Danmarks Tekniske Universitet

Few-shot NER for biomedical texts

W. Ø. Larsen, Simon s174450

Acknowledgements

I would like to thank my supervisor Ole Winther from DTU Compute to let me do this project with Silvi.ai and to always support new exciting projects. Thank you Rasmus Hvingelby for guiding me and helping me working on this project. I was able to further deepen my skills in the field of NLP, specifically NER - which is an ever-growing subject.

1 Introduction

Named Entity Recognition (NER) is a critical task in natural language processing that involves identifying and categorizing specific entities, such as persons, organizations, locations, medical codes, time expressions, quantities, percentages, etc., in text. The utility of this task becomes exceedingly vital in the field of bio-medicine, where recognizing and classifying medical terms and entities can enable more effective information extraction and data analysis. Pre-trained language models (e.g. BERT [1]) that have been pre-trained on huge amounts of labeled data have shown considerate performance in this domain. However, in practical applications, NER systems are usually expected to rapidly adapt to some new entity types unseen during training. It is costly while not flexible to collect a number of additional labeled data for these types.

Despite the significant progress in NER methods, the lack of annotated data for rare or novel entity types remains a challenge, especially in the ever-evolving bio-medical domain. This is where few-shot learning for NER proves its worth. By training models to learn from a small number of annotated examples, few-shot learning enables the rapid adaptation to new entity types with minimal manual labeling effort, paving the way for effective knowledge discovery from the vast bio-medical literature.

In this study, we will examine the performance of two different models in the context of Few-shot learning. We will be looking at ProtoBert [2] model, an extension based on prototypical networks [3] supported on a BERT encoder [1]. Additionally, we will be comparing the performance of this latter with a more complicated model based on Meta-learning. Specifically, the Decomposed Meta-Learning algorithm presented by Ma et al. [4]. This much more complex approach sequentially addresses few-shot span detection and few-shot entity typing with two decomposed models. They then combine the two to make one predictor and perform NER.

In this report, we aim to examine the performance of this decomposed meta-learning approach versus a more "simple" and straightforward implementation, like ProtoBert, in the few-shot NER setting. Specifically, we will be evaluating them on the PICO (Population, Intervention, Comparison, Outcome) Corpus, a collection of bio-medical research articles. We will also compare our results with Ma et al. [4] on the Few-NERD [5] data set with similar models. The primary objective of our research is to evaluate how well two models can perform an NER task on bio-medical texts in a Few-shot manner. The study is structured as follows:

1. We begin our report by clearly defining the task at hand, namely, Named Entity Recognition (NER) and its few-shot learning variant. In this section, we bring a concise explanation of what NER is, what Few-shot Named Entity Recognition signifies, and outline the scheme of entity type training we intend to adopt for our study.

2. We will then continue to describe the two distinct datasets we employed in our experiments – the Few-NERD dataset and the PICO Corpus. We start by offering a comprehensive description of the Few-NERD dataset and its relevance to our study. Subsequently, we describe the PICO Corpus, explaining its nature and the steps undertaken to adapt its format to align with the structure of the Few-NERD dataset.

3. Subsequently, we focus on the experimental setup. We introduce and detail the models employed in our analysis: the Decomposed Meta-Learning model as presented in the work of Ma et al. [4], and the baseline model, ProtoBert ([3], [1]). Following the model

definitions, we report on our experimental results, offering a comparative analysis of the performance of the two models under various scenarios.

4. Following our evaluation, we dedicate a segment for discussion, probing the strengths, limitations, and notable aspects of the methodologies tested and the outcomes observed.

5. Finally, the report concludes with proposed directions for future research and potential improvements, along with considerations for practical application of our approach in real-world bio-medical few-shot NER tasks.

As the field of bio-medicine continually expands, the need for efficient and adaptable NER systems is more pressing than ever. Through this study, we hope to contribute to this demand by advancing the application of meta-learning techniques for few-shot NER in bio-medical texts.

2 Task Definition

In this section, we will explain the concept of Named Entity Recognition (NER). In certain situations, models are trained and adjusted to carry out this task. However, they are also required to effectively apply their knowledge to domains slightly outside their training domain. They are given only a limited amount of contextual information, referred to as "few-shots," to make predictions. We will delve deeper into this aspect and discuss our approach to organizing the training, validation, and test data to accommodate this few-shot approach.

2.1 Named Entity Recognition

NER is formulated as a sequence labeling problem. Given an input sequence $\boldsymbol{x} = \{x_i\}_{i=1}^{L}$ with L tokens, an NER model is supposed to output a label sequence $\boldsymbol{y} = \{y_i\}_{i=1}^{L}$, where x_i is the i-th token, $y_i \in \mathcal{Y} \cup \{O\}$ is the label of x_i, $\mathcal{Y}$ is the pre-defined entity class set, and O denotes the non-entities.

In order for a model to properly "locate" where these entities should be labelled, during training the model will embed *spanned annotations*. These annotations are an integral part of NER. They introduce an extra layer of specificity in the labeling process. Span in this context refers to the portion of text that corresponds to a particular entity in the input sequence. Consequently, an annotation is the labeling of this identifies span with the appropriate entity type. This method allows for the model to encapsulate richer semantic information as spans often carry meaning collectively as a unit rather than as individual tokens. Which is crucial in the context of bio-medical texts since they can contain entities such as disease names or drug names which often span multiple tokens, and correctly classifying the full span of these entities is essential.

2.2 Few-shot Named Entity Recognition

In the Few-NERD paper by Ding et al. [5], they employ a Few-shot learning approach to train a Named Entity Recognition (NER) model. This approach, also known as an N-way K-shot setting, operates on "episodes" that are constructed iteratively. In each training episode, we randomly sample K examples (referred to as K-shot) for N classes

(referred to as N-way) from a specific domain. These examples form a support set denoted as $\mathcal{S}_{\text{train}} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N \times K}$. Additionally, we sample K' examples for each of the N classes to create a query set denoted as $\mathcal{Q}_{\text{train}} = \{(\mathbf{x}^{(j)}, \mathbf{y}^{(j)})\}_{j=1}^{N \times K'}$, ensuring that the support set and query set have no overlapping instances ($\mathcal{S}_{train} \cap \mathcal{Q}_{train} = \emptyset$).

During training, the Few-shot learning model learns to predict labels for the query set $\mathcal{Q}_{train}$ using the information provided by the support set $\mathcal{S}_{train}$. The supervision for both $\mathcal{S}_{train}$ and $\mathcal{Q}_{train}$ is available during the training process. Furthermore, a validation set (or dev set) is used to fine-tune the model on unseen examples and entities. Similarly, during the testing phase, the model is evaluated on unseen classes from the training phase. Few-shot learning systems are tasked with making predictions on the unlabeled query set $\mathcal{Q}_{test}$ using only a few labeled examples from the support set $\mathcal{S}_{test}$, where $\mathcal{S}_{test} \cap \mathcal{Q}_{test} = \emptyset$.

However, in sequence labeling problems such as Named Entity Recognition (NER), a sentence can contain multiple entities from different classes. Sampling examples at the sentence level is crucial for sequence labeling tasks, particularly for NER, due to the importance of contextual information. Consequently, the sampling process is more challenging compared to conventional classification tasks. Ding et al. [5] describe related work on methods used to sample examples and classes to form episodes, support sets and query sets. Although this can provide an interesting topic of discussion and research, it will not be considered in this report. The Few-NERD data set involves dense entity annotations. Therefore, they adopt a N-way $K \sim 2K$-shot setting. The primary principle of this setting is to ensure that each class in $\mathcal{S}$ contains $K \sim 2K$ examples, effectively mitigating the limitations of sampling. $\mathcal{E}_{test} = \{(\mathcal{S}_{test}, \mathcal{Q}_{test}, \mathcal{Y}_{test})\}$

This approach is not suitable for FEW-NERD, as it involves dense entity annotations. Therefore, in our report, just like Ding et al. we adopt an N-way $K \sim 2K$-shot setting. The primary principle of this setting is to ensure that each class in $\mathcal{S}$ contains $K \sim 2K$ examples, effectively mitigating the limitations of sampling. Subsequently, we will adopt the same sampling strategy for the PICO data set - as described in Section 3.

