

Titre: Entity Typing with Natural Language Inference for Fine-Grained
Title: Named Entity Recognition

Auteur: Simon Durocher
Author:

Date: 2023

Type: Mémoire ou thèse / Dissertation or Thesis

Référence: Durocher, S. (2023). Entity Typing with Natural Language Inference for Fine-Grained Named Entity Recognition [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. <https://publications.polymtl.ca/53392/>
Citation:

 **Document en libre accès dans PolyPublie**
Open Access document in PolyPublie

URL de PolyPublie: <https://publications.polymtl.ca/53392/>
PolyPublie URL:

Directeurs de recherche: Amal Zouaq
Advisors:

Programme: Génie informatique
Program:

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

**Entity Typing with Natural Language Inference for Fine-Grained Named
Entity Recognition**

SIMON DUROCHER

Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Mai 2023

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé :

**Entity Typing with Natural Language Inference for Fine-Grained Named
Entity Recognition**

présenté par **Simon DUROCHER**

en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

a été dûment accepté par le jury d'examen constitué de :

Daniel ALOISE, président

Amal ZOUAQ, membre et directrice de recherche

Michel DESMARAIS, membre

ACKNOWLEDGEMENTS

I profoundly thank my research director Amal Zouaq for continually realigning my work to achieve the desired objectives while believing in me throughout the project.

I also thank the Digital Research Alliance of Canada for providing me with the resources needed to achieve all my experiments.

Finally, I thank my family and friends for supporting me in their own unique ways, which I will always remember.

RÉSUMÉ

Le centre d'attention de ce mémoire est la *reconnaissance d'entités nommées à fin degré de granularité* (FgNER), qui consiste à détecter les mentions d'entités nommées dans des textes en anglais et à classer chacune d'entre elles avec un type précis provenant d'une taxonomie. Cette taxonomie décrit des types "fins" qui sont plus complexes à assigner aux mentions que les types traditionnels de la reconnaissance d'entités nommées.

Dans ce mémoire, notre premier objectif est de proposer un modèle d'inférence en langue naturelle (NLI) basé sur un modèle de langue pré-entraîné pour effectuer la tâche de typage d'entités à fin degré de granularité (ET) données en entrée. Nous proposons des patrons qui se basent sur les types pour améliorer l'inférence. Notre deuxième objectif est d'intégrer notre modèle d'ET dans un modèle complet de reconnaissance d'entités nommées (FgNER) tout en conservant des propriétés d'adaptabilité.

Nous proposons les trois intégrations suivantes: un modèle de détection de mentions d'entités nommées suivi de notre modèle d'ET, un modèle de reconnaissance d'entités nommées FgNER combiné avec notre modèle d'ET et enfin une approche d'apprentissage par transfert utilisant les poids de l'encodeur et du décodeur de notre modèle d'ET pour les poids d'un modèle de FgNER.

Notre recherche démontre que notre modèle de typage d'entités est comparable aux approches de l'état de l'art avec nos patrons de NLI. Nous avons également intégré avec succès notre modèle d'ET dans un modèle encodeur-décodeur de FgNER.

ABSTRACT

The centre of attention of this thesis is *Fine-grained Named Entity Recognition* (FgNER), which consists of detecting named entities in English texts and classifying each of them with a fine-grained type from a taxonomy. This taxonomy is composed of fine-grained types that are harder to assign to named entities than traditional types from the Named Entity Recognition task.

In this thesis, our first objective is to train a Natural Language Inference (NLI) model based on a pre-trained language model to achieve the fine-grained Entity Typing (ET) task. We propose templates based on types to improve inference. Our second objective is to integrate our best ET model inside a FgNER model while keeping the complete system adaptable.

We propose the three following integrations: using Named Entity Detection followed by ET, using a FgNER model combined with ET and finally a transfer learning from our ET model encoder and decoder’s weights to the weights of a FgNER model.

Our research shows that our ET model is comparable to state-of-the-art approaches when using our NLI templates. We also successfully integrated ET in an encoder-decoder FgNER model.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
RÉSUMÉ	iv
ABSTRACT	v
TABLE OF CONTENTS	vi
LIST OF TABLES	ix
LIST OF FIGURES	xi
LIST OF SYMBOLS AND ACRONYMS	xii
LIST OF APPENDICES	xiii
CHAPTER 1 INTRODUCTION	1
1.1 Elements of the Problem	1
1.2 Research Objectives	3
1.3 Thesis Outline	3
CHAPTER 2 BACKGROUND AND LITERATURE REVIEW	5
2.1 Fine-grained Named Entity Recognition and Other Relevant Tasks	5
2.2 Language Modelling with Transformers	7
2.2.1 The Tokenization Process	7
2.2.2 Attention	8
2.2.3 Encoders and Decoders	8
2.2.4 BERT	9
2.2.5 RoBERTa	10
2.2.6 BART	10
2.3 Design of Tagging Schemes	10
2.4 The Traditional Transformer-based NER Architecture	11
2.5 Prompt-based Techniques	11
2.6 State-of-the-Art Fine-grained Named Entity Recognition	12
2.6.1 Importance of Static Word Embeddings	12
2.6.2 Tagging Schemes for Fine-grained Named Entity Recognition	13

2.6.3	Conditional Random Fields	13
2.6.4	Biaffine Classifier	14
2.6.5	PL-Marker	14
2.6.6	Template Approaches for Fine-grained Named Entity Recognition . .	16
2.6.7	LightNER	17
2.7	State-of-the-Art Entity Typing	20
2.7.1	Masked Language Modeling for Entity Typing	20
2.7.2	Prompt Learning for Entity Typing	21
2.7.3	Natural Language Inference for Entity Typing	21
CHAPTER 3	OVERVIEW	24
3.1	Architecture Overview	24
3.1.1	Named Entity Detection Model	24
3.1.2	Entity Typing Model	26
3.1.3	Fine-grained Named Entity Recognition Model	27
3.2	Dataset	27
3.3	Evaluation Metrics	29
CHAPTER 4	NATURAL LANGUAGE INFERENCE FOR FULLY-SUPERVISED ENTITY TYPING	34
4.1	Supervision from General Natural Language Inference	34
4.2	Natural Language Inference Templates for Entity Typing	34
4.3	Natural Language Inference Datasets for Entity Typing	37
4.4	Training with Natural Language Inference Statements	38
4.5	Entity Typing with Natural Language Inference in Evaluation	39
4.6	Taxonomic and Example Statements Ensemble Ranking	40
4.7	Experimental Results	42
4.7.1	Comparative Results of Entity Typing with Natural Language Infer- ence Model	42
4.7.2	SotA Entity Typing Comparison	48
CHAPTER 5	INTEGRATING ENTITY TYPING FOR FINE-GRAINED NAMED ENTITY RECOGNITION	52
5.1	Named Entity Detection with Entity Typing	52
5.1.1	Adapting the Dataset for Named Entity Detection	52
5.1.2	Named Entity Detection with Entity Typing During Evaluation . . .	53
5.1.3	Pruning Named Entities from Type Rankings	53

5.2	Fine-grained Named Entity Recognition with Entity Typing	53
5.2.1	Adapting LightNER to Few-NERD	54
5.2.2	Independent Named Entity Recognition and Entity Typing	54
5.2.3	Transfer Learning of Entity Typing through the Pretrained Language Model	57
5.3	Experimental Results	57
5.3.1	Models with ET Integrations	57
5.3.2	Results of ET Integrations	58
5.3.3	SotA in Fine-grained Named Entity Recognition	60
CHAPTER 6	DISCUSSION	64
CHAPTER 7	CONCLUSION	66
7.1	Summary of our work	66
7.2	Limitations	66
7.3	Future Research	67
REFERENCES		69
APPENDICES		74

LIST OF TABLES

Table 2.1	NLI example.	7
Table 2.2	Noising transformations used for pretraining BART.	10
Table 2.3	Tagging schemes. ORG and PER are used as abbreviations for organization and person , respectively.	11
Table 2.4	Natural Language Inference templates for Entity Typing. X is the sentence represented by a sequence of tokens, while $x_{i:j}$ is a subsequence of the sentence from the token x_i to the token x_j . Moreover, $M(t)$ is the natural representation of a type t	22
Table 3.1	Few-NERD taxonomy, composed of a two-level hierarchy. A total of 8 coarse-grained types and 66 fine-grained types are defined.	29
Table 3.2	Few-NERD datasets.	30
Table 3.3	IOB2 Span-strict matching.	32
Table 3.4	IOB2 Coarse-grained matching.	32
Table 3.5	IO Fine-grained matching.	33
Table 4.1	Multi-Genre Natural Language Inference training dataset.	34
Table 4.2	Natural Language Inference templates for Entity Typing. All templates involve type-related natural tokens or tokens like "Mysterio" that will necessarily be seen after the first epoch of training.	36
Table 4.3	Entity Type verbalizers for Natural Language Inference statements.	36
Table 4.4	ET-Few-NERD datasets. NLI pair examples contain correct and incorrect pairs, where there is one incorrect pair per correct pair in training. In validation and testing, all 66 type-related NLI pairs are evaluated for taxonomic and example statements.	38
Table 4.5	Ensemble Reranking Example.	41
Table 4.6	Entity Typing performances on ET-Few-NERD-Small.	44
Table 4.7	ET performances on ET-Few-NERD-SUP. For ET, fine-grained matching is equivalent to accuracy.	50
Table 4.8	Zero-shot performances on ET-Few-NERD-SUP. "S" denotes the use of self-supervised training.	51
Table 5.1	Semantic-aware answer space.	54
Table 5.2	Named Entity Recognition and Entity Typing ensemble.	57

Table 5.3	Parameters of models involved in FgNER integrations. Small is for Few-NERD-Small and SUP is for Few-NERD-SUP. BERT-CRF-NER follows the same hyperparameters as BERT-CRF-NED. Additional hyperparameters for the LightNER architecture follow the study [1].	58
Table 5.4	Results of different ET integrations on Few-NERD-Small. ET is used as a shorter name for our best ET model bart-large-taxo-ex , while NEE stands for NER and ET Ensemble and NEP for Named Entity Pruning.	59
Table 5.5	Training parameters of PL-Marker. For a single sentence, all possible spans are split into different groups. The group size corresponds to the number of named entity span markers inside a single group. The sentence is forwarded with a single group at a time for all the existing groups, covering all named entity spans.	61
Table 5.6	Fine-grained Named Entity Recognition performances on Few-NERD-SUP of our SotA models and our Entity Typing integration of bart-large-taxo-ex . ET is used as a shorter name for our best ET model bart-large-taxo-ex , while NEE stands for NER and ET Ensemble and NEP for Named Entity Pruning.	61
Table A.1	Performances of other models on Few-NERD-SUP. ET is used as a shorter name for our best ET model bart-large-taxo-ex , while NEE stands for NER and ET Ensemble and NEP for Named Entity Pruning.	74

LIST OF FIGURES

Figure 1.1	Example of the FgNER task.	1
Figure 1.2	Example of taxonomy adaptation.	2
Figure 2.1	Decomposition of the task of FgNER.	6
Figure 2.2	Overview of relevant tasks.	7
Figure 2.3	The Transformer architecture.	9
Figure 2.4	Biaffine classifier. Feed Forward Neural Network (FFNN) start and end produce a start and end span representation vector, respectively.	14
Figure 2.5	Pack-Levitated Markers. We show only the first group of many that would be packed at the end of the sentence. The representation of a possible span i is achieved with the marker tokens $[O_i]$ and $[/O_i]$, which correspond to the start and end of a possible named entity span. Moreover, $^{(s)}$ and $^{(e)}$ represents the start and end, respectfully.	15
Figure 2.6	LightNER architecture.	18
Figure 3.1	Overview.	25
Figure 3.2	Architecture with independent Named Entity Detection and Entity Typing.	26
Figure 3.3	Entity Typing integrations for Fine-grained Named Entity Recognition.	28
Figure 4.1	NLI with RoBERTa.	40
Figure 4.2	NLI with BART.	40
Figure 4.3	Taxonomic and Example Ensemble.	42
Figure 4.4	Performance of different Pretrained Language Models during Entity Typing training. They both use taxo and dep as NLI statements.	45
Figure 4.5	Performance difference of using MNLI during ET training. They both use taxo and dep as NLI statements.	46
Figure 4.6	False positives before and after training. Fine types are sorted in ascending order of their verbalizer's word count. The frequencies are normalized by the total amount of false positives produced by the corresponding model. The base model is bart-large-mnli	47
Figure 4.7	Performance of different Natural Language Inference statements during Entity Typing training.	49
Figure 5.1	Architecture for independent Named Entity Recognition and Entity Typing	55

LIST OF SYMBOLS AND ACRONYMS

BERT	Bidirectional Encoder Representation from Transformers
BPE	Byte-Pair Encoding
CRF	Conditional Random Fields
ET	Entity Typing
Few-NERD	Few-shot Named Entity Recognition Dataset
FgNER	Fine-grained Named Entity Recognition
IOB	Inside-outside-beginning
LightNER	Lightweight generative framework for prompt-guided low-resource NER
LITE	Language Inference-based Typing of Entities
MNLI	Multi-Genre Natural Language Inference
NED	Named Entity Detection
NEE	NER and ET Ensemble
NEP	Named Entity Pruning
NER	Named Entity Recognition
NLI	Natural Language Inference
NLP	Natural Language Processing
PLET	Prompt Learning for Entity Typing
PLM	Pretrained Language Model
RoBERTa	Robustly optimized BERT approach

LIST OF APPENDICES

Appendix A	Appendix	74
------------	--------------------	----

CHAPTER 1 INTRODUCTION

The increasing amount of unstructured information has led to the development of various natural language processing methods. Fine-grained Named Entity Recognition (FgNER) has been developed to help capture the semantic structure of a text, which is partly associated with its named entities, concepts and relations between them. FgNER focuses on finding named entities in each sentence and classifying them into predefined fine-grained categories, as shown in Figure 1.1. It is often part of an information extraction process that is valuable for various Natural Language Processing (NLP) tasks.

