Development of a Machine Learning Model for Named Entity Recognition in Cybersecurity

Robin BARSCZUS - Lila BOURDEAU
robin1.barsczus@ipsa.fr - lila.bourdeau@ipsa.fr
Institut polytechnique des sciences avancées (IPSA), Paris

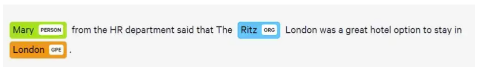
Abstract

This work focuses on developing a Named Entity Recognition (NER) system using a BERT-based model. The process involved preprocessing annotated data, addressing class imbalance and training the model over multiple epochs. Evaluation metrics and classification reports were used to assess the model's performance and identify areas for improvement.

1 INTRODUCTION

Named Entity Recognition (NER) is a sub-task of information extraction in Natural Language Processing (NLP) that classifies named entities into predefined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values and more. In the realm of NLP, understanding these entities is crucial for many applications as they often contain the most significant information in a text.

For example, consider the sentence: "Mary from the HR department said that The Ritz London was a great hotel option to stay in London."



Mary PERSON from the HR department said that The Ritz ORG London was a great hotel option to stay in London GPE .

Figure 1: Example

This image illustrates the process of Named Entity Recognition (NER) by showcasing how specific segments of a text query are identified and categorized into predefined entities.
"Mary" is labeled as PERSON. "The Ritz" is tagged as ORG, which stands for Organization. And "London" has been classified as GPE, which stands for Geopolitical entity.
NER systems automate this process by using machine learning models to extract structured information from unstructured text, making it highly relevant for tasks that require fast and accurate data processing, such as analyzing cybersecurity documents.
This diagram serves as a helpful visual to understand how our project similarly aims to extract meaningful entities, such as IP addresses and vulnerabilities, from cybersecurity texts.
In the field of cybersecurity, professionals often face the challenging task of analyzing vast volumes of textual data, such as incident reports, threat intelligence feeds, and technical blogs. Performing actionable insights manually from these sources is both time-consuming and susceptible to errors. This project aims to overcome these obstacles by developing a naming entity recognition (NER) model based on machine learning to automate the identification of cybersecurity-specific entities.
The goal is to save time and improve the efficiency of cybersecurity professionals by facilitating faster analysis of critical information.
This report focuses on leveraging pre-trained models from the Hugging Face Transformers library, fine-tuned on a curated dataset of cybersecurity texts.
Fine-tuning allows the model to adapt to the unique vocabulary and entity types of the domain. Additionally, this report provides a visual representation of the NER process to help readers understand how the system identifies and categorizes entities.

2 RELATED WORK

Named Entity Recognition (NER) is a fundamental task in Natural Language Processing (NLP) that focuses on identifying and classifying entities within unstructured text into predefined categories such as names of people, organizations, locations, and domain-specific entities. Over the years, NER has evolved significantly, driven by advancements in machine learning and deep learning.
Early NER systems were primarily rule-based, relying on handcrafted features, dictionaries, and linguistic rules. These approaches, while effective in specific contexts, suffered from limited adaptability and scalability. As statistical methods gained

prominence, algorithms like Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs) became widely used for NER. These methods introduced probabilistic modeling, allowing systems to generalize better across domains while reducing the reliance on manual rule design (Lafferty et al., 2001).

The advent of deep learning marked a paradigm shift in NER research. Recurrent neural networks (RNNs), particularly long-short-term memory (LSTM) networks, proved effective in capturing contextual information in sequences. Models such as BiLSTM-CRF (Huang et al., 2015) combined the strengths of deep learning and probabilistic modeling, achieving state-of-the-art performance on benchmark datasets.

Pre-trained language models further revolutionized NER. Models like BERT (Devlin et al., 2018) and its domain-specific adaptations (e.g., BioBERT for biomedical texts and SciBERT for scientific literature) enabled transfer learning by leveraging vast amounts of unlabeled data. Fine-tuning these models on NER datasets significantly improved their ability to handle nuanced and domain-specific entity recognition tasks. Recent efforts such as RoBERTa (Liu et al., 2019) and T5 (Raffel et al., 2020) have further pushed the boundaries of pre-trained architectures, with enhanced capacity to understand complex language patterns.

Domain-specific NER systems have also gained attention in specialized fields such as healthcare, finance, and cybersecurity. For example, Spacy and Hugging Face's Transformers library have enabled researchers to develop models tailored to unique challenges, such as recognizing chemical compounds in biomedical texts or malware names in cybersecurity contexts. However, these domain-specific systems often face challenges related to data scarcity, requiring innovative approaches like data augmentation, weak supervision, or active learning to create high-quality annotated datasets (Ratner et al., 2017).