Target Types $\mathcal{Y}$	[condition], [intervention]
Support set $\mathcal{S}$	(1) Phase II double - blind placebo - controlled randomized study of *armodafinil for brain radiation - induced fatigue* [condition]. (2) Efficacy of *denosumab* [intervention] for restoring normal mineral density in women receiving adjuvant aromatase inhibitors for early breast cancer.
Query Set $\mathcal{Q}$	(1) The Kampo medecine Goshajinkigan prevents neuropathy in breast cancer patients treated with docetaxel. (2) Ribociclib plus letrozole in early cancer: A presurgical, window - of - opportunity study.
Expected output	(1) The Kampo medecine Goshajinkigan prevents *neuropathy*[*condition*] in breast cancer patients treated with docetaxel. (2) *Ribociclib plus letrozole*[intervention] in early cancer: A presurgical, window - of - opportunity study.

Figure 1: An example of a simple 2-way 1-shot setting, here we have two entity classes and one example for each of them. Both in the support set and the query set. Different colors indicate different entity classes.

Courtesy of Ma et al. paper [4], we present an example on figure 2.2 of how an episode based on the PICO corpus can look like.

2.3 Meta-Learning

Meta-learning, often referred to as "learning to learn," is a rapidly evolving field in machine learning [6] [2] that aims to design models that can quickly adapt to new tasks with minimal training data. By training on a diverse range of related tasks, it acquires the ability to quickly adapt to new tasks by extracting and encapsulating transferable knowledge. By learning this aspect of higher-level knowledge from previous experience, it can leverage it to better generalize to new tasks.

This approach becomes particularly relevant and performance-critical in the context of few-shot learning, where the aim is to design models that can generalize effectively from a limited number of examples – often just a handful per class. Conventional machine learning models usually struggle in this setting due to the paucity of training data, which is less the case with meta-learning models.

As we described in the subsection above, in the context of few-shot learning problem, we are given an episode $\mathcal{E}$ from an episode distribution $\mathcal{P}(\mathcal{E})$. Each episode $\mathcal{E}_i$ contains a support set $\mathcal{S}_i$ and a query set $\mathcal{Q}_i$, where the support set is used for learning and the query set is used for testing. The objective is to learn a model f_θ parameterized by θ such that it minimizes the loss $L(\mathcal{E}_i, f_\theta)$ on new tasks.

$$L(\mathcal{E}_i, f_\theta) = \mathbb{E}_{(x,y)\sim\mathcal{Q}_i}[l(f_\theta(x), y)] \tag{1}$$

However, instead of optimizing for performance on training tasks, meta-learning optimizes on new episodes from $\mathcal{P}(\mathcal{E})$ with different entities and examples. This optimization often involves a bi-level optimization process, where the outer loop learns across tasks to update the model parameters θ, and the inner loop adapts the model to each specific task. A simplified visual sketch in Figure 2 can help better understand the process.

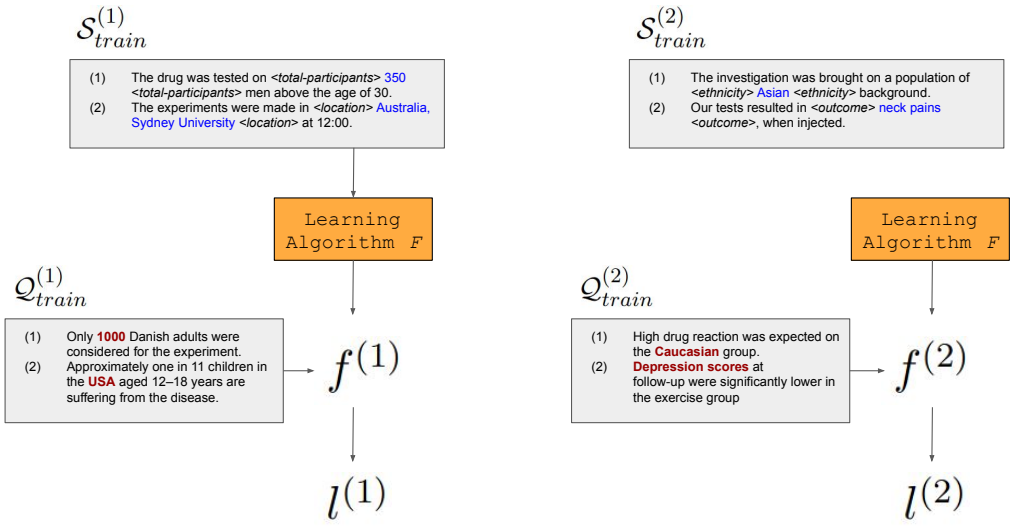


Figure 2: Meta-Learning sketch for two different episodes. This can also be sketched as a meta-testing phase.

Thus, by giving an N and K, a meta-learner model generates a task-specific model for a new task (or episode). This model gets updated by leveraging the support set $\mathcal{S}_{train}$, then

the task-specific model is tested on $\mathcal{Q}_{train}$ to get a test error. The meta-learner then learns to learn new tasks by considering how to reduce the test error on $\mathcal{Q}_{train}$ by updating on $\mathcal{S}_{train}$. Finally, the meta-learner is evaluated on test episodes in a similar format.

By learning to learn, meta-learning models can potentially provide a powerful tool for solving few-shot NER tasks, where the goal is to recognize new entity types from a small number of examples. This is particularly relevant in bio-medical NER, where new entity types keep emerging, and collecting a large number of annotated examples for each type is expensive.

One important downfall of meta-learning the reader must keep in mind, is its computational complexity during meta-training. Firstly, a model performs *inner-updates* and minimizes its error on randomly sampled support examples. Secondly, it evaluates its latter parameters on the query sets and performs *meta-update* by aggregating all these episodes. This involves a second order derivative with can become computationally inefficient. We go further in details in the explanation of this complexity in Appendix A.

3 Data Collections

We now shortly present the two data sets we will be evaluating our models on. Understanding their structure and how the domain knowledge is embedded within them is important when reviewing how our models perform. These two data sets offer a great example of knowledge transfer [7], also known as transfer learning. It refers to the process of transferring knowledge gained from domain to another related or unrelated domain. Thus, it involves leveraging the learned knowledge or representations from a source domain to improve performance on a target domain. This is why the Few-NERD data set, and its formatting in episodes, is an attractive candidate as a few-shot data set for NER. Therefore, we wanted to apply this similar formatting in the context of NER in bio-medical papers. Specifically, the PICO Corpus: A Publicly Available Corpus to Support Automatic Data Extraction from Biomedical Literature.

Before going into the description of the two data sets, we first want to present the assumptions made for both data sets. Both data sets contain types labeled in a hierarchical manner. Each data set contains a number of coarse-grained types, these latter contain fine-grained types. Our models will be trained to predict these fine-grained types. We proceed to describe how these fine-grain types are split for training.

Let all our entities belong to an entity set (denoted as $\mathcal{E}$). We split the entity set into three mutually disjoint subsets, respectively denoted as $\mathcal{E}_{train}$, $\mathcal{E}_{dev}$, $\mathcal{E}_{test}$, and $\mathcal{E}_{train} \cup \mathcal{E}_{dev} \cup \mathcal{E}_{test} = \mathcal{E}$, $\mathcal{E}_{train} \cap \mathcal{E}_{dev} \cap \mathcal{E}_{test} = \emptyset$. Under these circumstances, the instances in the training, development, and testing datasets consist solely of instances containing entities from $\mathcal{E}_{train}$, $\mathcal{E}_{dev}$, and $\mathcal{E}_{test}$, respectively. However, Named Entity Recognition (NER) is a sequence labeling problem, and it is possible for a sentence to contain multiple entities of different types. To prevent the introduction of new entity types during the training phase, we replace the labels of entities belonging to $\mathcal{E}_{test}$ with O (representing the absence of an entity) in the training set. Similarly, in the test set, entities belonging to $\mathcal{E}_{train}$ and $\mathcal{E}_{dev}$ are also replaced with O.

The paper Few-NERD [5] describes two splitting strategies with regards to building the three data sets. In an *intra*-manner or *inter*. The *intra*-manner relates to splitting the entities between train, development and test, such that they belong to different coarse-grained types. On the other hand, *inter*-splitting focuses on sharing the coarse-grained

types. In other words, we split the overall set of fine-grained entities between the three data sets (train, dev and test) . This means that $\mathcal{E}_{train}$ and $\mathcal{E}_{test}$ might share entities from the same coarse-grained type. In the field of bio-medicine, the *inter* setting makes more sense. It is common in real-word scenarios that NER models will be evaluated on classes that were also present during training. We will therefore only be focusing on *inter*-split entities for both Few-NERD and PICO.

