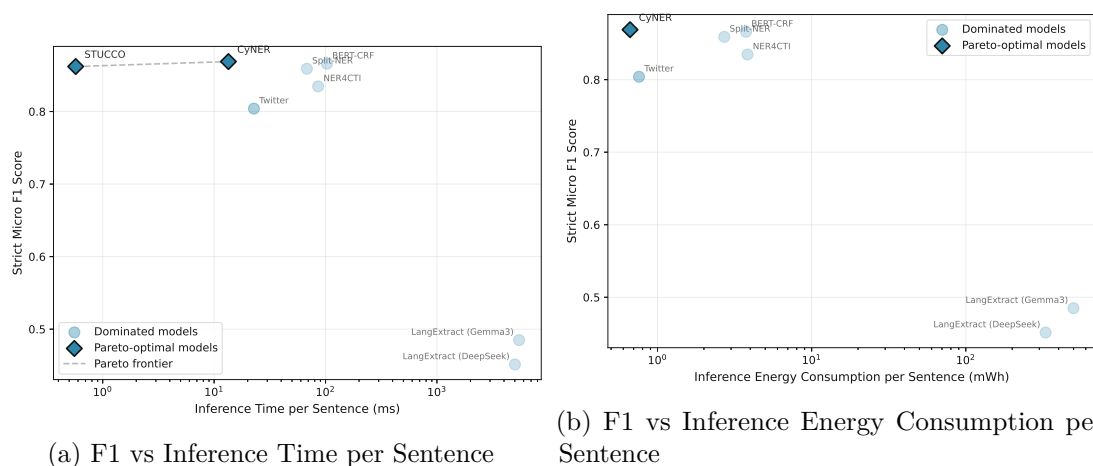


Figure 4.14: Comparison of strict micro F1 score with inference time and energy consumption on the DNRTI test dataset. x-axis is log-scaled for readability.

Twitter is the fastest model to train, completing training in under 30 minutes with an energy consumption of only 57 Wh, demonstrating that a BiLSTM-CRF architecture with word2vec embeddings can be trained efficiently even on CPU. This makes Twitter particularly attractive in scenarios where frequent retraining is required, for example due to changes in entity definitions or label mappings. BERT-CRF is the second fastest model to train at approximately 30 minutes and 124 Wh, while also achieving one of the highest F1 scores overall, making it a strong alternative to Twitter when higher accuracy is required.

CyNER and Split-NER both require around two hours of training, with energy consumption of 613 Wh and 486 Wh, respectively. While CyNER incurs higher training costs, it also achieves the best overall performance, representing a clear accuracy-cost trade-off rather than inefficiency. STUCCO, despite its excellent inference efficiency, requires over three hours to train. However, as it was trained on CPU due to the lack of a GPU-compatible implementation, its training time could potentially be reduced with an implementation utilizing a GPU.

The most complex model, NERACTI, is by far the slowest and most energy-intensive, requiring over eight hours of training and consuming 2.6 kWh, more than four times the energy required by CyNER. This substantial cost is not matched by a corresponding performance gain, reinforcing the observation that increased architectural complexity can dramatically raise computational cost without improving effectiveness. Overall, Figure 4.15b shows that Twitter, BERT-CRF and CyNER form the Pareto frontier for training efficiency: Twitter minimizes training cost, BERT-CRF provides an excellent balance of cost and performance and CyNER achieves the highest accuracy among the efficient models.

Notably, Twitter occupies a unique position in the cost-performance landscape. It is

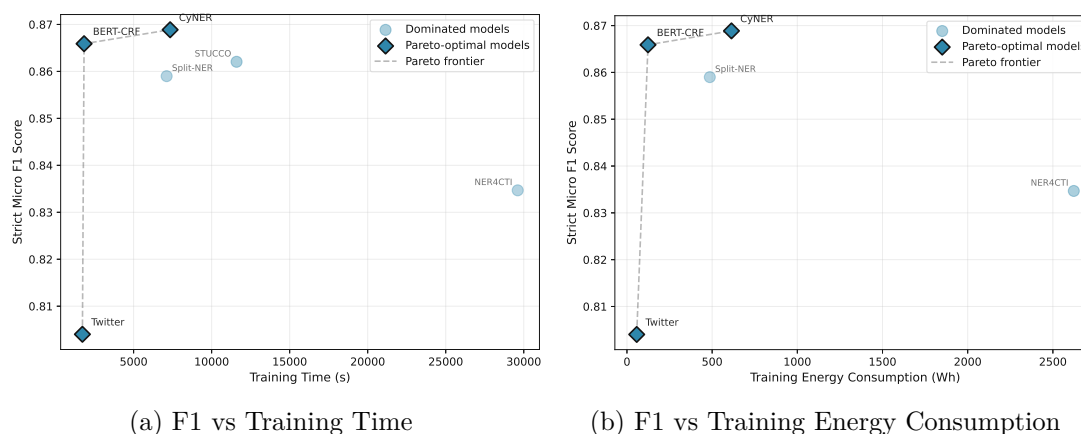


Figure 4.15: Comparison of strict micro F1 score with training time and energy consumption on the DNRTI test dataset.

Pareto-optimal for training but outperformed for inference by STUCCO and CyNER, making it well suited for settings that require frequent retraining but do not demand high-throughput inference. Table 4.6 again summarizes the runtime and energy consumption metrics obtained during model training and inference on the DNRTI test dataset.

Table 4.6: Cost metrics runtime and energy consumption are ordered by inference time per sentence on the DNRTI test dataset. Energy consumption measurements are not available for STUCCO because the framework for collecting the statistics does not support Python 2. LangExtract does not report training costs, as it is not trained. The column “Pre-trained models used” indicates which publicly available pre-trained models are employed by the CTI-NER approaches. The cost of pre-training these models is not included in the subsequent calculations. STUCCO and Twitter are trained on CPU, whereas all other models are trained on GPU (see Section 4.2.1).

Model	Pre-trained models used	Strict Micro F1	Inference Time per Word (ms)	Inference Energy Consumption per Word (mWh)	Inference Time per Sentence (ms)	Inference Energy Consumption per Sentence (mWh)	Training Time (s)	Training Energy Consumption (Wh)	
Bridges et al. (2014)	NLTK averaged perceptron tagger eng	0.862	<b>0.021</b>		<b>0.572</b>		11597.329		
Fujii et al. (2022)	BERT	<b>0.869</b>	0.504	<b>0.025</b>	13.437	<b>0.664</b>	7340.404	613.392	
Dionisio et al. (2019)	Twitter	0.804	0.853	0.028	22.770	0.758	<b>1719.206</b>	<b>57.306</b>	
Arora and Park (2023)	Split-NER	PoS, BERT	0.859	2.550	0.102	68.032	2.711	7116.411	486.262
Liu et al. (2022)	NER4CTI	PoS, word2vec, BERT	0.835	3.217	0.144	85.824	3.835	29612.061	2622.717
Chen et al. (2023)	BERT-CRF	BERT	0.866	3.859	0.141	102.966	3.752	1825.641	123.632
LangExtract w/ DeepSeek-R1:32b	deepseek-r1:32b	0.451	187.885	12.352	5012.764	329.556			
LangExtract w/ Gemma3:27b	Gemma3:27b	0.485	204.767	18.714	5463.196	499.285			

### Context Length-dependent Cost

Table 4.7 compares performance and cost metrics for the STIXnet dataset when tagging sentences individually without contextual information (SSS) versus tagging entire documents jointly (SWD), with inference costs reported on a per-word basis to account for the large variation in sentence lengths. Compared to the DNRTI test set, the per-word inference cost on STIXnet is higher across models and dataset composition. One explanation could be the smaller number of sentences and tokens in STIXnet, since fixed overheads such as model initialization and invocation are distributed over a smaller number of sentences and therefore tokens.

From an inference perspective, tagging entire documents is consistently faster and more energy-efficient per word than tagging individual sentences for almost all models. The only exception is Split-NER, for which document-level tagging is slightly more expensive than sentence-level tagging. The largest relative speedups are observed for the Baselines, which achieves up to a 3.6 times reduction in inference time per word and for the Twitter model, which shows a comparable 3.7 times improvement. Figure 4.17 visualizes these differences and shows that the relative ordering of models remains consistent for both inference time and energy consumption. However, despite these efficiency gains, document-level tagging leads to substantially worse tagging performance across all models (see Table 4.7 and Figure 4.13).

The training cost comparison, shown in Figure 4.16, exhibits a similar pattern to inference. Training with document-level inputs is generally more efficient on a per-word basis than sentence-level training. Nevertheless, this reduction in computational cost does not translate into improved effectiveness.

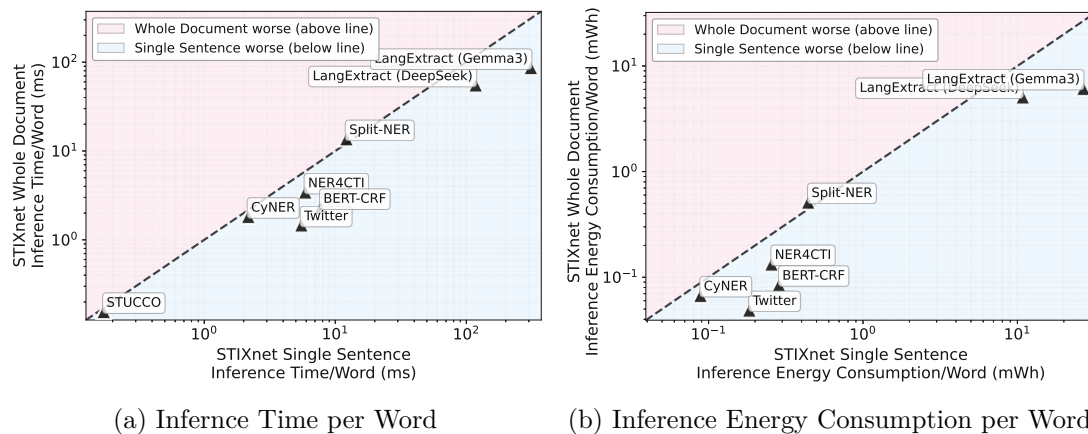


Figure 4.16: Comparison of inference time and energy consumption between SSS test dataset and SWD test dataset. Axes are log-scaled for readability.

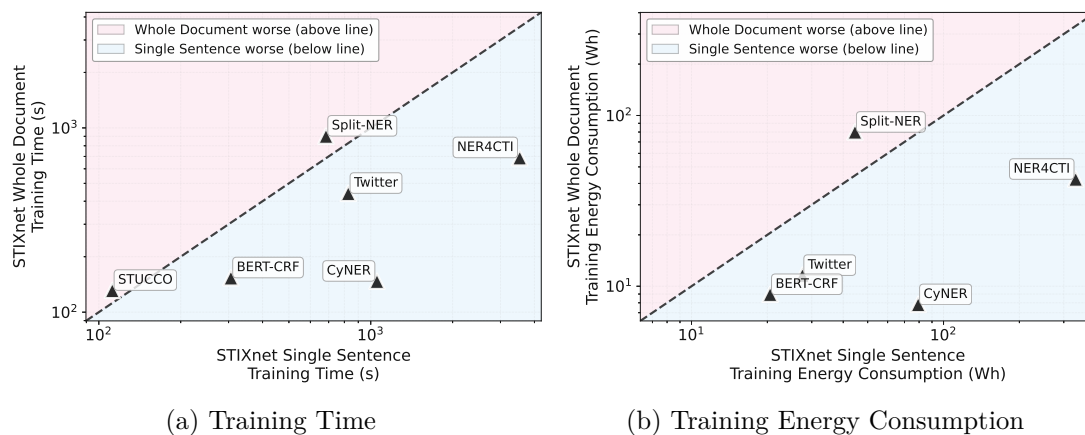


Figure 4.17: Comparison of training time and energy consumption between SSS test dataset and SWD test dataset. Axes are log-scaled for readability.

Table 4.7: Comparison of cost metrics for the STIXnet dataset when tagging sentences individually without contextual information (SSS) versus tagging entire documents jointly (SWD dataset). Ordered by SWD Inference Time per Word. Energy consumption measurements are not available for STUCCO because the framework for collecting the statistics does not support Python 2. LangExtract does not report training costs, as it is not trained. STUCCO and Twitter are trained on CPU, whereas all other models are trained on GPU (see Section 4.2.1).

Model	SSS Strict Micro F1	SSS Inference Time per Word (ms)	SSS Inference Energy Consumption per Word (mWh)	SSS Training Time (s)	SSS Training Energy Consumption (Wh)	SWD Strict Micro F1	SWD Inference Time per Word (ms)	SWD Inference Energy Consumption per Word (mWh)	SWD Training Time (s)	SWD Training Energy Usage (Wh)
Bridges et al. (2014) STUCCO	<b>0.838</b>	<b>0.171</b>		112.094		0.620	<b>0.156</b>		<b>131.362</b>	
Dionisio et al. (2019) Twitter	0.750	5.510	<b>0.184</b>	826.651	27.554	0.809	1.476	<b>0.049</b>	441.562	11.619
Fujii et al. (2022) CyNER	0.795	2.169	0.089	1054.134	79.326	<b>0.810</b>	1.842	0.068	147.241	<b>7.831</b>
Chen et al. (2023) BERT-CRF	0.759	7.720	0.286	<b>305.952</b>	<b>20.508</b>	0.651	2.317	0.085	153.686	8.977
Liu et al. (2022) NER4CTI	0.636	5.901	0.255	3531.133	334.535	0.414	3.465	0.134	689.213	42.461
Arora and Park (2023) Split-NER	0.809	12.162	0.442	684.507	44.528	0.687	13.773	0.516	902.917	80.129
LangExtract w/ DeepSeek-R1:32b	0.536	117.939	10.839			0.190	55.602	5.104		
LangExtract w/ Gemma3:27b	0.564	310.953	26.783			0.326	87.294	6.157		

## Discussion

This chapter discusses the findings presented in the previous section in relation to the research questions outlined in Section 1.2. The results indicate that simpler models outperform more complex ones. LLM-based NER approaches perform worse than tailored CTI-NER approaches. Performance on longer sequences that incorporate more context is worse than single-sentence tagging and BERT-based approaches achieve strong performance while incurring moderate cost.

All evaluations were performed on either the DNRTI or STIXnet dataset. Because STIXnet was only used for context comparison due to its small size and uneven entity type distribution, the majority of the analysis relied on the DNRTI dataset. As a result, the findings are only valid for this dataset and may differ on other datasets. Differences in annotation practices, labeling consistency and entity distribution could cause certain architectures to appear weaker due to dataset-specific characteristics rather than inherent model limitations.

In addition, the set of entity types to be tagged may favor some architectures over others. For example, IoCs are entities that are largely detectable using regular expressions and methods that benefit from surface-form regularity likely perform better on these entities. Conversely, when a dataset is very small, such as STIXnet, LMs like BERT tend to underperform because the available data is insufficient for effective training given their model size.

Dataset splitting also plays a critical role. In the literature, available datasets such as DNRTI are typically split so that entity types are evenly distributed across the training, validation and test sets. However, this approach does not account for the fact that the exact same entities present in the test set may already appear in the training set. The analysis of entity frequency attributes in Section 4.2.2 demonstrates this clearly. Tagging performance dropped by approximately 15 percentage points when the model had not

---

encountered an entity during training. This indicates that data leakage artificially inflates model performance.

Furthermore, the analysis of entity consistency shows that higher consistency in how an entity is labeled corresponds to better model performance. This trend was observed consistently across all evaluated approaches.

Across all perspectives from which performance scores were analyzed in Section 4.2.2, a clear trend emerged: simple architectures outperform more complex ones. Furthermore, encoder-based LMs such as BERT are the most effective embedding-and-encoding option and achieve the best performance when paired only with a Softmax classification or CRF layer. This aligns with the dominant use of BERT in the approaches identified in the SLR (see Figure 4.6a and 4.8). In contrast to the SLR, where most approaches use BiLSTMs (see Figure 4.7), this work shows that comparable or even superior performance can be achieved without a BiLSTM. The emergence of CRF and Softmax as the most effective decoders again coincides with their frequency of use in the literature (see Figure 4.6c).

Larger LMs such as RoBERTa provide a performance advantage, but this improvement is not substantial, as shown by the CyNER versus BERT-CRF results in Section 4.2.2. Despite using much larger LLMs and being relatively simple to set up and run, the LLM-based baseline underperformed significantly compared to tailored CTI-NER approaches. Also, only two out of 101 approaches [FAR24; Hu+24b] in the SLR achieved their best results using a LLM.

Addressing RQ1, fine-tuning on CTI data to train token-level classifiers for the NER task is necessary and represents a defining characteristic that differentiates CTI-NER approaches from the baseline. However, NER4CTI shows that generally fine-tuning a LM on very limited training data, such as DNRTI with approximately 175 thousand words, does not improve performance and instead leads to degradation. In addition, such pretraining increases the risk of data leakage and artificially inflated performance scores, increasing the risk of poor real-world performance on truly unseen data.

Another observation is that tagging all sentences of a document jointly performs substantially worse than tagging each sentence independently without surrounding document context, as shown in Figure 4.13. One contributing factor could be that the SWD training dataset contains only 36 samples, which is likely insufficient to achieve performance comparable to the SSS dataset. In addition, longer input sequences introduce more noise, which can negatively affect tagging performance. For LLMs in particular, performance degrades markedly on SWD, likely because extended context increases the risk of hallucination and over-generation of entities.

With respect to RQ2, increasing sequence length to incorporate additional context and exploit larger context windows generally reduces performance. In contrast, encoder-based LMs with simple decoders achieve the strongest results. Distilled LLMs such as Gemma3:27b and DeepSeek-R1:32b perform particularly poorly when used with prompt-based approaches such as LangExtract [Ham+25]. It is worth noting that the LLMs may already have been pretrained on the datasets used, while BERT and RoBERTa have earlier

---

cutoff dates than the publication of these datasets. Moreover, the reasoning-enabled LLM DeepSeek-R1 underperforms compared to the non-reasoning LLM Gemma3. Despite its age, the statistical learner STUCCO performs on par with BERT-based approaches.

In response to RQ3, the cost-performance analysis reveals distinct Pareto frontiers for training and inference, with different models excelling depending on the optimization objective. For inference, STUCCO and CyNER dominate all other approaches. STUCCO achieves the fastest inference at under one millisecond per sentence while maintaining competitive performance, which makes it the optimal choice when minimizing runtime is the primary concern. CyNER is approximately 24 times slower, but it achieves the highest strict micro F1 score and exhibits the lowest energy consumption among models with available measurements.

For training, the Pareto frontier consists of three models. Twitter minimizes training cost at under 30 minutes and only 57 Wh, which makes it well suited for scenarios that require frequent retraining, such as evolving entity definitions. BERT-CRF provides a strong balance between cost and performance at approximately 30 minutes and 124 Wh while achieving one of the highest F1 scores. CyNER represents the high-accuracy endpoint, requiring around two hours and 613 Wh but delivering the best overall effectiveness.

Notably, Twitter occupies a unique position in the cost-performance landscape. It is Pareto-optimal for training but dominated during inference by both STUCCO and CyNER, which makes it particularly suitable for deployment scenarios that prioritize retraining flexibility over high-throughput prediction. Conversely, STUCCO's exceptional inference speed comes at the cost of longer training times of over three hours on CPU, positioning it as the preferred choice for stable production environments in which models are trained infrequently but must process large volumes of text efficiently.

This highlights a limitation of the cost measurements. The evaluated approaches were implemented and designed over a time span of roughly a decade, with some methods, such as STUCCO, only available for CPU execution. Over this period, both software and hardware have evolved substantially. Although all experiments were conducted on the same hardware, the approaches rely on different software stacks. The versions of technologies such as Python, PyTorch, TensorFlow and Hugging Face vary across implementations, as does the experience and expertise of the original developers. Consequently, the measured costs may partially reflect differences in engineering quality and implementation efficiency rather than differences in model or architectural quality alone.