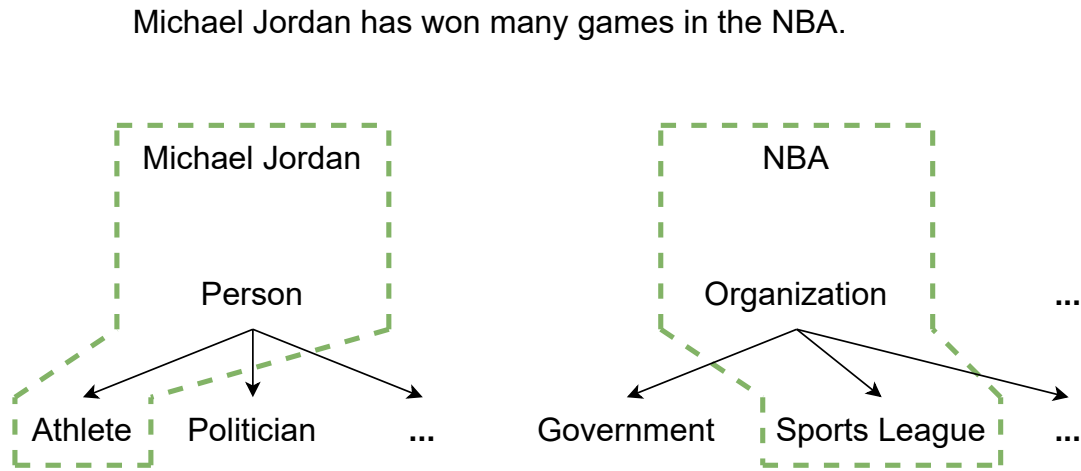


Figure 1.1 Example of the FgNER task.

1.1 Elements of the Problem

One of the main challenges of FgNER is the substantial amount of types in which a named entity can be classified. Like in most machine learning problems, the task can either be done in a low-resource setting with few-shot learning or in a fully-supervised setting where a large amount of data is available. Being able to deal with either of these settings is interesting because various emerging real-life scenarios can be covered. Hence, finding a solution that works well in both cases is hard.

The problem of FgNER or even Named Entity Recognition (NER) in general has been, for a long time, approached by classifying each word either with a specific type (defined in a taxonomy) or as outside of a named entity. Architectures built around this paradigm have used dedicated layers for this classification, where some weights are directly associated with

a specific type. Doing this has resulted in a model that is not as flexible as desired for some of the existing real-world settings. For example, we might want to perform an adaptation of a trained FgNER model to a dataset where types are structured differently, as shown in Figure 1.2. But to do so, these label-specific classifiers normally need complete retraining over the new dataset and taxonomy.

ESPN's Michael Jordan documentary debuted in 2020.

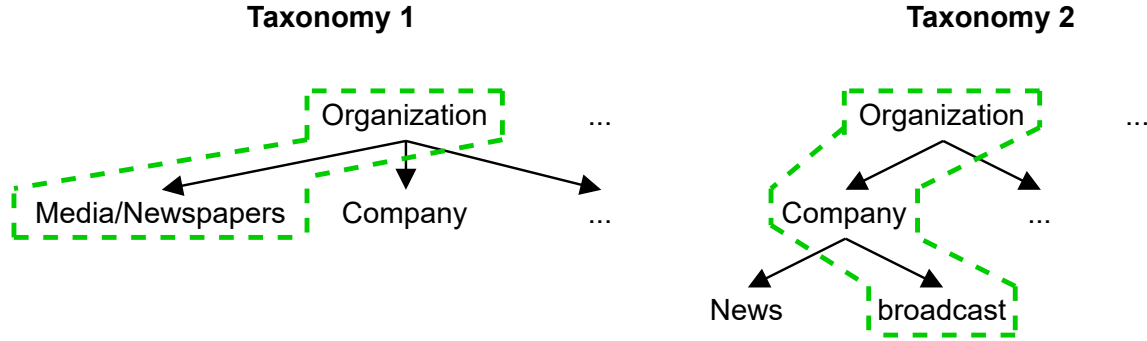


Figure 1.2 Example of taxonomy adaptation.

More recently, prompt-based approaches have emerged as paradigms to leverage the knowledge of Pretrained Language Models (PLMs) mainly by modifying the input. Different NLP prediction tasks can be transformed into a textual form and cleverly reuse the MLM pretraining as a strategy to model the probability of a text. For example, the simple task of sentiment classification requires to either predict a sentence as positive or negative. The input can be transformed to "[Sentence]. This sentence is [MASK].", where "[MASK]" would prompt the PLM for probable tokens mapped to "positive" and "negative" and then be compared to solve the task. This way, the architecture achieving the task remains simple, compared to other approaches that would add different additional task-specific layers. These models perform well in few-shot scenarios. Some of the best methods that achieve few-shot learning in NER are often behind in performance when evaluated in a fully supervised setting. Understanding how to improve the performances of these types of models becomes important. In this context, some approaches in Entity Typing (ET) have been leveraging prompt-based techniques to achieve better results than fine-tuning the models over fully-supervised settings [2]. ET predicts fine-grained types given the sentence and named entities as input. The prompt-based approach for ET involves Natural Language Inference (NLI), where different logical statements can be created regarding ET. While ET is directly related to FgNER, to our knowledge, there are no techniques that use these prompt-based ET models for FgNER.

This could potentially be enough to close the gap between a prompt-based approach and a traditional fine-tuning approach in fully-supervised FgNER.

1.2 Research Objectives

With all the specific problems derived from fully-supervised FgNER mentioned above, some research to validate the application of different promising strategies needs to be done. In this work, we first try to understand the impacts of SotA research in ET and then try to integrate it into a fully-supervised FgNER model. Hence, we take the direction of prompt-based approaches through the use of NLI for ET. We try to answer the following research questions regarding NLI-based ET:

1. Can Natural Language Inference strategies improve fully-supervised ET for a large dataset?
 - 1.1. Can we achieve Entity Typing with Natural Language Inference over a fully supervised dataset?
 - 1.2. What are new strategies in Natural Language Inference that can improve Entity Typing performance?

Afterwards, the use of our developed ET system in the broader task of FgNER remains important. This leads to the following question:

2. How can we integrate Entity Typing to achieve adaptable fully-supervised Fine-grained Named Entity Recognition?

These objectives will guide the rest of this work, where answering the first question will help us achieve our second objective.

1.3 Thesis Outline

In this thesis, we present in Chapter 2 a literature review relevant to FgNER. This includes recent important approaches for representing and understanding the text. It also includes different SotA strategies for either representing named entity spans or typing these named entities.

In Chapter 3, we show a general architecture and an overview of our different strategies to achieve FgNER, which integrates models able to correctly detect named entities and models for fully-supervised ET.

Since our approach involves the use of an ET model, we explain in Chapter 4 the design of different NLI-based ET strategies. Furthermore, we compare the performances of our models with other SotA ET models.

To achieve FgNER, we propose in Chapter 5 different integrations of our best NLI-based ET model in a FgNER architecture. This entails the development of strategies that capture and leverage the abilities of each module to enhance the overall performance. We subsequently test our models against SoTA models on the fully-supervised FgNER setting.

Finally, we discuss in Chapter 6 and 7 the results regarding our research objectives obtained from our proposed strategies. We also present the limitations of these strategies and then show some future directions for research valuable to fully-supervised FgNER.

CHAPTER 2 BACKGROUND AND LITERATURE REVIEW

In this chapter, we present different key components of recent advancements related to achieving the task of FgNER. We first define the important tasks that take part in this study. We continue with the description of the recent approaches in language modelling and understanding through the use of the Transformer architecture. Some general NER tagging schemes are presented, followed by a general approach to NER and a description of prompt-based methods since many FgNER models use this paradigm. Then, we discuss some SotA techniques in FgNER and in ET.

2.1 Fine-grained Named Entity Recognition and Other Relevant Tasks

In this section, we present how the specific setting of the FgNER task differentiates itself from the traditional setting of NER and define other important tasks related to FgNER.

Traditional Named Entity Recognition focuses on finding named entities and assigning each of them a type from a limited set of types. These coarse-grained types include **person**, **organization**, **location** and **miscellaneous**. These categories have been popularized primarily because of the CoNLL-2003 NER shared task [3]. One interesting coarse type is the **miscellaneous** type because it gathers other types not categorized in either **person**, **organization** or **location**.

Formally, the common setting of the NER task is described as a sequence labelling problem. A given dataset is formed of M examples $\{(\mathbf{X}_m, \mathbf{Y}_m)\}_{m=1}^M$ where a single sentence $X = [x_n]_{n=1}^N$ and labels $Y = [y_n]_{n=1}^N$ with N being the number of tokens. Each value y is from the set of labels $\mathcal{Y}$, which is composed of types from a typeset $\mathcal{T}$ and an outside label designating a "non-entity". For a new example i , the task is to be able to predict the correct $\mathbf{Y}_i$ for a sentence $\mathbf{X}_i$.

When dealing with traditional NER, all named entity spans are flat. In other words, there is no overlap between spans in terms of their starting and ending position. In contrast, nested NER relaxes this rule by allowing spans inside another. For example, the sentence "The Washington Monument is an obelisk-shaped building." contains the named entity "Washington Monument", which itself contains "Washington". Because our centre of attention in this study is about recognizing and fine-typing named entities, we consider flat NER as our setting, even though nested NER is a generalization of flat NER. In the case where there are nested entities, flat NER datasets will always take the larger span as the correct named

entity.

In FgNER, named entities are classified from a wide and deeper range of types. Most common datasets provide a strict taxonomy of types, where named entities can be assigned a fine-grained type. Taxonomies that describe named entities are defined with a hierarchical structure. More precisely, the most common taxonomies, such as Ontonotes [4], FIGER [5] and Few-NERD [6] follow a tree structure. In these datasets, each named entity is part of only one leaf of the taxonomy. This leaf is a fine type which can only have one direct parent associated with a coarser category. Throughout this research, we use a two-level taxonomy composed of coarse-grained and fine-grained types.

This results in paths of type like **person-athlete**. The addition of a fine type is made when enough examples that correspond specifically to this type appear in the dataset. Some datasets will just disregard those named entities since they already have enough coverage of frequent types with their taxonomy. However, similar to **miscellaneous** in NER, some taxonomy like Few-NERD [6] will define a type, called **other**, inside coarse-grained categories such that these remaining named entities can still be annotated.

As shown in Figure 2.1, FgNER can also be decomposed into two tasks. The first task is Named Entity Detection (NED), which outputs a named entity span given the sentence as input. The second task is Entity Typing. ET is the identification of the named entity type once its spans are known and can be solved as an independent task, taking the sentence and entity mentions as input and generating entity types as output. Assigning a fine-grained type indirectly involves identifying the correct coarse-grained type.

