The Rise of Hydra-BERT

A Multiheaded Approach for Multiclass Event Extraction on a Single Language Model Body

Master's thesis in Computer science and engineering

Christian Weckner

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Master's thesis 2024

The Rise of Hydra-BERT

A Multiheaded Approach for Multiclass Event Extraction on a Single Language Model Body

Christian Weckner

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

The Rise of Hydra-BERT
A Multiheaded Approach for Multiclass Event Extraction on a Single Language Model Body
Christian Weckner

Supervisor: Lovisa Hagström, Department of Computer Science and Engineering
Advisor: Aron Lagerberg, Recorded Future
Examiner: Richard Johansson, Department of Computer Science and Engineering

Master's Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A "cyber" interpretation of the Hydra, a multiheaded dragon found in Greek mythology. Generated using ChatGPT and Dall-e, powered by GPT-4.

Typeset in LaTeX
Gothenburg, Sweden 2024

The Rise of Hydra-BERT
A Multiheaded Approach for Multiclass Event Extraction on a Single Language Model Body
Christian Weckner
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Every day, millions of pieces of text hit the internet. A fraction of these describe events which can be invaluable in the right context. Recorded Future uses a platoon of event extraction models, attempting to find information nuggets in a sea of digital noise. Each model is only trained on a specific event type, leaving a lot of potential data synergies unexplored. This thesis proposes an alternative model, trained on all event types. The model should be able to detect events and tag roles equal to or better than models dedicated to a specific event type. It should also be a continual learner, not deteriorating on old event types as new ones are added.

The resulting model, called Hydra2, was trained on six different event types. It outperformed the baseline models in all event detection and role tagging tasks. Furthermore, the observed increase in performance also hints at hidden similarities among the event types utilized in these tasks. A smaller version, called Hydra2b, showed potential for continual learning, though further studies are required before declaring it a definite success.

Keywords: NLP, event extraction, event detection, role tagging, hydra, continual learning.

Acknowledgements

I want to start off by expressing my deepest gratitude to my supervisor Lovisa Hagström and advisor Aron Lagerberg for their immense support during this thesis. I also want to thank Tobias Norlund for helping me get the project started, and the rest of the Text Analytics team for always being available for questions. Furthermore, I want to thank my friends and colleagues at Chalmers, Recorded Future and anywhere else they might be found. Finally, a big thank you to my family for always supporting me.

Christian Weckner, Gothenburg, 2024-02-09

Contents

List of Figures

List of Tables

1

Introduction

This master's thesis is in collaboration with Recorded Future. Recorded Future is a Cyber Security company, specializing in the collection, processing, analysis, and dissemination of threat intelligence. Different clients are interested in different kinds of threats. One might have a special interest in credit card frauds, wheres another wants to stay up to date on the geopolitical situation of the area surrounding an office site. This information is often available somewhere in the millions of data streams feeding into the Internet, but finding the relevant information pieces, and connecting them to paint a larger picture is a tedious task.

In the case of threat intelligence, the desired 'gold nuggets' are events: text strings that contain an entity and describe something that happens to that entity. Picking out events from text is a research field in itself, called event extraction, which contains many different subtasks. At Recorded Future, event extraction is used to (1), classify the text, (e.g. as a Disaster or Cyber Attack), and (2), based on the classification, assign labels to the different roles involved. These two tasks are more formally known as event detection and role tagging, respectively. As an example, the text "A 4.5 magnitude earthquake struck south of Reykjavik." would be classified by (1) as Disaster, and (2) would assign Reykjavik the role Location.

Thankfully, the rise of transformer-based language models has simplified the task. Recorded Future employs a platoon of BERT-based classifier models to sift through text sources to identify different kinds of interesting events. Each model is trained to detect a specific event type, and classify the text as either <EVENT TYPE> or <NEGATIVE>. Once the text has passed through each model, (at most) one of them will have returned a positive prediction and tagged roles.

A system like this is easy to maintain; each model can be retrained independently of the others, and to add a new event type is as simple as adding a new model trained on data specific to that type. The drawback to this approach is scalability. Each model added to production increases the resource usage linearly. This increase might be unnecessarily large, as many models use the same language model body.

In this project, we propose a multi-class model as an alternative. The idea is to use a single model, trained in a multi-class environment, to classify the text and label the roles in a single pass. This model will be trained on the same data as the current models, but pooled rather than separated by event type. A single model could reduce resource usage, consequently lowering costs and environmental impact, while also

potentially achieving better performance by leveraging hidden synergies in the data.

1.1 Problem Statement

Recorded Future has over 40 disjoint event classifiers in production. The current system is highly flexible, as any number of new event classifiers can be added to production without affecting the others. The problems aimed to be solved are:

1. Reduce these models into a single multitasking classifier, and evaluate if it performs better than the disjoint classifiers. Specifically, we will look at whether pooling the data will improve performance, which is the basis of our first research question: *Does a joint model draw benefit from data containing all event types?*

2. Given that a joint model performs on par or better than the disjoint classifiers, it should be possible to add new event classes without hampering the performance of preexisting classes for it to be considered a viable contender. This will form the basis for our second research question: *Is such a model a continual learner?*

1.2 Limitations

The main limitation in this project was time. Due to the time frame, a number of deeper investigations into the hidden potential of pooled event types were scrapped. Similarly, more rehearsal dataset configurations could have been explored for research question 2 if more time were available.

1.3 Ethics Statement

When working with machine learning, there is always a risk of the model finding unintended patterns in data. In the field of threat intelligence, a biased model can produce faulty intel, which in turn may have grave consequences in the real world. As we in this project will rely exclusively on data curated by Recorded Future, we will also adhere to the practices and guidelines already in place to ensure our models are as unbiased as possible.

1.4 Contributions

The main contributions made by this study are:

1. Evidence supports the benefits of training an event extraction model on a pooled dataset, as it improved performance in both the event detection tasks and the role tagging tasks.

2. Support exists for the theory that fine-tuned, encoder-based language models are continual learners, with strong evidence observed in sentence-level tasks

(such as event detection) and weaker evidence in token-level tasks (such as role tagging).

2

Theory

In this chapter we introduce the theoretical context required for the concepts mentioned in subsequent parts of the thesis. These concepts are the transformer architecture, the language models BERT, RoBERTa and DistilBERT, the concepts of continual learning and catastrophic forgetting, event extraction with subfields event detection and role tagging, and finally the performance metrics used for classification tasks.

2.1 Transformers

The Transformer is a neural network architecture created for sequence-to-sequence tasks, such as translation [1]. It quickly became the default architecture for language models, with models like BERT [2] and its spinoffs [3] [4], the GPT family [5] and LLaMA [6] being based on transformers.

Prior to the Transformer, leading sequence-to-sequence models typically employed an encoder to generate representation vectors of an input sequence. These representations were then fed into a decoder, which utilized them to create an output sequence based on the specified task. The encoder and decoder were interconnected through an attention mechanism, managing dependencies between the input and output. Typically, both the encoder and decoder were constructed using recurrent neural networks (RNNs) or convolutional neural networks (CNNs). The Transformer also adopts an encoder-decoder structure, but with stacks of attention layers and point-wise, fully connected layers instead of RNNs.

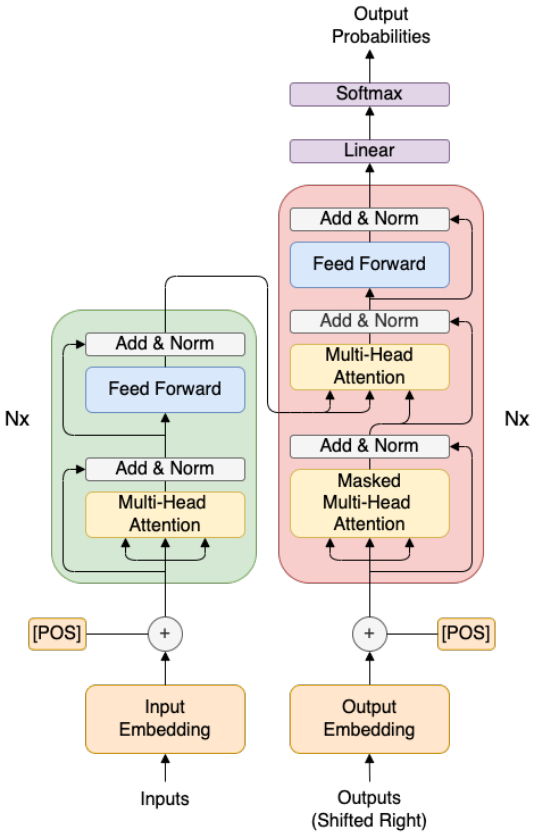

Figure 1: The transformer-architecture, with the encoder in to the left, and the decoder to the right.

2.1.1 Attention

Attention functions are used by a model to identify which parts of the input data it should focus on. The attention function maps a query and a set of key-value pairs to an output, all of which are vectors.

Scaled Dot-Product Attention

The attention variant used in the transformer is called Scaled Dot-Product Attention, and is computed with formula 2.1.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

Where Q, K and V are matrices containing query, key and value vectors respectively. The query and key vectors are of dimension d_k and the value vectors of d_v.

The dot-products between the queries and keys are computed, and then scaled down with $\frac{1}{\sqrt{d_k}}$ to counteract large dot products, which will result in extremely small gradients when put into the softmax function. The softmax function returns the weights for the values.

The transformer uses multiple attention heads in parallel, which all looks at different a different subspace of the queries, keys and values. This was shown to produce more diverse representations than just using a single head. The multihead attention is computed using formula 2.2:

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \\ \text{where} \quad \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.2}$$

with the parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$. In the paper, $h = 8$ parallel heads were used, with $d_k = d_{\text{model}}/h = 64$.

Self-attention

In the encoder, the attention mechanisms are used for self-attention. This means that the queries and keys and values are the same vectors. The attention mechanism will thus compute how important every other word in the input are to each other.

Encoder-Decoder Attention

The decoder employs "encoder-decoder attention" layers. The key-value pairs come from the output of the encoder, and the queries come from the previous decoder layer. This allows the decoder to attend to all positions of the input.

2.1.2 Encoder

The purpose of the encoder is to transform the input into information-rich vectors, called feature vectors or feature embeddings. In the Transformer, the encoder is comprised of a stack of $N = 6$ identical layers, each consisting of 2 sub-layers. The first sub-layer is a multi-headed self-attention mechanism, and its output is normalized before being fed into the second sub-layer. This subsequent sub-layer is a position-wise fully connected feed-forward network, and its output is also normalized. The output of each layer feeds into the subsequent one, continuing until the final layer, which, in turn, feeds into each layer of the decoder. The encoder can also be used independently from the rest of the Transformer. The language model BERT (section 2.2) is based solely on the encoder and performs tasks by feeding the feature embeddings to a task-specific output layer.

2.1.3 Decoder

Similar to the encoder, the Transformer decoder comprises $N = 6$ identical layers. In contrast to the encoder's two sub-layers, the decoder introduces a third sub-layer that conducts multi-head attention over the encoder's output. The multi-head self-attention sub-layer has been modified to restrict the positions it can attend to, allowing it to focus solely on preceding ones. This modification ensures that the output depends only on previously known predictions. Just as the encoder can

be used without the decoder, the decoder can be used without the encoder. The language model GPT [5] uses a stack of transformer decoders to great effect.

2.2 BERT

Bidirectional Representations from Transformers, or BERT, is a pre-trained transformer-based language model introduced in 2019 by Devlin, Chang, Lee, *et al.* [2]. What sets BERT apart from other pre-trained language models is the context captured in its language representations. BERT was the first model to generate representations with deep bidirectional context, a property allowing each word's representation to encompass information from all other words in the text. These bidirectional representations are thus more suitable for tasks where the entire context of the text is critical, such as question-answering tasks.

BERT was trained in two phases: a pre-training phase, during which the model learns to construct bidirectional embeddings, and a fine-tuning phase, where the model's parameters are tweaked to accommodate a downstream task that uses the embeddings.

The final BERT model was tested on four different benchmark suites (GLUE [7], SQuAD v1.1 [8], SQuAD v2.0, SWAG [9]) to evaluate its capabilities on tasks such as general language understanding and question answering. It advanced the state of the art in eleven NLP tasks, thereby reinforcing the viability of pre-trained models for NLP applications.

2.2.1 Architecture

BERT is based on the transformer encoder, which is identical to the one introduced in presented in section 2.1.2. BERT was configured in multiple sizes, but the two primarily presented were $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$. $\text{BERT}_{\text{BASE}}$, with its 12 transformer blocks[1], 768 neurons per layer, 12 attention heads, and 110M total parameters, is the exact same size as OpenAI's Decoder-based GPT model [5]. This choice was deliberate in order to compare a model based on an encoder to one based on a decoder.