## Future Work

While this thesis addresses all research questions and identifies which types of approaches perform best on a CTI-news dataset, several opportunities for future research remain. These opportunities arise primarily from the practical boundaries of the present study and indicate directions for extending and refining the proposed approach.

The current lack of a dataset that satisfies all criteria outlined in Section 3.2.3 represents a key starting point for improving CTI-NER evaluation. Future work should therefore focus on the manual annotation of large corpora of cybersecurity news articles, with multiple annotators labeling the same complete documents to ensure consistent tagging informed by full document context. In addition, the dataset split remains a critical aspect for future research. Methods for splitting NER datasets should be developed to prevent data leakage. One possible approach is to evaluate tagging performance exclusively on unseen entities.

If existing datasets are employed, assessment frameworks can be developed that outline methods for evaluating tagging consistency and quality and for providing key statistics to support rapid dataset selection. One possible approach is to use LLMs to annotate datasets without access to training examples, relying only on the predefined entity type definitions and their world knowledge. Then one could compare the resulting FPs and FNs. A high number of FPs would indicate that the LLM over-generates entities, whereas a high number of FNs would suggest that the LLM fails to recognize entities as annotated in the dataset. These measures could serve as initial indicators of dataset quality. An increase in FPs may indicate incomplete annotation, with relevant entities missing from the dataset, while an increase in FNs may suggest inconsistencies or inaccuracies in the existing annotations. When using a highly capable model with a large number of parameters, such indicators may be meaningful for dataset assessment.

Building on the potential of larger LLMs, additional evaluation directions emerge. In this work the baseline massively underperformed. Employing larger LLMs could provide

---

new insights into the baselines performance characteristics. Nevertheless, such models incur substantially higher costs, dramatically exceeding the already large cost differences observed in this study. Furthermore, the evaluation of LLMs with reasoning capabilities could be made more robust by comparing a single LLM with reasoning explicitly enabled and disabled, such as gpt-oss.

Moreover, the performance of the baseline dropped markedly as entity length increased. Future work could investigate this behavior in more detail. One possible approach is to compare the baseline performance with other prompt-based approaches using the same LLMs and to analyze the resulting differences. In addition, an in-depth span-boundary error analysis could provide further insights. Complementary to this, the use of a carefully constructed dataset containing long entities and handcrafted samples may also contribute to a deeper understanding of this effect.

This work showed that incorporating additional context and thus using longer input sequences, led to degraded performance. Future work could investigate whether this behavior is specific to the architectural components investigated in this study or whether alternative solutions exist that can tag multi-page documents in a single pass.

To make the cost measurements more robust, future work could upgrade all approaches to a unified SOTA software stack in order to eliminate bias introduced by differences in coding practices and software packages.

To further facilitate the automation of Security Operations Centers (SOCs) tasks, future work should not only focus on NER alone, but also on Relation Extraction (RE), as cybersecurity information exchange standards such as STIX 2.1 define not only entity types but also the relationships between them. This would enable the construction of KGs that allows SOC analysts to grasp coherencies even more quickly.

## Conclusion

This thesis conducted a SLR of 101 publications on CTI-NER, of which 12 provided publicly available implementation code for detailed evaluation. To support the breadth of the SLR, a LLM-based feature extraction pipeline was developed. The SLR revealed that BERT-based LM dominate the field, with BiLSTM encoders and CRF or Softmax decoders representing the most common architectural choices. Notably, only two publications reported achieving their best results using LLMs. The subsequent empirical evaluation of six selected approaches extends these findings and demonstrates a clear and consistent pattern: simpler architectures outperform more complex alternatives. Encoder-based LMs combined with a simple decoding layer, specifically CyNER with Softmax and BERT-CRF, achieve the highest effectiveness and are recommended for production systems requiring reliable entity extraction, while STUCCO presents a compelling option when long entity spans are prevalent. In contrast, LLM-based extractors consistently underperformed across all evaluated metrics and incurred substantially higher computational costs, which renders them unsuitable for CTI-NER. The expectation that document-level context would improve tagging performance is not supported. Instead, extended context often degrades results on the evaluated architectures, particularly for LLM-based approaches that are prone to hallucination and over-generation. A persistent weakness across all models is the inability to generalize effectively to unseen entities. All findings must be interpreted within the constraints of the evaluation. Dataset quality and annotation ambiguity limit the generalizability of the results and some cost-performance differences may reflect implementation overhead rather than true architectural distinctions. For practitioners designing CTI-NER datasets, consistent labeling emerges as the most important factor for model performance and sophisticated dataset splitting is necessary to prevent overfitting. Looking forward, the field requires higher-quality, consistently annotated datasets and more robust methods for incorporating document-level context before further architectural advances can be meaningfully evaluated. Until these foundational issues are addressed, encoder-based LMs with CRF or Softmax decoding remain the most effective and efficient

---

approaches for CTI-NER.

# LM-based Feature Extraction

Figure A.1 presents the developer prompt used for all feature extraction queries. Table A.1 lists the features of the data extraction form, along with the corresponding questions and guidelines applied for both manual and LM-based extraction. Rows highlighted in green indicate data extracted through the LLM.

Table A.1: Data Extraction Form

Feature Name	Question	Guideline
<b>Approach/Architecture</b>		
Approach name	How do the authors of the paper name their framework, method or tool?	I am focused on the name of the tool in general. If there are multiple tool names, e.g. one for the framework and one for the NER approach, write both down.
High-level architecture description	Describe the sequence of components used in the approach/architecture given the format defined in the guideline.	<p>Mixed approaches process data through components arranged in stages. Specify the NER architecture as a <b>single-line Sequence</b> using <b>dashes</b> between stages and <b>plus signs</b> (“+”) within a stage. The canonical stages are:</p> <ol style="list-style-type: none"> <li>1. <b>Embeddings</b></li> <li>2. <b>Model</b> (or <b>Encode</b>)</li> <li>3. <b>NER extraction</b> (or <b>Decode</b>) — include <b>post-processing</b> here via + (e.g., CRF+Rules)</li> </ol> <p><b>Sequence line (mandatory):</b> The <b>very first line</b> of <b>Answer</b> must be the Sequence. If a stage is truly absent, write None for that stage; if unspecified, mark Not reported.</p> <p><b>NER subsystem isolation rule</b></p> <p>The Sequence must be <b>NER-only</b> (representations → encoder(s) → tagger/decoder → post-processing). Upstream/downstream modules (e.g., retrieval, RAG, relevance filters/classifiers, relation extraction, summarization) are <b>out of scope</b>. If needed for orientation, list them once under “<b>Context (out of scope)</b>” without internal detail.</p>

Continued on next page

Feature Name	Question	Guideline
		<p><b>Bracket annotations (required for each component token)</b></p> <p>After <b>each component token</b>, add parentheses with the concrete mechanism/instantiation, e.g.: <code>characterEmbedding(CNN)</code>, <code>wordEmbedding(GloVe-6B)</code>, <code>contextualizedEmbeddings(BERT-base-uncased)</code>, <code>BiLSTM(2-layer)</code>, <code>Self-Attention(scaled-dot)</code>, <code>CRF(linear-chain)</code>, <code>rules(Regex,Gazetteer)</code>. If authors use a <b>generic name</b> (e.g., “contextualized embeddings”), keep it <b>and</b> add a clarifier showing <b>how it is created and what concepts are behind it</b>. If the bracketed mechanism is <b>not stated</b> in the paper and comes from general knowledge, append <code>*</code> to that bracket item (e.g., <code>contextualizedEmbeddings(LanguageModel*)</code>). Provide a matching [Background note] (<math>\leq 25</math> words) in <i>Explanation &amp; key points</i> and expand in <i>Long explanation</i>. Background items are <b>not evidence</b>.</p> <p><b>Component disambiguation &amp; role tags</b></p> <p>If a component appears in multiple roles, add <b>role tags</b> and list them separately (e.g., <code>BiLSTM[sequence](2-layer)</code> vs <code>BiLSTM[char](1-layer)</code>). <b>Do not merge</b> roles.</p> <p><b>Creation method (concise)</b></p> <p>For <b>each component</b>, state <b>how it is realized</b> (e.g., <i>char embeddings via CNN; subword via BPE/WordPiece; contextual via named LM; decoder via CRF; rules via regex/gazetteer</i>).</p> <p><b>Module wiring (one-to-two-line summary)</b></p> <p>In the <i>Long explanation</i>, briefly explain <b>how modules connect/fuse</b> (concat, residual, attention fusion, stacking) and <b>what each module contributes</b>. Keep this concise.</p> <p><b>Final-architecture rule</b></p> <p>If multiple variants exist, the Sequence must reflect <b>only the final/best NER model</b> used for main results. Add “<b>Not selected variants</b>” listing other NER component combos <b>tested but excluded</b>.</p> <p><b>Evidence mapping (concise)</b></p> <p>Provide at least <b>one quote per stage</b> (or per ambiguous token) from <b>body text or Markdown tables</b> (<math>\leq 120</math> chars). Do <b>not</b> quote figures/“Key insights”.</p> <p><b>Context (out of scope)</b></p> <p>Optionally list non-NER pipeline elements (retrieval, RAG, relevance classifier, relation extraction) in a simple bullet list <b>without</b> internal details.</p> <p><b>Output alignment</b></p> <p>Follow the global per-field format (<b>Answer; Explanation &amp; key points; Long explanation / description; Evidence</b>).</p> <ul style="list-style-type: none"> <li>• <b>Answer:</b> Begin with the <b>NER-only Sequence</b> (dash/plus with brackets).</li> <li>• <b>Explanation &amp; key points:</b> One sentence summarizing; include any [Background note] entries referenced by <code>*</code>.</li> <li>• <b>Long explanation / description:</b> 1–2 paragraphs on creation methods and wiring; add <b>Not selected variants</b> and optional <b>Context (out of scope)</b>.</li> <li>• <b>Evidence:</b> Minimal quotes that prove the components named in the Sequence.</li> </ul> <p><b>No preamble:</b> Apply all steps internally; <b>do not</b> print any plan/progress text.</p> <p><b>Two-pass is internal:</b> <b>Do not</b> echo the “Two-pass extraction” steps.</p> <p><b>Answer-first rule:</b> The response must begin with – <b>Answer:</b> ... (best test-set scores if available; else Not reported).</p> <p><b>Empty-case policy:</b> If no metrics/tables are present in the provided chunks:</p> <ul style="list-style-type: none"> <li>• Set <b>Answer = Not reported</b>.</li> <li>• In <b>Long explanation</b> add:             <ul style="list-style-type: none"> <li>– <b>Experiment catalog:</b> None found in provided chunks</li> <li>– <b>Table→Experiment mapping:</b> None</li> <li>– <b>Exported evaluation tables:</b> None</li> <li>– <b>Notes:</b> brief reason (e.g., only figures available; captions absent).</li> </ul> </li> </ul>

Continued on next page

Feature Name	Question	Guideline
Description	Describe the <b>NER architecture or approach</b> presented in this paper.	<p><b>Extraction question</b></p> <p>Describe the <b>NER architecture</b> presented in this paper.</p> <p><b>Guideline</b></p> <p>Provide a <b>sequential, but flexible</b> overview using the paper's own stage names. Explain how each part works by itself and how it integrates with the others. Include the <b>model name</b> if the authors provided one.</p> <p><b>Scope</b></p> <p>Describe <b>NER architecture only</b>. <b>Exclude</b> evaluation metrics, training/implementation details (hardware, schedules), dataset-selection procedures, and relevance/observation classifiers <b>unless</b> they directly define the NER architecture.</p> <p><b>NER subsystem isolation rule</b></p> <p>Extract and describe the <b>NER subsystem only</b> (representation → encoder(s) → tagger/decoder → post-processing). Treat upstream/downstream modules (e.g., retrieval, relevance filtering/classifiers, relation extraction, summarization) as <b>out of scope</b>. Mention them once in "<b>Context (out of scope)</b>" only if needed for orientation; <b>do not</b> describe their internals.</p> <p><b>Component disambiguation</b></p> <p>If a component appears in multiple roles (e.g., BiLSTM), create <b>separate entries</b> with <b>role tags</b> (e.g., BiLSTM[char-level], BiLSTM[sequence]). <b>Do not merge</b> these roles.</p> <p><b>Module card (expanded detail)</b></p> <p>For <b>each NER component</b>, write a <b>short paragraph</b> (≥3 sentences) covering:</p> <ul style="list-style-type: none"> <li>• <b>Inputs &amp; interface</b> (shapes/types if stated)</li> <li>• <b>Mechanics/algorithm</b> (how it works)</li> <li>• <b>Purpose</b> (authors' rationale)</li> <li>• <b>Wiring</b> (how it connects/fuses with prior/next modules)</li> <li>• <b>Key settings</b> (only if reported)</li> </ul> <p>If authors use only a generic name (e.g., "contextualized embeddings"), keep it and add a clarifier via [Background note] ≤25 words. Background notes are <b>explanatory only</b> (not evidence).</p> <p><b>Final-architecture rule</b></p> <p>If multiple variants exist, describe <b>only the final/best NER model</b> used for the main results. Add a subsection "<b>Not selected variants</b>" listing NER components/combo <b>tested but excluded</b>. If "best" is not explicit, choose the model tied to <b>primary results</b> or named the <b>main/full model</b>.</p> <p><b>Evidence mapping</b></p> <p>Provide at least <b>one quote per module</b> (≤120 chars) from <b>body text or Markdown tables</b>. Do <b>not</b> quote figures/"Key insights".</p> <p><b>Context (out of scope)</b></p> <p>Optionally list non-NER pipeline elements (e.g., retrieval, relevance classifier, relation extraction) in a simple bullet list <b>without</b> internal details.</p> <p><b>Output alignment</b></p> <p>Conform <b>exactly</b> to the global per-field format (<b>Answer; Explanation &amp; key points; Long explanation / description; Evidence</b>).</p> <ul style="list-style-type: none"> <li>• <b>Answer:</b> Start with the <b>NER-only sequence</b> line.</li> <li>• <b>Explanation &amp; key points:</b> One sentence summary + concise bullets highlighting the core NER blocks and their interactions.</li> <li>• <b>Long explanation / description:</b> The <b>module cards</b> (one per NER component), plus <b>Not selected variants</b> and (optional) <b>Context (out of scope)</b>.</li> <li>• <b>Evidence:</b> Per-module quotes (and any additional supporting quotes), drawn only from body text or Markdown tables.</li> </ul>

Continued on next page

Feature Name	Question	Guideline
Type of task	How is NER formulated (sequence-label, QA, etc.)?	<p><b>Primary formulation rule:</b> Report the <b>formulation used by the final/best CTI-NER model</b>. If multiple formulations are discussed, put the <b>primary</b> one in <b>Answer</b> and list others under “<b>Not selected formulations</b>” (with evidence).</p> <p><b>Inference from components (when unnamed):</b> If the paper <b>does not explicitly name</b> the formulation, <b>infer it</b> from the architecture/decoder and <b>cite</b> the components that justify it. Append (inferred from components) to the <b>Answer</b>. If still ambiguous, return Not reported.</p> <p><b>Disambiguation cues:</b></p> <ul style="list-style-type: none"> <li>• <b>Sequence labeling:</b> BIO/BILOU tags; token classifier or <b>CRF</b> over tag sequence</li> <li>• <b>Span classification:</b> start/end heads; candidate spans classified by type</li> <li>• <b>QA/MRC:</b> question/template per entity type; extractive answers as spans</li> <li>• <b>Generative extraction:</b> seq2seq/pointer network that <b>generates</b> entity strings</li> </ul> <p><b>Output alignment</b></p> <ul style="list-style-type: none"> <li>• <b>Answer:</b> single phrase + short clause (append inferred from components when applicable).</li> <li>• <b>Explanation &amp; key points:</b> one sentence stating <b>why</b> (point to the component/decoder).</li> <li>• <b>Long explanation / description:</b> rationale tying components to the chosen formulation and listing <b>Not selected formulations</b> (if any) with evidence.</li> <li>• <b>Evidence:</b> quotes naming decisive component(s) (e.g., “linear-chain CRF”, “start/end classifier”, “we prompt with templates ...”).</li> </ul>
Knowledge graph	Does the model query or create a knowledge graph? Describe briefly.	Note how the approach integrates/uses the KG.
Knowledge base	Does the model use a knowledge base? Describe briefly.	Note how the approach integrates/uses the KB. Ignore knowledge graphs (KGs).
Multilingual	Which languages are supported?	List ISO codes or names; “English-only” if unspecified. You may note training corpora languages if explicitly stated.
Entity types	Which entity types are recognised?	<p><b>BIO/IOB handling:</b> Strip B-/I-/S-/E- prefixes; exclude O/PAD/UNK/pipeline markers. Report base types only.</p> <p><b>Per-dataset separation:</b> If multiple datasets are used, list types per dataset; <b>do not</b> merge.</p> <p><b>Final model tie-in:</b> In <b>Answer</b>, report the types predicted by the final/best model. In <b>Long explanation</b>, include a per-dataset table and note differences.</p> <p><b>Provenance:</b> For each type (or set), state provenance (STIX 2.1 / Author-defined / Other standard).</p> <p><b>Unification:</b> If the final model unifies types, add “<b>Final model label set</b>” with a supporting quote.</p>
Relation extraction	Does the model also extract relations between entities?	Yes/No + brief explanation of the method.