3.1 Few-NERD

Few-NERD is a large-scale human-annotated dataset specifically designed for few-shot Named Entity Recognition (NER). It contains 188.2k sentences extracted from Wikipedia articles, with 491.7k entities manually annotated by well-trained annotators. Few-NERD is considered one of the largest human-annotated NER datasets available. The dataset incorporates an annotation schema that includes 8 coarse-grained entity types and 66 fine-grained entity types. This annotation schema was carefully designed through several pre-annotation rounds to ensure accurate and comprehensive labeling of the entities. Below on Figure 3, a schema of the entity types in Few-NERD presented by Ding et al. [5].

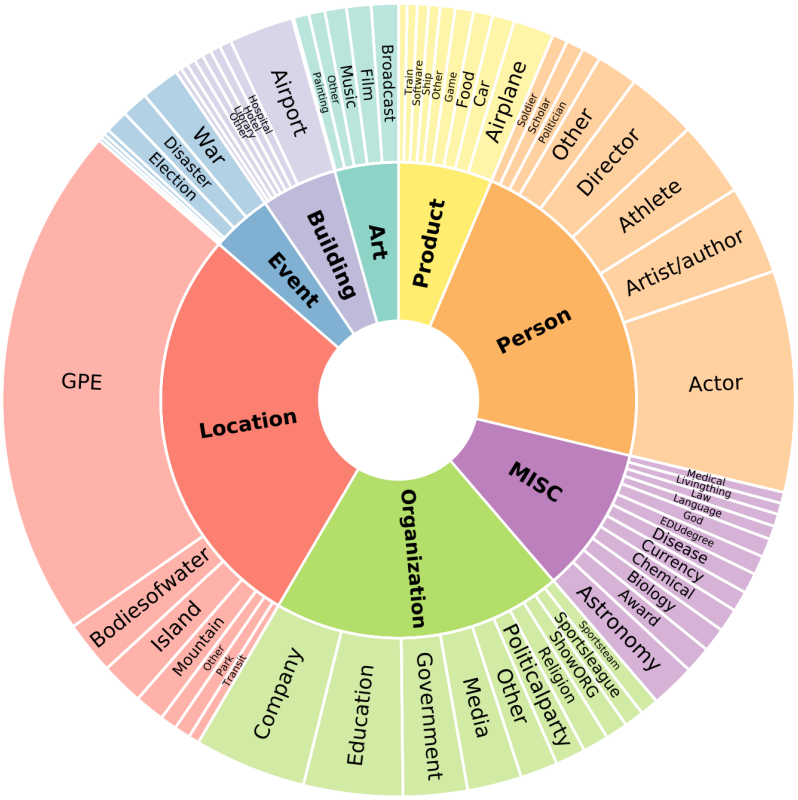


Figure 3: Overview of Few-NERD coarse-grained and fine-grained entities.

3.2 PICO Corpus

The PICO (Population, Intervention, Comparison, Outcome) Corpus [8] is a prominent dataset in the field of biomedical natural language processing (NLP). It serves as a valuable resource for researchers and practitioners working on tasks related to evidence-based

medicine and clinical research. Comprising annotated abstracts from scientific articles in the medical domain, the PICO Corpus focuses on the four essential components of a clinical research question: population, intervention, comparison, and outcome. The corpus contains 1011 manually annotated abstracts. The corpus has 17,739 entities, their frequency is described in table 1.

Table 1: Frequency of each entity in PICO

Sub-category	Number of examples
Participants (P)	
total-participants	847
intervention-participants	674
control-participants	647
age	210
eligibility	864
ethnicity	83
condition	321
location	168
Intervention & Control (IC)	
intervention	1011
control	949
Outcomes (O)	
outcome	978
outcome-measure	413
iv-bin-abs	288
cv-bin-abs	258
iv-bin-percent	561
cv-bin-percent	520
iv-cont-mean	154
cv-cont-mean	154
iv-cont-median	140
cv-cont-median	133
iv-cont-sd	69
cv-cont-sd	67
iv-cont-q1	3
cv-cont-q1	3
iv-cont-q3	3
cv-cont-q3	3

4 Models

In this sub-section, we present the overall approach of ProtoBert, a baseline model that we have chosen for our study. ProtoBert, developed by Ding et al. [5], is a powerful few-shot learning algorithm. In a similar aspect as Ma et al., we perform benchmarking of ProtoBert and other models against the Decomposed Meta-Learning (DML) algorithm. We will provide a brief description of the structure and algorithm of DML in this sub-section.

4.1 ProtoBERT

The ProtoBERT model [2] is one of the baseline models utilized in our study, drawing inspiration from the accomplishments of various meta-learning approaches in the field of few-shot learning ([9], [10]). This model capitalizes on the principles of a prototypical network, introduced by Snell et al. in 2017 [3], whilst incorporating the BERT encoder [1].

In this approach, the central element is the generation of a prototype, denoted as $\boldsymbol{z}$, for each unique entity type. This prototype is computed by determining the average of the embeddings of all tokens identified with a particular entity type. The averaging operation is performed on the support set $\mathcal{S}$, where, for the i-th type, the prototype is denoted as $\boldsymbol{z}_i$, and the corresponding support set is $\mathcal{S}_i$,

$$\boldsymbol{z}_i = \frac{1}{|\mathcal{S}_i|} \sum_{x \in \mathcal{S}_i} f_\theta(x). \tag{2}$$

For each token x in the query set $\mathcal{Q}$, the model calculates the distance between x and all existing prototypes. The euclidean distance serves as the metric function $d\left(f_\theta(x), \boldsymbol{z}\right) = \|f_\theta(x) - \boldsymbol{z}\|_2^2$.

Upon determining the distances between the token x and all prototypes, the model calculates the prediction probability of x being classified under each type. In the training phase, the model parameters are updated for each meta-task. During the testing phase, the model predicts the label of the prototype nearest to the token x.

Formally, given a support set $\mathcal{S}$ with types $\mathcal{Y}$ and a query x, the prediction process can be articulated as follows:

$$\begin{aligned} y^* &= \arg\min_{y \in \mathcal{Y}} d_y(x), \\ d_y(x) &= d\left(f_\theta(x), \boldsymbol{z}_y\right). \end{aligned} \tag{3}$$

Here, y^* denotes the predicted entity type for token x, which is determined by finding the entity type y in $\mathcal{Y}$ that minimizes the distance $d_y(x)$. This method thus utilizes the concept of prototypical networks to tackle the challenges of few-shot NER, incorporating them into the powerful architecture of the BERT model.

4.2 Decomposed Meta-Learning

Leveraging the power of Meta-Learning, Ma et al. 2 tackle an issue which can arise with prototypical networks like ProtoBERT. The latter exhibits limitations in its reliance on span-level metric learning. Firstly, during the decoding process, special care must be taken to handle overlapping spans, which occurs when multiple spans share some common tokens. For example, consider the sentence "The man ate in the restaurant." If we have two spans, one starting from "man" and ending at "ate" and another starting from "ate" and ending at

"in" we need to handle the overlap between the token "ate" This overlapping span requires careful consideration to avoid ambiguity.

Secondly, the class prototype corresponding to non-entities, represented by the "*O*" class, often contains noise. This noise arises because common non-entity words in a large vocabulary rarely share significant similarities. To give an example, if we consider the sentence "I bought an apple," the non-entity word "an" does not exhibit any noteworthy shared characteristics with other non-entity words. As a result, the class prototype for non-entities tends to be noisy and less reliable for accurate representation.

The paper addresses these issues by proposing a decomposed meta-learning framework. The decomposition presents two parts. Firstly a *few-shot entity span detection model* and secondly a *few-shot entity typing model*, respectively trained and tested via meta-learning. The figure provided by Ma et al. - publicly available on their GitHub - offers a good understanding of these two independent models and their training

Few-shot span detector
As we explained earlier, the decomposed meta-learning approach tackles the issue of overlapping spans by modelling the few-shot span detection as a sequence labeling problem. In other words, the authors divide the input sequence into individual tokens and assign a label to each token based on whether it belongs to a named entity span or not. To do this, during training, they use a class-agnostic detection model, specifically the model-agnostic meta-learning algorithm (MAML) [11]. By model-agnostic detection, we mean that the model considers all named entity spans as the same class and focuses only on identifying their location without considering their specific entity types.