BERT's input representation allows for two types of inputs: sentence and sentence-pairs. A sentence is a sequence of tokens, and a sentence pair is thus two token sequences concatenated into a single sequence. The tokens used comes from the WordPiece vocabulary [10], containing 30K tokens. Each sequence begins with a special [CLS] token, and sentences are separated using a special [SEP] token. The final representation of each token is composed of the corresponding token and its segment- and position embeddings. BERT's training was split into two phases: pre-training and fine-tuning. The pre-training is aimed towards the encoder, and consists of two tasks.

[1] A transformer block is the same as the attention/feed forward-layer pair encountered in the encoder, 1

2.2.2 Pre-Training

The first pre-training task was Masked LM. The purpose of this task was to learn a model capable of producing deep bidirectional language representations. 15 % of the input tokens were masked at random, and the model was then tasked to predict what was behind the mask. In 80% of the cases, the mask was a [MASK]-token. In 10% of the cases, the mask was another token chosen at random. In the final 10% of cases, the token selected for masking remained unmasked. Having this split mitigated a mismatch between pre-training and fine-tuning, as the [MASK]-token does not appear in the fine-tuning stage. The second pre-training task was Next Sentence Prediction (NSP). This tasks trains the model to understand relationships between sentences. The model was presented with two sentences, A and B, and tasked with predicting whether sentence B follows sentence A. During training, this was true for 50% of the cases.

Two datasets were used for pre-training. The first one was BookCorpus [11], a collection of indie books containing 800 million words. The second dataset was created by extracting text passages from English Wikipedia, resulting in a dataset of 2.5 billion words. The total dataset on which BERT was trained contained 3.3 billion words, or around 16 GB of data.

2.2.3 Fine-Tuning

The fine-tuning stage is aimed at training the model towards a specific task. This is done by adding an output layer to the encoder, and providing the model with inputs and target outputs. BERT was fine-tuned to 11 different NLP tasks, evaluating performance on tasks like entailment classification, sentiment classification and identifying plausible sentence continuations, among others.

2.3 BERT Variants

After the introduction of BERT, a multitude of variants were created for different purposes. Some were trained on other languages, and others for specific domains. Here, we will present the two variants that will be investigated for the thesis, as they are the models used by Recorded Future for event detection.

2.3.1 DistilBERT

DistilBERT [4] is a BERT-inspired pre-trained language model with a focus on model size. Research at the time indicated that an increase in parameters for pre-trained language models led to improved performance. However, models were growing in size, raising concerns about energy consumption during training and the potential hardware requirements that might hinder widespread adoption.

The proposed solution was DistilBERT. Like BERT, DistilBERT is Transformer-based but has its number of layers reduced by a factor of 2 (N=6). DistilBERT is trained to mimic BERT using a method called Knowledge Distillation [12]. In this

process, the teacher (BERT) makes soft predictions on a dataset, and the student (DistilBERT) is tasked with matching these predictions.

The final DistilBERT model underwent evaluation on tasks similar to those of BERT, specifically GLUE and SQuAD. The results demonstrated that DistilBERT retained up to 97% of BERT's performance. In addition to classification performance, DistilBERT was compared to BERT and ELMo for inference speed. All three models had to make predictions running on an iPhone 7 to showcase on-device performance. DistilBERT made predictions 46% faster than ELMo and 61% faster than BERT.

In summary, DistilBERT was 40% smaller than BERT but managed to retain 97% of language understanding while making predictions 60% faster.

2.3.2 RoBERTa

RoBERTa [3] was created with the thesis that the original BERT model was undertrained. First, the team re-implemented $\text{BERT}_{\text{BASE}}$ according to the specifications in the paper. Second, the size of the training data was increased from 16GB to 160GB. In addition to the BookCorpus and Wikipedia datasets used by when training the original BERT, three additional datasets containing news articles and web based texts were used. Third, the masked language modeling (MLM) objective was modified. When training the original BERT, the mask for each input sequence was computed once during the data pre-processing and, thus, remained the same for that sequence across all training epochs. For RoBERTa, the data was duplicated 10 times before the mask was computed, resulting in 10 different masks for each input sequence, adding variation and making the training data more reusable. The Next Sentence Prediction task was deemed unnecessary after some experiments, and was removed from the training. Fourth, the batch size was increased and epoch length decreased proportionally. Finally, the original token library was switched out in favor of a library containing 50K tokens, as opposed to the 30K in the WordPiece library.

RoBERTa was evaluated on the GLUE, SQuAD, and RACE [13] benchmarks for language understanding, question-answering capabilities, and next sentence identification, respectively. It outperformed $\text{BERT}_{\text{LARGE}}$ in all tasks, setting state-of-the-art (SotA) scores in 4 out of 9 GLUE tasks and achieving results that matched the state of the art on SQuAD and RACE.

2.4 Event Extraction

Event extraction is an NLP research area that focuses on detecting and organizing event information from text. An event can be described as something that happens, involving people or a place, or a change of state. Event extraction consists of many different sub-tasks, such as event detection, trigger extraction, and role tagging.

Despite being a decades-old task, event extraction is still a troublesome challenge, as it requires the system to have a semantic understanding of the text. This has been

alleviated by the introduction of transformer-based language models such as BERT, as the deep context found in the embedding helps with the task. [14]

2.4.1 Event Detection

Event detection involves a classification task where the model determines whether the input text describes a given event. Since the whole sentence is taken into consideration, event detection is considered a sentence-level task. In language models such as the BERT family, the input sentence is tokenized and then encoded into feature vectors. The feature vector corresponding to the start-of-sequence token, [CLS], contains enough contextual information about the sentence to be sufficient for the classification layer (event detection head) to make the classification.

2.4.2 Role Tagging

Role tagging (sometimes called role labeling) is a relaxed variant of semantic role labeling (SRL) [15]. In SRL, the task is to identify the semantic role each word in a sentence belongs to, dividing the sentence into non-overlapping phrases. In role tagging, the task is to instead identify the pre-defined event role to which each word belongs. One method of accomplishing this is BIO-tagging (also known as IOB2-tagging). Roles are represented by three types of tags: Beginning, Inside, and Outside. For each role X, a B-X and an I-X tag are created. The first word of a role belonging to type X is tagged with B-X, and any subsequent words of the role are tagged with I-X. Words that do not fit into any roles are tagged with O for outside.

By using a method like BIO-tagging, role tagging effectively becomes a token-level classification task, where each tag is a class. It is thus suitable to use a model like BERT for role tagging.

```
 Event Type: Disaster
      Roles: DisasterType
             Location
BIO-classes: B-Dis (Beginning of DisasterType)
             I-Dis (Inside of DisasterType)
             B-Loc (Beginning of Location)
             I-Loc (Inside of Location)
             O     (Outside)

O    B-Dis        O     O        O     B-Loc   I-Loc  I-Loc  O
An   earthquake   has   struck   the   South   of     France .
```

Figure 2: A BIO-tagged string, of type Disaster.

2.5 Continual Learning

Continual Learning is a subfield of NLP that focuses on improving existing models by extending their capabilities, such as adding new classes to a classification model. One major problem in continual learning is the loss of performance on old tasks when training the model to perform new tasks.

2.5.1 Catastrophic Forgetting

A major problem with continual learning in neural networks is the phenomenon *Catastrophic Forgetting* [16]. When a network trained on a specific task is retrained on a different task, the network becomes unable to perform the original task. As the networks learns a new task, it overrides the learned weights associated with the previous task. However, the forgetting only occurs when training is sequential, i.e. one task after another. If the network is trained on both tasks in parallel, the network weights will be tuned for both tasks, thus preventing the forgetting. [17]

2.5.2 Methods for Continual Learning

A common method for having a model learn new tasks without forgetting the old is using some sort of replay-based strategy, where a subset of the original training data is mixed into the training data of the new task, effectively forcing the model to rehearse the old tasks.

Rehearsal datasets for instruction based language models

In Scialom, Chakrabarty, and Muresan [18], the transformer-based text-to-text model T0 was used to investigate the continual learning capabilities of fine-tuned language models. The model was initially trained on 50 different tasks using 50 datasets and then evaluated on 4 zero-shot tasks, which were distinctly different from its training tasks. Subsequently, it was retrained on 12 new datasets, incorporating varying amounts of the original 50 datasets as rehearsal data to prevent catastrophic forgetting. The model was evaluated on 8 new tasks. It was demonstrated that including as little as 1% of the original training data was sufficient for the model to achieve good scores on the new tasks while maintaining its performance on the old tasks. Notably, the sizes of both the old and new datasets were standardized to approximately 100,000 entries each, meaning that 1% of a dataset equated to 1,000 entries.

Continual event detection using prototypes

In Cao, Chen, Zhao, *et al.* [19], continual learning on a BERT-based event detection model was investigated. Two challenges were identified when teaching a model a new event type using replay-based methods; *semantic ambiguity* and *class imbalance*.

The issue of semantic ambiguity arises when trigger words found in an event type has multiple semantic meanings, and thus can trigger multiple different events. Including

these in the rehearsal data can thus cause confusion in the model, as it can find it difficult to determine which event the specific entry describes, leading to poorer performance and generalization.

When employing a replay-based method, the dataset containing the new event type is often larger than the replay dataset containing samples from old event types. This imbalance can bias the training toward the new event types, resulting in the model catastrophically forgetting the old event types.

The proposed solution was a knowledge consolidation network (KCN), used to preserve knowledge. It featured two components:

- Prototype Enhanced Retrospection: a module which computes a prototype of each event type using the training data. The prototype can be regarded as the essential representation of the event type. Using this prototype, the module then picks out the N most representative examples of the event type, and stores them for rehearsal purposes. As only the most representative examples are stored, ambiguous cases are disregarded in favor of clearer instances, thereby resolving the issue of semantic ambiguity.
- Hierarchical distillation: a distillation module, which makes the current model mimic the predictions of the old model, ensuring good performance on old classes. This effectively solves the problem of class imbalance.

Experimental results demonstrated that using a KCN resulted in outperforming state-of-the-art models on the ACE and TAC KBP benchmarks.

In this project

While our problem was more similar to Cao, Chen, Zhao, *et al.* [19], we will attempt to solve it using the method described by Scialom, Chakrabarty, and Muresan [18], as adding new modules to the current infrastructure was deemed to be out of scope, and a purely data-based solution was more desirable.

2.6 Performance Metrics

The predictive performance of classification models is measured using the following terms: true positive (TP) for correct positive classifications, false positive (FP) for incorrect positive classifications, true negatives (TN) for correct negative classifications, and false negatives (FN) for incorrect negative classifications. On their own, these terms don't say much about a model, but together they can inform us about the model's behavior, e.g., whether it is overly cautious or confident in a specific class.

2.6.1 Accuracy

Accuracy is a measure of the model's general ability to correctly predict a class. High accuracy indicates that the model can predict a class with few-to-no errors. However,

lower accuracy does not provide information about whether the model struggles with false positives or false negatives. For such insights, other measurements are required.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{2.3}$$

2.6.2 Precision

Precision is a measure of the model's ability to correctly predict the positive class. High precision indicates that the model made few to no false-positive predictions.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.4}$$

2.6.3 Recall

Recall is a measure of how well the model identifies all occurrences of a class. A high recall indicates that the model made few to no false-negative predictions.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.5}$$

2.6.4 F1

F1 is an average measurement of a model's predictive ability for a class. It is computed as the harmonic mean of precision and recall.

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.6}$$

Macro-F1

Macro-F1 is a helpful measurement for multi-task models. It is the arithmetic mean of all the per-class F1 scores.

$$\text{Macro-F1} = \frac{\sum_{i=1}^{n} \text{F1}_i}{n} \tag{2.7}$$

3

Methods

This study is split into two parts, to investigate the two research questions. The first part focuses on training multitasking models in a multi-class environment. The second part focuses on continual learning, and adding new classes to an already trained classification model.

This chapter details the methodologies used for these parts. We describe the datasets used (3.1), the preprocessing (3.2), the model configurations (3.3) and their training (3.4), and finally how the models were evaluated (3.5).

3.1 Data

For this study we used twelve different data sets; training and evaluation sets for each of the six event types. The data has been collected from various sources, such as news feeds and social media feeds. It should be noted that the training and evaluation data sets have been curated using active learning methods. In production, the models will see thousands, even millions of negative examples for every positive case.

Between them, the training data sets contain 82,746 data points, of which 28,866 are in English. The remaining data points are a mix of Arabic, French, German, Italian, Persian, Portuguese, Russian, Simplified Chinese, Spanish and Ukrainian.

Event Type	**Roles**
Cyber Attack	Attacker, Method, Target
Disaster	DisasterType, Location
Disease Outbreak	Disease, Location
Person Threat	TargetObject, TargetPointer
PSW	Location, ThreatType, RecommendedAction
Terror Incident	Location

Table 3.1: The event types used for this study, and their roles.

All entries adhere to the same format, regardless of event type. An entry contains an untagged and a tagged text fragment. The fragment is typically a sentence. The tagged fragment has been annotated with labels for the entities present in the fragment, indicating what type of entity it is, and what its ID is in the RF-database. The entry also contains which event type class the fragment belongs to, as well as role

labels. Three of the event types, Cyber Attack, Disaster and Public Safety Warning (PSW) have more than one non-negative class. We did not take these into account when evaluating the performance of the models.