#### Approach/Architecture Components

Continued on next page

Feature Name	Question	Guideline
Rule-based components	Are rule-based components a part of the proposed approach? If so, name/describe them.	<p><b>Answer-first:</b> Begin <b>Answer</b> with Yes / No / Not reported -- &lt;one-line summary&gt;, e.g., Yes -- regex patterns + vendor gazetteer for organization entities.</p> <p><b>Final-first rule:</b> In <b>Answer</b>, list <b>rule-based components used by the final/best CTI-NER model first</b>; place others under “<b>Not selected / auxiliary</b>” (with evidence).</p> <p><b>Synonym normalization:</b> Treat <i>rules/heuristics/patterns/templates, dictionary/lexicon/gazetteer/whitelist/blacklist, ontology mapping, labeling functions (LFs)</i> as <b>rule-based</b> when deterministic.</p> <p><b>Not algorithms (scope boundary):</b> Exclude <b>deterministic algorithms</b> (e.g., Aho–Corasick, KMP, Rabin–Karp, deterministic tokenizers). Those belong to <b>Non-Machine Learning (ML) deterministic algorithms</b>. This feature covers <b>handcrafted/programmatically generated rules and lookups</b>.</p> <p><b>Illustrative, not limiting:</b> regex/pattern templates; gazetteers/lexicons (vendor lists, CWE/CVE-derived); ontology-based mapping (e.g., STIX label → entity type); dictionary lookups; rule engines; <b>Snorkel labeling functions</b>.</p> <p><b>Role tags (where used):</b> Tag each component with its locus: [preprocess], [extraction], [post].</p> <p><b>Describe each component (when stated):</b></p> <ul style="list-style-type: none"> <li>• <b>What it matches</b> (entity types/strings/pattern families)</li> <li>• <b>Conflict resolution</b> (priority, longest-match, fallback)</li> <li>• <b>Coverage/precision notes</b> (e.g., “high precision for vendor names”)</li> <li>• <b>Source/provenance</b> (gazetteer/ontology origin, update policy)</li> </ul> <p><b>Generated rules:</b> If rules are <b>machine-induced</b> (e.g., CNN/LM creates regex, Snorkel LFs), note the <b>generator</b> and <b>selection criteria</b>.</p> <p><b>Inference-time vs data-only:</b> If rules are used <b>only</b> for augmentation/weak labeling, state “<b>Labeling/augmentation only (not used at inference)</b>”.</p> <p><b>Evidence:</b> Provide <b>≥1 verbatim quote per rule component</b> (≤120 chars).</p> <p><b>Background note (optional):</b> If a gazetteer/ontology is named but not described, add [Background note] ≤25 words (explanatory only; not evidence).</p> <p><b>Output alignment</b></p> <p>Conform <b>exactly</b> to the global per-field format (<b>Answer; Explanation &amp; key points; Long explanation / description; Evidence</b>).</p> <ul style="list-style-type: none"> <li>• <b>Answer:</b> Yes/No/Not reported -- &lt;summary&gt;, then a concise, comma-separated list of final/best rule components with <b>role tags</b>, e.g., regex[extraction] (IoCs patterns), gazetteer[preprocess] (vendor list), ontology-mapping[post] (STIX→entity)</li> <li>• <b>Explanation &amp; key points:</b> One sentence on <b>why</b> rules are used and where; bullets if helpful (conflict policy, coverage, provenance).</li> <li>• <b>Long explanation / description:</b> Per component: role, what it matches, conflict policy, coverage/precision notes, provenance, whether generated or handcrafted, and whether used at inference or only for labeling. Include <b>Not selected / auxiliary</b> rules if present.</li> <li>• <b>Evidence:</b> One quote <b>per component</b> (≤120 chars).</li> </ul>

Continued on next page

Feature Name	Question	Guideline
Statistical learners	Are statistical learners a part of the proposed approach? If so, name/describe them and their peculiarities.	<p><b>Guideline</b></p> <p><b>Final-first rule:</b> In <b>Answer</b>, list <b>statistical learners used by the final/best CTI-NER model first</b>. Place learners <b>tested but not used</b> under “<b>Not selected learners</b>” (with evidence).</p> <p><b>Include CRF when it’s the decoder:</b> If a <b>CRF</b> is used on top of a neural encoder as the <b>NER decoder</b>, it <b>counts as a statistical learner</b>. List it here (e.g., <code>CRF[decoder]</code> (<code>linear-chain</code>, <code>Viterbi</code>)).</p> <p><b>Role tagging:</b> If the same learner appears in multiple roles, add a role tag, e.g., <code>SVM[filter]</code>, <code>CRF[decoder]</code>, <code>MaxEnt[tagger]</code>.</p> <p><b>Synonym normalization (recognition):</b> Treat <b>Maximum Entropy / MaxEnt / log-linear</b>, <b>CRF / conditional random field (linear-chain)</b>, <b>HMM / n-th order HMM</b>, <b>SVM / support vector machine</b>, <b>Perceptron / averaged perceptron</b>, <b>Naive Bayes</b>, <b>Logistic regression</b> as equivalent names.</p> <p><b>Illustrative, not limiting:</b> <code>CRF (linear-chain/higher-order)</code>, <code>HMM</code>, <code>MaxEnt/log-linear</code>, <code>SVM</code>, <code>Perceptron</code>, <code>Naive Bayes</code>, <code>Logistic regression</code>; <b>TF-IDF/BM25</b> when used as <b>features</b> or <b>candidate generator</b> for NER (otherwise see <i>Context</i>).</p> <p><b>Peculiarities to capture (when stated):</b> order (<code>CRF/HMM</code>), feature templates, decoding (<code>Viterbi/beam</code>), kernel &amp; <math>C</math> (<code>SVM</code>), regularization, calibration, class weighting, constraints, <b>history-based</b> tagging (<code>MaxEnt</code>).</p> <p><b>Scope boundary:</b> <b>Exclude</b> deep neural modules and LMs/Transformers. Include <b>TF-IDF/BM25 only if they feed the NER learner</b>; otherwise list them once under <b>Context (out of scope)</b>.</p> <p><b>Evidence:</b> Provide <b>≥1 verbatim quote per learner</b> (≤120 chars) from <b>body text</b> or <b>Markdown tables</b>. No figures/captions.</p> <p><b>Output alignment</b></p> <p>Conform <b>exactly</b> to the global per-field format (<b>Answer; Explanation &amp; key points; Long explanation / description; Evidence</b>).</p> <ul style="list-style-type: none"> <li>• <b>Answer:</b> concise, comma-separated list with brief clarifiers, e.g., <code>CRF[decoder]</code> (<code>linear-chain</code>, <code>first-order</code>; <code>Viterbi</code>), <code>MaxEnt[tagger]</code> (<code>history-based</code>), <code>SVM[filter]</code> (<code>RBF kernel</code>).</li> <li>• <b>Explanation &amp; key points:</b> One sentence summarizing which learners are <b>in the final model</b>, plus bullets for key peculiarities (if helpful).</li> <li>• <b>Long explanation / description:</b> For each learner: role, inputs/features, peculiarities (order, kernel, decoding, constraints), and how it integrates with the NER pipeline. Add <b>Not selected learners</b> (if any).</li> <li>• <b>Evidence:</b> One quote <b>per learner</b> from body text/Markdown tables.</li> </ul> <p><b>Context (out of scope)</b></p> <p>If TF-IDF/BM25 or other scoring is used <b>only</b> for retrieval unrelated to NER, mention it here in one bullet without details.</p>

Continued on next page

Feature Name	Question	Guideline
Prompt engineering	Does the approach employ natural language inputs/outputs to extract entities? If so, describe the prompt design.	<p><b>Answer-first summary:</b> Start <b>Answer</b> with Yes / No / Not reported -- &lt;one-line prompt style&gt;, e.g., Yes -- few-shot instruction prompt with JSON schema.</p> <p><b>Exhaustive listing (final-first):</b> If multiple prompts are used, <b>list all</b>. Put <b>final/best inference prompts first</b>; move others to “<b>Not selected / auxiliary prompts</b>” (e.g., augmentation, weak labeling, evaluation checker).</p> <p><b>Prompt taxonomy:</b> Classify as applicable: <b>Inference, Training/finetune templates, Augmentation/labeling, Retrieval-augmented (RAG), Evaluation/checker, Post-processing/validation</b>.</p> <p><b>Design facets to capture (when stated):</b></p> <ul style="list-style-type: none"> <li>• <b>Purpose</b> (what the prompt achieves)</li> <li>• <b>Template structure</b> (slots/variables, question templates)</li> <li>• <b>Entity schema encoding</b> (label list, verbalizers, definitions)</li> <li>• <b>Format constraints</b> (JSON schema, tag spans, delimiter tokens)</li> <li>• <b>Example count</b> (k-shot; positions of exemplars)</li> <li>• <b>Instruction style</b> (instructional vs Q/A vs generative extraction)</li> <li>• <b>Decoding params</b> (temperature/top-p/max tokens/stop)</li> <li>• <b>Context sources</b> (RAG retrieval, section-limited context)</li> <li>• <b>Safety/guardrails</b> (bans, refusal handling)</li> <li>• <b>Prompt generation</b> (handcrafted, programmatic, auto-generated)</li> </ul> <p><b>Example prompt policy:</b> If an <b>example prompt</b> is provided, include it (or <b>multiple excerpts</b> ≤120 chars each).</p> <p><b>Scope boundary:</b> Count <b>only natural-language inputs/outputs</b> used for <b>entity extraction</b> or its immediate setup. <b>Exclude</b> bare regex/rules unless they are part of an NL prompt; exclude UI strings/config keys.</p> <p><b>Disambiguation:</b> If style is <b>not explicitly named</b>, <b>infer from usage</b> and append (inferred from components); support with quotes for the premises.</p> <p><b>Not inference-time:</b> If prompts are used <b>only</b> for augmentation/labeling, state “<b>Augmentation only (not used at inference)</b>”.</p> <p><b>Evidence:</b> Provide ≥1 verbatim quote per prompt type (≤120 chars) from <b>body text</b> or <b>Markdown tables</b>.</p> <p><b>Output alignment</b></p> <p>Conform <b>exactly</b> to the global per-field format (<b>Answer; Explanation &amp; key points; Long explanation / description; Evidence</b>).</p> <p><b>Expected content</b></p> <ul style="list-style-type: none"> <li>• <b>Answer:</b> Yes/No/Not reported -- &lt;one-line prompt style&gt; and, if Yes, a concise list of prompt types used by the <b>final/best</b> model first (e.g., Inference: few-shot JSON; RAG: section-limited).</li> <li>• <b>Explanation &amp; key points:</b> One sentence summarizing <b>what the prompts do and why</b>; bullets (if helpful) for schema, examples, constraints, decoding params.</li> <li>• <b>Long explanation / description:</b> <ul style="list-style-type: none"> <li>– <b>Per prompt type:</b> purpose, template (slots), schema encoding, examples (k-shot), constraints (JSON/tags), decoding params, context retrieval, safety/guardrails, and <b>how/where in the pipeline</b> it’s applied.</li> <li>– <b>Not selected / auxiliary prompts:</b> augmentation/labeling or checker prompts not used at inference.</li> <li>– Mark (inferred from components) where applicable.</li> </ul> </li> <li>• <b>Evidence:</b> Short prompt excerpts (≤120 chars each) and any lines naming the prompt type/usage.</li> </ul>

Continued on next page

Feature Name	Question	Guideline
Reinforcement Learning	Is Reinforcement Learning (RL) applied? If so, summarise algorithm and purpose.	<p><b>Answer-first:</b> Begin <b>Answer</b> with Yes / No / Not reported -- &lt;one-line RL summary&gt;, e.g., Yes -- PPO fine-tuning of decoder to maximize span-F1.</p> <p><b>RL definition (scope):</b> Count <b>only</b> methods with an explicit <b>policy, actions, rewards</b>, and a named <b>RL algorithm</b> (policy gradient or value-based).</p> <p><b>Synonyms (recognition boost):</b> Consider <b>REINFORCE/policy gradient, actor-critic (A2C/A3C), PPO/TRL, QLearning/DQN, bandits, reward shaping, RLHF/RLAIF, DPO/KTO</b> as RL family <b>when the paper frames them as RL</b>.</p> <p><b>Not RL (explicit exclusions):</b> Exclude <b>self-training, curriculum learning, early stopping, hard-negative mining, loss reweighting, scheduled sampling</b> (unless explicitly cast as RL), <b>maximum-margin</b> methods, and vague “reinforcement signals” without a named RL algorithm.</p> <p><b>Final-first rule:</b> List RL used by the <b>final/best CTI-NER model</b> first; place other RL uses under “<b>Not selected / auxiliary</b>” (e.g., for augmentation or ablations).</p> <p><b>Where applied:</b> State the locus: <b>decoder/tagger, prompting</b> (prompt optimization), <b>retrieval/reranking, data selection/augmentation</b>, or <b>training control</b>. Note whether RL affects <b>inference</b> or <b>training only</b>.</p> <p><b>What to capture (when stated):</b></p> <ul style="list-style-type: none"> <li>• <b>Algorithm &amp; variant:</b> PPO/REINFORCE/A2C/DQN/bandit, etc.</li> <li>• <b>State:</b> token/span sequence, prompt template, candidate entities, etc.</li> <li>• <b>Action space:</b> tag assignment, span proposal, prompt edit, retrieval choice.</li> <li>• <b>Reward:</b> span-F1, exact-match, boundary reward, latency/precision trade-off, human feedback.</li> <li>• <b>Objective &amp; purpose:</b> e.g., improve boundary recall, reduce hallucinations, calibrate precision.</li> <li>• <b>Regime:</b> on/off-policy; batch/online; key <b>hyperparams</b> (<math>\gamma</math>, clip <math>\varepsilon</math>, KL <math>\beta</math>) if provided.</li> <li>• <b>Integration:</b> where RL sits in the NER pipeline and how it interacts with other components.</li> </ul> <p><b>Evidence:</b> Provide <math>\geq 1</math> <b>verbatim quote per RL use</b> (<math>\leq 120</math> chars) naming the <b>algorithm</b> and/or <b>reward</b>.</p> <p><b>Background note (optional):</b> If an RL method is named without explanation, add [Background note] <math>\leq 25</math> words. Background is <b>not evidence</b>.</p> <p><b>Output alignment</b></p> <p>Follow the global per-field format (<b>Answer; Explanation &amp; key points; Long explanation / description; Evidence</b>).</p> <p><b>Scope (final first):</b> Report deep-learning components used by the <b>final/best CTI-NER model first</b>. Then list components <b>tested but not used</b> in a short “<b>Not selected variants</b>” note (with evidence).</p> <p><b>Allowed (illustrative, not limiting):</b> <b>CNN/Char-CNN, (Bi)LSTM, GRU, vanilla RNN, MLP/FFN heads, attention modules attached to RNN/CNN</b> (additive/dot-product), <b>gating/fusion networks</b>.</p> <p><b>Explicit exclusions:</b> Exclude all <b>Transformer-based</b> models/layers (e.g., <b>BERT, RoBERTa, DeBERTa, GPT, Transformer encoders/decoders, self-attention stacks</b>), <b>LLMs/PLMs</b>, and <b>pretrained LM embeddings</b>. Also exclude <b>non-deep</b> components (e.g., <b>CRF, HMM, SVM, rules, gazetteers</b>). If the paper states <b>BiLSTM-CRF</b>, report <b>BiLSTM only</b> here; CRF is out of scope. Transformer/LM items belong to the <b>Embeddings/features</b> feature and must <b>not</b> appear here.</p> <p><b>Synonyms:</b> Treat <i>neural network, sequence model, recurrent model</i> as <b>equivalent</b>.</p> <p><b>Role tagging:</b> If a component appears in multiple roles, add a role tag, e.g., BiLSTM[sequence], CNNchar.</p> <p><b>Creation/realization:</b> When the paper explains <b>how</b> a component is realized, include it in parentheses, e.g., CNNchar (conv-k3), BiLSTM (2-layer), Attention (additive).</p>
Deep-learning components	Are deep-learning components a part of the proposed approach? If so, name/describe them.	<p><b>Scope (final first):</b> Report deep-learning components used by the <b>final/best CTI-NER model first</b>. Then list components <b>tested but not used</b> in a short “<b>Not selected variants</b>” note (with evidence).</p> <p><b>Allowed (illustrative, not limiting):</b> <b>CNN/Char-CNN, (Bi)LSTM, GRU, vanilla RNN, MLP/FFN heads, attention modules attached to RNN/CNN</b> (additive/dot-product), <b>gating/fusion networks</b>.</p> <p><b>Explicit exclusions:</b> Exclude all <b>Transformer-based</b> models/layers (e.g., <b>BERT, RoBERTa, DeBERTa, GPT, Transformer encoders/decoders, self-attention stacks</b>), <b>LLMs/PLMs</b>, and <b>pretrained LM embeddings</b>. Also exclude <b>non-deep</b> components (e.g., <b>CRF, HMM, SVM, rules, gazetteers</b>). If the paper states <b>BiLSTM-CRF</b>, report <b>BiLSTM only</b> here; CRF is out of scope. Transformer/LM items belong to the <b>Embeddings/features</b> feature and must <b>not</b> appear here.</p> <p><b>Synonyms:</b> Treat <i>neural network, sequence model, recurrent model</i> as <b>equivalent</b>.</p> <p><b>Role tagging:</b> If a component appears in multiple roles, add a role tag, e.g., BiLSTM[sequence], CNNchar.</p> <p><b>Creation/realization:</b> When the paper explains <b>how</b> a component is realized, include it in parentheses, e.g., CNNchar (conv-k3), BiLSTM (2-layer), Attention (additive).</p>

Continued on next page

Feature Name	Question	Guideline
Embeddings/ Features used	Which embeddings or handcrafted features feed the model?	<p>List <b>distinct embeddings/features</b> that feed the NER model.</p> <p><b>Answer priority:</b> List <b>embeddings/features used by the final/best CTI-NER model first</b>, then list <b>all other tested embeddings/features</b>.</p> <p><b>Synonyms:</b> Treat <i>embeddings, features, handcrafted features, featurizers, lexicons/gazetteers</i> as <b>equivalent</b>.</p> <p><b>Illustrative, not limiting:</b> GloVe, fastText, Word2Vec, ELMo, BERT/RoBERTa/DeBERTa, domain-adapted LMs, character CNN, subword BPE, TF-IDF, POS, orthographic, <b>gazetteers/lexicons</b>.</p> <p><b>Canonicalization &amp; dedup:</b> Use <b>canonical names</b> (e.g., “GloVe 6B”, “BERT-base-uncased”, “fastText”). <b>Deduplicate</b> aliases/synonyms.</p> <p><b>Creation method (when special):</b> If authors use a <b>special creation/derivation</b> (e.g., domain-adapted LM, continued pretraining, retrofitting, in-house fastText), briefly state <b>how it was created</b> (corpus/procedure) <b>as part of the item</b>.</p> <p><b>Fusion note:</b> If multiple embeddings feed the model, indicate <b>fusion</b> (e.g., concat, sum, attention gating) <b>when stated</b>.</p> <p><b>Two-pass recall sweep:</b> Scan <b>Methods/Architecture/Implementation</b> and <b>Appendix tables</b>; perform a <b>second pass</b>.</p> <p><b>Evidence per item or group:</b> Provide <b>≥1 verbatim quote</b> (≤120 chars) for each <b>distinct embedding/feature</b> (or one quote for a group).</p>
<b>Approach/Architecture Components: Language Model</b>		
LM-based	Does the system rely on a language model (LM)? Name it.	Look for e.g. BERT, Mistral, Meta (Llama) or OpenAI (e.g. GPT-4o) models. Note links to their repositories (e.g., GitHub, HuggingFace). Name all the LMs evaluated and state which one is the best. Do not just write the general term but its precise variation (e.g., bert-base-cased).