Despite significant progress, challenges remain in the field of NER. Handling ambiguous and overlapping entities, adapting to rapidly evolving vocabularies in specialized domains, and addressing biases in training datasets are active areas of research. Additionally, the growing need for multilingual NER has led to the development of models capable of handling diverse linguistic structures, such as XLM-R (Conneau et al., 2020).

In summary, NER has undergone substantial evolution from rule-based systems to advanced pre-trained models, with ongoing research focusing on addressing its limitations and expanding its applicability across domains and languages. These advancements have cemented NER as a critical component of modern NLP applications.

3 DATA EXPLORATION AND PREPROCESSING

This dataset is focused on the cybersecurity domain, specifically to analyze incidents related to cybersecurity breaches, techniques, and entities involved. Each entry in the dataset is structured with a unique_id, tokens (which represent individual words or terms), and ner_tags (which are labels indicating the type of information that each token represents).

Here is an example of a dataset entry:

{"unique_id": 74,
"tokens": ["Most", "recently", ",", "we", "observed", "them", "using", "yet", "another", "technique", "coupled", "with", "watering", "hole", "attacks", "."],
"ner_tags": ["O", "O", "O", "O", "O", "B-Entity", "B-Action", "B-Entity", "I-Entity", "I-Entity", "I-Entity", "I-Entity", "I-Entity", "I-Entity", "I-Entity", "O"]}

Here is an explanation of the tags used in the dataset:

- **B-Entity:** Indicates the beginning of a named entity related to the cybersecurity domain, such as a person, a tool, or a specific technology. Here, "them" and "another".
- **I-Entity:** Denotes subsequent tokens that are part of an entity, following the first token. Here, "technique", "coupled", "with", "watering", "hole" and "attacks".
- **B-Action:** Marks the beginning of an action or verb, indicating an event or activity, such as an attack or process in the cybersecurity context. Here, "using"
- **I-Action:** Similar to B-Action, but used for additional tokens that continue describing the same action.
- **B-Modifier:** Specifies a word that modifies or provides more context to an action or entity, such as adverbs or adjectives.
- **O:** Stands for "Other," which includes tokens that don't fit into any specific category (e.g., common words like "in," "as," or "the").

These tags adhere to the BIO (Begin-Inside-Outside) schema, a widely recognized standard for NER tasks. The BIO schema offers a straightforward and effective approach to marking entity boundaries and types, which is especially crucial in complex fields like cybersecurity, where entity boundaries may overlap or appear ambiguous.

For our study, we utilized three distinct datasets:

- **NER_Training:** This dataset was used to train the model and contains tokens along with their corresponding NER tags, enabling the model to learn entity patterns.
- **NER_Validation:** Employed during training, this dataset serves to validate the model's performance and includes labels for comparison purposes.
- **NER_Testing:** Reserved for testing, this dataset excludes labels to mimic real-world scenarios where the model predicts entities from unseen data.

To better understand the distribution of NER tags in our datasets, we applied a filtering step to both the training and validation datasets. Specifically, we removed sentences where all NER tags were labeled as `O`, which indicates that the tokens in those sentences do not correspond to any named entity. This preprocessing step was crucial for two reasons:

- It allowed us to focus on sentences containing meaningful entity annotations, ensuring a more accurate analysis of the tag distribution.
- It reduced noise in the datasets, as sentences with only non-entity tokens do not contribute to learning entity boundaries or types.

By refining the datasets in this way, we gained a clearer understanding of the distribution of named entities and ensured that the subsequent modeling efforts were focused on relevant data.

And we obtain these distributions (Figures 2 and 3):