Event Type	**Count**
Cyber Attack	**2,245**
- CyberAttack	1,162
- CyberCapability	171
- Negative	912
Disaster	**9,496**
- Disaster	4,124
- Traffic	276
- Negative	5,095
Disease Outbreak	**5,000**
- DiseaseOutbreak	1,907
- Negative	3,093
Person Threat	**2,241**
- PersonThreatPositive	921
- Negative	1,320
Public Safety Warning	**4,922**
- PSW	413
- AllClear	138
- Negative	4,371
Terror Incident	**4,962**
- TerrorIncident	4,124
- Negative	838
Total	**28,866**

Table 3.2: The event types used in the project (**bold**), and the classes of each event type. While all classes are present, we will not consider secondary classes such as Cyber Capability when evaluating the models.

3.1.1 Pooled Datasets

The experiments required multiple event types in a single dataset. As the base datasets (described in table 3.2) were homogeneous with regards to event types, they had to be pooled together. Constructing these pooled sets was split into two tasks; 1) merging schemas, and 2) merging data files.

1. **Merging schemas**: The schemas are integral for creating the correct model heads. Therefore, all events need to be present in the field corresponding to Event Heads. Each event head is paired to a Role Head, which classifies the

roles associated with the event. The roles are specified in another schema field, and are grouped by event type.

2. **Merging data files**: The data files used are JSONL-files. One line corresponds to one data entry. Therefore merging the data files was just a matter of appending one file to another.

In certain experiments, a dedicated head was created for the role "Location". The purpose of this was to investigate the possible gains of having a head trained on all occurrences of Location, regardless of event type. This was achieved by disassociating the role from the event type in the schema, effectively breaking it out into its own role. This is demonstrated in figure 2. After breaking out Location, the data was reprocessed, relabeling the old Location role as the new one.

```
merged_schema
├─ event_types
│  ├─ CyberAttack
│  ├─ Disaster
│  └─ DiseaseOutbreak
└─ role_labels
   ├─ CyberAttack.Attacker CyberAttack.Target CyberAttack.Method
   ├─ Disaster.DisasterType Disaster.Location
   └─ DiseaseOutbreak.DiseaseName DiseaseOutbreak.Location
```

Figure 1: The resulting schema after merging schemas for CyberAttack, Disaster and DiseaseOutbreak

```
merged_schema_location
├─ event_types
│  ├─ Disaster
│  └─ DiseaseOutbreak
└─ role_labels
   ├─ Disaster.DisasterType
   ├─ DiseaseOutbreak.DiseaseName
   └─ Location.Location
```

Figure 2: A schema where the role Location has been disassociated from the Event Types. This will result in a dedicated role head for Location, in addition to the role heads for Disaster and DiseaseOutbreak.

3.1.2 Rehearsal Datasets

To mitigate the model forgetting old classes when learning new, we created rehearsal data sets. These data sets contain a fraction of the original data used to train the model. In Scialom, Chakrabarty, and Muresan [18], it was shown that 1% rehearsal data was sufficient to keep the model from forgetting. However, as our substantially smaller than the assumed size of 100K data points in [18], we also created rehearsal sets of sizes 5% and 10%. When creating these rehearsal sets, we made sure to keep the ratio approximately the same between classes for the different event types. If 1% of any class count came out to less than 1 entry, this number was inflated to 1, to make sure each class was represented.

Event type	Event classes	Data points			
		1%	5%	10%	100%
CyberAttack	Cyber Attack	11	58	116	1,178
	CyberCapability	1	8	17	171
	Negative	9	45	91	912
Disaster	Disaster	41	206	412	4,124
	Traffic	2	13	27	276
	Negative	50	245	509	5,095
PSW	PSW	4	20	41	413
	AllClear	1	6	13	138
	Negative	43	218	437	4,371
TerrorIncident	TerrorIncident	41	206	412	4,124
	Negative	8	41	83	838

Table 3.3: The number of entries from each class in the different rehearsal datasets, and the original count in the last column.

3.1.3 Training, test and evaluation sets

The dataset used in training was split into a training and a test. A 90-10 split was employed, with 90% of the data going to the training set, and the remaining 10% going to the test set. This split was applied to the data set after all event types had been pooled together, and was thus applied to the data set as a whole, rather than to each individual event type data set.

Event Type	Count
Cyber Attack	738
Disaster	1,491
Disease Outbreak	1,409
Person Threat	750
PSW	796
Terror Incident	248

Table 3.4: The sizes of the different English evaluation data sets used.

For each event type, there exists evaluation data sets, listed in table 3.4. While training data sets are multi-lingual, the evaluation sets are divided by language. For this study, we evaluated only English datasets to narrow the scope.

3.2 Preprocessing

Before the actual training step, the training data is processed in two steps, normalization and tokenization.

3.2.1 Normalization

The data is normalized to reduce the risk of the model overfitting to entities in the data. As mentioned in 3.1, entities in the text fragments are tagged, making it easy to replace them. The entities are hidden with masks specific to the type of the entity, as seen in figure 3.

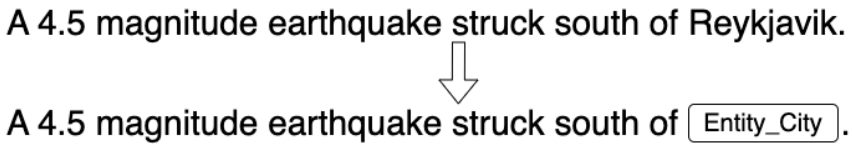

Figure 3: A text fragment containing a City-entity, before and after normalization.

3.2.2 Tokenization

To account for the masks, a set of custom tokens are added to the tokenizer. The tokenizers used are the built-in tokenizer of each model.

3.3 Model

In this study, we used XLM-RoBERTa$_{\text{LARGE}}$ and DistilBERT-base-uncased as our base models. XLM-RoBERTa$_{\text{LARGE}}$ is a RoBERTa variant which has been pre-trained on a corpus featuring multiple languages, instead of only English. DistilBERT-base-uncased is the same model as mentioned in 2.3.1. The models were chosen for this study as both are already used by Recorded Future in some capacity. For simplicity, we will refer to them as RoBERTa and DistilBERT.

3.3.1 Baselines

The current production setup at Recorded Future is one model per event-type. The baseline models mimics this setup. For each event-type, models are fine-tuned, evaluated and averaged. This procedure is performed on both RoBERTa and DistilBERT. Though the training data sets contain multiple languages, only English evaluation data sets are used. We trained five instances of each event type on each base model, in order to get an average score.

3.3.2 Multitask Model

The multitasking model was configured in two variants. The first variant consisted of a single body onto which we attached one event head and one role head for each event type. As this model has multiple heads, it is referred to has Hydra1.

The second multitask model, referred to as Hydra2, was also configured with one event head and one role head for each event type. In addition to these heads, an extra role head was added, dedicated to the role Location. As a result, the other role heads no longer needed to identify the role Location.

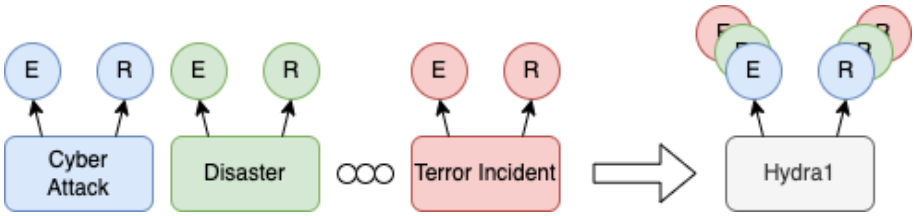

Figure 4: The Hydra1 model, with event and role heads for each event type on a unified body.

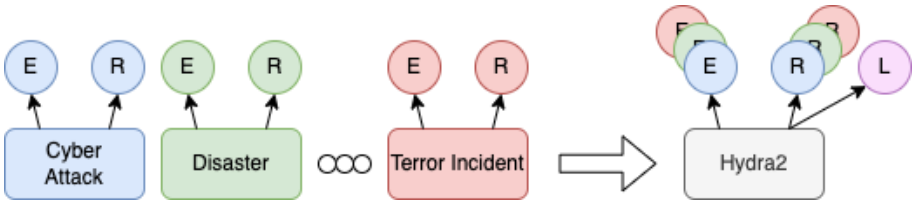

Figure 5: The Hydra2 model, where the role Location has been given a dedicated role head.

Like with the baselines, we trained five instances of Hydra1 and of Hydra2 for each model body, and drew averages.

3.3.3 Continual Learning Model

For the second half of the study, we investigated how well the best performing variants retained knowledge after being retrained on a new event type. Since the focus was on behavior rather than actual performance, only one instance of each model was trained, but with different dataset configurations.

A new model was trained on a subset of event types: Cyber Attack, Disaster, Public Safety Warning, and Terror Incident. This smaller Hydra-model served as baseline for the subsequent test. As with the previous experiments, the new Hydra was configured on both a DistilBERT body and a RoBERTa body.

After the initial training, the model was retrained on the two left out event types; Disease Outbreak and Person Threat. Disease Outbreak was chosen as it contains the role Location, and is one of the larger datasets. Person Threat was chosen for the opposite reasons; it lacks the role Location, and is one of the smaller datasets. These differences can hopefully be used to draw conclusions as to how data similarities and dataset sizes affect the model when retraining. The experiment plan is shown in figure 6.

The models were evaluated based on their scores relative to the benchmarks set by the baselines.

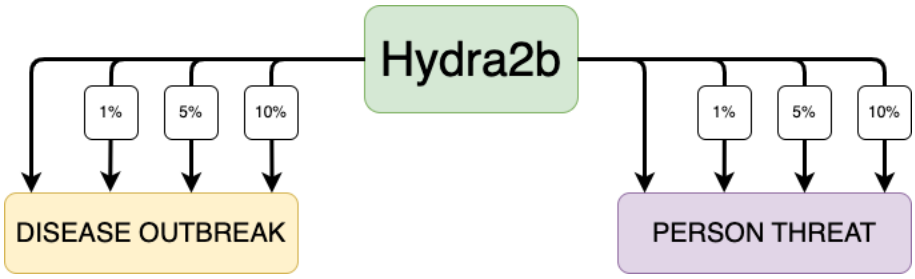

Figure 6: The plan for retraining a smaller Hydra2 using different sizes of rehearsal datasets.

3.3.4 Heads

Each head attached to a model is an output layer, connected to the final hidden layer of the model's body. The size of each head is proportional to the number of classes it has to identify. The event detection heads thus have either 2 or 3 neurons, depending on whether the head only has to identify positive and negative, or if there is an additional class, such as Cyber Capability.

The role tagging heads have 2 neurons for each role associated with the event type: 1 for the B-class and 1 for the I-class (see section 2.4.2), as well as an additional neuron for the O-class. In the Hydra2 model, the role Location has a dedicated role head and has been disassociated from the event type-specific role heads. As a consequence, the other role tagging heads have had their size reduced by 2 neurons. The exception to this is the role head associated with the Terror Incident event, which has been removed completely, as Location was the only role in that event type. As a result, the Hydra2 model is slightly smaller, with 6 neurons fewer than Hydra1.

3.4 Training

This section describes the hardware infrastructure and techniques used to fine-tune the different BERT models.

3.4.1 Hardware

Recorded Future uses Amazon Web Services G5-instances[1] for training ML-models. We were given access to this system to use for all of our training needs. The G5 system utilizes a NVIDIA A10G Tensor Core GPU, with 24 GB of graphics memory, making it more than sufficient for our needs.

3.4.2 Fine Tuning

The RoBERTa and DistilBERT models were fine-tuned in a multi-task setting, by attaching multiple pairs of output layers to the models' bodies. The pairs consists of an event head, for event detection, and a role head, for role tagging. The model was

[1] https://aws.amazon.com/ec2/instance-types/g5/

then tuned with the goal of maximizing its macro-F1 score. Macro-F1 was chosen as target metric as it is dependent on the classification performance of all heads.

The models were trained for a maximum of 15 epochs, with an early stop patience of 3 epochs. The learning rate was fixed to 1e-5. As these parameters are the default ones used at Recorded Future, we did not tweak them. All hyperparameters are displayed in table 3.5.

Hyperparameter	Values
Max Epochs	15
Early Stop Patience	3
Min Epochs	3
Optimizer	adamW
Learning Rate	0.00001
Batch Size	16
Target Metric	Macro-F1

Table 3.5: The hyperparameters used for training all models.

3.5 Evaluation

The evaluation of the models was conducted using the same methods already in use at Recorded Future. Given that event detection and role tagging are both classification tasks, our training approach will predominantly rely on precision and recall as the primary evaluation metrics.

All event types have their own evaluation datasets, shown in table 3.4. Despite having a dedicated Location head, the Hydra2 model will be evaluated on an event type-by-event type-basis, to better gauge how the model performs on each event types evaluation dataset.