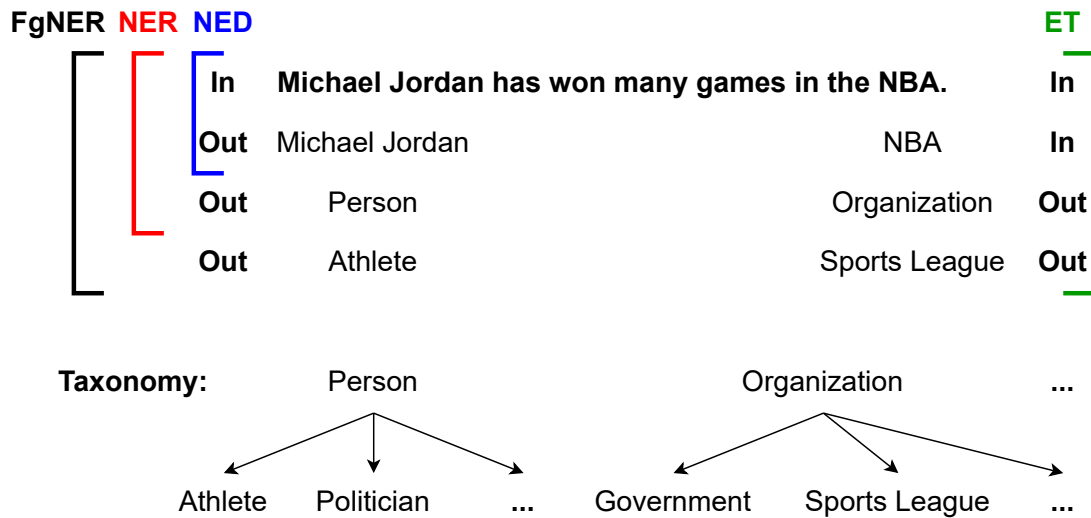


Figure 2.1 Decomposition of the task of FgNER.

As shown in Figure 2.2, one approach useful for performing ET is the task of Natural Language Inference (NLI). NLI is the task of validating the entailment of a hypothesis given some context. In other words, the goal is to assert the logical inference residing in a statement composed of a premise and a hypothesis. As we will see in this work, NLI is a useful task for entity typing. Table 2.1 shows possible statements designed for NLI, where final predictions can be used as predictions for the task of ET.

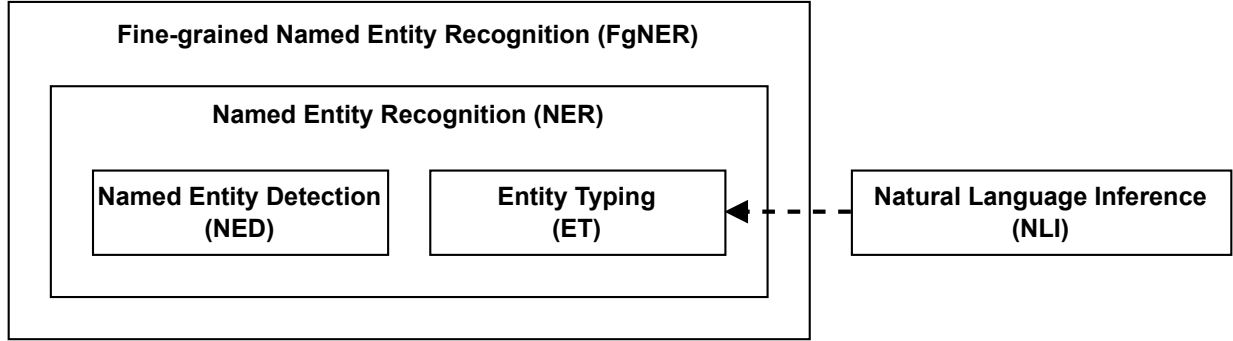


Figure 2.2 Overview of relevant tasks.

	NLI Statement	
	Premise	Hypothesis
Entailment	Micheal Jordan has won many games.	Micheal is a person.
No entailment	Micheal Jordan has won many games.	Micheal is a location.

Table 2.1 NLI example.

2.2 Language Modelling with Transformers

Regardless of the setting in which FgNER is performed, capturing the context in which these named entities occur remains essential. This is why, as in many other NLP tasks, language modelling can directly contribute to our task. The Transformer architecture [7] is one of the main models to capture a powerful understanding of natural language. We present here the key elements of this architecture while introducing different general models that have been notorious in the domain of NLP and that we also use throughout our research.

2.2.1 The Tokenization Process

Before training any Transformer-based models, a vocabulary $\mathcal{V}$ needs to be constructed. The algorithm used to build vocabularies of models such as RoBERTa [8] or BART [9] is called

Byte-Pair Encoding (BPE) [10]. Given a large dataset composed of sequences of characters, the algorithm uses an iterative process to merge the most common pattern composed of characters or character sequences into its own unique representation. This process continues until the desired size of the vocabulary is achieved. The tokenization of the dataset can then be completed, which ultimately compresses the dataset's size and enables mapping through the unique id of each token from the text to the Transformer architecture.

2.2.2 Attention

The attention mechanism is an important component in all Transformer-based architectures. More specifically, it enables any part of the input to attend to other parts and even itself, which results in a complexity of N^2 , where N is the length of the sequence. To achieve this, each token has its own randomly initialized embedding, which is transformed with a positional encoding. This results in x_i , which represents both the token and its position in the sentence. Moreover, the key K , the query Q , and the value V matrices are created to produce the attention $A \in \mathbb{R}^{N \times N}$ as shown below:

$$A(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

where $K, Q \in \mathbb{R}^{d_k}$ and $V \in \mathbb{R}^{d_v}$. As for d_k and d_v , they are predefined dimensions that are set to be large enough for the mechanism to capture valuable information. The scaling factor $\sqrt{d_k}$ helps decrease the potential large values that the dot product between these vectors can create. For example, two randomly initialized vectors of size d_k with a mean of zero and a variance of one would produce a vector with a variance of d_k when combined with the scaled dot product. The Transformer-based model also uses different attention heads, called multi-head attention. Each one learns to attend differently to the input, which can be combined to form a better representation of the sentence itself. Overall, the attention mechanism is a relatively good strategy to capture long-range dependencies, even though the complexity, as mentioned previously, is the main limitation of its capabilities.

2.2.3 Encoders and Decoders

The encoder and decoder are two major parts of the Transformer architecture, as shown in Figure 2.3. The encoder has the role of converting the input into embedding vectors that are representative of the information contained in the input. Achieving a dense representation of the input requires the combination of the multi-head attention mechanism through a feed-forward network.

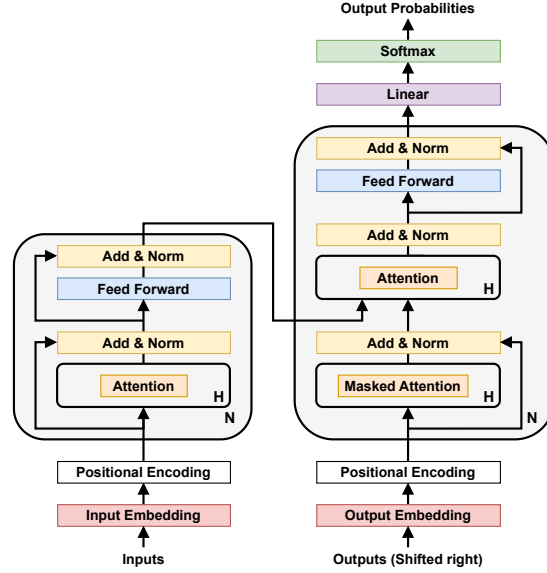


Figure 2.3 The Transformer architecture.

For the decoder, a similar architecture is used to capture a representation of the output and input given to the model. The goal of the decoder is to maximize the probability of the next output token given the encoded input and a representation of the previous output. To this end, vectors representing the next output tokens are masked from the attention of the previous output tokens during training. Moreover, a right shift of the output embeddings is made because the first output produced by the decoder should only involve the output from previous positions.

Solely based on the different paradigms present in the Transformer architecture, several models have emerged in the last few years that are now widely used.

2.2.4 BERT

BERT is an encoder architecture that uses two self-supervised learning objectives over a large amount of textual data. The first objective involves Mask Language Modeling (MLM), which is to predict a random token that has been switched with the "[MASK]" token inside a sentence. The model also performs the Next Sentence Prediction (NSP) pretraining objective, where, given two sentences as input, the model must predict if the second sentence follows or not the first sentence.

2.2.5 RoBERTa

The Robustly Optimized Bert Pretraining approach (RoBERTa) [8] is an amelioration of the BERT encoder. First, it demonstrates that the importance of the NSP objective can be insignificant in many tasks. Replacing this with better and larger training of the task of MLM is done instead. RoBERTa involves the MLM pretraining objective with the dynamic masking of tokens. Instead of fixing a mask to generate before the training for all examples, the masked token is chosen at random from epoch to epoch on a given example. This solves the problem of underfitting from BERT and provides better results with fewer data.

2.2.6 BART

Based on BERT and the decoder-only model GPT [11], BART [9] can produce accurate results on both discriminative and generative tasks. BART proposes a more general pre-training scheme than BERT that also includes some generation tasks, which results in better performances than BERT or GPT. The model follows the complete Transformer architecture, which is both the encoder and decoder. Instead of only masking tokens independently like BERT, BART incorporates other noising transformations and trains the language model to remove the noise produced by these transformations. Possible transformations are presented in Figure 2.2.

Noising Transformation	Concept	Example
True sentence	ABC.DEF.	Alice loves books. Bob hates them.
Token Masking	A_C.DE_.	Alice loves [MASK]. Bob hates [MASK].
Token Deletion	AC.DE.	Alice books. Bob hates.
Sentence Permutation	DEF.ABC	Bob hates them. Alice loves books.
Document Rotation	F.ABC.DE	them. Alice loves books. Bob hates
Text Infilling (# missing tokens?)	A_.DEF_.	Alice [MASK]. Bob hates them [MASK].

Table 2.2 Noising transformations used for pretraining BART.

2.3 Design of Tagging Schemes

Considering NER as a sequence labelling problem, the simplest form of annotation is the Inside-Outside (IO) tagging scheme, as shown in Table 2.3. The IO tagging scheme only indicates if a token is inside **I** a named entity or outside **O**. As for the Inside-Outside-Beginning

2 (IOB2) tagging scheme, the additional tag B indicates the beginning of each named entity. This enables the model to clearly identify the span boundaries of two named entities when they are consecutive and are of the same type.

	The	NBA	Chicago	Bulls	are	known	because	of	Michael	Jordan
IO	0	ORG	ORG	ORG	0	0	0	0	PER	PER
IOB2	0	B-ORG	B-ORG	I-ORG	0	0	0	0	B-PER	I-PER

Table 2.3 Tagging schemes. **ORG** and **PER** are used as abbreviations for **organization** and **person**, respectively.

2.4 The Traditional Transformer-based NER Architecture

One of the most common NER models is the SeqTagger [12]. This model is available in the HuggingFace Transformers library through BertForTokenClassification [13]. It applies a label classification for each input token. The model forwards tokens through BERT to create a hidden representation for each of them. Then, each of these hidden representations is sent to a linear layer dedicated to classification. A softmax can then be used to independently produce a probability distribution over the labels for each word.

2.5 Prompt-based Techniques

Any recent PLM stemming from the Transformer architecture has been trained on large amounts of data through self-supervised objectives. These self-supervised objectives are optimized for the probability $P(X; \theta)$ where X is the text and θ is the model. Normally, a fully supervised approach would optimize for the probability $P(Y|X; \theta)$, where Y is a desired output for a specific objective. However, some tasks will not provide enough annotations, which is often the case when dealing with large typesets as in FgNER. Hence, prompt-based approaches are redesigning X towards the specific objective, so a PLM can use its text modelling capabilities to benefit the task at hand.

A common procedure is to design a specific template $T(X)$ that will create a relation between the input X and some answer Z produced by the PLM. Afterwards, the goal is to correctly map the generated answer Z to a desired output Y .

For example, we can consider the simple task of sentiment classification, which is to classify a sentence as positive or negative. This task could be designed with the following template: "[Sentence]. The sentiment of this sentence is [MASK]". The model can then consider a

manually curated subset of the model’s vocabulary composed of tokens specifically related to a positive or negative sentiment as an answer to fill the "[MASK]" token. The token predicted by our model will thus directly represent if the sentence is positive or negative.