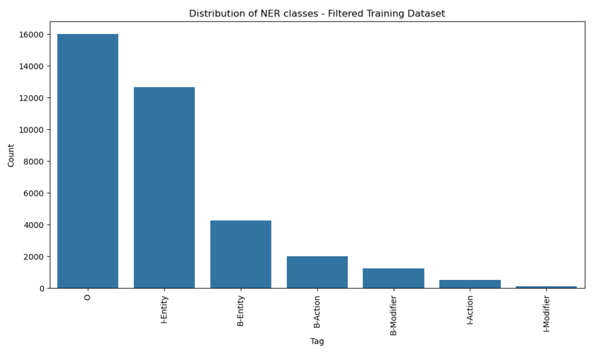


Figure 2: Distribution of the filtered training dataset

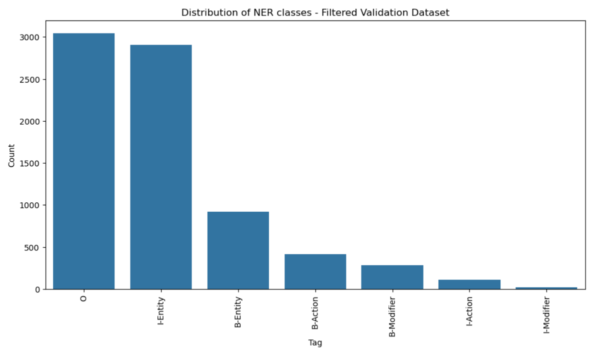


Figure 3: Distribution of the filtered validation dataset

4 MODEL SELECTION

4.1 BERT Model

The choice of BERT for token classification in this project is primarily motivated by its ability to capture contextual language understanding through a bidirectional transformer architecture. This characteristic is essential for Named Entity Recognition (NER), where the meaning of entities often depends on surrounding words. BERT's pre-training on massive corpora using a masked language modeling objective allows it to transfer general linguistic knowledge effectively, reducing the dependency on large labeled datasets. The architecture is naturally suited for token-level tasks, with a simple classification head that predicts NER tags while leveraging BERT's rich embeddings. Additionally, incorporating a weighted loss function addresses the challenge of imbalanced NER datasets, ensuring attention to underrepresented tags. Compared to alternatives like BiLSTM-CRF or unidirectional transformer models, BERT provides dynamic, context-aware embeddings that adapt to complex sentence structures, significantly enhancing entity recognition. Training from scratch was also ruled out due to the substantial computational and data requirements, making BERT's pre-trained model an efficient and effective choice for achieving state-of-the-art performance in NER tasks.

We used BERT (Bidirectional Encoder Representations from Transformers), specifically the pre-trained `bert-base-cased` model, fine-tuned for token classification. This decision was guided by several factors:

- **Contextual understanding:** BERT is designed to capture bidirectional context, making it highly effective for tasks like Named Entity Recognition (NER) where understanding the context around each word is crucial.
- **Pre-training on large corpora:** Pre-trained BERT has already learned language representations from a vast corpus, reducing the amount of data needed for fine-tuning in domain-specific tasks.
- **Flexibility for token-level tasks:** By using a token classification head, BERT can be

adapted for NER by assigning a class label to each token in a sequence.

4.2 Training explanations

The model was fine-tuned using the following procedure:

4.2.1 Loss function

The Cross-Entropy Loss function was used to optimize the model during training. This function measures the difference between the predicted probabilities for each token's class (output logits) and the true class labels. To address the issue of class imbalance—a common challenge in NER tasks due to the disproportionate representation of certain entity types—we computed class weights. These weights were derived from the frequency of each tag in the training data. By applying these weights, the loss function penalized misclassifications of rare classes more heavily than those of common ones, ensuring a balanced learning process.

4.2.2 Optimizer

We chose the AdamW optimizer, an extension of the Adam optimizer, designed to work effectively with weight decay. This choice ensures efficient gradient updates while preventing overfitting by penalizing large weight magnitudes. The learning rate was set to 2e-5, a value commonly used for fine-tuning pre-trained models like BERT, as it balances between convergence speed and stability.

4.2.3 Training loop

The training loop iteratively updated the model's weights over multiple epochs. The key steps are as follows:

1. **Forward pass**: For each mini-batch, the model processed the input IDs and attention masks, generating logits (predicted probabilities) for each token.

2. **Loss calculation**: The logits were compared with the true labels to compute the Cross-Entropy Loss. Only active tokens (non-padding tokens) contributed to the loss.

3. **Backward pass**: Gradients of the loss with respect to the model's parameters were computed using backpropagation.

4. **Optimizer step**: The optimizer updated the model's weights based on the computed gradients.

5. **Progress Tracking**: The `tqdm` library was used to display a real-time progress bar, showing batch-level loss updates and tracking overall progress within each epoch.

Hyperparameters

The training process was guided by the following hyperparameters:

- **Batch size**: 4 sentences per batch, balancing memory usage and computational efficiency.

- **Number of epochs**: 20 passes over the entire training dataset to ensure sufficient time for model convergence.

- **Sequence length**: 256 tokens per sentence. This length was chosen to accommodate most sentences in the dataset while managing memory usage. Truncated or padded sequences ensured uniformity in input lengths.

This detailed training process was critical for fine-tuning the BERT model, allowing it to learn patterns in the data effectively and generalize well to unseen examples.

5 EVALUATION OF THE MODEL

To evaluate the model, a dedicated validation dataset, separate from the training set, was used to assess the model's performance throughout the training process. The evaluation employed metrics such as precision, recall, F1-score, and support for each tag, computed using the `classification_report` function from the `sklearn` library. The model's predictions, initially generated as logits for each token, were converted into class labels by applying the `argmax` function, which selects the most probable tag for each token.

5.1 F-1 Score

To assess the performance of the model, the F1-score was selected as the primary evaluation metric. Unlike accuracy, which focuses solely on the proportion of correct predictions, the F1-score offers a more comprehensive evaluation by combining precision, the fraction of correctly identified entities out of all entities predicted by the model and recall, the fraction of correctly identified entities out of all actual entities in the dataset.