4

Results

4.1 Comparison of multi-headed event extraction models

Our study compared four variants of multi-tasking event detection models; Hydra1 and Hydra2 on a RoBERTa base, and Hydra1 and Hydra2 on a DistilBERT base, on a set of 6 different event types.

The baseline models were also RoBERTa and DistilBERT based. To mimic the current production setup, each baseline model was only trained on extracting and tagging roles for a single event type. In contrast Hydra1 and Hydra2 were trained on all event types. The event types and their roles are as described in table 3.1 .

4.1.1 Event Detection

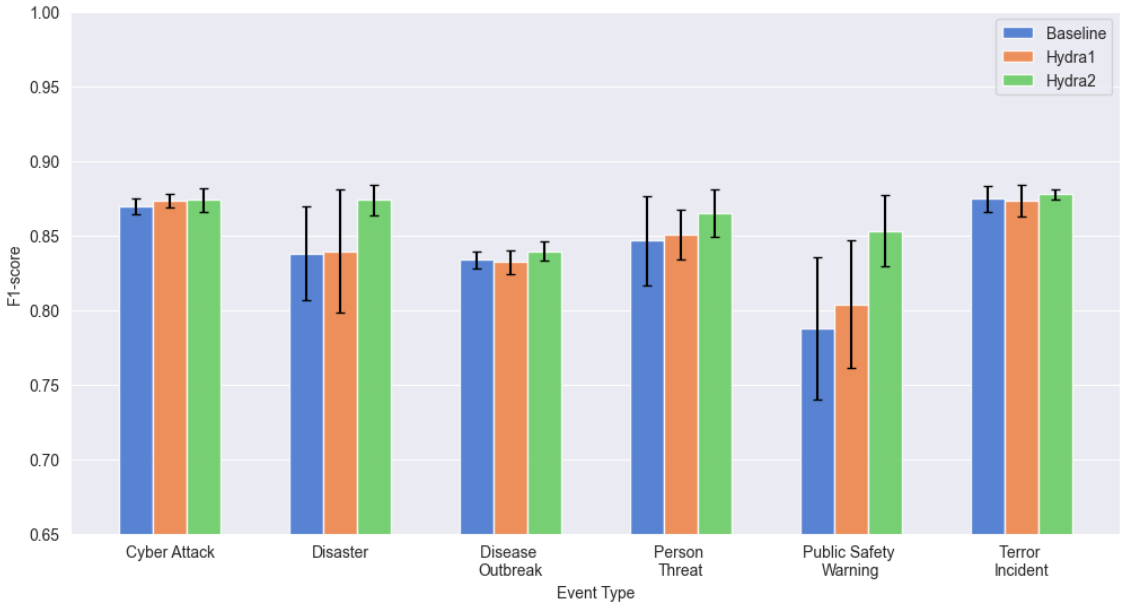

Figure 1: F1-scores for each event type's primary class on the Roberta based models.

The best performing variant was Hydra2, which was on par with or outperformed the baseline across all event types except one, on both RoBERTa (1) and DistilBERT (2)

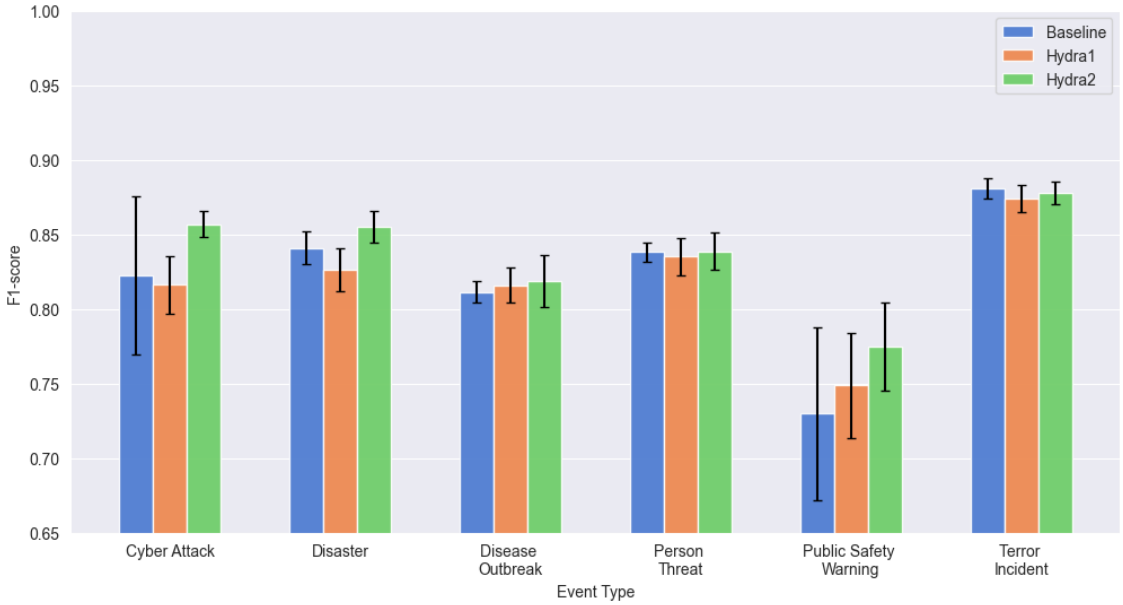

Figure 2: F1-scores for each event type's primary class on the DistilBERT based models.

bodies. The exception was Terror Incident on the DistilBERT body, where Hydra2 scored just under the baseline with a similar standard deviance. It should be noted that the standard deviances of the Hydra2 models are always similar to, or noticeably smaller than the ones seen in the baselines and Hydra1 models.

There seemed to be no connection between dataset size (3.2) and performance, except in Public Safety Warning, which was heavily imbalanced, where the negative class outnumbered the positive class by a factor of 10 (4,371 to 413). It was also in this class where the biggest increase in performance was observed; Hydra2 scored approximately 7 points higher than the baseline on both the RoBERTa and DistilBERT bodies.

Of the two variants, it was clear that Hydra2 is a stronger contender than Hydra1 when it comes to event detection. As for DistilBERT vs RoBERTa, the DistilBERT-based model performed well, but the RoBERTa-based Hydra2 was simply better at this task.

4.1.2 Role Tagging

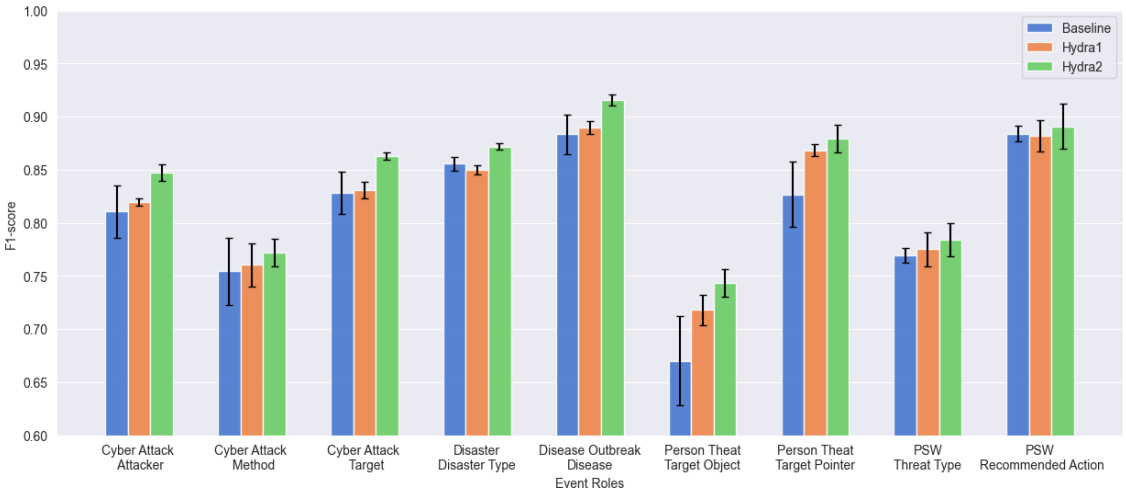

Figure 3: F1-scores for each event type's roles (excluding Location) on the RoBERTa based models.

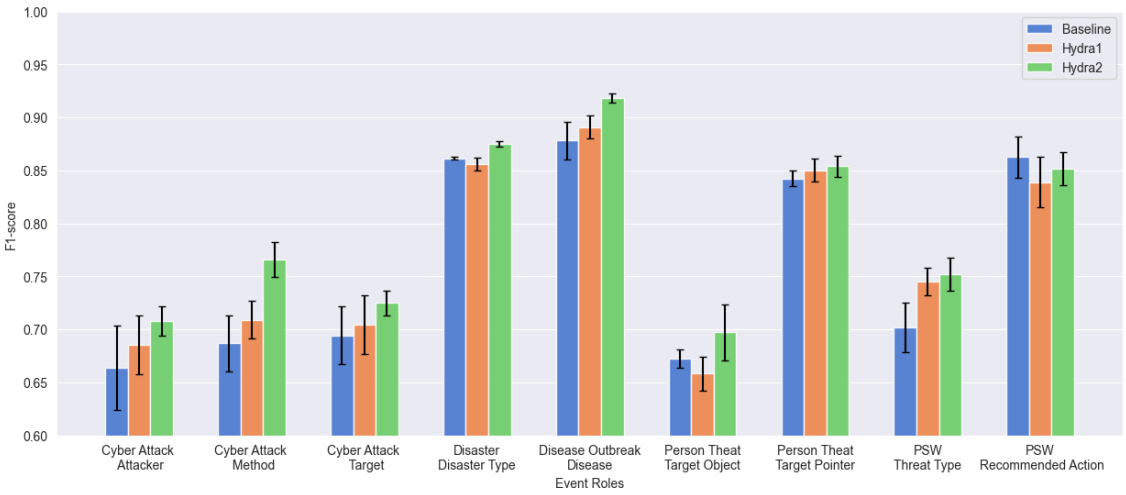

Figure 4: F1-scores for each event type's roles (excluding Location) on the DistilBERT based models.

In the role tagging task, the RoBERTa-based Hydra2-model (3) was once again the bert performer; posting both scores higher than the competitions, and in general with smaller deviations as well. The one exception was PSW's RecommendedAction, where the baseline was both high, and had a small deviation. The Hydra1-model performed well, often outperforming the baselines, but fell short against Hydra2 in every role.

The DistilBERT-based models (figure 4) showed a similar picture, with Hydra2 being the best in all but one role. The role which it performed worse in was the same as for RoBERTa-Hydra2; PSW's RecommendedAction.

While the DistilBERT-models were almost on par with the RoBERTa-models in event detection, this was not the case in role tagging. The DistilBERT-models perform far below the level of the RoBERTa-models when tagging the roles of Cyber Attack. Hydra2 manages to reach the same score on the Method-role, but falls short on Attacker and Target. This is similar to how the baseline and Hydra1-models struggled with Cyber Attack in the event detection task.

In figures 3 and 4, the role Location was left out. As Location is a special role, due to being present in multiple event types, it was evaluated separately to better showcase if directing the role to the Hydra2-models Location head made any impact on performance. The results are displayed in figures 5 (RoBERTa) and 6 (DisilBERT).

All models performed similarly when tagging Location in the Disaster and Disease Outbreak datasets. In the PSW dataset, the Hydra-models scored a few points higher than the baseline models. The truly noticeable difference, however, was on the Terror Incident dataset. RoBERTa Hydra2 scored nearly 10 points over the baseline, and with a fraction of the deviance. DistilBERT Hydra2 performed similarly, beating the baseline hand Hydra1 with a margin.

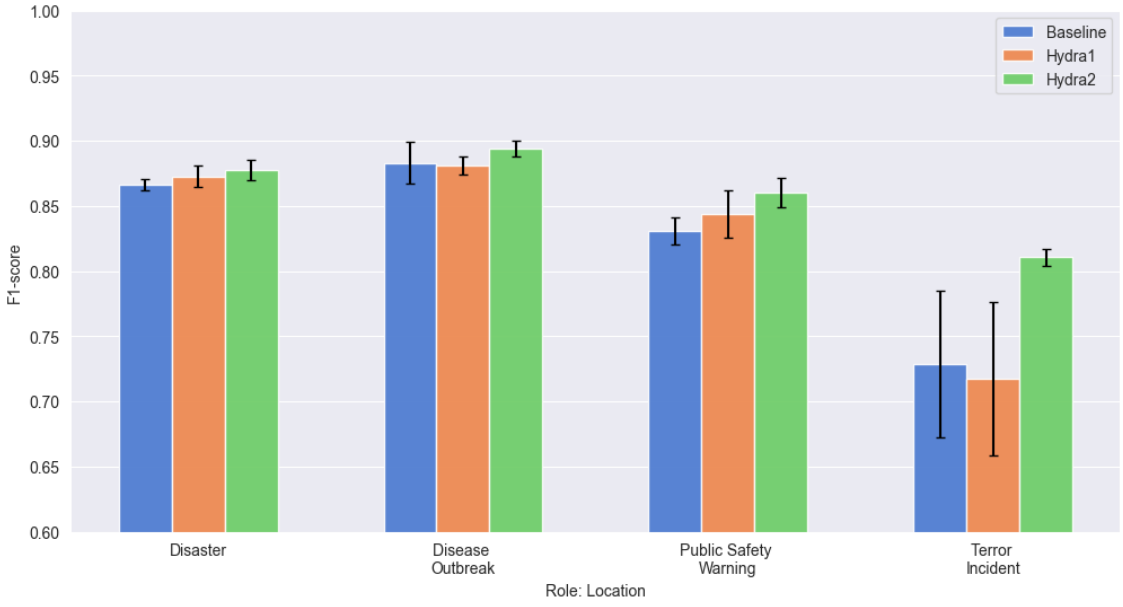

Figure 5: F1-scores for the role Location for the RoBERTa based models.