Once the named entity spans are detected, they are passed to a separate typing model, that performs entity typing i.e. assigning a specific entity type to each named entity span using the BIOES scheme (we shortly explain this scheme in Appendix B. This avoids the "*O*" prototype assignment, since the approach only aims to **locate** the named entity spans in the input sequence, without assigning any entity type labels to the tokens that are not part of any named entity span. Therefore, there is no need to learn a prototype for the "*O*" label, and the model can focus on learning the representations of the named entity spans only.

Few-shot entity typer
Now, for assigning the entities, the paper uses the same approach as standard prototypical networks [3]. Although, they bring a modification to ProtoNet, and propose MAML-ProtoNet, to address the disparity between entities coming from source domains and target domains. Unlike ProtoNet, which solely relies on the similarity of input spans with the representations of prototypes from the support examples, MAML-ProtoNet goes further. It also uses the support examples to adjust the shared embedding space of spans and prototypes.

Specifically, MAML-ProtoNet clusters representations of spans belonging to the same entity class together, while dispersing representations of spans from different entity classes. This improves the prediction accuracy by aligning spans within the same class and creating robust boundaries between spans of different classes.

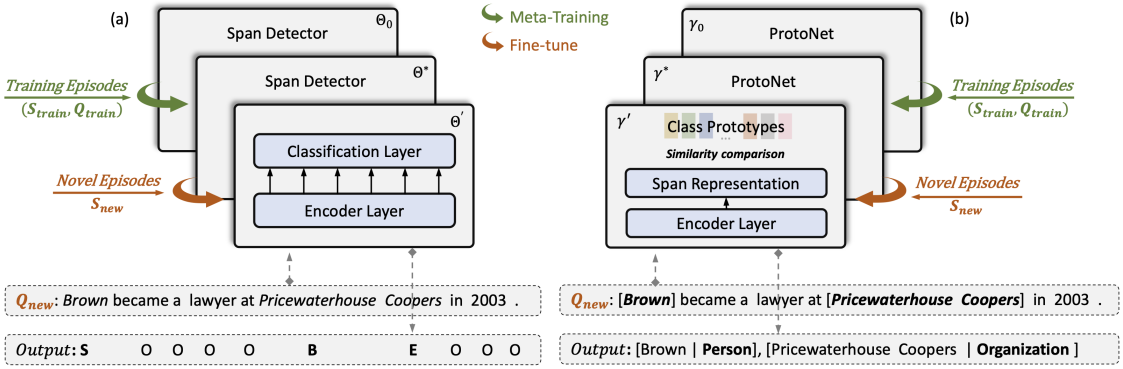


Figure 4: Decomposed Meta-Learning framework from Ma et al. [4]. "(a) entity span detection with parameters Θ and (b) entity typing with parameters γ. Two modules are trained independently using $(\mathcal{S}_{train}, \mathcal{Q}_{train})$. At meta-test time, these two modules firstly are finetuned on the support set $\mathcal{S}_{new}$, then given a query sentence in $\mathcal{Q}_{new}$, the spans detected by (a) are sent to (b) for entity typing."

5 Results

This section will present the results that have been achieved in the course of this semester. Three parts will be presented: the first one will be an overall result of the ProtoBERT model vs. the DML model with various N-way K-shot configurations and see the their influence on performance. Secondly, we will show a more detailed demonstration of the best configuration for both ProtoBERT and DML on Few-NERD and PICO. Lastly, we will see how well the best ProtoBERT and DML can perform in a configurable shot context, i.e. when a 5-way 1-shot model has been trained, how will it perform in a 5-way 5-shot context?

The experiments have been run on DTU Compute 'gpuv100' HPC clusters' [12]. We performed our evaluations on the 6th version of Few-NERD and the latest available PICO Corpusversion, publicly available on their respective GitHubs. The DML model trains ≈6hrs on a Tesla V100 GPU. Whereas the ProtoBERT took ≈2hrs. We are aware that Ma et al. were able to perform the training for DML in ≈1hr on a Tesla V100 GPU, however, our GPUs are shared with other students too, thus, retaining us from allocating too many GPUs for our own usage.

In a similar fashion as the testing on Few-NERD [5], we sampled 20000, 1000 and 5000 episodes for train, dev and test respectively, in PICO. Both data sets are sentence-levelled, this means that the model receives episodes as one sentence and not as a paragraph.

Disclaimer

It is also important to underline that the PICO Corpus only has 26 entities compared to 66 fine-grained entity types in Few-NERD. It was therefore required to limit ourselves to 7-way training in order to have a non-overlapping distribution of entity types between train, dev and test. Additionally, the distribution of the number of entities per class in PICO is sparse - as seen in Appendix D - therefore, when performing N-way K-shot experiments with $N \geq 7$ and $K \geq 2$ we have to discard the four outcome classes "*iv-cont-q1*", "*iv-cont-q3*", "*cv-cont-q1*" and "*cv-cont-q3*". Finally, in the Few-NERD data set, DML was trained on 5000 epochs. In the PICO data set context, this created considerable overfitting issues. Therefore, DML was only trained on 600 epochs for the PICO corpus.

5.1 The N-way K-shot results

Table 2 and 3 report the results of the DML and ProtoBERT approaches. It can be seen that ProtoBERT outperforms DML up to 4.6 F1 scores in the best setting (5 way 1-shot).

DML	
N-way K-shot	**F1-score**
5-way 1-shot	64.7 ± 0.3
5-way 5-shot	**71.5 ± 0.5**
10-way 1-shot	58.6 ± 0.4
10-way 5-shot	68.1 ± 0.1

ProtoBERT	
N-way K-shot	**F1-score**
5-way 1-shot	38.8 ± 1.5
5-way 5-shot	**58.8 ± 0.4**
10-way 1-shot	32.4 ± 0.8
10-way 5-shot	52.9 ± 0.4

Table 2: F1 scores with standard deviations on Few-NERD for inter settings. The best results are in **bold**. The standard deviation was extracted through 5 seed evaluations.

Additionally, the ProtoBERT model on the PICO Corpus outperforms ProtoBERT on Few-NERD by 0.3 F1 scores. Of course, standard deviation could reveal something different, with more time the results could differ.

Overall, ProtoBert outperforms DML in the PICO Corpus, when looking at Table 3. An exception is visible for 7-way 2-shot and 3-shot. But DML remains sovereign in performance

DML	
N-way K-shot	**F1-score**
5-way 1-shot	13.27 ± 3.96
5-way 5-shot	**31.7 ± 30.08**
6-way 1-shot	14.73 ± 7.99
7-way 1-shot	10.63 ± 3.70
7-way 2-shot	19.57 ± 6.99
7-way 3-shot	15.83 ± 14.60
7-way 5-shot	12.17 ± 5.67

ProtoBERT	
N-way K-shot	**F1-score**
5-way 1-shot	17.93 ± 3.72
5-way 5-shot	**35.9 ± 30.25**
6-way 1-shot	15.97 ± 0.87
7-way 1-shot	10.3 ± 5.22
7-way 2-shot	16.97 ± 2.89
7-way 3-shot	19.8 ± 11.10
7-way 5-shot	20.67 ± 14.25

Table 3: F1 scores on PICO Corpus for inter settings. The best results are in **bold**. The standard deviation was not extracted due to time constraints.

on the Few-NERD data set as presented by Ma et al. [4]. We however wish to note that DML showed increased F1-scores when presented with more data (From an F1-score og 7.8 to 10.1 in 7-way 1-shot scenario), instead of 20000 for the training data set like in Few-NERD context. This could suggest that with more time and investigation DML results could show more potential. On the other hand, ProtoBERT showed decreased performance when presented with more training data (from 12.6% to 12.3% F1-score).

It is clear to see that increasing the number of shots for any model, yields better performance. This is expected, the more support examples the model gets, a better embedding and understanding it will get on the given task. A clear linear relationship also emanates from the number of classes the model is trained on. A 5-way 1-shot model will perform better than a 7-way (or 10-way) 1-shot model. The addition of extra classes to a given task increases the complexity for the model. It has more classes to learn and to recognise. These results are visualised in both data sets. Note that it is reasonable to compare performance on both data sets, taken into account that both data sets portray a similar distribution of sentence length as seen in Appendix C. It is interesting to notice that when starting to train a model in 7-way 1-shot manner on the PICO Corpus (Table 3) and increasing the number of shots, the F1-scores increase when adding one more shot but then act differently depending on the model. For DML, the F1-score starts to decrease again. On the other hand, ProtoBERT's F1-scores reach a plateau for 2-shot and 3-shot and then increases again. One could argue that the plateau between 2-shot and 3-shot could be a threshold for the ProtoBERT model to overstep in order to improve its performance (going from 14.8 to 16.0 once given 5-shots).