Instead of generating a mapping between Z and Y , a simpler prompt-based approach directly validates if some predefined answers are probable or not. More precisely, a mapping $M(Y)$ from $Y \in \mathcal{T}$ to a textual representation can be used instead of generating the answer tokens Z . Hence, no generation is necessary for the model and we use the probability $P(T(X; M(Y)); \theta)$ for prediction. For the example regarding the task of sentiment classification described above, a possible template could be: "[Sentence]. The sentiment of this sentence is positive.". Instead of filling a token as done previously, a score would be produced from this template, which can be a value inside the range $[0,1]$ where 1 represents a valid template, meaning that the sentence is probably positive, while 0 would indicate that the sentence is probably negative.

2.6 State-of-the-Art Fine-grained Named Entity Recognition

In this section, we cover different elements that have been beneficial for the task of Fine-grained Named Entity Recognition, which, more than often, come from better designs aimed at the general task of NER.

First, we introduce the importance of word embeddings for FgNER in real-world scenarios. Secondly, we present some powerful techniques to represent types, named entities and the relations between them. This includes tagging schemes that ultimately model the type and span information of named entities when FgNER is framed as a token classification problem. It also encompasses strategies that either design the dependencies between predicted labels or named entity spans themselves, where some have demonstrated great capabilities for FgNER. Finally, we show different prompt-based strategies favourable for low-resource settings, which is often the case for FgNER because of the lack of high-quality annotations when covering large typesets.

2.6.1 Importance of Static Word Embeddings

Even if dynamic contextual embeddings from PLMs have provided significant improvements in NER over the last years, the use of static embeddings is still beneficial for new systems.

GloVe [14] is a common approach for static embeddings, which relies on the co-occurrence of words. It has been shown [15] that some improvement is observed when Glove embeddings are combined with BERT embeddings over the fine-grained dataset Ontonotes 5.0 [16]. Moreover, training static embeddings over a large amount of text from the general domain can make

the embedding useful for domains where context provides little to no information or takes an unusual form like HTML.

In the same train of thought, ET models predicting only from the named entity words have higher F1-scores than models with no access to words of the named entity but only the context in which the named entity takes place [17]. Nonetheless, contextual embeddings help disambiguate between the different possible senses that a named entity could take. It has been shown that separating the modelling of the dynamic and static named entity representations can help improve the performance [17].

2.6.2 Tagging Schemes for Fine-grained Named Entity Recognition

A design which benefits FgNER is the separation of tag information [18]. For example, the tagging **B-facility-goe-park** for "South" in "South Park is located in the United States" can be separated as such: **B**, **facility**, **goe**, **park**. Formally, it is possible to decompose a tag y in its components $\mathbf{C}^y = \{c_k\}_{k=1}^K$, where c_k is the index of a component type k . In the context of the previous example, the components would be $\mathbf{C}^{\text{B-facility-goe-park}} = \{c_1 = \text{B}, c_2 = \text{facility}, c_3 = \text{goe}, c_4 = \text{park}\}$. Since each component of a tag y has its own embedding, a combination of these components is necessary to form the complete tag embedding. A summation of the component's embedding seems to perform better than averaging them [18].

2.6.3 Conditional Random Fields

As presented earlier for traditional NER architectures 2.4, a token classifier like SeqTagger independently produces a probability distribution over the labels for each word. This poses the risk of producing some avoidable mistakes. For example, when using IOB2, the model can possibly generate the tag for entity "Michael Jordan" with "Michael" as **B-person** and Jordan as **B-person** (instead of **I-person**), which is incorrect. This highlights the problem of producing independent predictions.

Conditional Random Fields (CRF) [19] can avoid such mistakes by adding more constraints when generating labels [20]. This involves some dependency between our predictions. Because modelling dependencies between all labels is expensive, a simple strategy is to use a linear-chain CRF. The dependency between labels is minimal with this strategy because it assumes that labels follow a Markov chain of the first order. This only means in our case that a transition between a label prediction y_i given only the last label prediction y_{i-1} is enough to model the whole label dependencies.

The Viterbi algorithm is used for decoding CRFs and comes with a complexity of $O(|\mathcal{Y}|^2 * N)$,

where N is the sentence length and $\mathcal{Y}$ represents the labels. Consequently, dealing with a lot of labels is not ideal because of the quadratic complexity brought by $\mathcal{Y}$. In the architecture of the common token classifier model, the linear-chain CRF would replace the softmax layer.

2.6.4 Biaffine Classifier

There are other NER strategies from directly predicting a label for each word. The biaffine model [21] scores pairs of start and end tokens from separate label classifiers which also include the outside label. First, as shown in 2.4, the model uses different embeddings from the input, which includes BERT for contextual representation and other techniques for word-level and character-level representations. All these embeddings are combined with a bidirectional Long-Short-Term-Memory Network [22] to produce a single representation of the input. A representation is created for all possible named entity starts and ends. The model can then rank all combinations of start-end pairs. Evaluating all combinations enables the model to perform the harder task of nested NER, which is valuable. The traditional task of flat NER is achieved by adding restrictions to the biaffine model's predictions. It has been shown that this system is less affected by the domain and the granularity of types over other NER approaches [21], which means that it can deal well with FgNER.

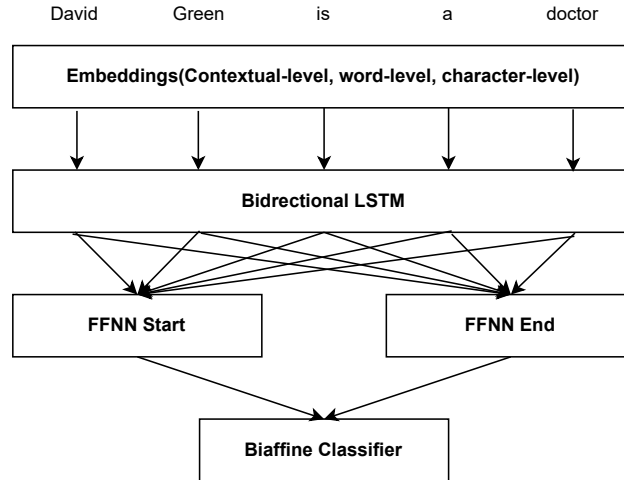


Figure 2.4 Biaffine classifier. Feed Forward Neural Network (FFNN) start and end produce a start and end span representation vector, respectively.

2.6.5 PL-Marker

Packed Levitated Marker (PL-Marker) [23] is another SotA approach that addresses the span representation problem. This approach can keep some dependencies between span pairs by

adding all possible span markers at the end of the token sequence. These levitated markers enable the model to achieve a span-wise representation of the entity type prediction, which has shown some considerable gain in terms of FgNER performance. This representation also makes the model applicable to nested NER. We present in Figure 2.5 PL-Marker for a flat NER setting with RoBERTa as the encoder.

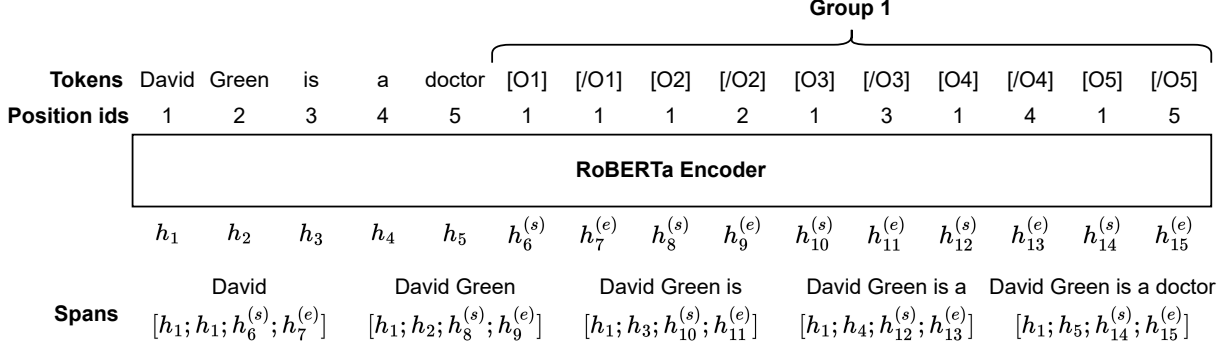


Figure 2.5 Pack-Levitated Markers. We show only the first group of many that would be packed at the end of the sentence. The representation of a possible span i is achieved with the marker tokens $[O_i]$ and $[/O_i]$, which correspond to the start and end of a possible named entity span. Moreover, $^{(s)}$ and $^{(e)}$ represents the start and end, respectfully.

One important strategy from PL-Marker is to assign to each marker the same position embedding of the token they are representing. For example, in Figure 2.5, for the first possible span in the sentence, which is only the token "Michael", its corresponding span marker start ($[O1]$) and end ($[/O1]$) both have the same position id as "Michael", which is 1.

Normally, the strategy of levitated markers would be to add all possible span representations at the end of the sentence. However, the number of possible spans increases rapidly with the length of the sentence. This becomes a problem since the maximum token size of a Transformer-based architecture like RoBERTa might be too low for a long sentence. To alleviate this problem, markers are packed in smaller and position-related groups of size K , resulting in inputs with lengths smaller than the maximum token size.

Another strategy to reduce the complexity produced by a large number of named entity spans is to directly reduce this number. First, PL-Marker limits the possible size of the span to a specific value. Moreover, we can use a simpler and faster model to produce a limited amount of potential spans and then apply PL-Marker over these spans, which has been shown to achieve the same performance with a three-time speed increase.

To help the optimization process during the training phase, PL-Marker initializes each start

marker with the "[MASK]" embedding and the end marker with the "entity" embedding. Instead of a random weight initialization, these relevant embeddings can be leveraged from the RoBERTa pretraining, following a similar objective of a prompt-based approach and providing value to the NER task without the need for a large amount of data.

2.6.6 Template Approaches for Fine-grained Named Entity Recognition

Question Answering Templates for Named Entity Recognition. A study [24] has formulated NER as a Machine Reading Comprehension (MRC) task. More formally, it is possible to form a question Q for each type t of a typeset $\mathcal{T}$. For example, a question like "Which person is mentioned in the text?" can be created for the type **person**. This simple prompting approach would follow the template "[CLS] <X> [SEP] <Q>". An encoder model like BERT is used to produce valuable embeddings for the rest of the model. The model can then produce probability distributions of start P_{start} and end P_{end} for each token in a sentence X . Finally, a binary classifier, comparable to what was done with the biaffine classifier, is used to match the start and end of the most probable $\hat{P}_{start}$ and $\hat{P}_{end}$. Indeed, this methodology can also be applied to nested NER. Since MRC with NER leads to data imbalance from the numerous negative examples, using a dice-loss [25] can decrease the importance of examples with high-confidence predictions.

Template BART. Template BART [26] is an approach developed for NER, which fills a positive template T^+ like "<X> [SEP] < $x_{i:j}$ > is a < t > entity." with every possible span for < $x_{i:j}$ > of X and each type t from a typeset $\mathcal{T}$. Furthermore, a negative template T^- like "< $x_{i:j}$ > is not an entity." is added for training. For the sentence "Michael Jordan has won many games.", a valid positive template would be "Michael Jordan has won many games [SEP] Michael Jordan is a athlete entity.", and a valid negative template would be "Michael Jordan has won many games [SEP] won is not an entity.". We can see from this example that other templates could easily be generated for both positive and negative examples. A technique equivalent to Template-BART but for multiple templates at the same time is done by Multi-template BART. This is achieved by training each template individually followed by combining the model's predictions through entity-level voting. According to the authors, the increase in performances coming from Multi-template BART is because different types of knowledge are captured by each template. It is important to consider, as in other multi-prompt techniques, that this methodology increases the overall complexity when using many templates.

InstructionNER. Instead of using question templates to find named entities, InstructionNER [27] proposes to use a template composed of the context, an instruction, and options. An example of this can be the following: "Sentence: Michael Jordan has won many games. Instruction: please extract the entities and their types from the input sentence, all entity types are in option. Options: athlete, politician, government, sports league, [other types]". The key idea proposed is to formulate a template as a sequence with instructions and options to produce a sequence of entity names and types, all with natural text. This multi-task instruction-based strategy follows a generative framework with models like BART and T5 [28]. BART has difficulty with instruction based pretraining, which is not the case for T5. The problem is caused by the noising scheme of BART, which can lead the model to generate an entire sequence for denoising while T5 has been trained to denoise specific tokens. This is more aligned with the task of filling tokens inside predefined templates.