The F1-score is computed as the harmonic mean of precision and recall, ensuring a balanced assessment:

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

This metric is particularly suitable for Named Entity Recognition (NER) tasks, where data often exhibits significant class imbalances, such as the dominance of the "O" tag. By emphasizing both false positives and false negatives, the F1-score ensures a robust evaluation of the model's capability to accurately detect and classify entities.

5.2 Loss analysis over epochs

The model shows a consistent decrease in the average loss over 20 epochs, as summarized below:

Epoch	Average Loss
1	1.7450
5	0.7057
10	0.2548
15	0.1152
20	0.0491

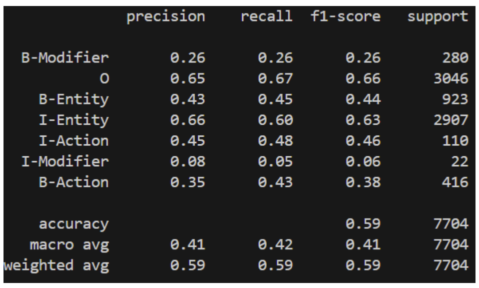
Table 1: Average loss per epoch

Phases of training

- **Initial phase (Epochs 1-5):** Rapid decrease in loss, indicating the model is effectively learning basic patterns in the training data.
- **Intermediate phase (Epochs 6-15):** Slower decrease in loss, showing the model is gradually converging.
- **Final phase (Epochs 16-20):** Minimal loss values, but classification metrics suggest potential overfitting or class imbalance.

5.3 Classification Report

The classification report provides detailed performance metrics for each class:



```
              precision    recall  f1-score   support

  B-Modifier       0.26      0.26      0.26       280
           O       0.65      0.67      0.66      3046
    B-Entity       0.43      0.45      0.44       923
    I-Entity       0.66      0.60      0.63      2907
    I-Action       0.45      0.48      0.46       110
  I-Modifier       0.08      0.05      0.06        22
    B-Action       0.35      0.43      0.38       416

    accuracy                           0.59      7704
   macro avg       0.41      0.42      0.41      7704
weighted avg       0.59      0.59      0.59      7704
```

Figure 4: Example

Analysis of the report

- **Class "O":** High precision and recall (0.65 and 0.67), as this class is over-represented in the dataset (support = 3046).
- **B-Entity and I-Entity Classes:** Moderate performance (F1-score $\approx 0.44 - 0.63$), but further improvement is possible.
- **Underrepresented Classes:** Classes like I-Modifier (support = 22) have very low metrics, reflecting difficulty in learning due to data imbalance.
- **Macro vs. Weighted Averages:** Weighted averages are higher than macro averages due to the dominance of frequent classes.

5.4 Key observations

- **Overall performance:** The model achieves moderate accuracy (59%) and weighted F1-score (0.59), largely driven by the dominant class "O".
- **Class imbalance:** Rare classes like I-Modifier and B-Modifier are poorly learned, leading to low precision and recall.
- **Complexity in entities:** The model struggles to distinguish between overlapping or complex entity classes like B-Action and B-Entity.

6 CONCLUSION AND RECOMMENDATIONS FOR IMPROVEMENT

To enhance the model's performance, several strategies can be implemented. Addressing class imbalance is crucial, as the underrepresented classes hinder overall accuracy. Techniques such as loss weighting can assign higher importance to rare classes during training, while data augmentation can artificially expand the dataset for these labels. Improving the dataset itself is another vital step. Ensuring the quality and consistency of annotations, especially for complex or nuanced labels, will help the model learn more effectively. Additionally, annotating more data for underrepresented classes would increase their support, making them easier to recognize.

Hyperparameter optimization can further refine the model. Experimenting with different learning rates, batch sizes, and the number of training epochs could lead to better convergence. Exploring advanced architectures, such as domain-adapted transformers or ensemble models, may also capture finer patterns and contextual dependencies. Finally, incorporating detailed evaluation tools, such as confusion matrices, can provide insight into class-specific errors and guide targeted improvements.

By implementing these recommendations, the model's overall performance can be enhanced, achieving better balance across all classes and addressing its current limitations effectively.

To conclude, this project aimed to develop a Named Entity Recognition (NER) model using fine-tuning of a BERT-based architecture. The results demonstrate moderate success, with the model effectively identifying common entities while struggling with less frequent and contextually nuanced labels. The consistent reduction in loss indicates robust training, while the evaluation metrics highlight areas for improvement, particularly for minority classes.

Future work could focus on addressing class imbalance through techniques such as data augmentation, weighted loss adjustments, or oversampling. Additionally, leveraging advanced architectures, such as domain-adapted models or ensemble methods, might enhance performance. Despite its limitations, the model provides a solid foundation for further refinement and practical application in entity recognition tasks.