Moreover, we note that for both data sets, the 5-way 5-shot configuration yields the best results. In the context of the PICO Corpus (which is of interest in this report), we present how the inter-classes across the three coarse-grained domains ("Intervention & Control", "Participants" and "Outcomes") have been split between train, dev and test in Table 4. We also provide a bar plot in Figure 5 of the distribution of unique annotations across the three PICO coarse-grained entities between the three data sets (Train, dev and test). As seen by the bar plots, there are not enough unique annotations to fulfill the 20000, 5000 and 1000 respective samples for train, dev and test. This is due to class imbalance. To tackle it, we oversampled the minority classes when making the episodes for the different splits and discarded the classes if they had less than $2 * K$ examples available. Therefore, a model could have overfitted to an entity like "*iv-cont-sd*" by seeing the same annotation several times. Making the model more inclined to annotate numerical values.

Furthermore, the models did not get to train or validate on the "Intervention & Control"

fine-grained entities. However, it is interesting to notice that there might be underlying correlations between the class "intervention" and "intervention-participants". Likewise for "control" and "control-participants". Henceforth, the model can benefit from the possible underlying relationships between entities across domains (i.e. "Intervention & Control", "Participants" or "Outcomes) and perform beter if correlation is present.

	Intervention & Control	**Participants**	**Outcomes**
Train		- total-participants - eligibility - ethnicity - age - control-participants	- iv-cont-mean - cv-cont-sd - iv-cont-q3 - cv-bin-percent - iv-bin-percent - cv-cont-q1 - iv-cont-median
Dev		- intervention-participants	- iv-cont-sd - cv-cont-median - iv-cont-q - iv-bin-abs - outcome
Test	- intervention - control	- location	- cv-cont-q3 - cv-bin-abs - cv-cont-mean

Table 4: Random inter-splitting of classes between data sets in a 5-way 5-shot context.

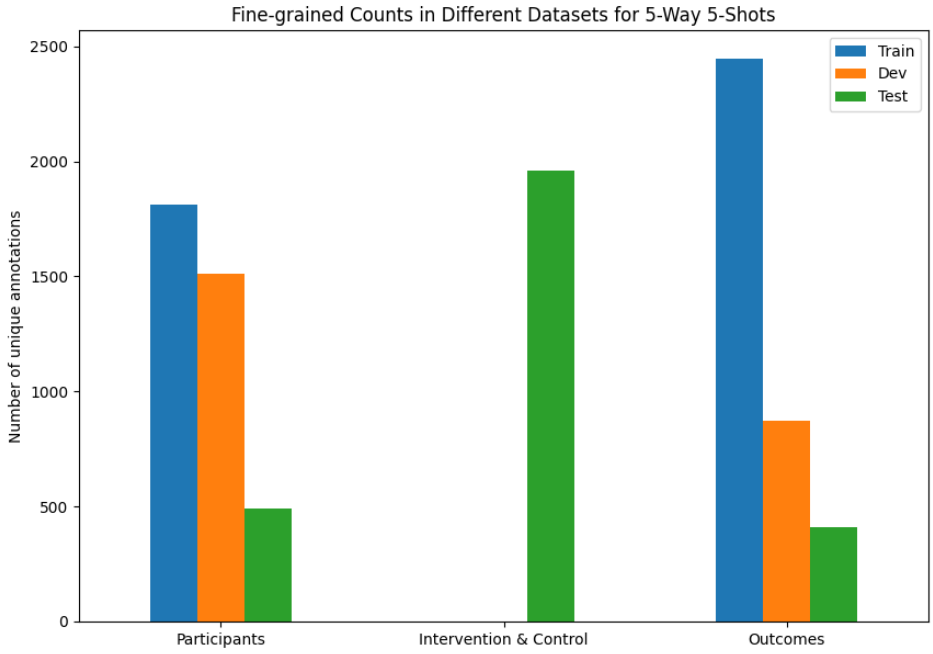
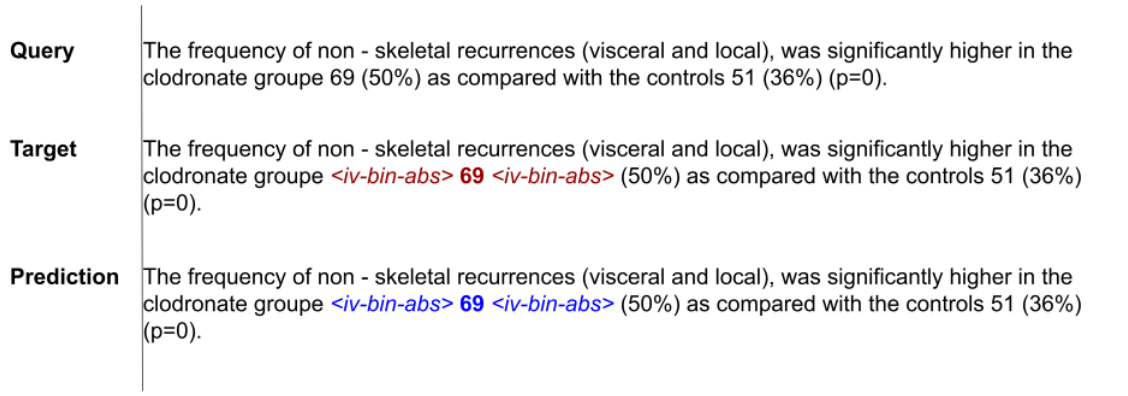
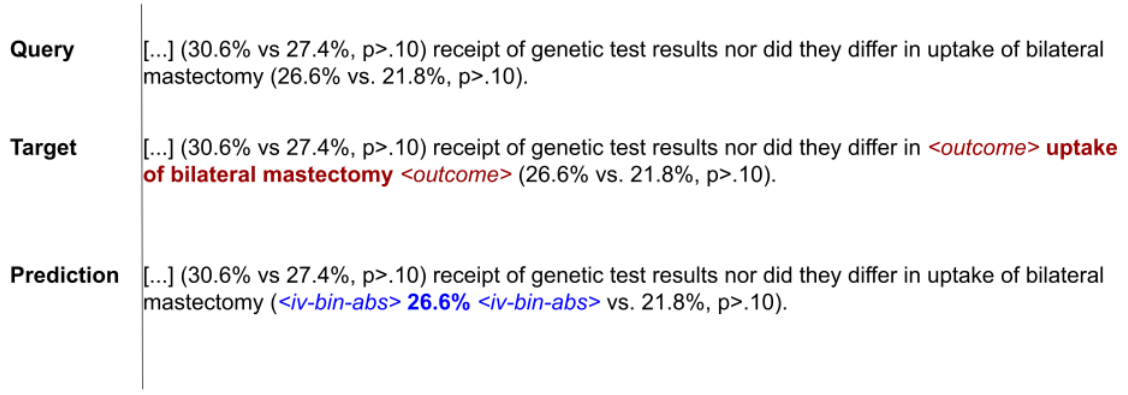


Figure 5: Number of unique annotation examples presented between the three data sets for the 5-way 5-shot scenario.



(a)



(b)

Figure 6: (a) A correct prediction made by the 5-way 5-shot DML model on the PICO Corpus validation set with the entity *iv-bin-abs*. (b) Incorrect prediction made by the same model with the entity *outcome*.

To get a better understanding on how the 5-way 5-shot model performs on the test set, we present two different predictions made by the model during the validation phase in Figure 6a and 6b.

As we can see the model had a lot of entities that involved numbers, the incorrect prediction shown in subfigure 6b could be justified. After being fine-tuned and trained on numerical entities from the "Outcome" domain, we suggest that the model could more enclined in predicting a numerical outcome like "*iv-bin-abs*" instead of a textual outcome.

Furthermore, we wanted to push our experiments further and imagined two contrasting scenarios:

1. How would a 5-way 1-shot model perform if it is given 5-way 5-shot scenario?
2. Conversely, how would a 5-way 5-shot model perform in a 5-way 1-shot task?

In the following part, we will show the results for these experiments. Take into account that the class distribution for the test set in these two scenarios are was at first the same as during the model's training, i.e. like in Table 4 and on a different seed (see Appendix E.