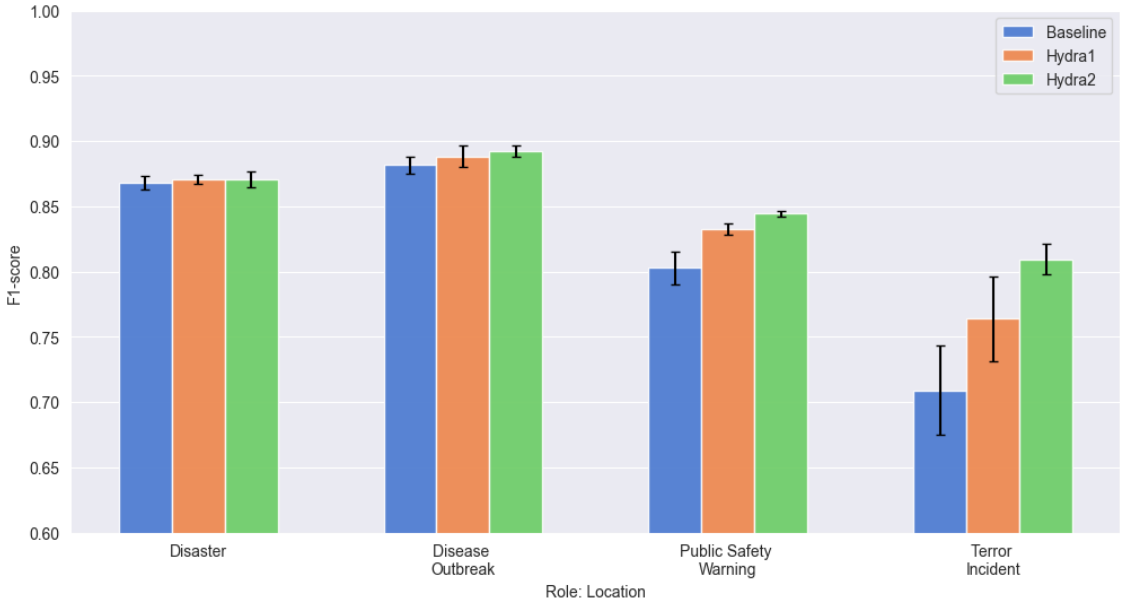

Figure 6: F1-scores for the role Location for the DistilBERT based models.

4.2 As a Continual Learner

In this section, we present the results of the experiments relating to the model learning new event types. We will only present the results of continual learning-experiments performed on Hydra2-models here, as Hydra2 was the better variant in the first round of experiments. The F1-scores for the Hydra1b models are available in Appendix B.

4.2.1 New Event Types

The new event types added to the Hydra2b model were Disease Outbreak and Person Threat. The results of the four RoBERTa instances for each event type, shown in Figures 7 and 8, showed promising performance. For each body-event pair, the average score from the corresponding Hydra2 model was used as a proxy for the best performance that could be expected.

On both the RoBERTa and DistilBERT models, the event detection performance for Disease Outbreak was close to the Hydra2 scores. The RoBERTa instances were within 1% of the Hydra2 score, while the DistilBERT instances were within 3% at worst. However, the event detection performances for Person Threat were less impressive. In the RoBERTa models, a significant drop in performance was observed when retraining with 1% rehearsal data. Nevertheless, the instances with 5% and 10% rehearsal data achieved scores within 2% of the Hydra2 scores, indicating that the earlier drop was an anomaly. As for the DistilBERT models, their scores were distinctly lower than those of Hydra2, being within 8% at worst.

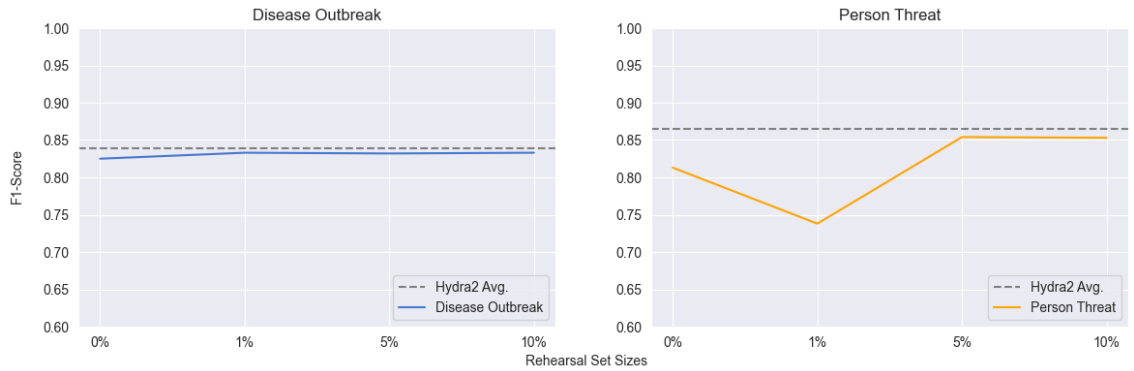

Figure 7: Event Detection scores for the RoBERTa-based Hydra2b-models.

Figure 8: Event Detection scores for the DistilBERT-based Hydra2b-models.

The models' role tagging scores, 9, on Disease Outbreak was close to that of Hydra2, being within 6 at worst for Disease, and within 5% for Location.

On the Person Threat roles, the performance was notably worse, with an 11% deviation from Hydra2 performance on TargetObject at worst. For the TargetPointer role, the worst dip was still within a 7% range of performance and could, just like the dip seen in the Event Detection task, be a one-time anomaly, as the performance with less and more rehearsal data achieved better scores.

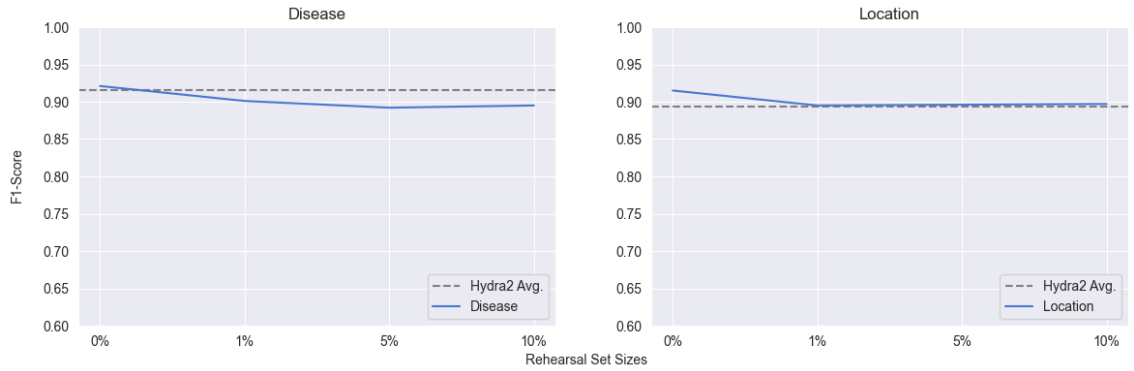

Figure 9: Role Tagging scores for Disease Outbreak on the RoBERTa-based Hydra2b model.

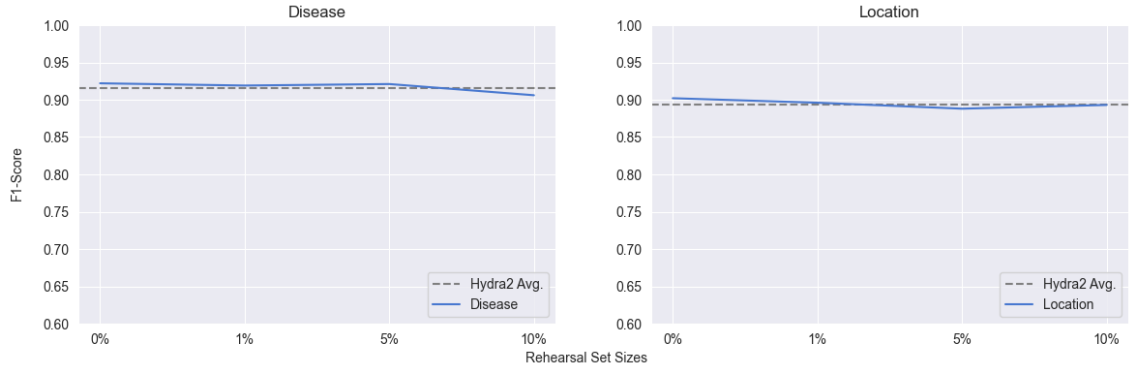

Figure 10: Role Tagging scores for Disease Outbreak on the DistilBERT-based Hydra2b model.

The role tagging performance on both model bodies was almost identical to their respective Hydra2 scores across all instances. The only exception occurred with the Disease role on the RoBERTa body, where the performance dropped to 3% below Hydra2, which was still considered very much acceptable.

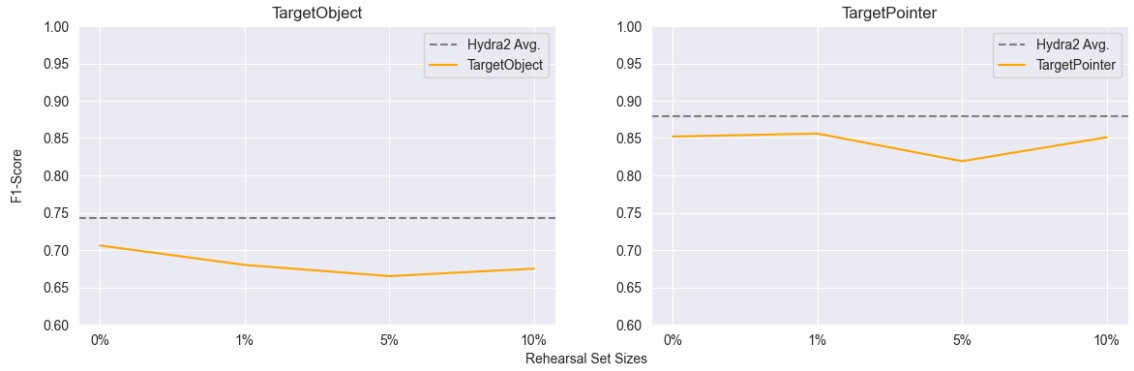

Figure 11: Role Tagging scores for Person Threat on the RoBERTa-based Hydra2b model.

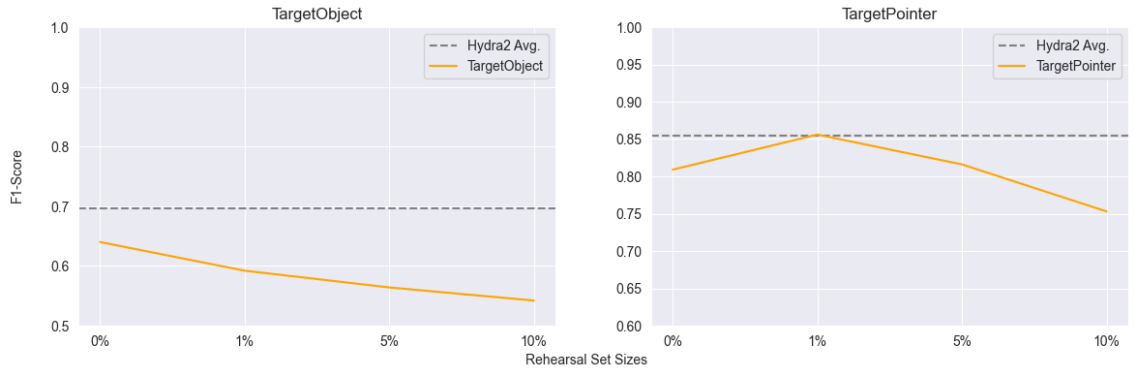

Figure 12: Role Tagging scores for Person Threat on the DistilBERT-based Hydra2b model.

On the Person Threat roles, the role tagging performance was not as promising as that for Disease Outbreak. The scores for TargetObject across all instances were significantly lower than those for Hydra2, as were the scores for TargetPointer on DistilBERT, with only one instance approaching the Hydra2 performance. The exception to these lower performances was TargetPointer on RoBERTa, where the scores were within 1.5% of Hydra2, except for the instance with 5% rehearsal data.