2.6.7 LightNER

In relation to what has been said about prompt-based approaches, LightNER [1] incorporates different ideas to accelerate the learning process of a NER model, which is valuable for FgNER. Given a sentence, the model will forward it first to the encoder, where some representation can be obtained. At the beginning of the decoding process, the BART decoder is given the encoded sentence and the starting state $\langle s \rangle$. The decoder will then produce tokens autoregressively until all named entities that seem probable are generated. As presented in Figure 2.6, the beneficial properties of LightNER are the use of learnable prompts to guide the attention mechanisms of a frozen encoder-decoder model. Moreover, a semantic answer space helps the adaptation process of the model to another dataset.

Learnable Continuous Prompts. Similar to the technique of Adapters [29], LightNER freezes both the encoder and decoder of the generative model and uses learnable continuous prompts to guide the attention mechanism. For each layer l of the encoder and decoder specifically, two matrices ϕ_v^l and ϕ_k^l are modulating the attention, where ϕ_v^l is concatenated to the value and ϕ_k^l to the key. Each one of them is in $\mathbb{R}^{|P| \times d}$, where $|P|$ is the length of the prompt and d is the size of the hidden representation at layer l . Performance remains fairly constant with different values of P . For the common NER dataset CoNLL-2003, the best value of P is around 10. Overall, these learnable continuous prompts limit the training cost while achieving reasonable performance compared to what a fully supervised model could obtain with a fine-tuned encoder and decoder.

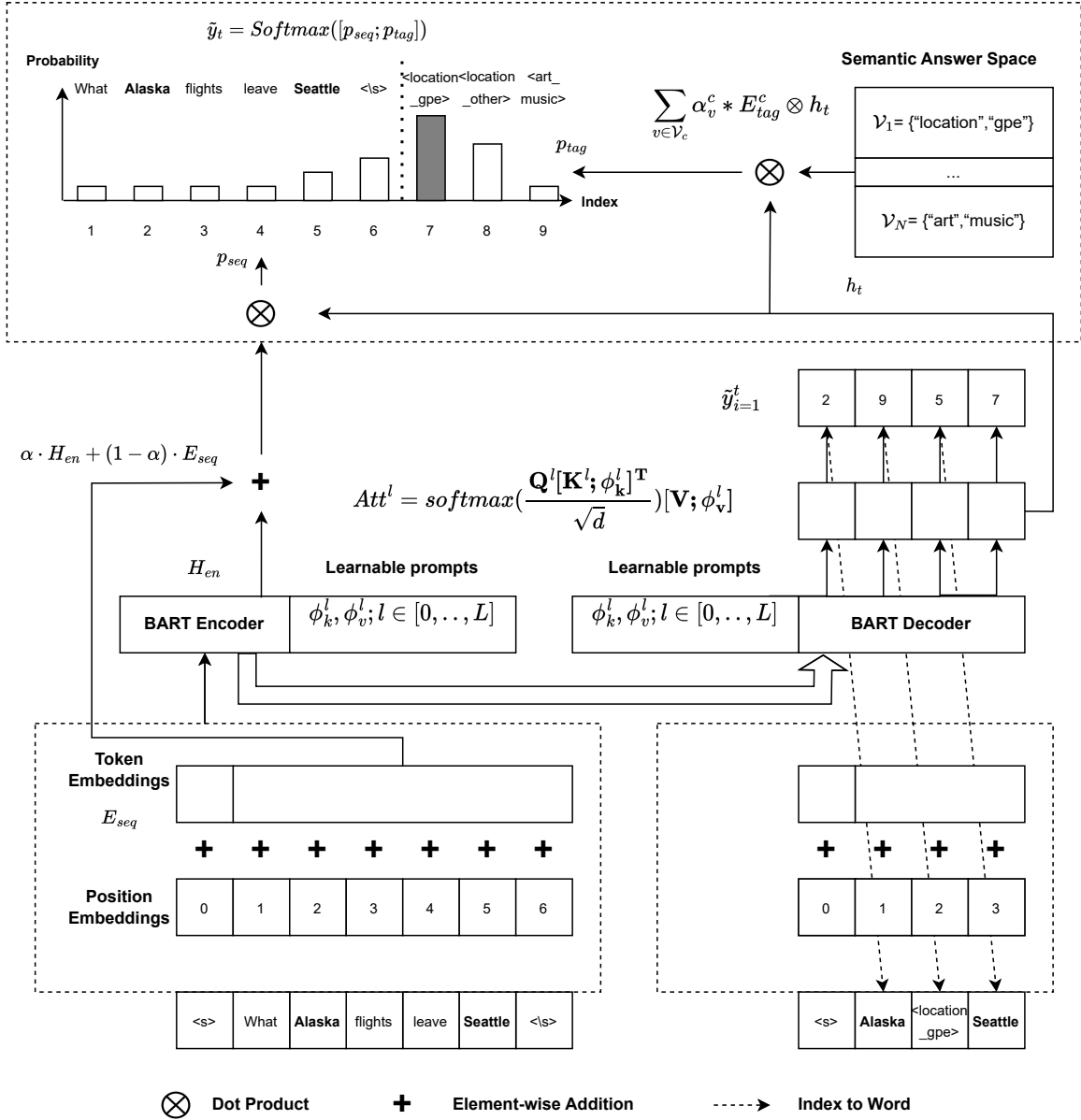


Figure 2.6 LightNER architecture.

Semantic Answer Space. Type-specific parameters require complete retraining when adapting the model to a new typeset. One major component that makes this model adaptable to a different typeset is the semantic answer space. Using a semantic answer space for types allows the model to have domain adaptation capabilities, which is relevant for real-world applications. This is often more valuable in few-shot settings where high-resource domain training is adapted to the types of low-resource domains.

Each type is decomposed into tokens taken from the model’s vocabulary. For example,

Michael Jordan is tagged with the fine type **person-athlete**. The resulting answer space of this specific type is the tokens "person" and "athlete". A semantic answer space is built for the entire typeset. By using type-related tokens, it leverages prior information useful for a few-shot setting and can be tuned to achieve comparable performance to other SotA models in a fully-supervised setting.

The semantic answer space does not take into account the ordering of the words, since a set of words is constructed for each type. However, producing this set implies that words appearing twice in the type label are now only there once. This alleviates a bias of starting with more weight for the redundant word, like "art" in **art-writtenart**, which could have led the model to over-predict this specific type instead of another fine type within the same coarse-grained category **art**.

Generating Predictions. It is important to understand how LightNER does FgNER since this generative framework makes FgNER achievable for encoder-decoder architectures, which is impossible when viewing FgNER only as a token classification problem. We present in Figure 2.6 how the model uses the hidden representation of its encoder and decoder to generate either span or type index predictions.

For the prediction of the start or end of named entity spans, some modifications to the encoder representation are made. More precisely, the final encoder’s representation is reweighted with the token embeddings themselves. This new representation is then combined with the h_t through a cross-product. Given the current generated output state, this gives probabilities over all positions in the sentence and can represent the start or the end of a span.

For the generation of type categories, the use of the decoded output h_t is done. The probability of a type c is obtained by taking a weighted average of each token embedding E^c multiplied by h_t .

Combining both index pointer and entity types probabilities with a softmax is necessary.

In Figure 2.6, the index of **location-gpe** type is generated from the previously generated states, which contain the entity span indexes of "Alaska" and the encoded sentence. A ratio between the number of tokens from the generated span types and the sequence’s length can be defined. The generator will keep producing tokens until the probability of a new index is lower than all other previous predictions or when the maximum length is achieved.

Overall, this model is few-shot friendly because of its prompt-guided attention mechanism and its domain adaptation capability resulting from the use of a semantic answer space. LightNER performances are still lower than a traditional sequence labelling NER model,

mainly because only 2.2% of the overall parameters are fine-tuned compared to a traditional model that trains all its parameters. However, it can still achieve relatively good fully supervised performance compared to other few-shot-friendly models.

2.7 State-of-the-Art Entity Typing

An important part of our study is dedicated to ET, which can be described formally as the prediction of a type t from a typeset $\mathcal{T}$ for every named entity $x_{i:j}$ in a given sentence X . The main difference with NER is that the named entity span in ET is also given as input. Moreover, each named entity is evaluated independently from the other named entities in the sentence.

There exist some ET settings like Ultra Fine-grained Entity Typing (UFET) [30] where no strict hierarchy is provided between type levels. This specific dataset provides different levels of types: coarse, fine to ultra-fine. Ultra-fine types are the most common free-form noun phrases also called headwords. Many of these headwords can be assigned to a single named entity since the goal is to cover the context in which a named entity is presented. The small number of examples per ultra-fine type makes it a specific setting compared to other ET datasets.

Recent approaches in ET mostly use prompt-based methods to leverage the PLM’s knowledge about the type or the task, which is not directly captured by a common classifier model for ET.

2.7.1 Masked Language Modeling for Entity Typing

Masked Language Model for Entity Typing (MLMET) [31] is a template-based approach that leverages Hearst patterns [32] and the masking ability of PLMs to generate weaker annotations from relevant headwords. Hearst patterns are used to describe hypernym and hyponym relationships. Some adequate patterns are " $\langle M \rangle$ and any other $\langle H \rangle$ ", " $\langle M \rangle$ and some other $\langle H \rangle$ " or " $\langle H \rangle$ such as $\langle M \rangle$ ", where H denotes a hypernym and M denotes a hyponym. For example, we could have M as "Michael Jordan" and H as "athlete". These patterns are applied to an ET dataset, which means that named entities are already detected but not typed. M can be filled by the named entity mention, while H can be replaced by "[MASK]" and filled with an appropriate token. With a correct mapping from what is generated by the mask and ultra-fine types provided by UFET, these predictions from Hearst patterns can serve as weak labels. A simple ET model can then be weakly trained on this dataset by following the input format "[CLS] $\langle X \rangle$ [SEP] $\langle x_{i:j} \rangle$ [SEP]",

where "[CLS]" is used for classification, and $x_{i:j}$ is the named entity. Continuing the training over human-annotated labels from UFET can produce some significant improvements over previous methods. This indicates that even for settings where fine-grained entity types are scarce, template engineering can help enhance supervision by providing weak but useful labels.

2.7.2 Prompt Learning for Entity Typing

Similarly, Prompt-Learning for Entity Typing (PLET) [2] uses hard and soft templates to achieve the task of ET. For this strategy, a hard ET template follows the pattern "[CLS] <X> [SEP] In this sentence, < $x_{i:j}$ > is [MASK].". An example of using this template can be: "[CLS] Michael Jordan has won many games. [SEP] In this sentence, Michael Jordan is [MASK]". In contrast, the idea of soft templates is similar to the learnable continuous prompts used to guide the attention mechanism in LightNER. However, learned vectors are from the token embedding space itself, replacing hard tokens. A soft template equivalent to this hard template would be "[CLS] <X> [SEP] < $x_{i:j}$ > [P_1] ... [P_l] [MASK].", where l is a hyper-parameter. In the context of PLET, the token "[MASK]" is limited to a vocabulary $\mathcal{V}^*$ composed of each natural type name and related nouns to each type in $\mathcal{T}$. The confidence score for a specific type y is the confidence scores obtained over the specific vocabulary $\mathcal{V}_y = w_1, w_2, \dots, w_k$ where $\mathcal{V}_y \in \mathcal{V}^*$:

$$P(y|X) = \frac{1}{k} \sum_i^k \lambda_i P([\text{MASK}] = w_k | T(X)),$$

where λ_i is a learnable weight for the specific token and $T(X)$ is the template generated for X . Their results have demonstrated that the small values of l are desirable for the soft prompts. Although hard prompts often achieve better results by a small margin over the soft templates, their performance can relatively drop for another dataset. This variance indicates that hard and soft templates are somewhat dependent on the designs used for the dataset.

2.7.3 Natural Language Inference for Entity Typing

LITE [33] addresses the ultra-fine ET task by reformulating it into the NLI problem.

Designing prompt templates for ET that also fit the premise-hypothesis pattern of NLI is possible. As described in 2.4, three templates of premise-hypothesis pairs for training have been formulated.

As with other prompt-based approaches, all these natural templates can leverage the model

Taxonomic Statement

$\langle X \rangle$ [SEP] $\langle x_{i:j} \rangle$ is a $\langle M(t) \rangle$.

- ✓ Michael has won many games. Michael is a person.
- ✗ Michael has won many games. Michael is a location.

Contextual Explanation

$\langle X \rangle$ [SEP] In this context, $\langle x_{i:j} \rangle$ is referring to $\langle M(t) \rangle$.