The result for testing **on the same seed** are presented in table 5. By testing on the same seed, we mean that the test was performed with the same entity distribution as presented in the bar plot 5

Model	Trained in an N-way K-shot scenario	Tested in an N-way K-shot scenario	New F1-scores	Old F1-scores
ProtoBERT	5-way 5-shot	5-way 1-shot	54.8	51.7 +/- 27.85
ProtoBERT	5-way 1-shot	5-way 5-shot	15.6	19.0
DML	5-way 5-shot	5-way 1-shot	44.9	54.5
DML	5-way 1-shot	5-way 5-shot	28.0	13.8

Table 5: Testing 5-way 5-shot models in 5-way 1-shot scenario (and vice-versa) for ProtoBERT and DML on PICO data set.

On top of this, we also provide a bar plot of the entity distribution for the 5-way 1-shot model in Figure 7.

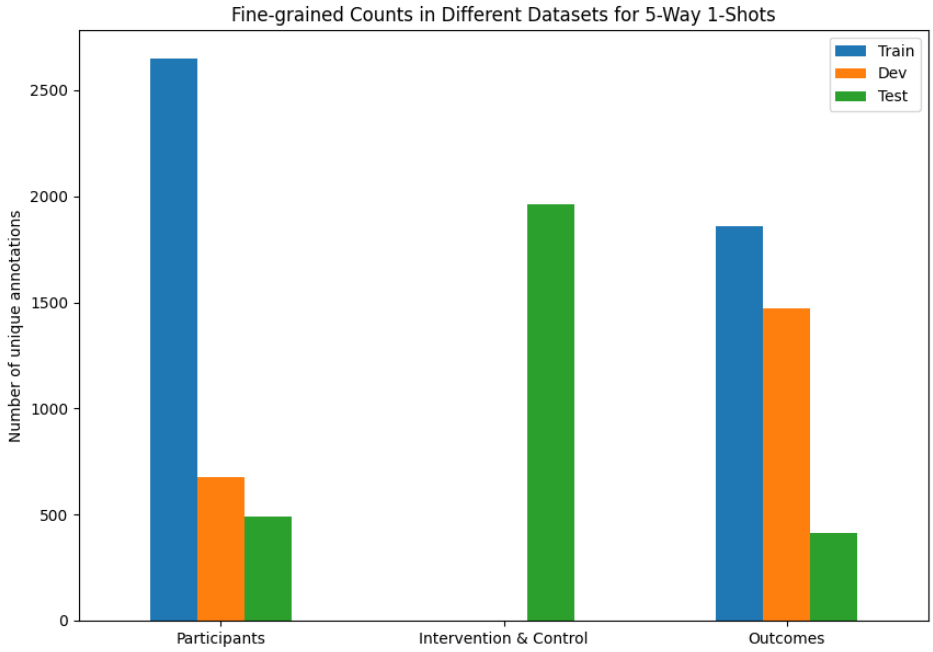


Figure 7: Number of unique annotation examples presented between the three data sets for the 5-way 1-shot scenario.

At first, we notice the clear decreasing performance of a 5-way 5-shot model once presented with a 5-way 1-shot scenario, across both models. Which makes sense, the model has been trained to see at least 5 examples for each new class, and now it only gets one. We also see that the DML shows an increasing performance one being trained on only 5-way 1-shot and then tested in a 5-shot scenario - going from a 13.8 F1-score to 28.0. This makes sense, the model gets more support examples to be able to generalize the new task at hand. However, we notice that this is not the case for ProtoBERT model, going from 59.1 to 54.8. We suggest that this could be due to the combination of two things. At first, the 5-way

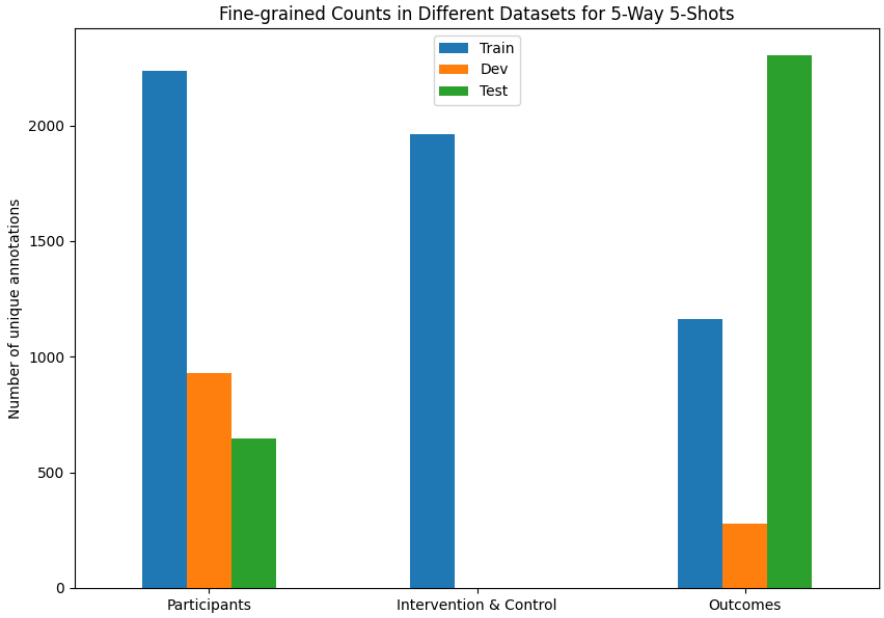
1-shot ProtoBERT gets its hyperparameter fine-tuned more in the "Participants" entity as seen in Figure 5, which is an entity that is way more present in the development set for the 5-way 5-shot setting. Since the 5-way 1-shot ProtoBERT already trained a lot in the Participants entity, as seen in Figure 7, this could indicate that the model might have overfit to the "Participants" class and has less focused on capturing the underlying relationships across the three domains (like with "intervention-participants" and "intervention" in the coarse-grain type "Intenvention & Control"). The second reason one could argue one is the complexity of ProtoBERT. Being much less complex than DML, it might be more inclined to overfit when presented with an imbalanced distribution of data.

To leverage randomness and generality, the same experience was made but with a different entity distribution seed. In a similar manner as before, we present the results in table 6 below.

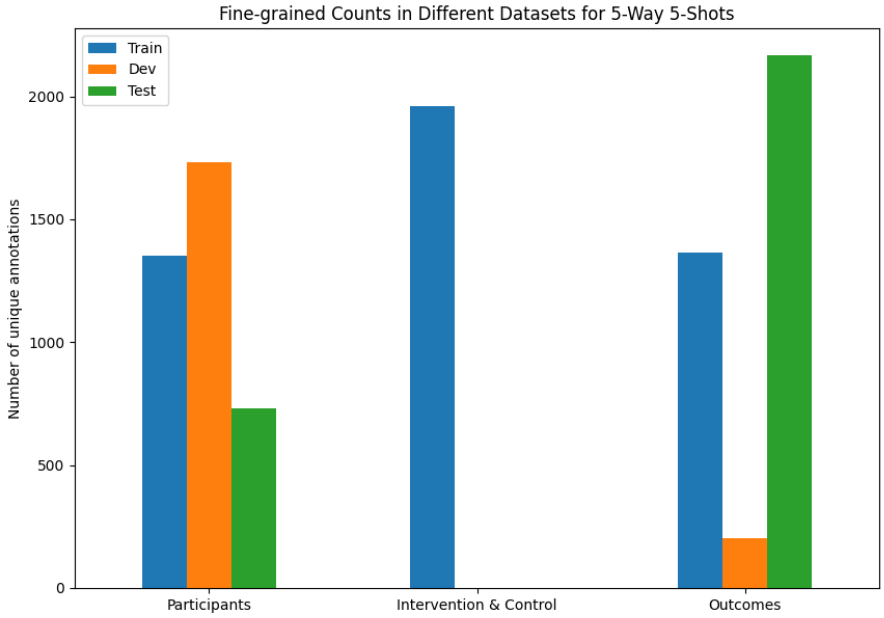
Model	**Trained in an N-way K-shot scenario**	**Tested in an N-way K-shot scenario**	**New F1-scores**	**Old F1-scores**
ProtoBERT	5-way 5-shot	5-way 1-shot	48.6	59.1
ProtoBERT	5-way 1-shot	5-way 5-shot	28.6	19.0
DML	5-way 5-shot	5-way 1-shot	37.0	54.5
DML	5-way 1-shot	5-way 5-shot	18.2	13.8

Table 6: Testing 5-way 5-shot models in 5-way 1-shot scenario (and vice-versa) for ProtoBERT and DML on PICO data set on a different seed.