4.2.2 Old Event Types

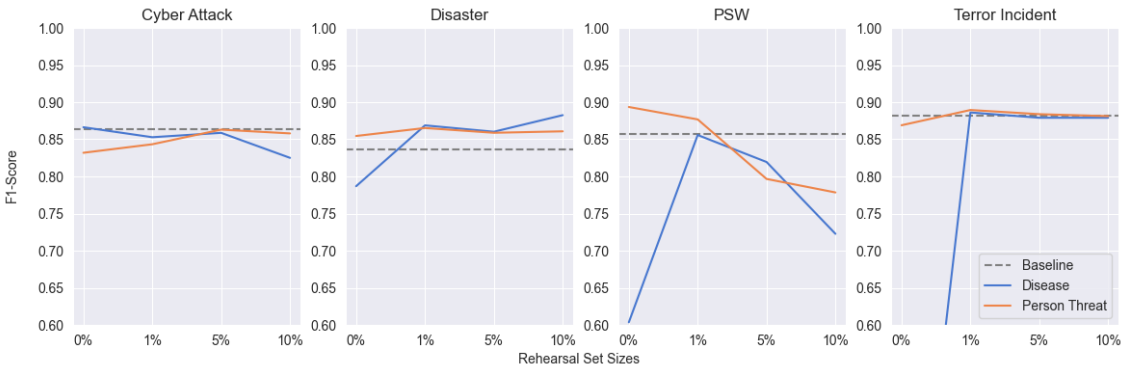

Figure 13: F1-scores for the main classes of Cyber Attack, Disaster, PSW and Terror Incident after retraining on Disease Outbreak (blue) or Person Threat (orange). The model used was a smaller Hydra2 based on RoBERTa.

Presented in figure 13 are the results of retraining on either Disease Outbreak or Person Threat with different rehearsal set sizes.

On the Cyber Attack and Disaster data sets, none of the models showed signs of catastrophic forgetting. On PSW and Terror Incident, the model retrained on Disease Outbreak without rehearsal data scored noticeably lower than the baseline. The Disease Outbreak model trained on 1% data recovered when 1% was used, and remained stable on Terror Incident. For both models, the performance on the PSW data set declined as the rehearsal sets increased in size.

A closer look at the precision and recall scores (Table 4.1) from the model retrained without rehearsal data shows stable precision scores and low recall scores. This cautious behavior is an indication that the model has forgotten about the class, and only assigns positive when the model is really confident. yo

Rehearsal Size	F1	Pr	Re
Baseline	0.8571	0.8638	0.8505
0%	0.6035	0.9451	0.4433
1%	0.8557	0.8140	0.9021
5%	0.8194	0.8588	0.7835
10%	0.7227	0.7914	0.6650

(a) PSW on Disease Outbreak-model

Rehearsal Size	F1	Pr	Re
Baseline	0.8812	0.7911	0.9944
0%	0.0842	0.7273	0.0447
1%	0.8861	0.7956	1.0000
5%	0.8790	0.7876	0.9944
10%	0.8790	0.7876	0.9944

(b) Terror Incident on Disease Outbreak-model

Table 4.1: F1, Precision and Recall-scores for the RoBERTa Disease Outbreak-model on PSW and Terror Incident.

Presented in figure 14 are the results of retraining on either Disease or Person Threat with different rehearsal set sizes.

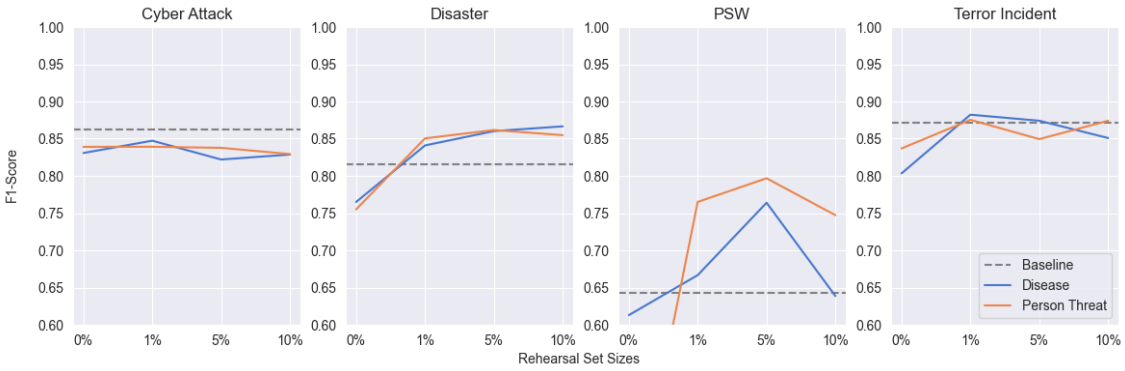

Figure 14: F1-scores for the main classes of Cyber Attack, Disaster, PSW and Terror Incident after retraining on Disease (blue) or Person Threat (orange). The model used was a smaller Hydra2 based on DistilBERT.

The DistilBERT-based models showed stable performance, with few to none signs of forgetting the old event types. The exception was seen on the PSW-dataset, where the Person Threat model retrained without rehearsal data delivered an F1-score far lower than the baseline.

In table 4.2, we can once again observe a low recall-score (0.1907) for the DistilBERT model that exhibited signs of catastrophic forgetting. In table 4.2, we can once again observe a low recall-score (0.1907) for the DistilBERT model that exhibited signs of catastrophic forgetting. Unlike the RoBERTa-based models, where multiple models showed signs of forgetting, only the 0% Person Threat-model among the DistilBERT models displayed any signs.

Rehearsal Size	F1	Precision	Recall
Baseline	0.6429	0.8684	0.5103
0%	0.3045	0.7551	0.1907
1%	0.7650	0.8140	0.7217
5%	0.7968	0.8278	0.7680
10%	0.7472	0.8210	0.6856

Table 4.2: F1-, precision- and recall-scores for the DistilBERT Person Threat-model on PSW.

4.2.3 Role Tagging

Presented in this section are the results from the role tagging task of the Hydra2b-models. The models were evaluated on a pooled dataset, which is why the role Location has been removed from all event types. As Location was the only role in the event type Terror Incident, the event type is not present in the tables.

Event Roles	Baseline	0%	1%	5%	10%
Cyber Attack					
- Attacker	0.8276	0.7826	0.8029	0.7973	0.7700
- Method	0.7751	0.6642	0.7354	0.7273	0.7047
- Target	0.8368	0.7412	0.7708	0.8000	0.7891
Disaster					
- DisasterType	0.8582	0.7564	0.8539	0.8418	0.8537
Public Safety Warning					
- RecommendedAction	0.8867	0.4255	0.9058	0.8712	0.8653
- ThreatType	0.7430	0.6852	0.7259	0.7538	0.7601
Location	0.8633	0.8574	0.8608	0.8517	0.8584

Table 4.3: F1-scores from the role tagging task on the RoBERTa-based Hydra2b-model retrained on Disease Outbreak.

Event Roles	Baseline	0%	1%	5%	10%
Cyber Attack					
- Attacker	0.8276	0.7238	0.7786	0.8027	0.8069
- Method	0.7751	0.5519	0.7315	0.7584	0.7576
- Target	0.8368	0.7645	0.8053	0.8047	0.8451
Disaster					
- DisasterType	0.8582	0.8484	0.8579	0.8589	0.8536
Public Safety Warning					
- RecommendedAction	0.8867	0.8643	0.9066	0.9095	0.8768
- ThreatType	0.7430	0.7413	0.7684	0.7573	0.7647
Location	0.8633	0.8583	0.8551	0.8576	0.8576

Table 4.4: F1-scores from the role tagging task on the RoBERTa-based Hydra2b-model retrained on Person Threat.

In the two RoBERTa-based models retrained with 0% rehearsal data, noticeable forgetting occurs, but it is not catastrophic. The roles suffering the most significant drops, potentially considered catastrophic, are PSW RecommendedAction and Cyber Attack Method, with decreases of 46 and 22 points, respectively, as shown in tables 4.3 and 4.4. Conversely, several roles in the RoBERTa models exhibit minimal performance degradation. Specifically, in the model retrained on Person Threat

Event Roles	Baseline	0%	1%	5%	10%
Cyber Attack					
- Attacker	0.7039	0.5719	0.6485	0.6652	0.6709
- Method	0.7183	0.5013	0.6207	0.7445	0.7195
- Target	0.7008	0.5488	0.6611	0.6570	0.6964
Disaster					
- DisasterType	0.8697	0.5633	0.8495	0.8522	0.8557
Public Safety Warning					
- RecommendedAction	0.8553	0.1787	0.7872	0.8017	0.8497
- ThreatType	0.7374	0.4739	0.7318	0.7684	0.7299
Location	0.8644	0.8635	0.8614	0.8601	0.8644

Table 4.5: F1-scores from the role tagging task on the DistilBERT-based Hydra2b-model retrained on Disease Outbreak.

(4.4), the roles of Disaster DisasterType, PSW RecommendedAction, and PSW ThreatType experience a maximum drop of only 2 points, even without rehearsal data.

The role Location shows resilience in both RoBERTa models. For the model retrained on Disease Outbreak, this resilience is understandable given the inclusion of the Location role. However, its sustained performance in the Person Threat model is unexpected and intriguing, given that Location is not a part of the Person Threat dataset.

The resilience of certain roles, including Location, might be attributed to the overall size of the datasets involved and the inherent capacity of the RoBERTa model. For instance, the Person Threat dataset, one of the smaller datasets with 2,241 entries, demonstrates that a larger model like RoBERTa can maintain performance even with minimal data. This contrasts with the performance on the smaller DistilBERT model, where Location shows a drop in performance, highlighting the impact of model size on resilience to forgetting. Events with minimal role performance degradation, such as Disaster (9,496 entries) and PSW (4,922 entries), have larger datasets compared to Cyber Attack (2,245 entries), which sees the most significant performance drop.

Overall, at 10% rehearsal data, the performance on almost all roles had recovered to their initial performance. The most notable exception to this was the Cyber Attack roles on the RoBERTa models retrained on Disease Outbreak, which was still off by 5 points.

Event Roles	Baseline	0%	1%	5%	10%
Cyber Attack					
- Attacker	0.7039	0.5576	0.6388	0.6481	0.6885
- Method	0.7183	0.5808	0.6687	0.7213	0.7124
- Target	0.7008	0.5738	0.6317	0.6582	0.6770
Disaster					
- DisasterType	0.8697	0.6875	0.8466	0.8598	0.8549
Public Safety Warning					
- RecommendedAction	0.8553	0.7437	0.8353	0.8226	0.8189
- ThreatType	0.7374	0.3680	0.7071	0.7249	0.7268
Location	0.8644	0.6425	0.8501	0.8536	0.8581

Table 4.6: F1-scores from the role tagging task on the DistilBERT-based Hydra2b-model retrained on Person Threat.

5

Discussion

In this study, we evaluated the performance of two variations of multitasking multi-class models and investigated their capabilities as continual learners. In this chapter, we will discuss our findings.

5.1 Did pooling the data affect performance?

In the first part of the study, we pooled the data into two variants: one where all data was placed into the same dataset but otherwise remained unchanged, and the other where the role Location was directed to a dedicated role tagging head. The first variant was used to train the models called Hydra1, and the second was used to train Hydra2.

In the event detection task, the Hydra1 models performed either on par with the baselines, slightly better, or slightly worse. The same can be said about the role tagging task, where Hydra1 often scored slightly better than the baselines and with a lower standard deviation. Simply pooling the data did not have a noticeable impact on performance. This is not inherently a bad thing, as on-par performance still supports the idea of using a single model instead multiple models.

The Hydra2 models often performed remarkably better than both the baseline and their Hydra1 counterparts, both in the event detection task and in the role tagging task. Most noticeable were the performance gains when tagging the role Location on the Terror Incident dataset. Here, the benefits of training on more diverse data became apparent, as the dedicated Location-head outperformed the lowest-scoring opponent by 10 points. After discussing these results with the Text Analytics team, it was revealed that the Terror Incident dataset is likely older than the others and may contain data of lesser quality.

The success of the Hydra2 model cannot be solely attributed to pooling the data. Since Hydra2 featured a dedicated role head for Location, the other role heads were relieved of the need to identify Location. This resulted in smaller heads and, consequently, fewer parameters in the model, which freed up space for the model to create better overall representations. Thus, by adapting the model to the data, we were able to achieve improved scores across both tasks.

From these findings, we can safely say that pooling the data had no adversarial effects on performance, but to actually draw benefits, model adaptation was necessary.

Furthermore, the excellent performance of the Location-head and the insight about dataset quality further strengthens the idea of the model finding synergies in the data when pooling the event types together.

5.2 Did the models learn new event types without forgetting?

In the second part of the study, we trained Hydra models on a subset of our event types and then retrained them on the ones left out. The models were retrained with different amounts of rehearsal data, with configurations of 0%, 1%, 5%, and 10%. The aim of this part was to evaluate the models' capabilities as a continual learner, which includes both the ability to learn new event types, but also remember old event types.