Continued on next page

Feature Name	Question	Guideline
LM integration	How is the language model (LM) integrated (prompting, fine-tune, embeddings)? Is a pretrained language model (LM) used with the NER approach? If so, state <b>exactly how it is integrated and trained</b> (embeddings-only vs frozen encoder vs partial fine-tune vs full fine-tune) and whether <b>**quantization**</b> is used.	<p><b>Axis A — Lineage / Pre-adaptation (prior to NER):</b> - <b>DAPT (domain-adaptive pretraining):</b> extra <b>unsupervised LM</b> training on a <b>domain corpus</b>. - <b>TAPT (task-adaptive pretraining):</b> extra <b>unsupervised LM</b> training on the <b>task's unlabeled data</b>. - <b>STILTs / Intermediate-task training:</b> supervised training on a <b>labeled intermediate task</b> (e.g., NLI) prior to NER. - <b>Other:</b> describe (e.g., continued MLM on custom CTI text).</p> <p>Capture: corpus, objective (e.g., MLM), steps/epochs if stated. If none stated: Lineage=None. (Background references: DAPT/TAPT; STILTs)</p> <p><b>Axis B — NER-stage Training Regime (final/best model):</b> - <b>(E1) Embeddings-only (feature-based):</b> LM used to precompute static features; <b>not</b> in the training graph; no LM params updated. - <b>(E2) Encoder (frozen):</b> LM in the model; <b>all</b> LM layers frozen; train <b>head-only</b>. - <b>(E3) Partial fine-tune:</b> some LM params updated (e.g., <b>adapters/LoRA/QLoRA/IA3, last N layers, prefix/prompt-tuning</b>). - <b>(E4) Full fine-tune:</b> <b>all</b> LM layers trainable. - <b>Quantization:</b> note <b>int8/int4</b> and tool (e.g., <b>bitsandbytes</b>). QLoRA = E3 with Quantization=int4(bnb) and Adapters=QLoRA. - “End-to-end / jointly trained / all layers updated” → map to E4 only if gradients through LM are implied; otherwise classify per evidence.</p> <p>Final-first rule: Report the <b>final/best CTI-NER model</b> first; put other experimented lineages/regimes under “<b>Not selected setups</b>”.</p> <p><b>Recognition Cues (Generic):</b> - <i>Lineage:</i> “continued/further pretraining,” “MLM on domain/task data,” “intermediate task (e.g., NLI) before NER.” - <i>NER regime:</i> “frozen/linear probe/head-only,” “last N layers,” “adapters/LoRA/QLoRA,” “end-to-end/jointly trained.”</p> <p><b>Evidence:</b> Provide <b>≤1 short quote per asserted aspect</b> (lineage type + corpus/objective; NER regime; adapters/layers; quantization). Quotes must be from <b>body text or Markdown tables</b> (≤120 chars). No figures/captions.</p> <p><b>Ambiguity Handling:</b> If any field is unclear, set it to <b>Not reported</b> and briefly explain in <i>Long explanation</i>. You may add a <b>[Background note]</b> (≤25 words) to clarify concepts (explanatory only; not evidence).</p> <p><b>Output Alignment:</b> Follow the global per-field format: Answer; Explanation &amp; key points; Long explanation / description; Evidence.</p> <p><b>Examples (Illustrative Only):</b> - Yes — LM=BERT-base-uncased; Lineage=DAPT(MLM on CTI reports) → NER=Encoder(frozen); Trainable=head-only; Adapters=None; Quantization=None - Yes — LM=RoBERTa-large; Lineage=STILTs(NLI) → NER=Full; Trainable=all; Adapters=None; Quantization=None - Yes — LM=Llama-7B; Lineage=None → NER=Partial; Trainable=LoRA(τ=16); Adapters=QLoRA; Quantization=int4(bnb)</p>
Reasoning Language Model	Is a reasoning-enhanced language model (LM) used? Name it.	E.g., DeepSeek R1, OpenAI o3.
Context window	What input token limit is supported?	<p><b>Scope:</b> Report the <b>inference-time input token limit</b> used for <b>CTI-NER</b>. Ignore training-time truncation, batch sizes, or non-NER tasks.</p> <p><b>Normalization:</b> Return an <b>integer number of tokens</b> as the <b>Answer</b>. If multiple limits are reported, choose the <b>bottleneck</b> relevant to <b>CTI-NER</b> on the <b>final/best architecture</b>.</p> <p><b>Chunking:</b> If chunking/sliding windows are used, the <b>Answer</b> is the <b>per-chunk token limit</b>. In the <b>Long explanation</b>, note that longer documents are handled via chunking.</p> <p><b>Granularity:</b> If both document- and sentence-level limits appear, choose the limit <b>actually used at inference</b> for <b>CTI-NER</b>.</p> <p><b>Not reported handling</b></p> <p>If no limit is stated in the provided context, set <b>Answer = Not reported</b> and add a <b>[Background estimate]</b> (non-evidential) in <b>Explanation &amp; key points</b>; expand in <b>Long explanation</b>.</p>
<b>Evaluation design</b>		
Cost evaluation	Is computational cost analysed? How measured?	If “Yes”, describe how it was measured and present the results.

Continued on next page

Feature Name	Question	Guideline
Quantitative results	Which evaluation metrics are reported <b>and</b> for which evaluation task (experiment) was each metric used? Provide <b>all evaluation results</b> (all metric values), and state the <b>task/experiment</b> where each value was obtained.	<p><b>Two-pass extraction:</b></p> <ol style="list-style-type: none"> <li><b>Identify</b> all evaluation tasks/experiments (dataset, split, subset, setting, model variant).</li> <li><b>Collect</b> every reported <b>metric</b> → <b>value</b> for each identified experiment.</li> </ol> <p><b>Per-experiment separation: Do not merge</b> results across tasks/datasets/splits.</p> <p><b>Metric semantics:</b> Capture averaging (micro/macro), span level (token/entity), and matching policy (exact/partial/strict) when stated.</p> <p><b>Answer priority:</b> In <b>Answer</b>, list <b>best test-set scores per metric</b> for the final/best model first (with dataset/split/experiment).</p> <p><b>All values required:</b> In <b>Long explanation</b>, include <b>every reported value for every experiment</b>.</p> <p><b>Table→Experiment mapping (bullets):</b> Before exported tables, add bullets: <b>Table label, Experiment tags</b> (dataset, split, subset, variant), and <b>Association confidence</b>. If ambiguous, mark “caption/association uncertain.”</p> <p><b>Exported evaluation tables:</b> Insert the <b>Markdown tables exactly as retrieved. Do not reformat</b>. If a <b>caption</b> is in adjacent text, place it immediately <b>above</b> the table as an <i>italic</i> line.</p> <p><b>No inference of numbers: Do not</b> fabricate or estimate metrics.</p> <p><b>Two-track structure (no mixing):</b> Split output into <b>Author-stated conclusions</b> and <b>Model-inferred conclusions</b>.</p> <p><b>Evidence (authors):</b> For each <b>Author-stated</b> conclusion, include a <b>verbatim quote</b> (≤120 chars) from <b>body text</b> or <b>Markdown tables</b>.</p> <p><b>Model-inferred items:</b> Include <b>Premises, Reasoning (1–2 lines), Confidence</b>. Tie inferences to named experiments/variants.</p> <p><b>Conflicts &amp; scope:</b> If results conflict across datasets/splits, state the conflict and constrain scope accordingly.</p>
Interpretation	What conclusions do the authors explicitly draw from their evaluation results? What additional conclusions can reasonably be inferred?	<p><b>No inference of numbers: Do not</b> fabricate or estimate metrics.</p> <p><b>Two-track structure (no mixing):</b> Split output into <b>Author-stated conclusions</b> and <b>Model-inferred conclusions</b>.</p> <p><b>Evidence (authors):</b> For each <b>Author-stated</b> conclusion, include a <b>verbatim quote</b> (≤120 chars) from <b>body text</b> or <b>Markdown tables</b>.</p> <p><b>Model-inferred items:</b> Include <b>Premises, Reasoning (1–2 lines), Confidence</b>. Tie inferences to named experiments/variants.</p> <p><b>Conflicts &amp; scope:</b> If results conflict across datasets/splits, state the conflict and constrain scope accordingly.</p>
Evaluation strategy	How is the evaluation strategy designed?	Describe the experiments used to evaluate the model: high-level tests, folding, train/validation/test splits, ablation studies, significance tests.
Embedding-effect study	Did authors analyse embedding choice?	Yes/No; if Yes add a rationale.
<b>Dataset</b>		
Dataset available	Is the dataset publicly available? Provide link if given.	<p>Yes/No + URL.</p> <p><b>Provenance vs modifications:</b></p> <ul style="list-style-type: none"> <li><b>Original dataset:</b> name, source/provider, version/date (if stated).</li> <li><b>Author-modified dataset:</b> describe modifications (filtering, relabeling, merging, dedup, normalization).</li> </ul> <p>If merged, report each source dataset separately (role &amp; contribution). Mention non-CTI evaluations briefly in <i>Other evaluations</i>.</p>
Data source type	What text corpora compose the dataset?	<p><b>Scope (final dataset focus):</b> Report sources for the <b>CTI/cybersecurity dataset</b> used by the final/best model. For merges, list each source dataset separately.</p> <p><b>Per-source details (when stated):</b></p> <ul style="list-style-type: none"> <li><b>Category</b> (CTI report / vendor bulletin / blog / news / social / forum / dark web / other)</li> <li><b>Named provider or domain</b> (e.g., “Symantec”, “microsoft.com”)</li> <li><b>Time range, Language(s)</b></li> <li><b>Count / proportion</b> of documents</li> </ul> <p>Provide ≥1 quote per distinct source or source group.</p>
Class-imbalance considered	Did authors discuss class imbalance?	<p>If yes, describe approach and impact. Keep provenance vs modifications clear for reused/merged datasets. Mention non-CTI evaluations only briefly.</p>

Continued on next page

Feature Name	Question	Guideline
Dataset size	How many samples are included in the dataset?	Report size in the unit used by the authors (e.g., tokens, sentences, CTI reports). If split sizes are shown, report train/validation/test distinctly. Keep provenance notes for reused/merged datasets.
Sample size	What is the average token length per sample in the dataset?	<p><b>Scope:</b> For the CTI dataset used by the final/best model. For multiple datasets or merges, report each source separately; compute overall only if all counts provided.</p> <p><b>Granularity:</b> Derive <i>sample</i> from the paper's wording (sentence/document/span). Use the paper's term in the <b>Answer</b> with units (e.g., 20.1 tokens/sentence).</p> <p><b>Derivation allowed (from evidence):</b> If not given, derive using only paper-provided totals (show formula &amp; inputs).</p> <p><b>Tokenization semantics:</b> Use the authors' token definition; if unspecified, mark tokenization unspecified.</p> <p><b>Splits/tasks:</b> Return the full table of averages per split/task; do not collapse across splits.</p>
Dataset description	Note further dataset characteristics.	Add characteristics not covered elsewhere (not augmentation, source type, class-imbalance, size, tagging). E.g., dataset created by merging multiple sources; describe the merge at a high level.
Manually tagged	Was data manually (by humans) tagged?	Yes/No. Apply CTI dataset focus; keep provenance vs modifications clear for reused/merged datasets.
Tagging strategy	What tagging schema (e.g., BIO) does the dataset use?	Common: BIO, BIOES, or entity-only spans. Report per CTI dataset; keep provenance notes for reused/merged datasets.
<b>Code</b>		
Code available	Is source code shared? Give URL.	Yes/No + URL (usually GitHub).
Language & frameworks	Primary programming stack?	<p><b>Scope:</b> Report the <b>primary programming stack for the final/best CTI-NER implementation</b> only. Put alternatives under “<b>Not selected / prototypes</b>”.</p> <p><b>Examples:</b> Python, Java, <b>PyTorch</b>, <b>TensorFlow</b>, <b>Hugging Face Transformers</b>, spaCy, NLTK, Flair, Stanza, Apache OpenNLP.</p> <p><b>Canonicalization &amp; dedup:</b> Use canonical names (e.g., “Hugging Face Transformers”). Deduplicate synonyms.</p> <p><b>Inference policy:</b> If a library implies a language (e.g., OpenNLP ⇒ Java) but language is unstated, report the library and add language only as a [Background note].</p>
Execution platform	Declared hardware/software environment?	<p><b>Scope:</b> Hardware/accelerators, drivers/toolkits, OS used for the <b>final/best CTI-NER results</b>. Put other machines/cloud instances under “<b>Not selected environments</b>”.</p> <p><b>Examples:</b> <b>NVIDIA A100 80GB</b>, <b>CUDA 12.2</b>, <b>cuDNN 9</b>, <b>Ubuntu 22.04</b>, <b>Linux kernel 6.5</b>, <b>Windows 11 23H2</b>, <b>ROCm 6.0</b>.</p> <p>Exclude programming languages/libraries here (they belong to <i>Language &amp; frameworks</i>).</p>

Continued on next page

Feature Name	Question	Guideline
Hyper-parameters	List hyperparameters explicitly reported.	<p><b>Scope:</b> Collect all explicitly reported hyperparameters for the final/best CTI-NER model.</p> <p><b>Per-component grouping:</b> Define short component aliases (e.g., enc, tagger, crf, emb_word, emb_char). Do not mix hyperparameters across components.</p> <p><b>Answer format (qualified pairs):</b> Return comma-separated pairs as component.param=value. If unambiguous to a single component, the alias may be omitted.</p> <p><b>Embeddings included:</b> Include embedding hyperparameters when explicitly reported (dims, vocab/casing, BPE merges).</p> <p><b>Exhaustiveness check:</b> Sweep Methods, Implementation/Training details, and Appendix tables.</p> <p><b>Normalization:</b> Keep numbers as numbers; include units in values if given (e.g., dropout=0.1, max_seq_len=512 tokens).</p> <p><b>Evidence per item:</b> Provide one quote (<math>\leq 120</math> chars) for each unique param=value (or a single quote that lists multiple params). Quote only body text or Markdown tables.</p> <p><b>Not reported handling:</b> If a component has no hyperparameters stated, mark it in the Long explanation as component: None reported.</p>
<b>Research Context</b>		
CTI sub-field/Research objective	Which CTI use-case? What problem does the approach solve?	Record what the NER approach is intended to accomplish (e.g., log analysis; news analysis; converting unstructured CTI to STIX/MISP; KG construction; performance improvements).
Research gaps	Which gaps do authors identify?	List them.
Limitations	Author-stated limitations?	
Challenges	What challenges were encountered?	
Strengths	Claimed advantages?	

You are an **NLP expert specialized in Named Entity Recognition (NER)** and also a **cybersecurity domain expert**.

Extract the requested fields **strictly** from the **provided PDF context** (vector-store content only). **Source format note:** Papers are stored in **Markdown**. Treat **body text** and **Markdown tables** as authoritative sources.

**This is a one-shot request** — **do not ask** whether additional tasks should be done; **include everything** in this single response.

If a field is not stated in the provided context, answer exactly `Not reported`. **Never invent information**.

**Background clarifications** If the authors merely **name** a method/component without explaining it and it would not be understandable without background knowledge, add a **very brief** clarification from your own knowledge, labeled `[Background note] <=25 words>`. This note is **explanatory only** (not evidence), must **not** contradict the paper, and must **not** be used to fill missing fields or invent values. Evidence must come from **body text or Markdown tables**.

**Always quote** the **relevant** phrases (`<=120 chars`) for any populated field, drawn from main text or tables. **Do not quote** image descriptions and editorial “Key insights” callouts.

Provide minimal background (1 sentence) only to clarify terms already used by the authors.

If the paper is not solely about NER, **focus on the NER-related aspects**. **Focus on NER architecture and NER evaluation** (models, encoders/decoders, labeling schemes, losses, datasets/splits, metrics). **Exclude** dataset-creation pipelines and relevance/observation classifiers unless they directly affect NER architecture or evaluation. **Do not** focus on relation extraction.

Assume **Related Work** are **not loaded** into the vector store; **do not** use them for evidence.

**All output must be in Markdown**. Tables must be **Markdown tables**. Use bullet points where appropriate.

**No deferral** If evidence is missing or ambiguous, output `Not reported` and proceed. Do not postpone the answer, ask the user for input, or promise future retrieval.

**Per-field return format (Markdown)**

- **Answer:** `<value>` or `Not reported`
- **Explanation & key points:** A **single-sentence rationale** followed by 3–5 concise bullets (include bullets only if helpful; omit if trivial)
- **Long explanation / description:** **Exhaustive, fully detailed account** that explains **everything to the last detail** relevant to the feature
- **Evidence:**
  - `<verbatim quote 1>`
  - `<verbatim quote 2>`
  - `<verbatim quote 3>`

Figure A.1: LLM-based Feature Extraction Developer Prompt

## A.1 Evaluation

The final setup of the LM-based extraction pipeline is described in Chapter 3.1.3. To reach this stage, I validated the original pipeline using papers that included implementation code, which I manually assessed. If the key aspects of the manually assessed feature were present in the LM-extracted version, the extraction was considered correct.

The original configuration used paragraph-sized chunks without headings, limited retrieval to 10 chunks per query (compared to 20 in the final version), applied a “low” reasoning effort, and did not use a developer prompt. The validation highlighted the following issues:

- Responses were too shallow and omitted crucial information.
- There was ambiguity about what to extract when a paper described not only a NER approach but an entire pipeline, e.g. from web crawling to KG construction.
- Due to chunking at the paragraph level, the model often failed to correctly relate information that was distributed across multiple chunks.

By addressing these issues using the pipeline described in Chapter 3.1.3, the error rate was reduced by 77%, the number of papers with zero extraction errors increased from 0 to 7 out of 12, and the average number of errors per paper dropped to below 0.1 (see Figure A.2). An error is defined as an instance in which the LM makes one of the mistakes described in the list above. The LM did not produce hallucinations. Figure A.2 shows that the LLM-based extractor also performs well on 4 randomly selected publications (test set) [Che+21; Wan+21; Tho+23; You+24].

Figure A.3 shows the cost breakdown of extracting architecture-related features for all identified publications. The total cost (including VAT) was \$15.38 (€13.23). Due to cost considerations, GPT-5-mini was used for only three features, which incurred the same cost as extracting 17 features using GPT-5-nano. The use of the OpenAI Flex API<sup>1</sup> further reduced the token cost of GPT-5-mini and GPT-5-nano by half. Consequently, the largest cost component was the retrieval of chunks from the vector store, accounting for 58% of the total cost.

<sup>1</sup><https://platform.openai.com/docs/guides/flex-processing?api-mode=responses>

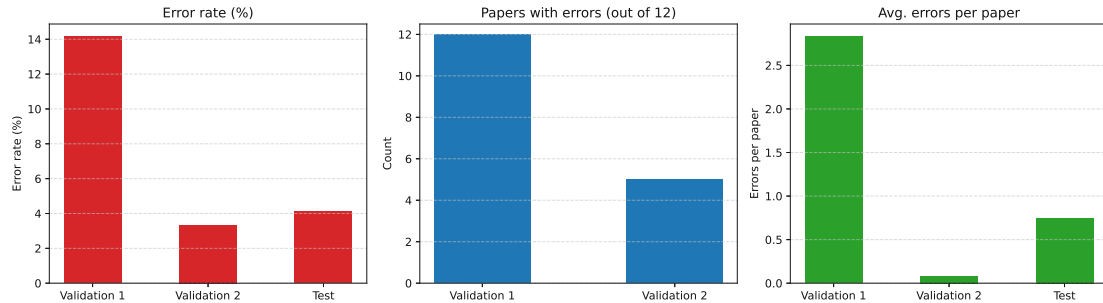


Figure A.2: Comparison of the initial LM-based extraction pipeline's (Validation 1) with the improved pipeline (Validation 2) on the validation set (publications with implementation code). Further Test shows the performance on 4 random selected papers.

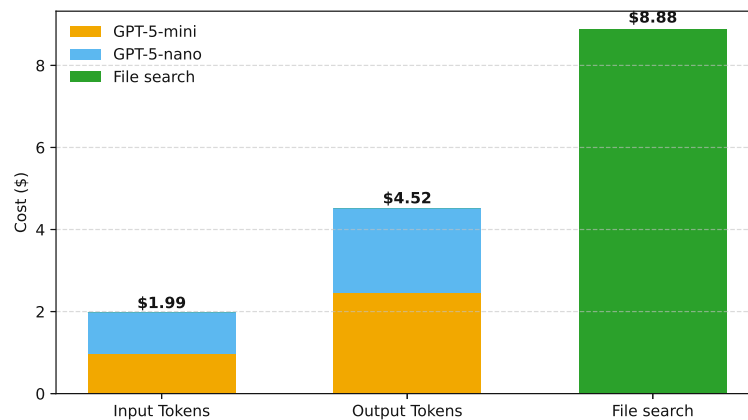


Figure A.3: Cost breakdown (incl. VAT) of extracting architecture-related features from all publications identified in the SLR

# High-level Architecture of NER Approaches that do not provide implementation Code

Table B.1 shows a high-level summary of the NER approach of each publication that do not provide implementation code identified in the SLR.