With this entity distribution amongst the data sets, the F1-scores highlight an expected behaviour. When going from 5-shot to 1-shot, the F1-score increases and conversely for the opposite scenario (F1-score decreases when given more shots). There are several argumentations for this behaviour. First thing we notice from looking at the bar plots in Figure 8, the test set in the new seed has only examples in "Outcomes" and "Participants" and not in "Intervention & Control" in comparison from before. Therefore, one could argue that from Figure 5 and 7 the models are finally leveraging their training and fine-tuning from the examples they were presented in "Outcomes" and "Participants". We speak of possible data leakage of the test set and entity class training overlapping from the new seed. Which was of course expected. Nevertheless, this experiment could be considered as a real-life scenario in the field of NER for bio-medicine.



(a) 5-way 1-shot



(b) 5-way 5-shot

Figure 8: Number of unique annotation examples presented between the three data sets for the two different scenarios on a different seed.

6 Discussion

In the following section a discussion of our results will be presented including suggestions for future work. We were able to see in the first place, that overall ProtoBERT performed best on the PICO Corpus (Table 3. In a similar setting as Few-NERD (i.e. same amount of data) and not taking into account classs distribution imbalance, ProtoBERT had better generalisation capabilities. This could be due to the over-complexity of DML, when presented with a less complex data set like PICO (26 classes vs. 66 on Few-NERD), the model could become too specialized to the PICO training data and memorize specific patterns or noise in it, resulting in poorer generalization to unseen data. However, it is important to note that the relationship between the number of classes, model complexity, and overfitting is not always straightforward. Another reason to a poor performance of DML can also be due to the quality of the data. In contrast to the Few-NERD data set, the PICO Corpus paper does not explore the knowledge correlation among types. As we mentioned earlier, knowledge transfer [7] is crucial for few-shot learning. It would be interesting to explore the knowledge correlation among all the entity types. We could observe that entity types sharing identical coarse-grained types typically have larger similarities, resulting in easier knowledge transfer. And on the other hand, some of the fine-grained types across coarse-grained types share little correlations due to distinct contextual features. This would be an ideal setup from the perspective of knowledge transfer. Therefore, in the future, one could investigate this on the PICO Corpus.

Another reason for poor performance could also be due to the class distribution amongst data sets, as similarly as we could to Few-NERD (inter) [5] we assigned roughly 50% fine-grained types of all the 3 coarse-grained types to $\mathcal{E}_{train}$, 26% to $\mathcal{E}_{dev}$ and 23% to $\mathcal{E}_{test}$, respectively. As a note, Few-NERD assigned 60%, 20% and 20% respectively. With more time, we would have tried tweaking these amounts and explore if the coarse information would affect the prediction of new entities. One other important factor to take into account in these results is the lack of standard deviation and mean F1-scores to most of the experiments. Due to the complexity of DML and its training time, we were not able to draw a standard deviation to our results in time. This would have highlighted more accurate results and a better idea of our models performance.

We were however able to get an idea of which configuration brought the best results in the PICO Corpus setting. The 5-way 5-shot scenario seemed like an interesting set-up to generalise on the PICO Corpus. With more time, we would have wished to perform more hyperparameter tuning to increase the F1-scores. Exploring the change of the number of epochs; we saw that it was important to drastically increase when working with a less large data set like PICO. The Decomposed Meta-Learning approach has a lot of hyperparameters to tune, which offers a lot of future work projects to improve its performance. The same applies for the ProtoBERT model. Additionally, as underlined in the results, increasing the training size for DML, presented better results. One could imagine DML offering similar results as ProtoBERT with more data. Furthermore, the 5-way 1-shot models tested on 5-way 5-shot setting showed quite some expected behaviour. I.e. increasing F1-score when acquired with more support examples. This shows that in the context of a literature review in bio-medicine for example, one could use a pre-trained ProtoBERT model and use it as a few-shot learner to improve literature review classification with transfer learning. With more time, this experiment would have also been applied to other N-way models, with increasing or decreasing shots.

7 Conclusion

In this study, we extensively investigated the application of Meta-Learning techniques, particularly the Decomposed Meta-Learning (DML) method, and ProtoBERT for Few-Shot Named Entity Recognition (NER) tasks.

Our empirical results highlighted the effectiveness of the ProtoBERT model when tested on the PICO Corpus, showing robust generalization capabilities. However, the DML model performance seemed to be affected by its complexity, which may be a subject for further investigation. Overfitting due to over-specialization to the training data is one possible explanation, revealing an interesting trade-off between model complexity and generalizability. A finer investigation into how model complexity, the number of classes, and overfitting interplay could be a future research direction.

Interestingly, we observed that the data quality and the class distribution could be another influencing factor for model performance. Investigating the knowledge correlation among entity types, as it pertains to knowledge transfer in few-shot learning, could be interesting to look at. Refining the class distribution to match the data set characteristics could also potentially improve model performance.

While our results provided insightful trends, the absence of standard deviation and mean F1-scores due to computational and time constraints, suggests that future studies could further validate these findings with more detailed statistical analysis.

One exciting result was the improved performance of the DML model with increased training size, suggesting that with enough data, DML could potentially rival or surpass the ProtoBERT performance. This prospect opens the door for future studies with larger data sets. However, over-complex is not always the right solution. The ProtoBERT models performance showed attractive results to the PICO Corpus NER taks.

In the 5-way 5-shot scenario, both ProtoBERT and DML models performed well, indicating potential in real-world applications such as bio-medical literature review tasks. It seems promising to further explore the impact of different parameters on model performance, including the number of epochs, the number of shots and other finetuning hyperparameters.

In conclusion, this study offered a comprehensive view of Meta-Learning methods' efficacy in Few-Shot NER tasks. It paves the way for further investigations into model complexity, data quality, class distribution, and hyperparameter tuning. It also opens up new potential avenues for utilizing Few-Shot learning models for practical bio-medical NER tasks. Future work could focus on fine-tuning these methods for specific tasks, further improving their effectiveness.

Bibliography

[1] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Ling uistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[2] Michael Tänzer, Sebastian Ruder, and Marek Rei. "Memorisation versus Generalisation in Pre-trained Language Models". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 7564–7578. DOI: 10.18653/v1/2022.acl-long.521. URL: https://aclanthology.org/2022.acl-long.521.

[3] Jake Snell, Kevin Swersky, and Richard Zemel. "Prototypical Networks for Few-shot Learning". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/cb8da6767461f2812ae4290eac7cbc42-Paper.pdf.

[4] Tingting Ma et al. "Decomposed Meta-Learning for Few-Shot Named Entity Recognition". In: *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 1584–1596. DOI: 10.18653/v1/2022.findings-acl.124. URL: https://aclanthology.org/2022.findings-acl.124.

[5] Ning Ding et al. "Few-NERD: A Few-shot Named Entity Recognition Dataset". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 3198–3213. DOI: 10.18653/v1/2021.acl-long.248. URL: https://aclanthology.org/2021.acl-long.248.

[6] Sachin Ravi and Hugo Larochelle. "Optimization as a Model for Few-Shot Learning". In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=rJY0-Kcll.

[7] Aoxue Li et al. "Large-Scale Few-Shot Learning: Knowledge Transfer With Class Hierarchy". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 7205–7213. DOI: 10.1109/CVPR.2019.00738.

[8] Faith Mutinda et al. "PICO Corpus: A Publicly Available Corpus to Support Automatic Data Extraction from Biomedical Literature". In: *Proceedings of the first Workshop on Information Extraction from Scientific Publications*. Online: Association for Computational Linguistics, Nov. 2022, pp. 26–31. URL: https://aclanthology.org/2022.wiesp-1.4.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks*. 2017. arXiv: 1703.03400 [cs.LG].

[10] Ning Ding et al. *Prototypical Representation Learning for Relation Extraction*. 2021. arXiv: 2103.11647 [cs.CL].

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1126–1135. URL: https://proceedings.mlr.press/v70/finn17a.html.

[12] DTU Computing Center. *DTU Computing Center resources*. 2022. DOI: 10.48714/DTU.HPC.0001. URL: https://doi.org/10.48714/DTU.HPC.0001.

A Meta-Learning Computational Complexity

One of the main computational challenges in meta-learning stems from the bi-level optimization structure inherent in its design. Specifically, this complexity arises from the nested nature of the optimization procedure, which often involves computing gradients through an inner optimization process, leading to the computation of second-order derivatives or Hessian-vector products.