When evaluating the models on Disease Outbreak, we observed scores that were on par with the original Hydra2 models, regardless of the rehearsal dataset size. However, the evaluation of Person Threat showed scores significantly lower than those of the original Hydra2. While there is a size difference between the respective training datasets, another explanation for the performance disparity could lie in event type similarities.

Out of the four event types that Hydra2b was trained on, three (Disaster, PSW, and Terror Incident) describe events happening in the real world, similar to Disease Outbreak. Person Threat, on the other hand, describes interactions between people and could, therefore, be considered different from Disaster, PSW, and Terror Incident. A model trained on similar event types might produce representations that are already in a usable state for the new event type. In contrast, an event type that is not at all similar might be more challenging to incorporate into the pretrained model.

It is important to note that these are merely speculations, as no formal event type similarity analysis was performed on the datasets. One future experiment could be to compute prototypes for each event type, as suggested by Cao, Chen, Zhao, *et al.* [19], and measure the cosine similarity between the prototypes.

As for investigating the forgetfulness, our experiments showed that the models were seemingly resilient to forgetting in the event detection task, even when no rehearsal data was included. There were, of course, some exceptions to this, as evident in figures 13 and 14.

In the role tagging task, we observed both resilience and drops in performance when no rehearsal data was present, as well as recovery when it was added.

In both tasks, when performance dropped, it did eventually recover. However, just 1% was not always sufficient, as evident when examining the results of the Cyber Attack roles in all models. In some cases, not even 10% was enough to fully recover. This might be due to role tagging being a token-level task and, therefore, more sensitive to changes in the overall representations than the event detection tasks.

Despite this, we still believe that the Hydra2 models have the potential to be continual learners, though there are some experiments that could strengthen our confidence:

1. Carefully select which examples are used in the rehearsal sets. In this study, the examples were chosen in order of appearance in the dataset file, and can thus be of questionable quality.
2. Were the rehearsal datasets too small to be used effectively? Would using a constant size, such as 500 or 1 000 entries in each rehearsal set, be better? In Scialom, Chakrabarty, and Muresan [18], 50 datasets were used when training the original model, each consisting of approximately 100 000 entries. 1% of this amounts to 50 000 entries in the rehearsal dataset, which is half the size of each new dataset used. In contrast, our 1% rehearsal set contained 211 entries, which is one 10th of the smaller of the two new datasets used (Person Threat).
3. Should the rehearsal datasets be rebalanced? In the PSW-rehearsal dataset, there were 10 times as many negative examples as there were of the main class. Would a rehearsal set with more emphasis on positive examples result in better recovery?
4. The training regime was unchanged when retraining. Is straight-up mixing the rehearsal data with the new training data the best approach? There is a risk of having batches without any rehearsal data. Would it be better to force some rehearsal data in with every batch?

5.3 RoBERTa or DistilBERT?

Hydra1, Hydra2 and Hydra2b were all trained on both a RoBERTa-body and a DistilBERT-body. The DistilBERT-models performance impressed, and came close to the scores of the RoBERTa counterparts in a number of tests.

The poor results in the role tagging test does DistilBERT a disfavor, as the RoBERTa beats it by over 10 points in two of the roles.

As the continual learning experiments were not entirely conclusive, it is difficult to say which model is better from a continual learning perspective. With this in mind, RoBERTa is the better body for our models, but DistilBERT has potential.

6

Conclusion

This study aimed to investigate whether a multitasking multiclass event extraction model was a feasible alternative to an army of dedicated event extraction models. To achieve this, we trained four different encoder-based models on pooled datasets and evaluated their performances on the tasks of event detection and role tagging. The models that performed best were further tested to evaluate their ability to learn new event types without negatively affecting their performance on previously learned event types.

The resulting model, named Hydra2, surpassed both the baseline models and its predecessor, Hydra1, in event detection and role tagging tasks. This performance enhancement stemmed from pooling the data and making slight modifications to the model architecture to better exploit similarities between event types. Despite being a minor adjustment, the introduction of a role head dedicated to a role found in multiple event types demonstrated the advantages of diversifying the training data. This approach could be extended to more roles and might also aid in distinguishing between structurally similar events that single-event-type models often struggle with.

A smaller model, named Hydra2b, was initially trained on a subset of event types and subsequently retrained on new event types, providing insights into its behavior as a continual learner. Although the scores for the new event types did not always match those of the baseline, the performance appeared to remain unaffected by the addition of rehearsal data into the training process.

The continual learning experiments were based on the claim that 1% rehearsal data was sufficient for an encoder-decoder based model performing generative tasks to operate as a continual learner. We successfully extended this claim to our encoder-based event extraction model for the sentence-level task of event detection. However, we were unable to do the same for the token-level task of role tagging, where not even 10% rehearsal data was enough to maintain performance across all roles. Nonetheless, these results suggest that achieving this might be possible with 10% rehearsal data under the right conditions, setting the stage for future research on the topic.

Bibliography

[1] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2023. arXiv: 1706.03762 [cs.CL].

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: 1810.04805 [cs.CL].

[3] Y. Liu, M. Ott, N. Goyal, *et al.*, *Roberta: A robustly optimized bert pretraining approach*, 2019. arXiv: 1907.11692 [cs.CL].

[4] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, *Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter*, 2020. arXiv: 1910.01108 [cs.CL].

[5] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[6] H. Touvron, T. Lavril, G. Izacard, *et al.*, *Llama: Open and efficient foundation language models*, 2023. arXiv: 2302.13971 [cs.CL].

[7] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, *Glue: A multi-task benchmark and analysis platform for natural language understanding*, 2019. arXiv: 1804.07461 [cs.CL].

[8] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, J. Su, K. Duh, and X. Carreras, Eds., Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392. DOI: 10.18653/v1/D16-1264. [Online]. Available: https://aclanthology.org/D16-1264.

[9] R. Zellers, Y. Bisk, R. Schwartz, and Y. Choi, *Swag: A large-scale adversarial dataset for grounded commonsense inference*, 2018. arXiv: 1808.05326 [cs.CL].

[10] Y. Wu, M. Schuster, Z. Chen, *et al.*, *Google's neural machine translation system: Bridging the gap between human and machine translation*, 2016. arXiv: 1609.08144 [cs.CL].

[11] Y. Zhu, R. Kiros, R. Zemel, *et al.*, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 19–27. DOI: 10.1109/ICCV.2015.11.

[12] G. Hinton, O. Vinyals, and J. Dean, *Distilling the knowledge in a neural network*, 2015. arXiv: 1503.02531 [stat.ML].

[13] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy, *Race: Large-scale reading comprehension dataset from examinations*, 2017. arXiv: 1704.04683 [cs.CL].

[14] V. D. Lai, *Event extraction: A survey*, 2022. arXiv: 2210.03419 [cs.CL].

[15] X. Carreras and L. Màrquez, "Introduction to the CoNLL-2004 shared task: Semantic role labeling," in *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, Boston, Massachusetts, USA: Association for Computational Linguistics, May 2004, pp. 89–97. [Online]. Available: https://aclanthology.org/W04-2412.

[16] R. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, pp. 128–135, May 1999. DOI: 10.1016/S1364-6613(99)01294-2.

[17] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*, vol. 24, Elsevier, 1989, pp. 109–165.

[18] T. Scialom, T. Chakrabarty, and S. Muresan, *Fine-tuned language models are continual learners*, 2022. arXiv: 2205.12393 [cs.CL].

[19] P. Cao, Y. Chen, J. Zhao, and T. Wang, "Incremental event detection via knowledge consolidation networks," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 707–717. DOI: 10.18653/v1/2020.emnlp-main.52. [Online]. Available: https://aclanthology.org/2020.emnlp-main.52.

A

Appendix 1

Presented here are the average F1, Precision and Recall-scores for all models used to investigate RQ1.

Event Type	**F1**	**Precision**	**Recall**
Cyber Attack Class	0.8692 ± 0.0052	0.8530 ± 0.0408	0.8898 ± 0.0499
- Attacker	0.8105 ± 0.0244	0.7933 ± 0.0428	0.8365 ± 0.0821
- Method	0.7543 ± 0.0316	0.7495 ± 0.0712	0.7653 ± 0.0459
- Target	0.8281 ± 0.0199	0.7839 ± 0.0362	0.8786 ± 0.0190
Disaster Class	0.8379 ± 0.0317	0.9129 ± 0.0288	0.7789 ± 0.0745
- DisasterType	0.8554 ± 0.0065	0.8139 ± 0.0128	0.9016 ± 0.0075
- Location	0.8662 ± 0.0046	0.8427 ± 0.0172	0.8917 ± 0.0210
Disease Outbreak Class	0.8334 ± 0.0054	0.7884 ± 0.0078	0.8842 ± 0.0200
- Disease	0.8832 ± 0.0188	0.8513 ± 0.0369	0.9185 ± 0.0094
- Location	0.8828 ± 0.0160	0.8699 ± 0.0373	0.8973 ± 0.0173
Person Threat Class	0.8462 ± 0.0298	0.8706 ± 0.0160	0.8266 ± 0.0700
- TargetObject	0.6698 ± 0.0420	0.7406 ± 0.0230	0.6155 ± 0.0723
- TargetPointer	0.8268 ± 0.0309	0.8421 ± 0.0225	0.8125 ± 0.0431
Public Safety Warning Class	0.7873 ± 0.0478	0.8925 ± 0.0249	0.7083 ± 0.0792
- Location	0.8307 ± 0.0104	0.7902 ± 0.0229	0.8761 ± 0.0146
- RecommendedAction	0.8838 ± 0.0076	0.8463 ± 0.0162	0.9254 ± 0.0209
- ThreatType	0.7696 ± 0.0068	0.7500 ± 0.0587	0.7976 ± 0.0583
Terror Incident Class	0.7083 ± 0.0792	0.7897 ± 0.0124	0.9799 ± 0.0236
- Location	0.7289 ± 0.0564	0.6905 ± 0.0676	0.7752 ± 0.0656

Table A.1: RoBERTa Baseline average scores for RQ1.

Event Type	**F1**	**Precision**	**Recall**
Cyber Attack Class	0.8733 ± 0.0045	0.8370 ± 0.0233	0.9141 ± 0.0257
- Attacker	0.8195 ± 0.0033	0.7690 ± 0.0181	0.8782 ± 0.0276
- Method	0.7603 ± 0.0203	0.7508 ± 0.0293	0.7708 ± 0.0260
- Target	0.8310 ± 0.0078	0.7837 ± 0.0190	0.8848 ± 0.0121
Disaster Class	0.8392 ± 0.0411	0.9129 ± 0.0099	0.7796 ± 0.0721
- DisasterType	0.8498 ± 0.0046	0.8140 ± 0.0130	0.8891 ± 0.0112
- Location	0.8724 ± 0.0082	0.8521 ± 0.0153	0.8939 ± 0.0122
Disease Outbreak Class	0.8321 ± 0.0079	0.7703 ± 0.0169	0.9057 ± 0.0278
- Disease	0.8895 ± 0.0059	0.8683 ± 0.0171	0.9121 ± 0.0093
- Location	0.8811 ± 0.0067	0.8568 ± 0.0172	0.9072 ± 0.0114
Person Threat Class	0.8501 ± 0.0166	0.8557 ± 0.0265	0.8470 ± 0.0487
- TargetObject	0.7179 ± 0.0143	0.7330 ± 0.0714	0.7118 ± 0.0563
- TargetPointer	0.8683 ± 0.0056	0.8704 ± 0.0209	0.8672 ± 0.0246
Public Safety Warning Class	0.8036 ± 0.0429	0.8691 ± 0.0216	0.7526 ± 0.0901
- Location	0.8439 ± 0.0180	0.7865 ± 0.0285	0.9108 ± 0.0111
- RecommendedAction	0.8819 ± 0.0144	0.8220 ± 0.0372	0.9534 ± 0.0310
- ThreatType	0.7752 ± 0.0161	0.7003 ± 0.0286	0.8694 ± 0.0233
Terror Incident Class	0.8732 ± 0.0109	0.7939 ± 0.0106	0.9709 ± 0.0350
- Location	0.7173 ± 0.0591	0.6880 ± 0.0855	0.7559 ± 0.0644

Table A.2: RoBERTa Hydra1 average scores for RQ1.