Table B.1: High-level Architecture of NER approaches that do not provide implementation code

Name & Publication	Embeddings/Features	Encode	Decode
Joshi et al. (2013) [Jos+13]	feature engineering: context (bigram, nearby/current tokens) in the form of 1-hot encoded vector		CRF + StanfordNLP Gazettters
Ma et al. (2018) [Ma+18]	char emb. (CNN) + word emb. (GloVe) + PoS emb. (StanfordCoreNLP) + generic NER emb. (StanfordCoreNLP) + chunk emb. (ApacheOpenNLP) + brown cluster emb.	BiLSTM	CRF
Gasmi et al. (2018) [GBL18]	word emb. (word2vec)	BiLSTM	CRF
Security OSIF (Li et al., 2018) [Li+18]			StanfordNLP NER (CRF) + Regex
Xiao (2018) [Xia18]	Domain Model (word2vec-CBOW, author-trained), Span Identification (word2vec skip-gram (NERTagger by Lample et al. (2016)))	Span Identification (BiLSTM (NER Tagger by Lample et al. (2016)))	Span Identification (CRF (NER Tagger by Lample et al. (2016) [ <b>&lt;empty citation&gt;</b> ])) → Span Classification (Domain Ontology (BabelNet) + Similar Words Extraction (Domain Model or CBOW) → Entity Type Voting)
Qin et al. (2019) [Qin+19]	char emb. (char-based word2vec → CNN)	BiLSTM + Feature Template	CRF

*(continues on next page)*

Name & Publication	Embeddings/Features	Encode	Decode
Li et al. (2019) [LGJ19]	word emb. (word2vec, author-trained) + char emb. (CNN)	BiLSTM → MHSA	CRF
Wang and Chow (2019) [WC19]	PoS emb. (ANNIE PoS Tagger)		JAPE (grammar and pattern-matching) + Gazetter + Regex
Gasmi et al. (2019) [GLB19]	word emb. (word2vec, author-trained)	BiLSTM	CRF
Satyapanich et al. (2019) [SFF19]	word emb. (word2vec, author-trained) + contextualized emb. (BERT 4th-to-last layer average) + linguistic feat. (PoS emb. (StanfordCoreNLP), syntactic dependencies (StanfordCoreNLP), Dbpedia Spotlight classes)	BiLSTM → Attention layer → FC layer	Softmax
Wu et al. (2020) [WLG20]	char emb. (1-hot enc. chars $\xrightarrow{\text{FFNN}}$ densemb. vector)	BiLSTM	CRF → Rules (BMM (Bidirectional Maximum Matching) + dictionary (TF-IDF-built))
Kim et al. (2020) [Kim+20]	char emb. (BOC(Bag-of-Characters) + meta features $\xrightarrow{\text{FFNN}}$ dense emb. vector) + word emb. (GloVe)	BiLSTM → FC layer	CRF + Regex
Yi et al. (2020) [Yi+20]	tokens + PoS emb. → Feature Templates → Feature Selection		CRF + Regex + Dictionary
Simran et al. (2020) [Sim+20]	word emb. (Keras rand. emb.)	BiGRU → CNN	CRF
Wang et al. (2020) [Wan+20b]	char emb. (BiLSTM) + word emb. (GloVe)	BiLSTM → FFNN	CRF
Wang et al. (2020) [Wan+20a]	word emb. (word2vec(author-trained) + BERT, concat) + char emb. (CNN)	BiLSTM → MHSA → FC layer	CRF w/ TFSL loss (proposed by authors)
Liu (2020) [Liu20]	Feature Templates + char emb. (CNN) + word emb. (not reported)	Attention fusion (FFNN) → BiLSTM	CRF
Binyamini et al. (2021) [Bin+21]	word emb. (word2Vec-CBOW, author-trained)	BiLSTM → FC layer	Softmax → CMI (Complete Missing Information; kNN → Logistic Reg.)
Wang et al. (2021) [Wan+21]	word emb. (BERT)	BiLSTM	CRF → Dictionary-based Validation
Li et al. (2021) [Li+21c]	word emb. (word2vec, author trained) + char emb. (CNN)	BiLSTM → dyn. Attention layer (MHSA → Sigmoid Gate → GRU)	LSTM → Softmax
Tsinganos and Mavridis (2021) [TM21]	word emb. (SpaCy-pretrained-English)	BiLSTM	Softmax
CyberEyes (Fang et al., 2021) [FZH21]	word emb. (word2vec, author-trained) + char emb. (CNN); document-level graph	BiLSTM → document-level GCN → BiLSTM	CRF
Ma et al. (2021) [Ma+21]	word emb. (1-hot enc. vocab → FC layer)	BiLSTM → concat(BiSLTM output + word emb.)	CRF
Gao et al. (2021) [GZL21]	1. word emb. (by Roy et al., 2017) [RPP17] → 2. domain dict emb. (8-dim. binary n-gram vector) →	1. BiLSTM → MHSA → concat (BiLSTM(word emb.) + MHSA output + BiLSTM(domain feat. emb)) 2. BiLSTM →	CRF
Chen et al. (2021) [Che+21]	word emb. (BERT, wordpiece)	ID-CNN (Iterated Dilated Convolutional Neural Networks)	CRF + Dictionary
Chiam et al. (2021) [Chi+21]	featureExtraction(PoS (not reported), n-gram, casing, suffix) for previous, current and next token		CRF
Zhou et al. (2021) [Zho+21]	word emb. (BERT-wwm (whole world masking), fine-tuned, wordpiece)	BiLSTM	CRF

(continues on next page)

Name & Publication	Embeddings/Features	Encode	Decode
Qi et al. (2022) [Qi+22]	char emb. (CNN) + Local-context Feature Tempaltes (identify negative words)	BiLSTM	CRF → validation(entity removal if negation) → category normalization(similarity)
EEMAP (Li et al., 2022) [Li+22]	word emb. (BERT) + PoS emb. (NLTK)	MHSA + BiLSTM	Softmax
Zuo et al. (2022) [Zuo+22]	word emb. (BERT, fine-tuned)	BiLSTM → Attention layer	CRF
Su et al. (2022) [Su+22b]	word emb. (RoBERTa layers 9-12 stacked)	SCNN (Semantic Convolutional Neural Network convolutes across Transformer layers for a single token, not across neighboring tokens like CNN are usually used.) → BiLSTM	CRF
Huang et al. (2022) [Hua+22]	Rule-based only: Coreference Resolution (EXTRACTOR by Satvat et al. (2021) [SGV21]) (replacing CTI-specific words with placeholders) → Dependency Parsing (Stanza) → Rule-based Triplet Extraction		
CTI View (Zhou et al., 2022) [Zho+22]	word emb. (BERT-base-cased)	BiLSTM → GRU	CRF + IoCs extraction (black/whitelisting IoCs from DBs, regex) + TTP extraction (not reported)
Wang et al. (2022) [Wan+22a]	word emb. (BERT)	BiLSTM	CRF → Knowledge Engineering Verification (Gazetter, Max-Matching)
Ramrakhiani et al. (2022) [Ram+22]	word emb. (CyBERT)	BiLSTM	CRF
Sandescu et al. (2022) [San+22]	word emb. (Bloom, SpaCy tok2vec)	residual CNN (SpaCy tok2vec)	SpaCy v3 NER (not further specified)
Chan et al. (2022) [Cha+22]	DeBERTa emb. (Huggingface)	DeBERTa encoder (Huggingface)	argmax (of logits, Huggingface) + IoCs extraction (extraction rules)
SecIE (Park & Lee, 2022) [PL22]	PoS emb (SyntaxNet), char n-gram features, word shape features		Lexio-Syntactic Pattern-based Method (Dependency Parsing, SyntaxNet) + Classification-based Extraction (Log. Reg.) + Regex for IoCs extraction + Dictionary
Zhang et al. (2022) [Zha+22]	word emb. (BERT)	BiLSTM → Self-Attention Layer	CRF
Srivastava et al. (2022) [SPG22]	word emb. (BERT-large-cased, wordpiece, fine-tuned)	FFNN	Softmax
Jo et al. (2022) [JLS22]	word emb. (BERT, fine-tuned)	BiLSTM	CRF
Guo et al. (2023) [Guo+23]	word emb. (BERT, wordpiece)	BiGRU → relation-based attention	BiGRU → CRF
FIEBD (Wang & Liu, 2023) [WL23]	word emb. (PERT <sub>large</sub> ) + char emb. (CNN) + dependency embedding (Stanford-CoreNLP) + PoS emb. (StanfordCoreNLP)	Graph feature encoder (stacked GAT (Graph Attention Network)) → BiGARU (A combination of GRU and GAT.) → MHSA	Entity Boundary Detection (BiLSTM) → CRF
ATDG (Cui et al., 2023) [CLH23]	word emb. (word2vec, skip-gram, author trained) + char emb. (DPCNN (Deep Pyramid Convolutional Neural Network))	BiGRU with MHSA	CRF
Hashemi Chaleshtori and Ray (2023) [HR23]	word emb. (XLNet-base-cased, BERT-base-cased, RoBERTa-large)	XLNet-base-cased, BERT-base-cased, RoBERTa-large → Dropout layer	Sequence Classification Layer (FC layer) → argmax → Ensemble majority-voting over all Transformer models
Høst et al. (2023) [HLM23]	word emb. (SecBERT)	SecBERT	Classification Head (not further specified)

(continues on next page)

Name & Publication	Embeddings/Features	Encode	Decode
Bose et al. (2023) [Bos+23]	word emb. (BERT-base-uncased)	BERT → BiLSTM	CRF
MTLWE (Lu et al., 2023) [Lu+23]	word emb. (BERT + dictionary(word2vec), Jieba tokenizer)	BiLSTM → MHSA	CRF
ExpSeeker (Du et al., 2023) [Du+23]	word emb. (BERT)	BiLSTM → Dropout → BiLSTM → Dropout	FC layer → CRF + Regex + Dictionary
Tho et al. (2023) [Tho+23]	word emb. (GloVe) + char emb. (BiLSTM)	BiLSTM	CRF
FineCTI-TER (Ma et al., 2023) [Ma+23]	word emb. (GloVe) + char emb. (CNN) + PoS emb. (skip-gram)	BiLSTM → MHSA	CRF
Qiao et al. (2023) [QZD23]	word emb. (BERT)	BiLSTM → Attention layer	CRF
Chang et al. (2023) [Cha+23]	CLN (conditional layer norm)(semantic emb. (CySecBERT → BiLSTM), char emb. (CNN), PoS emb. (NLTK)) + position emb. (logarithmic relative position coding)	FFNN → MGDCNN (Multi-Granularity Dilated Convolution Neural Network)	Biaffine Co-Predictor + MLP → WWRC (Word-Word Relation Classification)
SECURENET (Su & Ding, 2023) [SD23]	W2NER [Li+21b] with SecBERT and MLAT (Multi-Layer Adversarial Training)		
VIET (Zhang et al., 2023) [ZZZ23]	word emb. (word2vec, author-trained) + PoS emb. (not reported)	BiLSTM	FC layer → Softmax
Bekhouché & Adi (2024) [BA24]	average-based emb. fusion: BERT emb. + Domain-Specific emb. (by Roy et al.(2017) [<empty citation>])	BiLSTM	FC layer → Softmax
Yang et al. (2024) [Yan+24]	word emb. (RoBERTa) + PoS emb. (NLTK)	BiGRU with Attention layer	CRF
Li et al. (2024) [Li+24a]	word emb. (KE (Knowledge enhanced)-BERT by Liu et al. (2020) [<empty citation>])	BiLSTM → RGAN (Relation guided Attention network) → BiLSTM	CRF
Fieblinger et al. (2024) [FAR24]	Llama-2-70B-chat	Llama-2-70B-chat	Llama-2-70B-chat (One-shot, MS Guidance framework)
Kougioumtzidou et al. (2024) [Kou+24]	BERT-base-cased	BERT-base-cased	Softmax + Regex
Permana et al. (2024) [Per+24]	word emb. (300-dim, rand. init.)	BiLSTM → BiLSTM	FC layer → CRF
Faruk (2024) [Far24]		RoBERTa-base-squad2	
CYee (Mostafa et al., 2024) [MSZ24]	word emb. (BERT)	FC layer (ReLU, Dropout)	Softmax
Xie et al. (2024) [Xie+24]	word emb. (BERT)	BiLSTM → DRL (Deep Reinforcement Learning) layer ↔	CRF
CTI-JE (Liu et al., 2024) [Liu+24]	word emb. (SecBERT)	BiLSTM → FFNN + shared emb. w/ RE (2D Conv.) → DCNN → Channel Attention	FFNN + Biaffine Predictor
CyberEntRel (Ahmed et al., 2024) [AKH24]	word emb. (RoBERTa)	BiLSTM → Attention layer	CRF

(continues on next page)

Name & Publication	Embeddings/Features	Encode	Decode
DECEPT-CTI (Sayari et al., 2024) [Say+24]	word emb. (SecRoBERTa)	BiLSTM	CRF → MITRE ATT&CK Tactics & Techniques Mapping (SecRoBERTa + Huggingface Multi-Label Classification)
CSER (Marjan & Amagasa, 2024) [MA23]	morphological feature (word form), semantic feature (GloVe + random Keras emb. for OOV words), context feature (random Keras emb.)	morphological feature (BiLSTM), context feature (BiGRU → FFNN) → concat all 3 features	CRF
Cyst-MMET (Wang et al., 2024) [Wan+24a]	word emb. (SciBERT-Scivocab (uncased)) + char emb. (ELMo), image emb. (ResNet)	visual encoding (ResNet) → Multi-level Fusion Encoder (Module 1 (mod. MHSA) → Module 2 (similarity matrix → agg. most similar) Textual encoder ↗	Classification Head (FFNN + Sigmoid or argmax)
ITIRel (Zhu et al., 2024) [Zhu+24]	word emb. (BERT) + domain knowledge vector (ITI (IoT Threat Intelligence) dictionary, 1-hot enc)	Token Pair emb. Generator (FFNN(tanh)) → Token Pair Tagger (FFNN → Softmax)	transform highest prob. entity spans from matrix
JCLB (Hu et al., 2024) [Hu+24a]	word emb. (BERT)	BiLSTM	CRF + BRB (Belief rule base) (regex)
CTI-TFN (Lv et al., 2024) [Lv+24]	word emb. (SecureBERT + Glove + Elmo) + char emb. (random) → BiLSTM	FFNN → Fourier Transformation + Sentence emb. (TextCNN) → BiLSTM	CRF
LLM-TIKG (Hu et al., 2024) [Hu+24b]		GPT-4	GPT + Regex
SecNetPrompt (You et al., 2024) [You+24]	word emb. (BERT)	BERT	CRF + Label Word Enrichment (word2vec → KNN) + Regex
SecBABC (Chen et al., 2024) [CZX24]	word emb. (SecBERT)	BiLSTM w/ PDG (projected gradient descent)	CRF
Meng et al. (2024) [Men+24]	word emb. (BERT)	BiLSTM → MVCA (Multi-scale Void Channel Attention)(DCNN → FFNN (ReLU + Sigmoid))	CRF
Peng et al. (2024) [Pen+24]	word emb. (XLM-RoBERTa)	BiLSTM → concat(BiSLTM output + word emb.)	FC layer → CRF + Regex
Yi et al. (2024) [Yi+24]	word emb (BERT)	BiLSTM + GAT (Graph Attention)	CRF
TISCG (Du et al., 2024) [Du+24]	word emb. (SecureBERT)	Cross-BiLSTM + GGFF (Global Gated Feature Fusion)	Softmax
PER (Ullah et al., 2024) [Ull+24]	char emb. (BiLSTM) + word emb. (cybersecurity emb. by Park (2018) [<empty citation>]) + dependency emb. (Not reported)	BiLSTM → BiLSTM w/ Dependency Interaction	CRF
Xabcd (Wang et al., 2024) [Wan+24b]	word emb. (XLNet)	BiLSTM → Attention layer	CRF + Dictionary Verification
SEM (Chen et al., 2025) [Che+25]	1. word emb. (BERT) + char emb. (CNN) → 2. similar semantic emb. (Similar Word Sense Matching Algo → word2vec → Self-Attention layer)	1. BiLSTM → MHSA 2. →	CRF
CTIKG (Mouiche et al., 2025) [MMS25]	word emb. (SecureBERT)	BiGRU	FC layer → CRF

(continues on next page)

Name & Publication	Embeddings/Features	Encode	Decode
Zhao & Zheng (2025) [ZZ25]	word emb. (BERT)	FFNN → BiLSTM	CRF
CERM (Wu et al., 2025) [Wu+25]	LSIEM (Local semantic information extraction module) (word emb. (word2vec, author-trained))	LSIEM(→ CNN → MHSA) → IEM (Interrelations' extraction module) (GAN) → EIM (Entity Identification Module) (BiLSTM)	EIM (CRF)
TIKG (Mouiche & Saad, 2025) [MS25a]	word emb. (SecureBERT)	BiLSTM → Attention layer	FC layer → Softmax
HAG (Jia et al., 2025) [Jia+25]	word emb. (BERT)	BiGRU	CRF
SecLMNER (Zhang et al., 2025) [Zha+25]	LLM prompting-based simplification of input text with LLM-based verification → SecureBERT	SecureBERT	Classification Head (not further specified)
TIJERE (Mouiche & Saad, 2025) [MS25b]	word emb. (SecureBERT+)	BiGRU	FC layer → CRF
	context(unigram, bigram, nearby words) in the form of PoS(NLTK), BIO tags, surface features	IOB tagging(history-based MEM with Averged Perceptron)-Domain Tagging(history-based MEM with Averged Perceptron)	Greedy Decoder + Gazetter

## Baseline Entity Type Definitions (Prompts)

The following definitions were used for the entity types of the STIXnet dataset for prompting the LLMs Gemma3:27b and DeepSeekR1:32b using LangExtract:

- **intrusion-set:** A grouped set of adversarial behaviors and resources with common properties that is believed to be orchestrated by a single organization<sup>1</sup>.
- **attack-pattern:** Attack Patterns are a type of TTP that describe ways that adversaries attempt to compromise targets. Attack Patterns are used to help categorize attacks, generalize specific attacks to the patterns that they follow, and provide detailed information about how attacks are performed<sup>2</sup>.
- **malware:** Malware is a type of TTP that represents malicious code. It generally refers to a program that is inserted into a system, usually covertly. The intent is to compromise the confidentiality, integrity, or availability of the victim's data, applications, or operating system (OS) or otherwise annoy or disrupt the victim<sup>3</sup>.
- **tool:** Tools are legitimate software that can be used by threat actors to perform attacks. Knowing how and when threat actors use such tools can be important for understanding how campaigns are executed. Unlike malware, these tools or software packages are often found on a system and have legitimate purposes for power users, system administrators, network administrators, or even normal users. Remote access tools (e.g., RDP) and network scanning tools (e.g., Nmap) are examples of Tools that may be used by a Threat Actor during an attack<sup>4</sup>.

---

<sup>1</sup><https://oasis-open.github.io/cti-documentation/stix/intro.html>

<sup>2</sup>[https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#\\_axjjf603msy](https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#_axjjf603msy)

<sup>3</sup>[https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#\\_s5l7katgpbp09](https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#_s5l7katgpbp09)

<sup>4</sup>[https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#\\_z4voa9ndw8v](https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#_z4voa9ndw8v)

- **identity:** Identities can represent actual individuals, organizations, or groups (e.g., ACME, Inc.) as well as classes of individuals, organizations, systems or groups (e.g., the finance sector)<sup>5</sup>.
- **location:** A Location represents a geographic location. The location may be described as any, some or all of the following: region (e.g., North America), civic address (e.g. New York, US), latitude and longitude<sup>6</sup>.
- **vulnerability:** A Vulnerability is a weakness or defect in the requirements, designs, or implementations of the computational logic (e.g., code) found in software and some hardware components (e.g., firmware) that can be directly exploited to negatively impact the confidentiality, integrity, or availability of that system<sup>7</sup>.
- **campaign:** A Campaign is a grouping of adversarial behaviors that describes a set of malicious activities or attacks (sometimes called waves) that occur over a period of time against a specific set of targets<sup>8</sup>.
- **threat-actor:** Threat Actors are actual individuals, groups, or organizations believed to be operating with malicious intent. A Threat Actor is not an Intrusion Set but may support or be affiliated with various Intrusion Sets, groups, or organizations over time<sup>9</sup>.
- **domain-name:** In the Internet, a domain name is a string that identifies a realm of administrative autonomy, authority, or control. Domain names are often used to identify services provided through the Internet, such as websites, email services, and more<sup>10</sup>.
- **url:** A uniform resource locator (URL), colloquially known as web address, is a reference to a resource on the World Wide Web. A URL specifies the location of a resource on a computer network and a mechanism for retrieving it<sup>11</sup>.
- **indicator:** Indicators contain a pattern that can be used to detect suspicious or malicious cyber activity. For example, an Indicator may be used to represent a set of malicious domains<sup>12</sup>.
- **file\_paths:** A path (or filepath, file path, pathname, or similar) is a string that uniquely identifies an item in a hierarchical file system<sup>13</sup>.
- **sha256s:** Identify SHA-256 hashes.
- **tactic:** Tactics represent the “why” of an ATT&CK technique or sub-technique. It is the adversary’s tactical goal: the reason for performing an action. For example, an adversary may want to achieve credential access<sup>14</sup>.