If we consider the Model-Agnostic Meta-Learning (MAML) algorithm described in subsection 4.2 as an example. MAML aims to learn an initialization Θ that can be quickly fine-tuned for new tasks. It does this by using a two-level training process.

In the inner loop, the model parameters are updated given randomly sampled episodes $(\mathcal{S}_{train}^{(i)}, \mathcal{Q}_{train}^{(i)}, \mathcal{Y}_{train}^{(i)})$ and performing *inner-update*, giving us a new set of parameters Θ_i':

$$\Theta_i' = U^j(\Theta; \alpha, \mathcal{S}_{train}^{(i)}), \tag{4}$$

Where U^j denotes the j-step gradient updates with learning rate α to minimize the loss from the support set. We then proceed to evaluate the Θ' parameters on the query set $\mathcal{Q}_{train}^{(i)}$ and perform *meta-update*:

$$\min_{\Theta} \sum_i L(\Theta_i'; \mathcal{Q}_{train}^{(i)}). \tag{5}$$

Equation 5 involves a second order derivative. This results in computation complexity of $O(n^2)$ for each loop, where n is the number of parameters. However, the paper utilizes the first-order approximation to alleviate this computational bottlenieck:

$$\Theta \leftarrow \Theta - \beta \sum_i \nabla_{\Theta_i'} L(\Theta_i'; \mathcal{Q}_{train}^{(i)}). \tag{6}$$

where β denotes the learning rate used in meta-update.

B BIOES scheme

The BIOES tagging scheme is used for annotating and representing entities. It is a tag assigned to a token in a sequence that indicates wheter it belongs to an entity and, if so, the position and type of the entity. The tags are represented as follows:

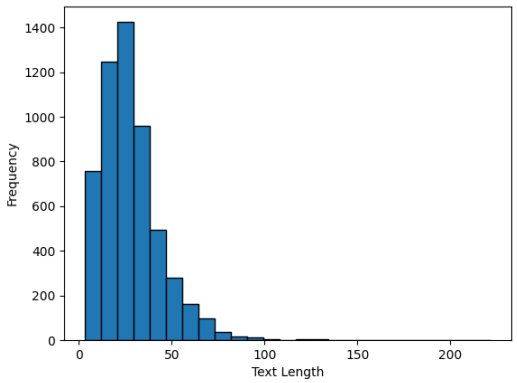
- B (Beginning): Marks the first word of an entity within a sequence. It indicates the beginning of an entity.
- I (Inside): Marks words that are inside an entity, following the initial word marked with the B tag.
- O (Outside): Marks words that do not belong to any entity. It represents non-entity words or tokens.
- E (End): Marks the last word of an entity within a sequence. It indicates the end of an entity.
- S (Single): Marks entities that consist of a single word, where the entity starts and ends within the same token.

An example can help visualize how it works:

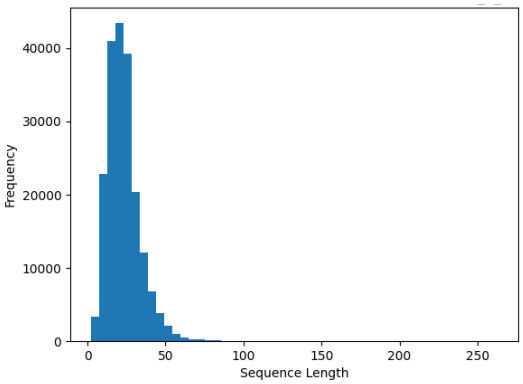
$$Simon_{[B]}Larsen_{[I]} \text{ studies at} DTU_{[B]}Campus_{[E]}.$$

C Sequence Length Distribution between Few-Nerd and PICO

As we can see from Figure 9, the sequence length is centered around 25 for both data sets.



(a) Sequence length distribution for Few-NERD training set



(b) Sequence length distribution for PICO overall

Figure 9: Sequence length distribution

D Number of examples per class in PICO

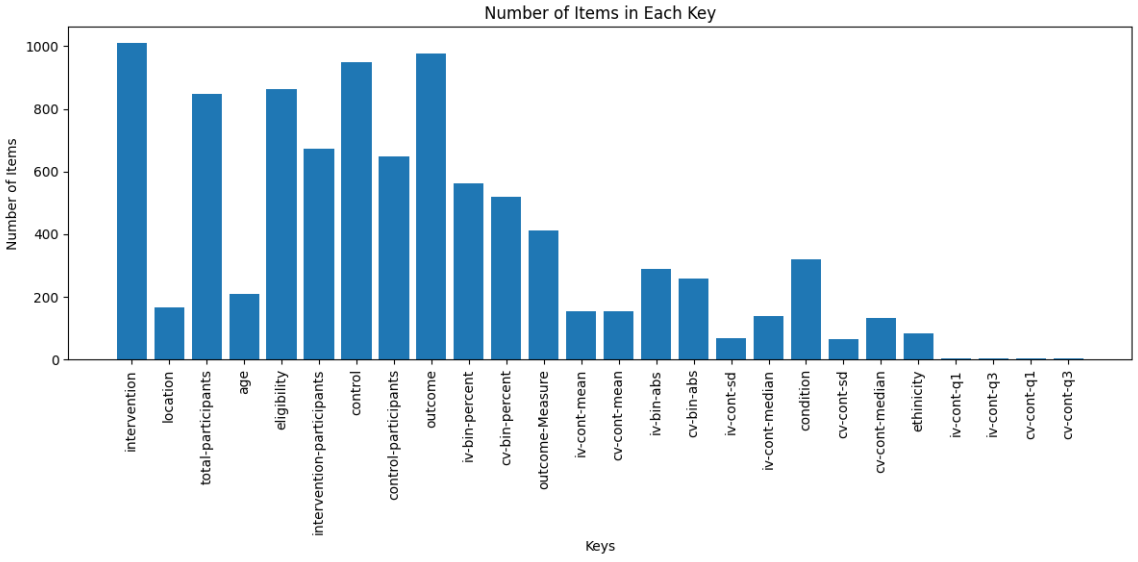


Figure 10: Number of entity distribution in the PICO Corpus

E Unique class annotations distribution for a different seed

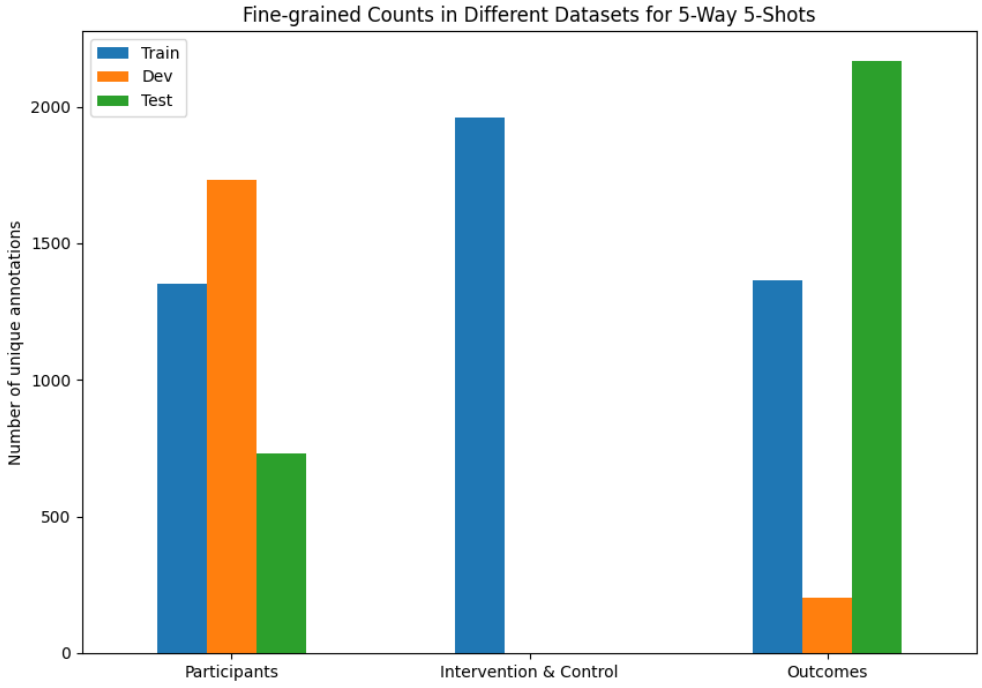


Figure 11: Number of unique annotations across PICO coarse-grained entities between training, dev, and testing data sets