Event Type	F1	Precision	Recall
Cyber Attack Class	0.8736 ± 0.0078	0.8100 ± 0.0171	0.9484 ± 0.0156
- Attacker	0.8470 ± 0.0080	0.7928 ± 0.0207	0.9096 ± 0.0136
- Method	0.7719 ± 0.0128	0.7575 ± 0.0470	0.7903 ± 0.0345
- Target	0.8628 ± 0.0037	0.8333 ± 0.0070	0.8946 ± 0.0101
Disaster Class	0.8737 ± 0.0104	0.8999 ± 0.0039	0.8491 ± 0.0207
- DisasterType	0.8718 ± 0.0033	0.8422 ± 0.0184	0.9041 ± 0.0152
- Location	0.8772 ± 0.0078	0.8534 ± 0.0166	0.9026 ± 0.0080
Disease Outbreak Class	0.8393 ± 0.0065	0.8393 ± 0.0065	0.9226 ± 0.0100
- Disease	0.9156 ± 0.0053	0.9049 ± 0.0110	0.9267 ± 0.0058
- Location	0.8937 ± 0.0059	0.8820 ± 0.0115	0.9059 ± 0.0051
Person Threat Class	0.8647 ± 0.0160	0.8634 ± 0.0230	0.8685 ± 0.0518
- TargetObject	0.7432 ± 0.0132	0.8046 ± 0.0383	0.6936 ± 0.0439
- TargetPointer	0.8792 ± 0.0131	0.8737 ± 0.0135	0.8850 ± 0.0226
Public Safety Warning Class	0.8528 ± 0.0239	0.8254 ± 0.0253	0.8845 ± 0.0562
- Location	0.8601 ± 0.0112	0.8093 ± 0.0206	0.9182 ± 0.0097
- RecommendedAction	0.8909 ± 0.0208	0.8322 ± 0.0338	0.9593 ± 0.0176
- ThreatType	0.7843 ± 0.0156	0.7069 ± 0.0293	0.8824 ± 0.0273
Terror Incident Class	0.8775 ± 0.0035	0.7867 ± 0.0082	0.9922 ± 0.0093
- Location	0.8104 ± 0.0064	0.7415 ± 0.0095	0.8936 ± 0.0090

Table A.3: RoBERTa Hydra2 average scores for RQ1.

Event Type	F1	Precision	Recall
Cyber Attack Class	0.8223 ± 0.0530	0.8433 ± 0.0241	0.8105 ± 0.1082
- Attacker	0.6636 ± 0.0395	0.6563 ± 0.0167	0.6751 ± 0.0761
- Method	0.6868 ± 0.0262	0.6498 ± 0.0348	0.7333 ± 0.0702
- Target	0.6943 ± 0.0272	0.6373 ± 0.0315	0.7641 ± 0.0413
Disaster Class	0.8408 ± 0.0107	0.8878 ± 0.0115	0.7993 ± 0.0284
- DisasterType	0.8613 ± 0.0014	0.8357 ± 0.0081	0.8887 ± 0.0072
- Location	0.8680 ± 0.0050	0.8450 ± 0.0079	0.8924 ± 0.0111
Disease Outbreak Class	0.8113 ± 0.0074	0.7720 ± 0.0139	0.8552 ± 0.0155
- Disease	0.8780 ± 0.0176	0.8224 ± 0.0347	0.9427 ± 0.0093
- Location	0.8814 ± 0.0067	0.8642 ± 0.0047	0.8994 ± 0.0111
Person Threat Class	0.8380 ± 0.0066	0.8472 ± 0.0134	0.8295 ± 0.0220
- TargetObject	0.6724 ± 0.0087	0.6995 ± 0.0320	0.6500 ± 0.0382
- TargetPointer	0.8423 ± 0.0075	0.8539 ± 0.0182	0.8312 ± 0.0095
Public Safety Warning Class	0.7297 ± 0.0578	0.8433 ± 0.0257	0.6495 ± 0.0992
- Location	0.8028 ± 0.0123	0.7351 ± 0.0293	0.8858 ± 0.0246
- RecommendedAction	0.8624 ± 0.0191	0.8270 ± 0.0303	0.9017 ± 0.0213
- ThreatType	0.7016 ± 0.0234	0.6712 ± 0.0742	0.7471 ± 0.0700
Terror Incident Class	0.8803 ± 0.0068	0.7978 ± 0.0105	0.9821 ± 0.0083
- Location	0.7090 ± 0.0343	0.7311 ± 0.0183	0.6896 ± 0.0552

Table A.4: DistilBERT baseline average scores for RQ1.

Event Type	F1	Precision	Recall
Cyber Attack Class	0.8161 ± 0.0194	0.8475 ± 0.0205	0.7891 ± 0.0496
- Attacker	0.6857 ± 0.0277	0.6253 ± 0.0456	0.7614 ± 0.0254
- Method	0.7089 ± 0.0176	0.6445 ± 0.0282	0.7889 ± 0.0272
- Target	0.7047 ± 0.0277	0.6530 ± 0.0398	0.7664 ± 0.0231
Disaster Class	0.8261 ± 0.0142	0.8851 ± 0.0121	0.7754 ± 0.0324
- DisasterType	0.8560 ± 0.0062	0.8227 ± 0.0064	0.8921 ± 0.0090
- Location	0.8706 ± 0.0032	0.8433 ± 0.0072	0.8999 ± 0.0102
Disease Outbreak Class	0.8159 ± 0.0119	0.7591 ± 0.0200	0.8824 ± 0.0163
- Disease	0.8909 ± 0.0106	0.8457 ± 0.0181	0.9415 ± 0.0099
- Location	0.8883 ± 0.0083	0.8691 ± 0.0267	0.9090 ± 0.0146
Person Threat Class	0.8351 ± 0.0126	0.8405 ± 0.0077	0.8302 ± 0.0275
- TargetObject	0.6580 ± 0.0160	0.6536 ± 0.0136	0.6645 ± 0.0433
- TargetPointer	0.8501 ± 0.0107	0.8576 ± 0.0122	0.8431 ± 0.0225
Public Safety Warning Class	0.7488 ± 0.0352	0.8314 ± 0.0146	0.6835 ± 0.0607
- Location	0.8324 ± 0.0042	0.7682 ± 0.0100	0.9085 ± 0.0068
- RecommendedAction	0.8388 ± 0.0240	0.7814 ± 0.0374	0.9068 ± 0.0273
- ThreatType	0.7450 ± 0.0128	0.6645 ± 0.0211	0.8482 ± 0.0163
Terror Incident Class	0.8741 ± 0.0089	0.7930 ± 0.0091	0.9743 ± 0.0297
- Location	0.7638 ± 0.0324	0.7521 ± 0.0209	0.7766 ± 0.0474

Table A.5: DistilBERT Hydra1 average scores for RQ1.

Event Type	F1	Precision	Recall
Cyber Attack Class	0.8567 ± 0.0085	0.8144 ± 0.0146	0.9043 ± 0.0233
- Attacker	0.7078 ± 0.0140	0.6728 ± 0.0214	0.7482 ± 0.0381
- Method	0.7659 ± 0.0166	0.7174 ± 0.0354	0.8236 ± 0.0328
- Target	0.7247 ± 0.0120	0.6543 ± 0.0234	0.8131 ± 0.0176
Disaster Class	0.8549 ± 0.0109	0.8605 ± 0.0168	0.8507 ± 0.0375
- DisasterType	0.8748 ± 0.0026	0.8438 ± 0.0140	0.9086 ± 0.0169
- Location	0.8702 ± 0.0060	0.8397 ± 0.0116	0.9032 ± 0.0093
Disease Outbreak Class	0.8186 ± 0.0172	0.7586 ± 0.0209	0.8923 ± 0.0599
- Disease	0.9179 ± 0.0043	0.8777 ± 0.0076	0.9620 ± 0.0029
- Location	0.8923 ± 0.0040	0.8534 ± 0.0142	0.9352 ± 0.0107
Person Threat Class	0.8383 ± 0.0125	0.8541 ± 0.0090	0.8237 ± 0.0300
- TargetObject	0.6970 ± 0.0267	0.6807 ± 0.0672	0.7209 ± 0.0420
- TargetPointer	0.8537 ± 0.0101	0.8451 ± 0.0151	0.8628 ± 0.0186
Public Safety Warning Class	0.7744 ± 0.0296	0.8337 ± 0.0260	0.7258 ± 0.0578
- Location	0.8442 ± 0.0024	0.7765 ± 0.0104	0.9250 ± 0.0100
- RecommendedAction	0.8518 ± 0.0156	0.7999 ± 0.0384	0.9127 ± 0.0196
- ThreatType	0.7519 ± 0.0152	0.6642 ± 0.0318	0.8682 ± 0.0230
Terror Incident Class	0.8778 ± 0.0077	0.7907 ± 0.0123	0.9866 ± 0.0064
- Location	0.8093 ± 0.0117	0.7369 ± 0.0181	0.8977 ± 0.0096

Table A.6: DistilBERT Hydra2 average scores for RQ1.

B

Appendix 2

Presented here are the F1-scores for the Hydra1b-models used to investigate RQ2.

Event Roles	**Baseline**	**0%**	**1%**	**5%**	**10%**
Cyber Attack	0.8737	0.3520	0.8333	0.8313	0.8484
- Attacker	0.8166	0.5291	0.7991	0.7871	0.7991
- Method	0.7755	0.6800	0.7233	0.7921	0.7390
- Target	0.8377	0.7388	0.8101	0.7477	0.7526
Disaster	0.6800	0.7902	0.8624	0.8608	0.8793
- DisasterType	0.8773	0.7919	0.8410	0.8311	0.8421
- Location	0.8701	0.8370	0.8509	0.8645	0.8588
Public Safety Warning	0.8333	0.1784	0.7528	0.8489	0.7937
- Location	0.8579	0.8124	0.8408	0.8514	0.8301
- RecommendedAction	0.9014	0.7709	0.8789	0.8577	0.8628
- ThreatType	0.7594	0.6645	0.7023	0.7404	0.7100
Terror Incident	0.8706	0.3172	0.8607	0.8642	0.8784
- Location	0.7796	0.7688	0.7940	0.8061	0.7928

Table B.1: RoBERTa Hydra1b F1-scores for RQ2, retrained on Disease Outbreak.

Event Roles	**Baseline**	**0%**	**1%**	**5%**	**10%**
Cyber Attack	0.8737	0.8508	0.8434	0.8646	0.8633
- Attacker	0.8166	0.7819	0.7736	0.7892	0.8387
- Method	0.7755	0.6612	0.7596	0.7568	0.8082
- Target	0.8377	0.7721	0.8078	0.7729	0.8140
Disaster	0.6800	0.7773	0.8593	0.8723	0.8734
- DisasterType	0.8773	0.8418	0.8418	0.8444	0.8536
- Location	0.8701	0.8525	0.8640	0.8616	0.8630
Public Safety Warning	0.8333	0.7129	0.8226	0.8033	0.7968
- Location	0.8579	0.8077	0.8378	0.8360	0.8441
- RecommendedAction	0.9014	0.8391	0.8808	0.8687	0.8520
- ThreatType	0.7594	0.7385	0.7475	0.7692	0.7629
Terror Incident	0.8706	0.8533	0.8900	0.8838	0.8866
- Location	0.7796	0.7601	0.8173	0.7807	0.7722

Table B.2: RoBERTa Hydra1b F1-scores for RQ2, retrained on Person Threat.

Event Roles	Baseline	0%	1%	5%	10%
Cyber Attack	0.8619	0.6626	0.8414	0.8514	0.8267
- Attacker	0.6928	0.6409	0.6344	0.6405	0.6773
- Method	0.7107	0.4533	0.7473	0.7670	0.7022
- Target	0.7322	0.6292	0.7020	0.6965	0.7267
Disaster	0.8295	0.8489	0.8551	0.8461	0.8604
- DisasterType	0.8714	0.8324	0.8486	0.8512	0.8440
- Location	0.8716	0.8399	0.8627	0.8639	0.8631
Public Safety Warning	0.7989	0.6667	0.7645	0.7341	0.7838
- Location	0.8270	0.7653	0.8395	0.8176	0.8275
- RecommendedAction	0.8543	0.7108	0.7911	0.8058	0.8092
- ThreatType	0.7325	0.6407	0.6846	0.6971	0.7340
Terror Incident	0.8812	0.8432	0.8832	0.8878	0.8894
- Location	0.8142	0.6899	0.7950	0.8080	0.8000

Table B.4: DistilBERT Hydra1b F1-scores for RQ2, retrained on Person Threat.

Event Roles	Baseline	0%	1%	5%	10%
Cyber Attack	0.8619	0.8425	0.8492	0.8462	0.8616
- Attacker	0.6928	0.5363	0.6530	0.6520	0.6605
- Method	0.7107	0.6762	0.6324	0.7354	0.7219
- Target	0.7322	0.5675	0.6944	0.7002	0.7105
Disaster	0.8295	0.7371	0.8421	0.8610	0.8614
- DisasterType	0.8714	0.6394	0.8481	0.8026	0.8586
- Location	0.8716	0.7681	0.8636	0.8598	0.8560
Public Safety Warning	0.7989	0.7524	0.7654	0.7049	0.7427
- Location	0.8270	0.6735	0.8109	0.8135	0.8119
- RecommendedAction	0.8543	0.6486	0.8016	0.7885	0.8293
- ThreatType	0.7325	0.4412	0.7129	0.6853	0.7211
Terror Incident	0.8812	0.7036	0.8693	0.8741	0.8794
- Location	0.8142	0.5103	0.7904	0.8049	0.7988

Table B.3: DistilBERT Hydra1b F1-scores for RQ2, retrained on Disease Outbreak.