The following definitions were used for the entity types of the DNRTI dataset for

<sup>5</sup>[https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#\\_wh296fiwplkp](https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#_wh296fiwplkp)

<sup>6</sup>[https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#\\_th8nitr8jb4k](https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#_th8nitr8jb4k)

<sup>7</sup>[https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#\\_q5ytzmajn6re](https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#_q5ytzmajn6re)

<sup>8</sup>[https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#\\_pcpvfz4ik6d6](https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#_pcpvfz4ik6d6)

<sup>9</sup>[https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#\\_k017w16zutw](https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#_k017w16zutw)

<sup>10</sup>[https://en.wikipedia.org/wiki/Domain\\_name](https://en.wikipedia.org/wiki/Domain_name)

<sup>11</sup><https://en.wikipedia.org/wiki/URL>

<sup>12</sup>[https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#\\_muftcrpnf89v](https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#_muftcrpnf89v)

<sup>13</sup>[https://en.wikipedia.org/wiki/Path\\_\(computing\)](https://en.wikipedia.org/wiki/Path_(computing))

<sup>14</sup><https://attack.mitre.org/tactics/enterprise/>

---

prompting the LLMs Gemma3:27b and DeepSeekR1:32b using LangExtract. The prompts are AI-generated.

- **HackOrg**: Named hacker or threat-actor groups/organizations (e.g. APT groups or cybercrime gangs) conducting the attacks.
- **OffAct**: The concrete malicious activity or attack (e.g., a spear-phishing campaign or DDoS attack) carried out by the actor.
- **SamFile**: Specific malicious samples, such as file names, malware binaries, and their associated hashes.
- **SecTeam**: Security vendors, research teams, or government security units that analyze or respond to the threat.
- **Tool**: Software, scripts, or malware families used as tools by the attacker during the operation.
- **Time**: Dates or times when attacks, campaigns, or relevant events occur.
- **Purp**: The attacker’s objective or purpose, such as espionage, data theft, or service disruption.
- **Area**: Geographic locations related to the threat, such as countries, regions, or cities.
- **Idus**: Industry sectors targeted or involved (e.g., finance, military, technology).
- **Org**: Legitimate organizations mentioned in the report, such as victim companies or service providers.
- **Way**: The way or method the attack is executed, i.e., the attack vector or technique used to gain access.
- **Exp**: Loopholes, vulnerabilities or exploits leveraged by the attacker, including both exploit names and identifiers (e.g., CVEs).
- **Features**: Distinctive behaviors of the attack.

The developer prompt for LangExtract is as follows: “Extract cybersecurity threat intelligence entities from the text.”

# Overview of Generative AI Tools Used

Generative AI tools were used to refine thesis wording, generate code for plots, analyze code repositories, do part of the SLR (see Section 3.1.3) and refine prompts for the Baseline (see Appendix C) and the LM-based SLR feature extractor (see Appendix A). Moreover, gpt-5-nano and -mini are used for the LM-based SLR feature extractor.

For refining the thesis wording, a OpenAI GPT<sup>15</sup> developed by the author was used with the prompt shown in Figure 1. It was used for a paragraph at a time.

The generative AI tools used were Claude Code and ChatGPT.

This GPT is designed to refine and enhance paragraphs for scientific writing. It will focus on improving sentence structure, grammar, spelling, and word choice to ensure that the text is clear, concise, and appropriate for a scientific context. The GPT will avoid adding new content or altering the original meaning, instead concentrating on making the writing more polished and suitable for academic or professional scientific publications. When correcting, it will ensure that complex scientific terminology and concepts are retained accurately and used appropriately. It will avoid colloquialisms, informal language, and subjective tone, maintaining an objective, formal, and precise style throughout. The GPT will not hesitate to ask for clarification if needed to ensure accuracy but will generally prioritize making educated assumptions to maintain flow. It will not expand abbreviations or acronyms if they are already present in the text. Responses will be direct and to the point, providing an improved version of the text with minimal explanation unless explicitly requested by the user. The GPT must not use semicolons in refined sentences and should replace them with alternative punctuation or sentence structures that maintain clarity and flow. The GPT will also revise text to prefer simple present for established facts, definitions, and interpretation (including what the paper, section, or figure shows). Use simple past for completed actions and specific observations (methods performed and results obtained). Use present perfect to summarize prior work or an ongoing research trend without a specific time point. Use future tense only for clearly stated future work or implications, and keep tense consistent within a sentence or paragraph.

Figure 1: Prompt of the GPT used by the author the refine the wording of this work.

<sup>15</sup><https://chatgpt.com/g/g-nkU6sSo3B-scientific-writing-refiner>

# Übersicht verwendeter Hilfsmittel

Generative KI-Werkzeuge wurden eingesetzt, um den Text der Arbeit zu verbessern, Code für Abbildungen zu generieren, Code-Repositorien zu analysieren, einen Teil der SLR zu übernehmen (siehe Kapitel 3.1.3) und Prompts für die Baseline (siehe Appendix C) sowie den LM-basierten SLR-Feature-Extraktor (siehe Appendix A) zu verbessern.

Zur Verfeinerung der Textformulierung wurde ein vom Autor entwickelter OpenAI GPT<sup>16</sup> mit dem in Abbildung 2 dargestellten Prompt verwendet. Jeder Absatz wurde einzeln überarbeitet.

Die konkret eingesetzten generativen KI-Werkzeuge waren Claude Code und ChatGPT.

This GPT is designed to refine and enhance paragraphs for scientific writing. It will focus on improving sentence structure, grammar, spelling, and word choice to ensure that the text is clear, concise, and appropriate for a scientific context. The GPT will avoid adding new content or altering the original meaning, instead concentrating on making the writing more polished and suitable for academic or professional scientific publications. When correcting, it will ensure that complex scientific terminology and concepts are retained accurately and used appropriately. It will avoid colloquialisms, informal language, and subjective tone, maintaining an objective, formal, and precise style throughout. The GPT will not hesitate to ask for clarification if needed to ensure accuracy but will generally prioritize making educated assumptions to maintain flow. It will not expand abbreviations or acronyms if they are already present in the text. Responses will be direct and to the point, providing an improved version of the text with minimal explanation unless explicitly requested by the user. The GPT must not use semicolons in refined sentences and should replace them with alternative punctuation or sentence structures that maintain clarity and flow. The GPT will also revise text to prefer simple present for established facts, definitions, and interpretation (including what the paper, section, or figure shows). Use simple past for completed actions and specific observations (methods performed and results obtained). Use present perfect to summarize prior work or an ongoing research trend without a specific time point. Use future tense only for clearly stated future work or implications, and keep tense consistent within a sentence or paragraph.

Abbildung 2: Prompt des GPTs des Autors, der zur Verbesserungen des Textes verwendet wurde.

<sup>16</sup><https://chatgpt.com/g/g-nkU6sSo3B-scientific-writing-refiner>

# List of Figures

2.1	Main approaches to NER [KMN24] . . . . .	6
4.1	Sankey diagram of the paper selection process for the SLR. Starting from multiple academic databases, duplicate entries were removed and papers were filtered out according to the criteria defined in Section 3.1.1, resulting in a final set of publications classified by code availability. . . . .	24
4.2	Distribution of publications per year . . . . .	25
4.3	Temporal distribution of publications discriminated by their usage of LMs. . . . .	26
4.4	Number of publications that use each method family. A single publication may contribute to multiple categories if it employed more than one method family. . . . .	26
4.5	Overview of components used in NER architectures, categorized by their roles in embedding, encoding and decoding. . . . .	27
4.6	Types of architectures for each NER pipeline step. A single publication may employ multiple embedding types. For encoding and decoding, each publication was assigned to a single type. . . . .	29
4.7	Most common enhancement types in NER architectures. Publications use more than one enhancement type. . . . .	31
4.8	Usage of different LMs across the identified publications . . . . .	31
4.9	Heatmap of strict micro-averaged F1 scores across entity types. “Frequency (train)” denotes the number of samples in the training dataset. Models are ordered in descending order according to their overall strict micro F1 score. Entity types are ordered top to bottom (best to worst) by their strict micro F1 score averaged across all models. . . . .	45
4.9	Heatmap of strict micro-averaged F1 scores across entity types. “Frequency (train)” denotes the number of samples in the training dataset. Models are ordered in descending order according to their overall strict micro F1 score. Entity types are ordered top to bottom (best to worst) by their strict micro F1 score averaged across all models. . . . .	46
4.10	This boxplot shows the strict micro F1 score of each bucket over all attributes per model. The best and worst performing attribute-bucket combinations are labeled. . . . .	47
		93

4.11	Each subfigure presents, for a single attribute, the change in performance across buckets for all models. Each attribute is subdivided into predefined buckets, as defined in Section 3.2.4. . . . .	48
4.12	Each subfigure presents, for a single model, the change in performance across buckets for all entity attributes, with each property subdivided into predefined buckets as defined in Section 3.2.4. . . . .	49
4.12	Each subfigure presents, for a single model, the change in performance across buckets for all entity attributes, with each property subdivided into predefined buckets as defined in Section 3.2.4. . . . .	50
4.12	Each subfigure presents, for a single model, the change in performance across buckets for all entity attributes, with each property subdivided into predefined buckets as defined in Section 3.2.4. . . . .	51
4.13	This scatterplot illustrates the performance of the models on the STIXnet dataset, measured by strict micro-averaged F1 score. Performance is shown for two settings: tagging each sentence individually (SSS test dataset) and tagging all sentences of a document jointly (SWD test dataset). . . . .	52
4.14	Comparison of strict micro F1 score with inference time and energy consumption on the DNRTI test dataset. x-axis is log-scaled for readability. . . . .	54
4.15	Comparison of strict micro F1 score with training time and energy consumption on the DNRTI test dataset. . . . .	55
4.16	Comparison of inference time and energy consumption between SSS test dataset and SWD test dataset. Axes are log-scaled for readability. . . . .	57
4.17	Comparison of training time and energy consumption between SSS test dataset and SWD test dataset. Axes are log-scaled for readability. . . . .	58
A.1	LLM-based Feature Extraction Developer Prompt . . . . .	79
A.2	Comparison of the initial LM-based extraction pipeline's (Validation 1) with the improved pipeline (Validation 2) on the validation set (publications with implementation code). Further Test shows the performance on 4 random selected papers. . . . .	81
A.3	Cost breakdown (incl. VAT) of extracting architecture-related features from all publications identified in the SLR . . . . .	81
1	Prompt of the GPT used by the author the refine the wording of this work.	91
2	Prompt des GPTs des Autors, der zur Verbesserungen des Textes verwendet wurde. . . . .	92

# List of Tables

3.1	Coverage of dataset criteria of the datasets employed . . . . .	17
3.2	Entity mention distribution across entity types for the SSS dataset . . . . .	18
3.3	Characterizing statistics of the STIXnet dataset (single sentences and whole documents) and the DNRTI dataset . . . . .	19
4.1	Method Families used in NER approaches . . . . .	36
4.2	High-level Architecture of NER approaches that provide implementation code . . . . .	37
4.3	Knowledge and Miscellaneous Properties . . . . .	38
4.4	LMs used in NER approaches . . . . .	40
4.5	Overall model performance on DNRTI test dataset across all models ordered by strict micro F1 score . . . . .	43
4.6	Cost metrics runtime and energy consumption are ordered by inference time per sentence on the DNRTI test dataset. Energy consumption measurements are not available for STUCCO because the framework for collecting the statistics does not support Python 2. LangExtract does not report training costs, as it is not trained. The column “Pre-trained models used” indicates which publicly available pre-trained models are employed by the CTI-NER approaches. The cost of pre-training these models is not included in the subsequent calculations. STUCCO and Twitter are trained on CPU, whereas all other models are trained on GPU (see Section 4.2.1). . . . .	56
4.7	Comparison of cost metrics for the STIXnet dataset when tagging sentences individually without contextual information (SSS) versus tagging entire documents jointly (SWD dataset). Ordered by SWD Inference Time per Word. Energy consumption measurements are not available for STUCCO because the framework for collecting the statistics does not support Python 2. LangExtract does not report training costs, as it is not trained. STUCCO and Twitter are trained on CPU, whereas all other models are trained on GPU (see Section 4.2.1). . . . .	58
A.1	Data Extraction Form . . . . .	66
B.1	High-level Architecture of NER approaches that do not provide implementation code . . . . .	82
		95

# Acronyms

- AI** Artificial Intelligence. 8
- BiGRU** Bidirectional Gated Recurrent Unit. 32, 37, 83–87
- BiLSTM** Bidirectional Long Short-Term Memory. 7, 26, 28, 30, 32–34, 37, 38, 52, 54, 60, 64, 82–87
- CNN** Convolutional Neural Network. 5, 7, 26, 28, 37, 67, 70, 73, 74, 82–87
- CRF** Conditional Random Field. 3, 5, 9, 10, 25–29, 31, 33, 34, 37, 38, 52–54, 60, 64, 82–87
- CTI** Cyber Threat Intelligence. 1–5, 13, 32, 33, 60, 62
- CTI-NER** CTI-focused NER. 6, 7, 12, 16, 24, 44–46, 53, 56, 59, 60, 62, 64, 65, 95
- DL** Deep Learning. 24–26, 36, 52, 53
- FC** fully connected. 34, 37, 38, 83–87
- FFNN** Feed Forward Neural Network. 26, 28, 37, 83–87
- GCN** Graph Convolutional Network. 28, 31, 35, 38, 39, 83
- GRU** Gated Recurrent Unit. 33, 83, 84
- IoCs** Indicators of Compromise. 5, 6, 26, 33, 35, 59, 70, 84
- KB** Knowledge Base. 31, 35, 36, 38, 39
- KG** Knowledge Graph. 2, 31, 32, 35, 36, 39, 63, 80
- LLM** Large Language Model. 2, 3, 5, 6, 8, 9, 14, 16, 25, 27, 29, 30, 36, 43, 46, 52, 53, 59–64, 88, 90

- LM** Language Model. 2, 13, 14, 24–26, 28, 30, 31, 33, 34, 36, 40, 52, 59, 60, 64, 66, 80, 93, 95
- LSTM** Long Short-Term Memory. 7, 28, 33
- MEM** Maximum Entropy Model. 31, 87
- MHSA** Multi-Head Self-Attention. 7, 8, 28, 33, 37, 83–87
- ML** Machine Learning. 70
- NER** Named Entity Recognition. 1–3, 5–9, 11–14, 16, 19, 25–32, 34–37, 40, 43, 44, 59, 60, 62, 63, 80, 82, 93, 95
- NLP** Natural Language Processing. 1, 7, 14, 32, 40, 79
- NN** Neural Network. 10, 26
- OOV** out-of-vocabulary. 20, 21, 27, 32–34, 37, 43, 45, 46, 48, 86
- OSINT** Open Source Intelligence. 4, 34
- PoS** Part-of-Speech. 28, 31–33, 35, 37, 38, 41, 42, 56, 82–85, 87
- RL** Reinforcement Learning. 25, 26, 36
- RNN** Recurrent Neural Network. 3, 5, 7, 26, 28, 32, 73
- SDO** STIX 2.1 Domain Object. 6, 7, 17, 18
- SLR** Systematic Literature Review. 2, 3, 11–13, 15, 16, 23, 24, 40, 60, 64, 81, 82, 93, 94
- SSS** STIXnet single sentence. 18, 44, 45, 52, 57, 58, 60, 94, 95
- STIX 2.1** Structured threat Information Expression (version 2.1). 6, 16, 32–34, 39, 63, 69
- SWD** STIXnet whole document. 41, 52, 57, 58, 60, 94, 95
- TTP** Tactic, Technique, Procedure. 1, 32, 35

# Bibliography

- [95] “Appendix C: Named Entity Task Definition (v2.1)”. In: *Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995*. 1995. URL: <https://aclanthology.org/M95-1024/> (visited on 01/02/2026).
- [Agh+22] Ehsan Aghaei et al. *SecureBERT: A Domain-Specific Language Model for Cybersecurity*. 2022. DOI: 10.48550/arXiv.2204.02685.
- [AI23] Vectra AI. *2023 State of Threat Detection*. Tech. rep. Vectra AI, 2023. URL: [https://cdn.prod.website-files.com/64e50cbe2b6f932c04238c14/689db628db970401f2bfde5b\\_6630da12bd1f4ccb222dffea\\_2023-state-of-threat-detection\\_comp.pdf](https://cdn.prod.website-files.com/64e50cbe2b6f932c04238c14/689db628db970401f2bfde5b_6630da12bd1f4ccb222dffea_2023-state-of-threat-detection_comp.pdf) (visited on 01/26/2026).
- [AKH24] K. Ahmed et al. “CyberEntRel: Joint extraction of cyber entities and relations using deep learning”. In: *Elsevier Computers and Security* 136 (2024). DOI: 10.1016/j.cose.2023.103579.
- [AL23] Dhananjay Ashok and Zachary C. Lipton. *PromptNER: Prompting For Named Entity Recognition*. arXiv:2305.15444 [cs]. 2023. DOI: 10.48550/arXiv.2305.15444.
- [AP23] J. Arora and Y. Park. “Split-NER: Named Entity Recognition via Two Question-Answering-based Classifications”. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Vol. 2. Association for Computational Linguistics (ACL), 2023, pp. 416–426. DOI: 10.18653/v1/2023.acl-short.36.
- [BA24] M.E.A. Bekhouche and K. Adi. “A BERT-Based Framework for Automated Extraction of Behavioral Indicators of Compromise from Security Incident Reports”. In: ed. by Mosbah M et al. Vol. 14551 LNCS. Springer Science and Business Media Deutschland GmbH, 2024, pp. 219–232. DOI: 10.1007/978-3-031-57537-2\_14.
- [Bay+22] Markus Bayer et al. *CySecBERT: A Domain-Adapted Language Model for the Cybersecurity Domain*. 2022. DOI: 10.48550/arXiv.2212.02974.