- ✓ Michael has won many games. In this context, Michael is referring to person.
- ✗ Michael has won many games. In this context, Michael is referring to location.

Label Substitution

$\langle X \rangle$ [SEP] $\langle x_{0:i} \rangle \langle M(t) \rangle \langle x_{j:n} \rangle$.

- ✓ Michael has won many games. Person has won many games.
- ✗ Michael has won many games. Location has won many games.

Table 2.4 Natural Language Inference templates for Entity Typing. X is the sentence represented by a sequence of tokens, while $x_{i:j}$ is a subsequence of the sentence from the token x_i to the token x_j . Moreover, $M(t)$ is the natural representation of a type t .

confidence to assess if a sentence X can be entailed by a hypothesis $H(t)$ which puts in relation the entity mentioned to a specific type t . For efficiency, LITE is built only around its best-performing template, which is the taxonomic statement. Since common fine-grained type sets have a hierarchical structure, type dependencies also need to be modelled. The technique used in LITE combines an ancestor's type description $H(c_{an})$ and a descendant's type $H(c_{de})$ in the NLI format, where the fine-grained descendant type is used as a premise for the coarse-grained ascendant type. For example, the named entity "London" in a sentence has the type description "London is a city" and "London is a location". A dependency is formed by entailing these two descriptions together and obtaining confidence over this entailment.

Another design choice for LITE is the use of negative sampling in their training strategy. More formally, the training example loss is formulated as such:

$$\mathcal{L}_t(X) = \left[\varepsilon(X, H(c^-)) - \varepsilon(X, H(c)) + \gamma \right]_+,$$

where $[x]_+ \equiv \max(x, 0)$ and γ is a non-negative constant. As for the negative type c^- , the entailment ε between the premise and hypothesis should be incorrect. A similar technique is used for the dependency objective:

$$\mathcal{L}_d(X) = \left[\varepsilon(H(c_{de}), H(c_{an})) - \varepsilon(H(c_{de}^-), H(c_{an})) + \gamma \right]_+,$$

where c_{de} and c_{an} correspond to the dependent and ancestor types, respectively.

Both of these margin ranking losses can be combined as follows:

$$\mathcal{L} = \frac{1}{|\mathbf{X}|} \sum_{X_m \in \mathbf{X}} \frac{1}{|\mathbf{C}_m|} \mathcal{L}_t(X_m) + \lambda \mathcal{L}_d(X_m),$$

where $\mathbf{C}_m$ is the set of true types assigned to the named entity m and λ is balancing how much the dependency loss influences the training. One major benefit of framing the entity typing problem as NLI is that it makes it easier to integrate new types, which is more desirable than fixed typesets in constantly changing real word domains.

CHAPTER 3 OVERVIEW

Our main research questions target the use of Entity Typing for Fine-grained Named Entity Recognition. Before considering the general task of FgNER, our first objective is to achieve ET with NLI on a large fully-supervised dataset. This involves the adaptation and the validation of NLI importance from what was achieved for low-resource ultra-fine ET to the task of fully-supervised ET. Since a simple adaptation might not yield the best performances, this first objective also includes the use of different strategies to maximize the benefits of NLI for ET in a fully-supervised setting. We propose NLI templates like the ontological and example statements to capture in diverse ways the semantic information given from a fully-supervised dataset. During the evaluation, different NLI-based rankings are unified to produce more accurate type rankings.

Our second objective is to integrate ET inside a FgNER system which overall doesn't include label-specific classifiers that would hinder the capacity of adapting automatically to a new typeset. Our approach to integrating ET for FgNER involves the coupling of ET with a NED or a NER system. We use a NED system to feed named entities to our best ET system, where we introduce a threshold over the generated ET rankings to prune falsely predicted named entities. We also achieve a combination of independent NER and ET models by using an ensembling over the predictions of both models while also using a threshold for pruning named entities. Moreover, we also perform transfer learning from our ET system to our NER system by using the same encoder-decoder architecture for both tasks.

3.1 Architecture Overview

In the following sections, we describe each architecture involved in performing the FgNER with ET. Building different integrations based on our best-trained ET model requires us to leverage all the predictions. The core parts of our FgNER architecture are presented in Figure 3.1.

3.1.1 Named Entity Detection Model

Different strategies are proposed for detecting named entities, which in our approaches can either be a NED or NER model. As presented in Figure 3.1, combining a model that produces named entity spans with an ET model requires that all predicted spans from a single sentence are divided into their own specific examples. Each example can then be evaluated

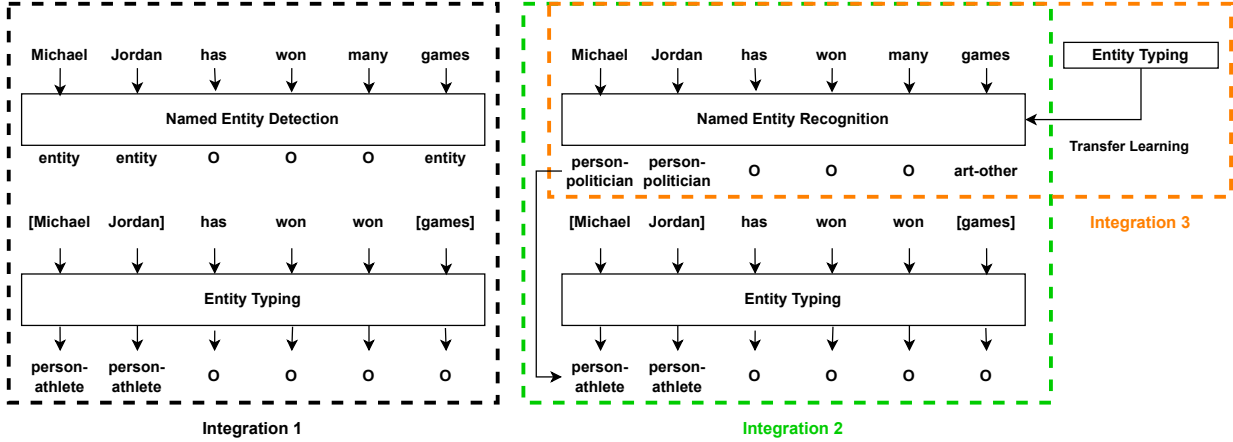


Figure 3.1 Overview.

independently of other predicted entity spans by the ET model. The predictions of our ET model can, later on, be combined to form the final sequence label predictions.

The main reason for using a NED model is to correctly capture named entity spans, regardless of their associated types. Our baseline for NED, BERT-CRF-NED, uses the HuggingFace Transformers library’s BertForTokenClassification with **bert-base-cased** weights, with a Conditional Random Field (CRF) layer. The overall NED architecture integrated into our FgNER system follows what is presented in Figure 3.2. We use BertForTokenClassification because it is one of the most common models for NER, which covers Named Entity Detection. We chose to use the pre-trained weights of the cased version of BERT because it is generally better than the uncased version **bert-base-uncased** for NER in English. We use the CRF layer because CRF can carefully model the dependencies between predicted labels, which enables our final FgNER model to achieve a relatively good span representation compared to advanced span modelling techniques described in the literature review. Also, this decoupling of the task of NED and ET can limit the complexity of our approach, which would be high when labelling with a CRF for the task of FgNER. To this end, we also compare this combination of NED and ET models with a simple NER baseline, BERT-CRF-NER, that follows the same setting as our NED architecture. More precisely, the only difference is that the typing is not delegated to an independent ET system but performed with additional parameters that specifically represent types in the linear and CRF layer.

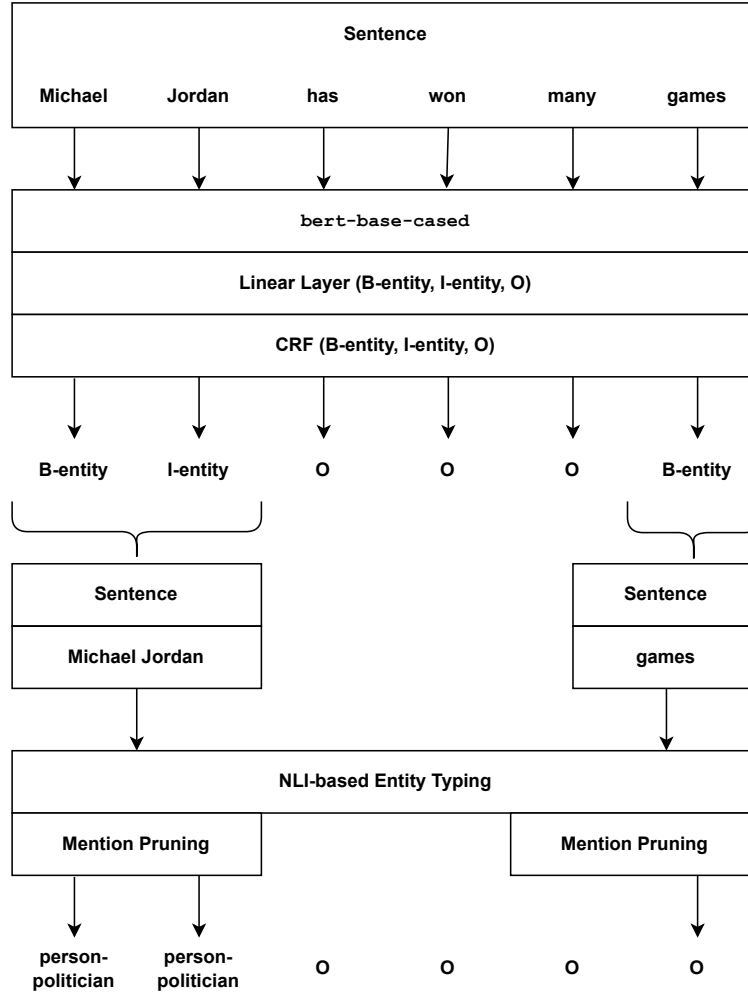


Figure 3.2 Architecture with independent Named Entity Detection and Entity Typing.

3.1.2 Entity Typing Model

Our solution to improve FgNER with type-free parameters revolves around the use of the NLI-based ET model. Since SotA performances have been achieved over ultra-fine ET with NLI-based methods, testing different strategies with NLI to achieve FgNER is essential in our research. Before considering additional strategies, the choice of the architecture and pretraining performed are important. The previous approach used **roberta-large-mnli** as their base model, where the pretraining of RoBERTa is continued over the MNLI dataset [34]. This is our PLM baseline for ET since it gives the best performances compared to **roberta-large**. Our end goal is to transfer as much as possible of what has been learned from the ET training to the rest of the FgNER system. One common technique is to use a model that has been pre-trained on ET as a starting point for the fine-tuning of the NER

task. This leads to the usage of a common architecture between our Named Entity Detection system and our ET system, which is the BART encoder-decoder architecture. We use BART because our type-adaptable system baseline for NER is LightNER. This model frames the NER task as a generation problem, where BART is a suitable architecture. To directly evaluate the impact of BART, `bart-large-mnli` is compared and evaluated with the same hyperparameters and training procedure done on `roberta-large-mnli`. Also, `bart-large` can validate if a general pretraining over the MNLI dataset can provide an improvement on the relatively large fully supervised dataset like Few-NERD.

3.1.3 Fine-grained Named Entity Recognition Model

In this work, we plan to enhance NER using the Entity Typing model described in the previous section. To do so, we employ the generative state-of-the-art LightNER model. LightNER serves as our baseline for performing FgNER without label-specific parameters. While this NER model can achieve reasonable performance and can easily be adapted to a new typeset, it is also few-shot friendly, which isn't the case for our previous NED model. This makes it more compliant to real-world low-resource scenarios, which is an important capability.

Our first approach follows a similar strategy to what has been presented for the NED and ET architecture. As illustrated in green in Figure 3.3, an independent integration of a LightNER with our best ET system is made, where both predictions of systems can influence the final results. Our second approach, in orange, involves the use of transfer learning from the ET model. Since both the ET model and LightNER use are based on the BART architecture, the weights of the encoder and decoder in LightNER can be replaced by what has been learned over the ET training.