- [Bin+21] H. Binyamini et al. “A Framework for Modeling Cyber Attack Techniques from Security Vulnerability Descriptions”. In: *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2021, pp. 2574–2583. DOI: 10.1145/3447548.3467159.
- [BKL10] Stevan Bird et al. *Natural Language Processing with Python*. O’Reilly Media, 2010. ISBN: 978-0-596-51649-9.
- [Bos+23] Avishek Bose et al. “Context-Augmented Key Phrase Extraction from Short Texts for Cyber Threat Intelligence Tasks”. In: *2023 IEEE International Conference on Intelligence and Security Informatics (ISI)*. 2023, pp. 1–6. DOI: 10.1109/ISI58743.2023.10297274.
- [Bri+14] Robert A. Bridges et al. *Automatic Labeling for Entity Extraction in Cyber Security*. 2014. DOI: 10.48550/arXiv.1308.4941.
- [Car+22] Angela Carrera-Rivera et al. “How-to conduct a systematic literature review: A quick guide for computer science research”. In: *MethodsX* 9 (2022), p. 101895. ISSN: 2215-0161. DOI: 10.1016/j.mex.2022.101895.
- [Cha+22] H.-J. Chan et al. “FeedRef2022: A Named Entity Recognition Dataset for Extracting Indicators of Compromise”. In: ed. by Tsumoto S et al. Institute of Electrical and Electronics Engineers Inc., 2022, pp. 2578–2584. DOI: 10.1109/BigData55660.2022.10020985.
- [Cha+23] Y. Chang et al. “Research on Unified Cyber Threat Intelligence Entity Recognition Method Based on Multiple Features”. In: Institute of Electrical and Electronics Engineers Inc., 2023, pp. 233–240. DOI: 10.1109/CAIT59945.2023.10469250.
- [Che+21] Y. Chen et al. “Joint BERT Model based Cybersecurity Named Entity Recognition”. In: *4th International Conference on Software Engineering and Information Management, ICSIM 2021; Virtual, Online; Japan; 16 January 2021 through 18 January 2021; Code 170192*. Association for Computing Machinery, 2021, pp. 236–242. DOI: 10.1145/3451471.3451508.
- [Che+23] S.-S. Chen et al. “Enhancing Cyber Threat Intelligence with Named Entity Recognition Using BERT-CRF”. In: Institute of Electrical and Electronics Engineers Inc., 2023, pp. 7532–7537. DOI: 10.1109/GLOBECOM54140.2023.10436853.
- [Che+25] L. Chen et al. “A Named Entity Recognition Model for Threat Intelligence Based on Similar Semantic Space Construction”. In: *5th International Conference on Computer Science and Management Technology, ICCSMT 2024; Xiamen; China; 18 October 2024 through 20 October 2024; Code 206507*. Association for Computing Machinery, Inc, 2025, pp. 323–327. DOI: 10.1145/3708036.3708092.

- [Chi+21] Y.J. Chiam et al. “Malware behavior profiling from unstructured data”. In: *11th International Conference on Soft Computing and Pattern Recognition, SoCPaR 2019, and 11th World Congress on Nature and Biologically Inspired Computing, NaBIC 2019; Hyderabad; India; 13 December 2019 through 15 December 2019; Code 243309*. Ed. by Abraham A et al. Vol. 1182 AISC. Springer, 2021, pp. 130–140. DOI: 10.1007/978-3-030-49345-5\_14.
- [CLH23] B. Cui et al. “ATDG: An Automatic Cyber Threat Intelligence Extraction Model of DPCNN and BIGRU Combined with Attention Mechanism”. In: ed. by Zhang F. et al. Vol. 14306 LNCS. Springer Science and Business Media Deutschland GmbH, 2023, pp. 189–204. DOI: 10.1007/978-981-99-7254-8\_15.
- [CR98] N. Chinchor and P. Robinson. “MUC-7 Named Entity Task Definition (version 3.5)”. In: *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*. 1998. URL: <https://aclanthology.org/M98-1028/> (visited on 01/02/2026).
- [CZX24] T. Chen et al. “PANSAs: A Framework for Threat Knowledge Graph Construction”. In: Institute of Electrical and Electronics Engineers Inc., 2024, pp. 763–768. DOI: 10.1109/CBASE64041.2024.10824404.
- [Dal14] Henry Dalziel. *How to Define and Build an Effective Cyber Threat Intelligence Capability*. O’Reilly, 2014. ISBN: 978-0-12-802730-1.
- [Dee+25] DeepSeek-AI et al. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. 2025. DOI: 10.48550/arXiv.2501.12948.
- [Det+23] Tim Dettmers et al. *QLoRA: Efficient Finetuning of Quantized LLMs*. 2023. DOI: 10.48550/arXiv.2305.14314.
- [Dev+19] Jacob Devlin et al. “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)*. Ed. by Jill Burstein et al. Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.
- [Dia13] Mona Diab. “SEM 2013: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)”. In: ACL, 2013. ISBN: 978-1-937284-48-0.
- [Dio+19] N. Dionisio et al. “Cyberthreat Detection from Twitter using Deep Neural Networks”. In: vol. 2019-July. Institute of Electrical and Electronics Engineers Inc., 2019. DOI: 10.1109/IJCNN.2019.8852475.
- [Du+23] Y. Du et al. “ExpSeeker: extract public exploit code information from social media”. In: *Applied Intelligence* 53.12 (2023), pp. 15772–15786. DOI: 10.1007/s10489-022-04178-9.

- [Du+24] C. Du et al. “Threat Intelligence Named Entity Recognition Based on Global Gated Feature Fusion”. In: Institute of Electrical and Electronics Engineers Inc., 2024, pp. 618–622. DOI: 10.1109/IoTAAI62601.2024.10692655.
- [FAR24] R. Fieblinger et al. “Actionable Cyber Threat Intelligence Using Knowledge Graphs and Large Language Models”. In: Institute of Electrical and Electronics Engineers Inc., 2024, pp. 100–111. DOI: 10.1109/EuroSPW61312.2024.00018.
- [Far24] Tanjim Bin Faruk. *Automated CVE Analysis: Harnessing Machine Learning In Designing Question-Answering Models For Cybersecurity Information Extraction*. 2024. DOI: 10.48550/arXiv.2412.16484.
- [FLN20] Jinlan Fu et al. “Interpretable Multi-dataset Evaluation for Named Entity Recognition”. In: ed. by Bonnie Webber et al. Association for Computational Linguistics, 2020, pp. 6058–6069. DOI: 10.18653/v1/2020.emnlp-main.489.
- [Flo25] Floyd. *flightower/ioc-finder*. 2025. URL: <https://github.com/flightower/ioc-finder> (visited on 09/08/2025).
- [Fuj+22] S. Fujii et al. “CyNER: Information Extraction from Unstructured Text of CTI Sources with Noncontextual IOCs”. In: ed. by Cheng C.-M and Akiyama M. Vol. 13504 LNCS. Springer Science and Business Media Deutschland GmbH, 2022, pp. 85–104. DOI: 10.1007/978-3-031-15255-9\_5.
- [Fuj+23] S. Fujii et al. “Extracting and Analyzing Cybersecurity Named Entity and its Relationship with Noncontextual IOCs from Unstructured Text of CTI Sources”. In: *Journal of Information Processing* 31 (2023), pp. 578–590. DOI: 10.2197/IPSJJIP.31.578.
- [FZH21] Y. Fang et al. “CyberEyes: Cybersecurity Entity Recognition Model Based on Graph Convolutional Network”. In: *Computer Journal* 64.8 (2021), pp. 1215–1225. DOI: 10.1093/comjnl/bxaa141.
- [GBL18] H Gasmi et al. “LSTM Recurrent Neural Networks for Cybersecurity Named Entity Recognition”. In: *ICSEA 2018 : The Thirteenth International Conference on Software Engineering Advances*. Ed. by L Lavazza et al. 2018, pp. 1–6.
- [GK25] Akshay Goel and Atilla Kiraly. *Introducing LangExtract: A Gemini powered information extraction library- Google Developers Blog*. 2025. URL: <https://developers.googleblog.com/en/introducing-langextract-a-gemini-powered-information-extraction-library/> (visited on 12/27/2025).
- [GLB19] H. Gasmi et al. “Information extraction of cybersecurity concepts: An LSTM approach”. In: *Applied Sciences (Switzerland)* 9.19 (2019). DOI: 10.3390/app9193945.

- [GS96] Ralph Grishman and Beth Sundheim. “Message Understanding Conference-6: A Brief History”. In: *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*. 1996. URL: <https://aclanthology.org/C96-1079/> (visited on 03/10/2025).
- [Guo+23] Y. Guo et al. “A framework for threat intelligence extraction and fusion”. In: *Computers and Security* 132 (2023). DOI: 10.1016/j.cose.2023.103371.
- [GZL21] C. Gao et al. “Data and knowledge-driven named entity recognition for cyber security”. In: *Cybersecurity* 4.1 (2021). DOI: 10.1186/s42400-021-00072-y.
- [Ham+25] Dzenan Hamzic et al. “Enhancing Cyber Situational Awareness with AI: A Novel Pipeline Approach for Threat Intelligence Analysis and Enrichment”. In: *Availability, Reliability and Security*. Ed. by Bart Coppens et al. Springer Nature Switzerland, 2025, pp. 44–62. ISBN: 978-3-032-00633-2.
- [Ham25] Dzenan Hamzić. *selfconstruct3d/AITSecNER · Hugging Face*. 2025. URL: <https://huggingface.co/selfconstruct3d/AITSecNER> (visited on 03/20/2025).
- [HLM23] A.M. Høst et al. “Constructing a Knowledge Graph from Textual Descriptions of Software Vulnerabilities in the National Vulnerability Database”. In: *24th Nordic Conference on Computational Linguistics, NoDaLiDa 2023; Torshavn; Faroe Islands; 22 May 2023 through 24 May 2023; Code 205927*. Ed. by Alumae T and Fishel M. University of Tartu Library, 2023, pp. 386–391.
- [HR23] F. Hashemi Chaleshtori and I. Ray. “Automation of Vulnerability Information Extraction Using Transformer-Based Language Models”. In: ed. by Katsikas S et al. Vol. 13785 LNCS. Springer Science and Business Media Deutschland GmbH, 2023, pp. 645–665. DOI: 10.1007/978-3-031-25460-4\_37.
- [HS97] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.
- [Hu+24a] C. Hu et al. “Joint contrastive learning and belief rule base for named entity recognition in cybersecurity”. In: *Cybersecurity* 7.1 (2024). DOI: 10.1186/s42400-024-00206-y.
- [Hu+24b] Y. Hu et al. “LLM-TIKG: Threat intelligence knowledge graph construction utilizing large language model”. In: *Computers and Security* 145 (2024). DOI: 10.1016/j.cose.2024.103999.
- [Hua+22] C.-C. Huang et al. “Building Cybersecurity Ontology for Understanding and Reasoning Adversary Tactics and Techniques”. In: ed. by Tsumoto S et al. Institute of Electrical and Electronics Engineers Inc., 2022, pp. 4266–4274. DOI: 10.1109/BigData55660.2022.10021134.

- [Hus+17] Ghaith Husari et al. “TTPDrill: Automatic and Accurate Extraction of Threat Actions from Unstructured Text of CTI Sources”. In: *Proceedings of the 33rd Annual Computer Security Applications Conference. ACSAC '17*. Association for Computing Machinery, 2017, pp. 103–115. ISBN: 978-1-4503-5345-8. DOI: 10.1145/3134600.3134646.
- [Ins24] SANS Institute. *The Importance of Cyber Threat Intelligence: Insights from Recent Nobelium Attacks*. 2024. URL: <https://www.sans.org/blog/the-importance-of-cyber-threat-intelligence-insights-from-recent-nobelium-attacks> (visited on 01/26/2026).
- [ISO25] ISO. *Threat intelligence and why it matters for cybersecurity*. 2025. URL: <https://www.iso.org/information-security/threat-intelligence> (visited on 01/26/2026).
- [Jia+25] J. Jia et al. “Hyper attack graph: Constructing a hypergraph for cyber threat intelligence analysis”. In: *Computers and Security* 149 (2025). DOI: 10.1016/j.cose.2024.104194.
- [Jin+23] Youngjin Jin et al. “DarkBERT: A Language Model for the Dark Side of the Internet”. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2023, pp. 7515–7533. DOI: 10.18653/v1/2023.acl-long.415. (Visited on 10/09/2025).
- [JLS22] H. Jo et al. “Vulcan: Automatic extraction and analysis of cyber threat intelligence from unstructured text”. In: *Computers and Security* 120 (2022). DOI: 10.1016/j.cose.2022.102763.
- [JM25] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd. 2025. URL: <https://web.stanford.edu/~jurafsky/slp3/> (visited on 01/02/2026).
- [Joh+16] Christopher S. Johnson et al. *Guide to Cyber Threat Information Sharing*. Tech. rep. NIST SP 800-150. National Institute of Standards and Technology, 2016, NIST SP 800–150. DOI: 10.6028/NIST.SP.800-150.
- [Jos+13] A. Joshi et al. “Extracting cybersecurity related linked data from text”. In: 2013, pp. 252–259. DOI: 10.1109/ICSC.2013.50.
- [KC07] Barbara Kitchenham and Stuart Charters. “Guidelines for performing Systematic Literature Reviews in Software Engineering”. In: (2007).
- [Kim+20] G. Kim et al. “Automatic extraction of named entities of cyber threats using a deep Bi-LSTM-CRF network”. In: *International Journal of Machine Learning and Cybernetics* 11.10 (2020), pp. 2341–2355. DOI: 10.1007/s13042-020-01122-6.

- [KLZ25] Anoop Kotha et al. *GPT-5 prompting guide / OpenAI Cookbook*. 2025. URL: [https://cookbook.openai.com/examples/gpt-5/gpt-5\\_prompting\\_guide](https://cookbook.openai.com/examples/gpt-5/gpt-5_prompting_guide) (visited on 08/20/2025).
- [KMN24] Imed Keraghel et al. *Recent Advances in Named Entity Recognition: A Comprehensive Survey and Comparative Study*. 2024. DOI: 10.48550/arXiv.2401.10825.
- [Kou+24] A. Kougioumtzidou et al. “An End-to-End Framework for Cybersecurity Taxonomy and Ontology Generation and Updating”. In: Institute of Electrical and Electronics Engineers Inc., 2024, pp. 247–254. DOI: 10.1109/CSR61664.2024.10679346.
- [Lan+23] Max Landauer et al. “Deep learning for anomaly detection in log data: A survey”. In: *Machine Learning with Applications* 12 (2023), p. 100470. ISSN: 26668270. DOI: 10.1016/j.mlwa.2023.100470.
- [Lan+24] Lukas Lange et al. *AnnoCTR: A Dataset for Detecting and Linking Entities, Tactics, and Techniques in Cyber Threat Reports*. 2024. DOI: 10.48550/arXiv.2404.07765.
- [Leg+20] Valentine Legoy et al. *Automated Retrieval of ATT&CK Tactics and Techniques for Cyber Threat Reports*. 2020. DOI: 10.48550/arXiv.2004.14322.
- [LGJ19] T. Li et al. “A Self-Attention-Based Approach for Named Entity Recognition in Cybersecurity”. In: Institute of Electrical and Electronics Engineers Inc., 2019, pp. 147–150. DOI: 10.1109/CIS.2019.00039.
- [Li+18] K. Li et al. “Security OSIF: Toward automatic discovery and analysis of event based cyber threat intelligence”. In: ed. by Loulergue F et al. Institute of Electrical and Electronics Engineers Inc., 2018, pp. 741–747. DOI: 10.1109/SmartWorld.2018.00142.
- [Li+21a] Jingye Li et al. *Unified Named Entity Recognition as Word-Word Relation Classification*. 2021. DOI: 10.48550/arXiv.2112.10070.
- [Li+21b] Jingye Li et al. “Unified Named Entity Recognition as Word-Word Relation Classification”. In: *AAAI Conference on Artificial Intelligence*. 2021.
- [Li+21c] T. Li et al. “Adversarial active learning for named entity recognition in cybersecurity”. In: *Computers, Materials and Continua* 66.1 (2021), pp. 407–420. DOI: 10.32604/cmc.2020.012023.
- [Li+22] Y. Li et al. “A Novel Threat Intelligence Information Extraction System Combining Multiple Models”. In: *Security and Communication Networks* 2022 (2022). DOI: 10.1155/2022/8477260.
- [Li+24a] K. Li et al. “A Knowledge-Enhanced Cyber Threat Intelligence Entity and Relation Extraction Method”. In: Institute of Electrical and Electronics Engineers Inc., 2024, pp. 375–384. DOI: 10.1109/ICFTIC64248.2024.10912987.

- [Li+24b] Q. Li et al. “Comprehensive vulnerability aspect extraction”. In: *Applied Intelligence* 54.3 (2024), pp. 2881–2899. DOI: 10.1007/s10489-023-05262-4.
- [Lib22] Matteo Liberato. “SecBERT: Analyzing reports with BERT-like models”. MA thesis. University of Twente, 2022.
- [Lim+17] Swee Kiat Lim et al. “MalwareTextDB: A Database for Annotated Malware Articles”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Regina Barzilay and Min-Yen Kan. Association for Computational Linguistics, 2017, pp. 1557–1567. DOI: 10.18653/v1/P17-1143.
- [Liu+22] P. Liu et al. “Multi-features based Semantic Augmentation Networks for Named Entity Recognition in Threat Intelligence”. In: vol. 2022-August. Institute of Electrical and Electronics Engineers Inc., 2022, pp. 1557–1563. DOI: 10.1109/ICPR56361.2022.9956373.
- [Liu+23] Nelson F. Liu et al. *Lost in the Middle: How Language Models Use Long Contexts*. 2023. DOI: 10.48550/arXiv.2307.03172.
- [Liu+24] Y. Liu et al. “CTI-JE: A Joint Extraction Framework of Entities and Relations in Unstructured Cyber Threat Intelligence”. In: ed. by Shen W et al. Institute of Electrical and Electronics Engineers Inc., 2024, pp. 2728–2733. DOI: 10.1109/CSCWD61410.2024.10580210.
- [Liu20] W. Liu. “Network Security Entity Recognition Methods Based on the Deep Neural Network”. In: ed. by Huang C et al. Vol. 1088. Springer, 2020, pp. 1687–1692. DOI: 10.1007/978-981-15-1468-5\_201.
- [LLP25] G. Liu et al. “Graph neural networks embedded with domain knowledge for cyber threat intelligence entity and relationship mining”. In: *PeerJ Computer Science* 11 (2025). DOI: 10.7717/peerj-cs.2769.
- [LMP01] John D. Lafferty et al. “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data”. In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. Morgan Kaufmann Publishers Inc., 2001, pp. 282–289. ISBN: 978-1-55860-778-1.
- [Lu+23] Y. Lu et al. “Cybersecurity Named Entity Recognition Based on Word-level Enhancement and Multi-task Learning”. In: *2023 7th International Conference on Deep Learning Technologies, ICDLT 2023; Dalian; China; 27 July 2023 through 29 July 2023; Code 193263*. Association for Computing Machinery, 2023, pp. 71–78. DOI: 10.1145/3613330.3613346.
- [Lv+24] H. Lv et al. “Joint Extraction of Entities and Relationships from Cyber Threat Intelligence based on Task-specific Fourier Network”. In: Institute of Electrical and Electronics Engineers Inc., 2024. DOI: 10.1109/IJCNN60899.2024.10650942.

- [Ma+18] C. Ma et al. “DM NLP at SemEval-2018 Task 8: Neural Sequence Labeling with Linguistic Features”. In: *NAACL HLT 2018 - International Workshop on Semantic Evaluation, SemEval 2018 - Proceedings of the 12th Workshop*. Ed. by Apidianaki M et al. Association for Computational Linguistics (ACL), 2018, pp. 707–711. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85067981267&partnerID=40&md5=93698283c85e8aee2349622ed8d40ddf>.
- [Ma+21] P. Ma et al. “Cybersecurity named entity recognition using bidirectional long short-term memory with conditional random fields”. In: *Tsinghua Science and Technology* 26.3 (2021), pp. 259–265. DOI: 10.26599/TST.2019.9010033.
- [Ma+23] C. Ma et al. “FineCTI: A Framework for Mining Fine-grained Cyber Threat Information from Twitter Using NER Model”. In: ed. by Hu J et al. Institute of Electrical and Electronics Engineers Inc., 2023, pp. 531–538. DOI: 10.1109/TrustCom60117.2023.00085.
- [MA23] M.A. Marjan and T. Amagasa. “CSER: Enhancing Cybersecurity Entity Recognition Through Multidimensional Feature Fusion”. In: ed. by He J et al. Institute of Electrical and Electronics Engineers Inc., 2023, pp. 1214–1221. DOI: 10.1109/BigData59044.2023.10386941.
- [MCV23] F. Marchiori et al. “STIXnet: A Novel and Modular Solution for Extracting All STIX Objects in CTI Reports”. In: *18th International Conference on Availability, Reliability and Security, ARES 2023; Benevento; Italy; 29 August 2023 through 1 September 2023; Code 191642*. Association for Computing Machinery, 2023. DOI: 10.1145/3600160.3600182.
- [Men+24] M. Meng et al. “Research on BERT-Based Named Entity Method for Network Security”. In: ed. by Xu B. Institute of Electrical and Electronics Engineers Inc., 2024, pp. 1155–1159. DOI: 10.1109/ITNEC60942.2024.10732930.
- [Mic25] Microsoft. *Chunk documents in vector search - Azure AI Search*. 2025. URL: <https://learn.microsoft.com/en-us/azure/search/vector-search-how-to-chunk-documents> (visited on 08/27/2025).
- [Mik+13] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. DOI: 10.48550/arXiv.1301.3781.
- [MMS25] Inoussa Mouiche et al. *Context-aware Entity-Relation Extraction Pipeline for Threat Intelligence Knowledge Graphs*. 2025. DOI: 10.36227/tehrxiv.173627493.39916970/v1.
- [MS24] Inoussa Mouiche and Sherif Saad. “TI-NERmerger: Semi-Automated Framework for Integrating NER Datasets in Cybersecurity:” in: *Proceedings of the 21st International Conference on Security and Cryptography*. SCITEPRESS - Science and Technology Publications, 2024, pp. 357–370. ISBN: 978-989-758-709-2. DOI: 10.5220/0012867900003767.

- [MS25a] I. Mouiche and S. Saad. “Entity and relation extractions for threat intelligence knowledge graphs”. In: *Computers and Security* 148 (2025). DOI: 10.1016/j.cose.2024.104120.
- [MS25b] Inoussa Mouiche and Sherif Saad. *TIJERE: A Novel Threat Intelligence Joint Extraction Model based on Analyst Expert Knowledge*. 2025. DOI: 10.36227/techrxiv.174286575.55673704/v1.
- [MSZ24] Y. Mostafa et al. “Automating Cyber Defense: Enhancing Threat Intelligence with AI-Driven Annotation”. In: Institute of Electrical and Electronics Engineers Inc., 2024. DOI: 10.1109/ICSPIS63676.2024.10812585.
- [Nat06] National Institute of Standards and Technology (US). *Minimum security requirements for federal information and information systems*. Tech. rep. National Institute of Standards and Technology (U.S.), 2006. DOI: 10.6028/NIST.FIPS.200.
- [Ope+25] OpenAI et al. *gpt-oss-120b & gpt-oss-20b Model Card*. 2025. DOI: 10.48550/arXiv.2508.10925.
- [Ozi25] Filipp Ozinov. *bakwc/JamSpell*. 2025. URL: <https://github.com/bakwc/JamSpell> (visited on 09/08/2025).
- [Pal+24] T. Paladini et al. “You Might Have Known It Earlier: Analyzing the Role of Underground Forums in Threat Intelligence”. In: *The International Symposium on Research in Attacks, Intrusions and Defenses*. Association for Computing Machinery, 2024, pp. 368–383. DOI: 10.1145/3678890.3678930.
- [Pen+24] Z. Peng et al. “Research on Knowledge Graph Construction for Smart Grid Cybersecurity”. In: *3rd International Conference on Cryptography, Network Security and Communication Technology, CNSCT 2024; Virtual, Online; China; 19 January 2024 through 21 January 2024; Code 201392*. Association for Computing Machinery, 2024, pp. 164–170. DOI: 10.1145/3673277.3673306.
- [Per+24] D.R. Permana et al. “An Enhanced Method with Part of Speech Tagging and Named Entity Recognition Techniques Towards Advanced Persistent Threat in Cyber Threat Intelligence: Work in Progress”. In: Institute of Electrical and Electronics Engineers Inc., 2024, pp. 493–498. DOI: 10.1109/EECSI63442.2024.10776424.
- [PL22] Y. Park and T. Lee. “Full-Stack Information Extraction System for Cybersecurity Intelligence”. In: *EMNLP 2022 - Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*. Association for Computational Linguistics (ACL), 2022, pp. 541–549. DOI: 10.18653/v1/2022.emnlp-industry.54.
- [Por25] Pierre-Antoine Porte. *Doing RAG on PDFs using File Search in the Responses API | OpenAI Cookbook*. 2025. URL: [https://cookbook.openai.com/examples/file\\_search\\_responses](https://cookbook.openai.com/examples/file_search_responses) (visited on 08/27/2025).

- [PSM14] Jeffrey Pennington et al. “Glove: Global Vectors for Word Representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162.
- [Qi+20] Peng Qi et al. “Stanza: A Python Natural Language Processing Toolkit for Many Human Languages”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, 2020, pp. 101–108. DOI: 10.18653/v1/2020.acl-demos.14. (Visited on 08/29/2025).
- [Qi+22] Y. Qi et al. “A General Construction Method of Cyber Security Knowledge Graph”. In: Institute of Electrical and Electronics Engineers Inc., 2022. DOI: 10.1109/BESC57393.2022.9995070.
- [Qin+19] Y. Qin et al. “A network security entity recognition method based on feature template and CNN-BiLSTM-CRF”. In: *Frontiers of Information Technology and Electronic Engineering* 20.6 (2019), pp. 872–884. DOI: 10.1631/FITEE.1800520.
- [QZD23] Z. Qiao et al. “Improving Cybersecurity Named Entity Recognition with Large Language Models”. In: Institute of Electrical and Electronics Engineers Inc., 2023. DOI: 10.1109/CSECS60003.2023.10428218.
- [Rad+18] Alec Radford et al. “Improving Language Understanding by Generative Pre-Training”. In: (2018).
- [Rad+19] Alec Radford et al. “Language Models are Unsupervised Multitask Learners”. In: (2019).
- [Ram+22] N. Ramrakhiani et al. “Extracting Entities and Events from Cyber-Physical Security Incident Reports”. In: *22nd IEEE International Conference on Data Mining Workshops, ICDMW 2022; Orlando; United States; 28 November 2022 through 1 December 2022; Category number CFP2256B-ART; Code 186593*. Ed. by Candan K.S et al. Vol. 2022-November. IEEE Computer Society, 2022, pp. 602–609. DOI: 10.1109/ICDMW58026.2022.00083.
- [Ran+21] Priyanka Ranade et al. “CyBERT: Contextualized Embeddings for the Cybersecurity Domain”. In: *2021 IEEE International Conference on Big Data (Big Data)*. 2021, pp. 3334–3342. DOI: 10.1109/BigData52589.2021.9671824. (Visited on 10/09/2025).
- [RPP17] Arpita Roy et al. *Learning Domain-Specific Word Embeddings from Sparse Cybersecurity Texts*. 2017. DOI: 10.48550/arXiv.1709.07470.
- [Sae+23] Saqib Saeed et al. “A Systematic Literature Review on Cyber Threat Intelligence for Organizational Cybersecurity Resilience”. In: *Sensors (Basel, Switzerland)* 23.16 (2023), p. 7273. ISSN: 1424-8220. DOI: 10.3390/s23167273.

- [San+22] C. Sandescu et al. “EXTRACTING EXPLOITS AND ATTACK VECTORS FROM CYBERSECURITY NEWS USING NLP”. In: *UPB Scientific Bulletin, Series C: Electrical Engineering and Computer Science* 84.2 (2022), pp. 63–78. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85131862306&partnerID=40&md5=d0f0034ef3cff027b5579a9858b3b752>.
- [SAR23] Ivano SARNO. *What Is a Large Language Model?* Text. 2023. URL: <https://knowledge-centre-translation-interpretation.ec.europa.eu/en/news/what-large-language-model> (visited on 01/03/2026).
- [Say+24] A. Sayari et al. “DECEPT-CTI: A Framework for Enhancing Cyber Deception Strategies through NLP-based Extraction of CTI from Unstructured Reports”. In: Institute of Electrical and Electronics Engineers Inc., 2024, pp. 286–293. DOI: 10.1109/NCA61908.2024.00049.
- [Sch23] Roie Schwaber-Cohen. *What is a Vector Database & How Does it Work? Use Cases + Examples / Pinecone*. 2023. URL: <https://www.pinecone.io/learn/vector-database/> (visited on 06/11/2024).
- [SD23] Y. Su and Y. Ding. “SECURENET: Advancing Cybersecurity Knowledge Graphs with Multi-Feature Perceptrons and Hierarchical Adversarial W2NER”. In: Institute of Electrical and Electronics Engineers Inc., 2023, pp. 633–636. DOI: 10.1109/IAECST60924.2023.10503315.
- [SFF19] T. Satyapanich et al. “Extracting Rich Semantic Information about Cybersecurity Events”. In: ed. by Baru C et al. Institute of Electrical and Electronics Engineers Inc., 2019, pp. 5034–5042. DOI: 10.1109/BigData47090.2019.9006444.
- [SGV21] Kiavash Satvat et al. *EXTRACTOR: Extracting Attack Behavior from Threat Reports*. 2021. DOI: 10.48550/arXiv.2104.08618.
- [Sim+20] K. Simran et al. “Deep Learning Approach for Enhanced Cyber Threat Indicators in Twitter Stream”. In: ed. by Thampi S.M et al. Vol. 1208 CCIS. Springer, 2020, pp. 135–145. DOI: 10.1007/978-981-15-4825-3\_11.
- [SP97] M. Schuster and K.K. Paliwal. “Bidirectional recurrent neural networks”. In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681. ISSN: 1941-0476. DOI: 10.1109/78.650093.
- [SPG22] S. Srivastava et al. “Study of Word Embeddings for Enhanced Cyber Security Named Entity Recognition”. In: *2022 International Conference on Machine Learning and Data Engineering, ICMLDE 2022; Dehradun; India; 7 September 2022 through 8 September 2022; Code 188151*. Ed. by Singh V. Vol. 218. Elsevier B.V., 2022, pp. 449–460. DOI: 10.1016/j.procs.2023.01.027.
- [SS21] I. Sarhan and M. Spruit. “Open-CyKG: An Open Cyber Threat Intelligence Knowledge Graph”. In: *Knowledge-Based Systems* 233 (2021). DOI: 10.1016/j.knsys.2021.107524.

- [Su+22a] Jianlin Su et al. *Global Pointer: Novel Efficient Span-based Approach for Named Entity Recognition*. 2022. DOI: 10.48550/arXiv.2208.03054.
- [Su+22b] Z. Su et al. “An identification method for threat intelligence in the security field”. In: Institute of Electrical and Electronics Engineers Inc., 2022, pp. 149–153. DOI: 10.1109/CECIT58139.2022.00035.
- [Tea+25] Gemma Team et al. *Gemma 3 Technical Report*. 2025. DOI: 10.48550/arXiv.2503.19786.
- [Tho+23] Nguyen Dai Tho et al. “Extracting Multiple Relations between Entities from Unstructured Threat Intelligence Reports”. In: *Journal of Science and Technology on Information security* (2023), pp. 5–13. ISSN: 2615-9570. DOI: 10.54654/isj.v3i20.976.
- [TM21] N. Tsinganos and I. Mavridis. *Building and evaluating an annotated corpus for automated recognition of chat-based social engineering attacks*. 2021. DOI: 10.3390/app112210871.
- [Ull+24] S. Ullah et al. “TTPatternMiner: Automated Learning and Characterization of Attack Pattern from Malicious Cyber Campaign”. In: Institute of Electrical and Electronics Engineers Inc., 2024, pp. 405–411. DOI: 10.1109/MILCOM61039.2024.10773782.
- [Vas+17] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: [https://proceedings.neurips.cc/paper\\_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html](https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html) (visited on 01/02/2026).
- [Wan+20a] X. Wang et al. “NER in Threat Intelligence Domain with TSFL”. In: ed. by Zhu X et al. Vol. 12430 LNAI. Springer Science and Business Media Deutschland GmbH, 2020, pp. 157–169. DOI: 10.1007/978-3-030-60450-9\_13.
- [Wan+20b] Xuren Wang et al. “DNRTI: A Large-Scale Dataset for Named Entity Recognition in Threat Intelligence”. In: *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 2020, pp. 1842–1848. DOI: 10.1109/TrustCom50675.2020.00252.
- [Wan+21] X. Wang et al. “A Method for Extracting Unstructured Threat Intelligence Based on Dictionary Template and Reinforcement Learning”. In: ed. by Shen W et al. Institute of Electrical and Electronics Engineers Inc., 2021, pp. 262–267. DOI: 10.1109/CSCWD49262.2021.9437858.
- [Wan+22a] X. Wang et al. “Cyber Threat Intelligence Entity Extraction Based on Deep Learning and Field Knowledge Engineering”. In: Institute of Electrical and Electronics Engineers Inc., 2022, pp. 406–413. DOI: 10.1109/CSCWD54268.2022.9776139.

- [Wan+22b] Xuren Wang et al. “APTNER: A Specific Dataset for NER Missions in Cyber Threat Intelligence Field”. In: *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. 2022, pp. 1233–1238. DOI: 10.1109/CSCWD54268.2022.9776031.
- [Wan+24a] B. Wang et al. “Fine-grained cybersecurity entity typing based on multimodal representation learning”. In: *Multimedia Tools and Applications* 83.10 (2024), pp. 30207–30232. DOI: 10.1007/s11042-023-16839-z.
- [Wan+24b] R. Wang et al. “Xabcd: A Better Model for Named Entity Recognition in Cyber Threat Intelligence”. In: Institute of Electrical and Electronics Engineers Inc., 2024, pp. 966–971. DOI: 10.1109/ICSP62122.2024.10743764.
- [WC19] T. Wang and K.P. Chow. “Automatic tagging of cyber threat intelligence unstructured data using semantics extraction”. In: ed. by Zheng X et al. Institute of Electrical and Electronics Engineers Inc., 2019, pp. 197–199. DOI: 10.1109/ISI.2019.8823252.
- [WL23] P. Wang and J. Liu. “A Cyber Threat Entity Recognition Method Based on Robust Feature Representation and Adversarial Training”. In: *12th International Conference on Computing and Pattern Recognition, ICCPR 2023; Qingdao; China; 27 October 2023 through 29 October 2023; Code 197800*. Association for Computing Machinery, 2023, pp. 255–259. DOI: 10.1145/3633637.3633677.
- [WLG20] H. Wu et al. “An Effective Approach of Named Entity Recognition for Cyber Threat Intelligence”. In: ed. by Xu B and Mou K. Institute of Electrical and Electronics Engineers Inc., 2020, pp. 1370–1374. DOI: 10.1109/ITNEC48623.2020.9085102.
- [Woi+25] Elisabeth Woisetschläger et al. “SC4OSINT: A Story Clustering Approach to Optimize OSINT Analysis”. In: *Availability, Reliability and Security: ARES 2025 International Workshops, Ghent, Belgium, August 11–14, 2025, Proceedings, Part II*. Springer-Verlag, 2025, pp. 5–24. ISBN: 978-3-032-00632-5. DOI: 10.1007/978-3-032-00633-2\_1.
- [Wu+25] C. Wu et al. “Cybersecurity entity recognition model for IoT via hierarchical attention mechanism”. In: *International Journal of Modern Physics C* (2025). DOI: 10.1142/S0129183124420130.
- [Xia18] Z. Xiao. “Towards a two-phase unsupervised system for cybersecurity concepts extraction”. In: ed. by Zhao L et al. Institute of Electrical and Electronics Engineers Inc., 2018, pp. 2161–2168. DOI: 10.1109/FSKD.2017.8393106.
- [Xie+24] Y. Xie et al. “BERT-BiLSTM-CRF-DRL—A Named Entity Recognition Method for Knowledge Graph”. In: Institute of Electrical and Electronics Engineers Inc., 2024, pp. 197–202. DOI: 10.1109/ICISCAE62304.2024.10761476.

- [Yan+24] C. Yang et al. “A Cybersecurity Entity Recognition Method for Enhancing Situation Awareness in Power Systems”. In: Institute of Electrical and Electronics Engineers Inc., 2024, pp. 212–217. DOI: 10.1109/NEESSC62857.2024.10733391.
- [Yi+20] F. Yi et al. “Cybersecurity Named Entity Recognition Using Multi-Modal Ensemble Learning”. In: *IEEE Access* 8 (2020), pp. 63214–63224. DOI: 10.1109/ACCESS.2020.2984582.
- [Yi+24] J. Yi et al. “Text Command Intelligent Understanding for Cybersecurity Testing”. In: *Electronics (Switzerland)* 13.21 (2024). DOI: 10.3390/electronics13214330.
- [You+24] Y. You et al. “P-TIMA: a framework of Twitter threat intelligence mining and analysis based on a prompt-learning NER model”. In: *Computer Journal* 67.12 (2024), pp. 3221–3238. DOI: 10.1093/comjnl/bxae084.
- [Zar+23] Urchade Zaratiana et al. *GLiNER: Generalist Model for Named Entity Recognition using Bidirectional Transformer*. 2023. DOI: 10.48550/arXiv.2311.08526.
- [Zha+22] K. Zhang et al. “Research on Named Entity Recognition Method of Network Threat Intelligence”. In: ed. by Lu W et al. Vol. 1699 CCIS. Springer Science and Business Media Deutschland GmbH, 2022, pp. 213–224. DOI: 10.1007/978-981-19-8285-9\_16.
- [Zha+25] Y. Zhang et al. “SecLMNER: A framework for enhanced named entity recognition in multi-source cybersecurity data using large language models”. In: *Expert Systems with Applications* 271 (2025). DOI: 10.1016/j.eswa.2025.126651.
- [Zho+21] S. Zhou et al. “Named Entity Recognition Using BERT with Whole World Masking in Cybersecurity Domain”. In: Institute of Electrical and Electronics Engineers Inc., 2021, pp. 316–320. DOI: 10.1109/ICBDA51983.2021.9403180.
- [Zho+22] Y. Zhou et al. “CTI View: APT Threat Intelligence Analysis System”. In: *Security and Communication Networks* 2022 (2022). DOI: 10.1155/2022/9875199.
- [Zhu+24] F. Zhu et al. “ITIRel: Joint Entity and Relation Extraction for Internet of Things Threat Intelligence”. In: *IEEE Internet of Things Journal* 11.11 (2024), pp. 20867–20878. DOI: 10.1109/JIOT.2024.3373799.
- [Zuo+22] J. Zuo et al. “An End-to-end Entity and Relation Joint Extraction Model for Cyber Threat Intelligence”. In: Institute of Electrical and Electronics Engineers Inc., 2022, pp. 204–209. DOI: 10.1109/ICBDA55095.2022.9760342.

- [ZZ25] J. Zhao and W. Zheng. “Cybersecurity Entity Recognition Method Combined with Feedforward Neural Network Enhancement”. In: ed. by Sun F et al. Vol. 1328. Springer Science and Business Media Deutschland GmbH, 2025, pp. 95–104. DOI: 10.1007/978-981-96-1698-5\_10.
- [ZZZ23] S. Zhang et al. “VIET: A Tool for Extracting Essential Information from Vulnerability Descriptions for CVSS Evaluation”. In: ed. by Atluri V and Ferrara A.L. Vol. 13942 LNCS. Springer Science and Business Media Deutschland GmbH, 2023, pp. 386–403. DOI: 10.1007/978-3-031-37586-6\_23.