3.2 Dataset

Fully supervised Few-NERD [6] (Few-NERD SUP) is the dataset chosen for this work since it is composed of a large amount of manually annotated fine-grained typed named entities. Based on FIGER [5], Few-NERD contains 66 different fine-grained entity types that cover the general domain. In this dataset, every named entity is annotated with a fine-grained entity type. The list of types is shown in Table 3.1. The dataset follows the IO tagging scheme. Because the taxonomy is completely hierarchical, every fine type has a unique coarser type associated. It has been shown with the development of this dataset that there is a lot more correlation between the context feature of two fine types with the same coarse type than two

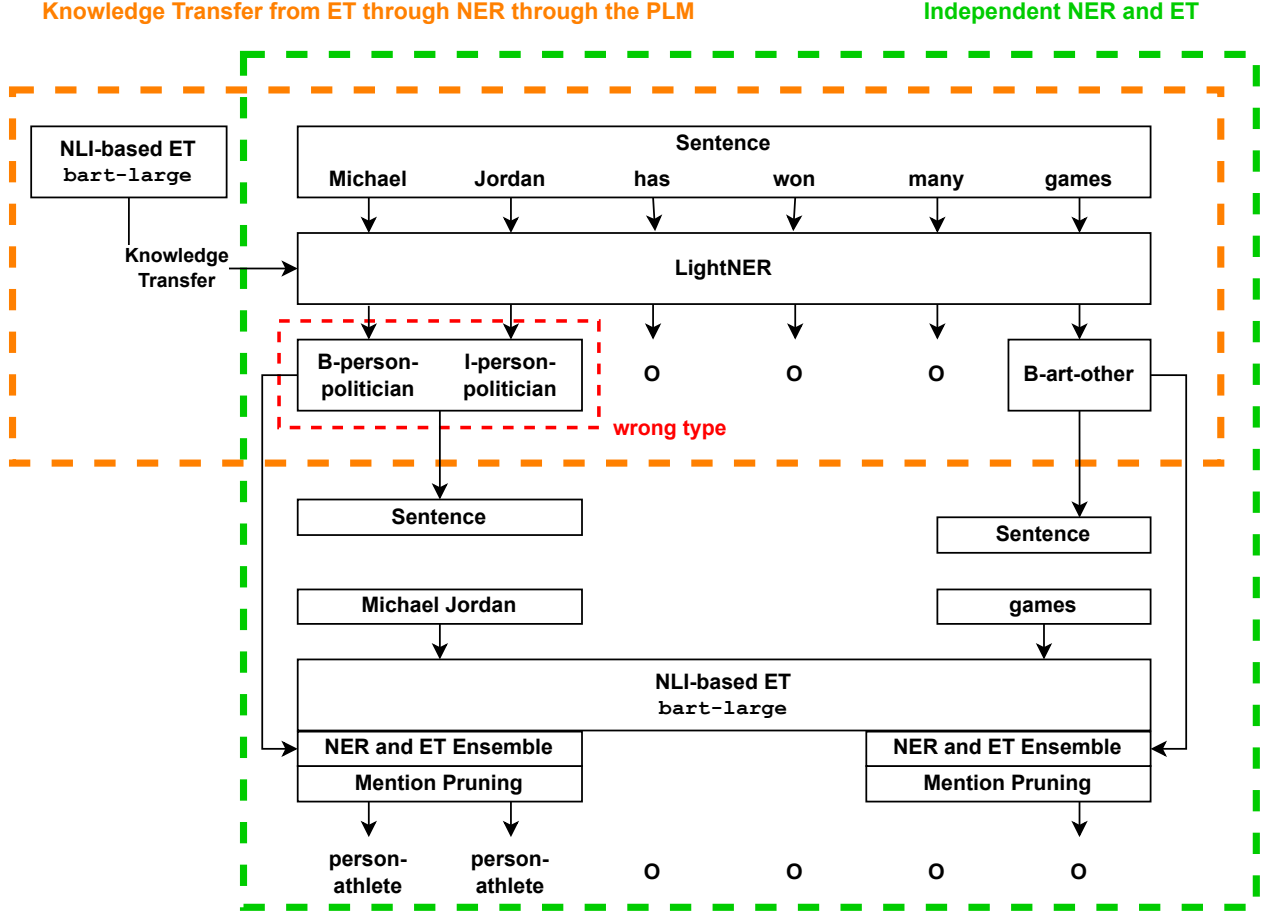


Figure 3.3 Entity Typing integrations for Fine-grained Named Entity Recognition.

types with a different coarse type. Hence, this dataset captures the essence of what makes FgNER hard, while dealing with a moderate typeset size. The presence of fine type **other** for each coarse category captures any entities that would not fit other fine types inside this specific coarse type. This ensures that a lot more entities are covered during the annotation process while keeping these sentences under the same coarse-grained category. Since some entity types do not occur as frequently as other coarse types but still need to be captured, they are placed in the coarse type **other**. Hence, this category is the one with the most unrelated context features between its fine types.

To reduce the energy consumption of running different experiments on the large Few-NERD-SUP dataset, we constructed the Few-NERD-Small dataset, which decreases the number of examples of each subset by an order of magnitude. For instance, the number of training examples goes from 131767 to 14537. As presented in the Algorithm 1, one heuristic to reduce the dataset size is to keep approximately the same relative type frequencies. For

Coarse-grained Type	Fine-grained Types
art (6)	broadcastprogram, film, music, other , painting, writtenart
building (8)	airport, hospital, hotel, library, other , restaurant, sportsfacility, theater
event (6)	attack/battle/war/militaryconflict, disaster, election, other , protest, sportsevent
location (7)	bodiesofwater, GPE, island, mountain, other , park, road/railway/highway/transit
organization (10)	company, education, government/governmentagency, media/newspaper, other , politicalparty, religion, showorganization, sportsleague, sportsteam
other (12)	astronomything, award, biologything, chemicalthing, currency, disease, educationaldegree, god, language, law, livingthing, medical
person (8)	actor, artist/author, athlete, director, other , politician, scholar, soldier
product (9)	airplane, car, food, game, other , ship, software, train, weapon

Table 3.1 Few-NERD taxonomy, composed of a two-level hierarchy. A total of 8 coarse-grained types and 66 fine-grained types are defined.

training, validation and testing subsets, a ratio of $\frac{131767}{10000} = 13.18$ has been used to reduce type frequencies. For example, there is a total of 759 **art-film** named entity spans in the test dataset of Few-NERD, whereas there is $\frac{759}{13.18} \approx 58$ in the Few-NERD-Small test dataset. The down-sampling algorithm is greedy, meaning that whenever a new example is randomly chosen from the unvisited examples, it is only kept if all its named entities are needed to achieve the desired type frequencies. Hence, examples with only one named entity have a smaller probability of being discarded by our greedy algorithm. This leads to a smaller named entity density in examples. All important descriptive statistics of our NER datasets are presented in Table 3.2.

3.3 Evaluation Metrics

As defined by previous studies [5, 35], ET and NER metrics designed for hierarchical typesets can either be strict or loose. In other words, the specific part of dealing with multi-label hierarchical classification from NER and ET systems brings to life new metrics that can deal with partially correct annotations. Let us consider $\mathcal{T}$ the set of all types, regardless of their level of granularity. Let us also define a segment of a text as e . Correct types for e can be defined as $t_e \subset \mathcal{T}$, while predicted types for e are defined as $\hat{t}_e \subset \mathcal{T}$. If e is a segment that

Algorithm 1: Dataset subsampling.**Input:** $D = \{D_{train}, D_{dev}, D_{test}\};$ N'_{train} : number of approximate desired training examples ; // 10000 for

Few-NERD-Small

 T : types;**Output:** $D' = \{D'_{train}, D'_{dev}, D'_{test}\};$ $N_{train} \leftarrow \text{CountExample}(D_{train});$ $R \leftarrow N'_{train}/N_{train};$ **foreach** $subset$ in $\{train, dev, test\}$ **do** $F'_{max,subset} \leftarrow \emptyset;$ **forall** $t \in T$ **do** $F'_{max,subset}(t) \leftarrow \text{Floor}(R * \text{CountEntities}(D_{subset}, t));$ $D'_{subset} \leftarrow \emptyset;$ $Visited \leftarrow \emptyset;$ **while** $\text{CheckStillMissingEntities}(D'_{subset}, F'_{subset})$ **do** $x \leftarrow \text{SelectRandomExample}(D_{subset});$ **if** $x \notin Visited$ **then** $F'_{subset} \leftarrow \text{CollectTypeFrequencies}(D'_{subset});$ $F'_{added} \leftarrow \text{CollectTypeFrequencies}(x);$ $F_{potential} \leftarrow F'_{subset} \cap F'_{added};$ **if** $F_{potential}(t) \leq F'_{max,subset}(t) \forall t \in T$ **then** $D'_{subset} \leftarrow D'_{subset} \cup x;$ $Visited \leftarrow Visited \cup x;$

Name	#Types		Subset	#Examples	#Entities	Ratio
	Cg	Fg				
Full	8	66	Total	188239	486059	2.58
			Train	131767	340387	2.58
			Dev	18824	48770	2.59
			Test	37648	96902	2.57
Small	8	66	Total	20900	36982	1.77
			Train	14537	25865	1.78
			Dev	2395	3729	1.56
			Test	3968	7388	1.86

Table 3.2 Few-NERD datasets.

does not represent any named entity, $\mathbf{t}_e = \emptyset$. Similarly, if e is not considered as a named entity span from the system’s prediction, the resulting predicted types are $\hat{\mathbf{t}}_e = \emptyset$. To better understand the loose metric, let’s suppose the existence of a segment e on Few-NERD as a named entity example. The correct types are $\mathbf{t}_e = \{t_{g_1}, t_{f_1}\}$, where t_{g_1} and t_{f_1} are the correct specific coarse and fine types, respectively. The predicted types are $\hat{\mathbf{t}}_e = \{t_{g_1}, t_{f_2}\}$. We observe that only the coarse type t_{g_1} is predicted correctly, while $t_{f_1} \neq t_{f_2}$. With loose metrics, this is not considered completely wrong, since the overlap $\mathbf{t}_e \cap \hat{\mathbf{t}}_e = \{t_{g_1}\}$, which is not empty. However, it is invalid for a strict metric, since $\hat{\mathbf{t}}_e \neq \mathbf{t}_e$. Here are the strict and loose metrics used to compare and evaluate each model, where T are all the true named entity segments and P are the predicted named entity segments:

Strict Metrics:

$$\text{precision} = (\sum_{e \in T \cap P} \delta_{\mathbf{t}_e, \hat{\mathbf{t}}_e}) / |P| \quad \text{recall} = (\sum_{e \in T \cap P} \delta_{\mathbf{t}_e, \hat{\mathbf{t}}_e}) / |T|, \text{ where } \delta_{i,j} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

Loose Macro Averaged Metrics:

$$\text{precision} = \frac{1}{|P|} \sum_{e \in P} \frac{|\mathbf{t}_e \cap \hat{\mathbf{t}}_e|}{|\hat{\mathbf{t}}_e|} \quad \text{recall} = \frac{1}{|T|} \sum_{e \in T} \frac{|\mathbf{t}_e \cap \hat{\mathbf{t}}_e|}{|\mathbf{t}_e|}$$

Loose Micro Averaged Metrics:

$$\text{precision} = \frac{\sum_{e \in P} |\mathbf{t}_e \cap \hat{\mathbf{t}}_e|}{\sum_{e \in P} |\hat{\mathbf{t}}_e|} \quad \text{recall} = \frac{\sum_{e \in T} |\mathbf{t}_e \cap \hat{\mathbf{t}}_e|}{\sum_{e \in T} |\mathbf{t}_e|}$$

F1 score:

$$\text{F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

The loose macro-averaged F1 score (loose MaF1) computes the metric independently for each type, which is aligned for an evaluation that doesn’t depend on class imbalance. In contrast, the loose micro averaged F1 score (loose MiF1) gives equal importance to each example. For the ET evaluation performance, the loose MiF1 is used by previous ET systems evaluated on Few-NERD. Hence, this metric is mostly used for the final ET comparison over the complete Few-NERD dataset. As for choosing the best checkpoint model, we follow LITE [33] by using the loose MaF1 on the validation dataset. Using the strict MiF1 for the evaluation is more aligned with the type hierarchy constraints, which is that a fine type from Few-NERD will always be paired with the same coarse type. As a matter of fact, the strict MiF1 is equivalent

to the strict accuracy, which is also provided in other studies.

To identify and compare what impacts performance in FgNER systems, metrics are computed for each type level in the type hierarchy. To align ourselves with previous approaches, entity span-level metrics are used instead of word-level metrics. Few-NERD annotations follow the IO tagging scheme over fine-grained types, which makes it unable to differentiate two consecutive named entities with the same fine-grained types, but can when two consecutive named entities are from different fine-grained types. Hence, it is possible to transform IO annotations as IOB2 for coarse-grained-level annotations or span-level annotations. Thus, more robust and significant evaluation metrics can be applied, while keeping the same number of entities. Furthermore, it is a necessity for NED systems to differentiate two consecutive named entities, which cannot be done with IO tagging. An ET system would have a lot of difficulties if these two consecutive named entities were given as one single named entity since it was trained to classify one named entity at a time. The advantage of computing this metric is to compare NED models with complete NER systems. Since some models include a NED submodule as part of the FgNER system, it is relevant to capture the potential impact of some strategies over detecting each correct named entity span. Overall, the less restricting metric is the IOB2 span-strict matching, as illustrated in Table 3.3, followed by the IOB2 coarse-grained-strict matching shown in Table 3.4 and finally with the fine-grained-strict matching as in Table 3.5, which is also the strict accuracy. The F1 score of the fine-grained matching is used for the validation and evaluation of performances.

	Michael	Jordan	has	won	many	games
Good	B-entity	I-entity	O	O	O	O
Bad	O	B-entity	I-entity	O	O	O

Table 3.3 IOB2 Span-strict matching.

	Michael	Jordan	has	won	many	games
Good	B-person	I-person	O	O	O	O
Bad	B-product	I-product	O	O	O	O

Table 3.4 IOB2 Coarse-grained matching.

Our evaluations will cover all the experiments done to analyze the impact of NLI-based strategies for ET as integration for achieving FgNER over some large fully-supervised datasets. We will first make a deep dive into the added value of each component of our ET model. Then, an analysis of each integration of our ET model inside some different FgNER systems will

	Michael	Jordan	has	won	many	games
Good	person-athlete	person-athlete	O	O	O	O
Bad	person-politician	person-politician	O	O	O	O

Table 3.5 IO Fine-grained matching.

be done to put in perspective the performances of our approaches in comparison to what has been done in FgNER.

CHAPTER 4 NATURAL LANGUAGE INFERENCE FOR FULLY-SUPERVISED ENTITY TYPING

As mentioned in the literature review, prompt engineering is a valuable paradigm in which a task is defined from the inputs. This can reduce the need for task-specific architecture design while also providing good few-shot and zero-shot performances due to the capability of leveraging knowledge captured in large pre-trained models. One specific technique for prompt engineering involves the creation of human-curated templates. The technique of template engineering remains really intuitive to use while prompting the model for task-related information. ET has the potential of leveraging template engineering with the use of NLI, which is the case for many other NLP tasks [36]. Our techniques use different forms of NLI statements that can together perform fine-grained ET.

4.1 Supervision from General Natural Language Inference

From previous works with NLI with ET on ultra-fine typing in relatively low resource settings, pretraining the PLM on general NLI has shown some gains in comparison to only performing the PLM’s normal pretraining. More precisely, this was done by using the Multi-Genre Natural Language Inference (MNLI) [34] dataset, described in Table 4.1. The main characteristics of this dataset are its large size and the multiple domains it covers. The task of NLI involves sentences that can either entail, contradict or be neutral to each other. The MNLI has a balanced dataset where each of these relationships occurs about one-third of the time. As shown also for other NLI datasets, models trained on this task tend to over-predict an entailment for longer sentences and over-predict a contradiction for sentences containing negation. These biases should be taken into consideration when evaluating models pre-trained on MNLI.

Genre	Fiction	Government	Slate	Telephone	Travel
Examples	77 348	77 350	77 306	83 348	77 350

Table 4.1 Multi-Genre Natural Language Inference training dataset.

4.2 Natural Language Inference Templates for Entity Typing

As described earlier, template engineering provides an easy way to naturally probe a PLM for task-related information. Template engineering involves the use of specific tokens to represent

the task to accomplish. A recasting of the ET task as an NLI problem can be achieved by adding semantic information inside the premise and hypothesis pairs. The semantic information is provided by verbalizers. Generally, verbalizers are an indirect way of mapping a class without any additional parameters by simply using some class-related tokens or embeddings provided by a PLM, which should yield some valuable representation. In our case, the type names or textual occurrences of a specific entity type are a source of verbalization. Since types in most taxonomies are already described in a form where their meaning is naturally understandable for humans, it is reasonable to utilize this information to benefit our model. Hence, all manually curated type names can be integrated into our NLI templates. Having intuitive verbalizers also aligns with the semantic and syntactic understanding learned during the pretraining process.

Manual template engineering with NLI as a prompting method has been proven helpful over different ET tasks like the Open Entity dataset [30] where the types are not bound to a clear taxonomy or Ontonotes [37] where a taxonomy is defined. The evaluation of NLI in a more complete fully-supervised setting, which Few-NERD can provide, remains important. As presented previously, it has been shown that with PLET [2], hard templates outperform soft templates for Few-NERD in a fully-supervised setting. One of the major causes is that learning continuous templates is valuable for noisy data that would not be as aligned as a high-quality and human-curated dataset. Hence, we decided to use hard manual NLI templates only for our study on Few-NERD. As shown with LITE [33], NLI templates also work better with curated annotations. A degradation of performance arises when training the model on more noisy data. Other techniques that utilize both curated and noisy data during their training process are not better than NLI-based training with only the curated data. Hence, performing ET with NLI templates on Few-NERD might achieve good performance relative to other ET systems. Existing and new NLI templates are combined and compared in our strategy to validate their individual importance. An overview of each template is presented in Table 4.2. All the templates remain the same over the different datasets and during the training and evaluation process. Representative examples of how we manually curated different verbalizers can be found in Table 4.3.

Taxonomic and Dependency Statements. Similarly to LITE [33], we train models using the taxonomic(**taxo**) and dependency(**dep**) NLI statements. As shown in 4.2, **taxo** validates the connection between a named entity and a type name, given the context in which this named entity takes place. Training with the taxonomic statement can help the model learn a natural relationship between the context, the entity span and the type. Modifications have been made to the type label such that the resulting verbalizers are more natural, which

Taxonomic Statement (taxo)	
Definition	Relates context, named entity, and type.
✓	Michael Jordan has won many games. Michael Jordan is a person athlete.
✗	Michael Jordan has won many games. Michael Jordan is a location other.
Dependency Statement (dep)	
Definition	Hierarchically relates types through the named entity.
✓	Michael Jordan is a athlete. Michael Jordan is a person.
✗	Michael Jordan is a athlete. Michael Jordan is a location.
Ontological Statement (onto)	
Definition	Directly hierarchically relates types.
✓	Michael Jordan is a athlete. A athlete is a person.
✗	Michael Jordan is a athlete. A athlete is a location.
Example Statement (ex)	
Definition	Relates context, the named entity and another named entity example that appears frequently and only for this type.
✓	Michael Jordan has won many games. Michael Jordan is like Mysterio.
✗	Michael Jordan has won many games. Michael Jordan is like Jeff Bridges.

Table 4.2 Natural Language Inference templates for Entity Typing. All templates involve type-related natural tokens or tokens like "Mysterio" that will necessarily be seen after the first epoch of training.

Type Label	Taxonomic Statement	Example Statement
art-broadcastprogram	art broadcast program	EastEnders
art-writtenart	art written	Glasgow Herald
location-bodiesofwater	location bodies of water	Colorado River
other-livingthing	other living thing	rat
person-artist/author	person artist or author	Britten

Table 4.3 Entity Type verbalizers for Natural Language Inference statements.

includes the addition of spaces between words and the replacement of slash by "or". The model can then learn information on the semantic relationship between named entities and types without assigning a label number to types as it would be for the linear classifier.

As for **dep**, this statement affirms the connection between a named entity and a coarse type, given a correct association between a fine type and the named entity. This can provide relationships between types through the named entity, which helps the model naturally learn the hierarchical structure of the type set. An additional verbalizer for each coarse type is necessary for this statement, which is simple given that the Few-NERD taxonomy only deals with coarse-grained types composed of one word.

Ontological Statement. The ontological statement **onto**, defined in Table 4.2, can be integrated into the training process like the dependency statement. It provides an alternative way to capture the semantic relations from the type hierarchy while producing more direct attention between types. Similar to the dependency statement, verbalizers are also generated for coarse-grained types. However, no named entity is present for this statement, which makes a stronger signal from the model’s attention towards the type verbalizers.

Example Statement. Instead of using a type verbalizer to form taxonomic, dependency or ontological statements, another way to do ET with NLI is to use named entities that are unique to a specific fine type. For each fine type, a specific named entity is chosen as a form of type verbalizer. This named entity is selected from the most frequent and unique named entities in the training dataset. More specifically, defined as the example statement **ex** in Table 4.2, each most frequent unique named entity can indirectly represent their corresponding fine type when generating a ranking. Instead of using the pattern "is a" which checks for a hierarchical relationship, the use of "is like" can act as a form of direct similarity between two named entities. Even if the model might not have seen it during the pretraining, the most frequent and unique named entities like "Mysterio" for the "person-athlete" type could after some epochs encompass some form of knowledge from the context related to this specific fine type containing this named entity. Directly using these type-representative examples for ET can be easily adapted to new datasets because each verbalizer is constructed directly from within the dataset.

4.3 Natural Language Inference Datasets for Entity Typing

Demonstrating the impact of ET with NLI for large fully-supervised fine-grained ET involves the use of the Few-NERD dataset. Constructed from Few-NERD-Small, the ET-Few-NERD-

Small dataset can be used to accelerate experiments. This is even more valuable for ET since the main bottleneck of FgNER is ET with NLI because of all the possible templates produced during the training and evaluation process. Examples containing multiple named entities are separated into many examples because type ranking templates like taxonomic and example statements are designed for typing one named entity at a time, making the entire ET system independent of previous and next entity typing predictions. Hence, a single NER example could result in a lot of NLI pair examples. Table 4.4 presents some descriptive statistics over ET datasets.

Name	#Types		Subset	#Entities	NLI Pair Examples			
	Cg	Fg			Taxo	Dep	Onto	Ex
Full	8	66	Total	486059				
			Train	340387	680774	680774	680774	680774
			Dev	48770	3218820			3218820
			Test	96902	6395532			6395532
Small	8	66	Total	36982				
			Train	25865	51730	51730	51730	51730
			Dev	3729	246114			246114
			Test	7388	487608			487608

Table 4.4 ET-Few-NERD datasets. NLI pair examples contain correct and incorrect pairs, where there is one incorrect pair per correct pair in training. In validation and testing, all 66 type-related NLI pairs are evaluated for taxonomic and example statements.

4.4 Training with Natural Language Inference Statements

Training an ET model with NLI involves a margin ranking loss per statement. As previously mentioned in the literature review, this loss is defined as the following:

$$\mathcal{L}_{\text{nli_statement}} = \varepsilon(P, H) - \varepsilon(P, H') + \gamma,$$

where γ is a non-negative constant, H is the entailing hypothesis of premise P and H' is the non-entailing hypothesis of P . This differs from the loss used for pretraining on the MNLI dataset since it involves the classification of pairs as either an entailment, a neutral relationship or a non-entailment. Each ranking loss is computed with a fixed margin value γ of 0.1, which was chosen from previous experiments done in LITE [33].

The margin ranking loss between a pair of positive and negative taxonomic statements is fully backpropagated, while the loss for the dependency statements has a backpropagation

weight of 0.05. We justify these hyperparameter values by following the setting of LITE [33]. Since the ontological statement is similar to the dependency statement, its loss weight is also 0.05. The example statement is similar to the taxonomic statement because it maps a named entity to a fine type. Hence, its loss has an important weight of 1.

The corresponding weighting for each loss is the following, whether each of them is actually computed or not, since possible training variations may not include all statements:

$$\mathcal{L} = \mathcal{L}_{\text{taxo}} + 0.05 * \mathcal{L}_{\text{dep}} + 0.05 * \mathcal{L}_{\text{onto}} + \mathcal{L}_{\text{ex}}$$

By using the margin ranking loss, we need to sample a negative type for each statement involved in a single named entity. We use the same random negative type for similar statements. Therefore, we make sure that the sampling is identical for similar statements, which is valuable for comparative purposes. For example, given the correct fine type **person-athlete**, a negative sampling of **person-politician** could have been done for the taxonomic and example statements. However, the negative fine type has the same coarse type as the positive fine type. Since the dependency and ontological statements need a negative coarse type, the negative sampling for these two statements is independent of the sampling made for the taxonomic and example statements. That way, the sampling for each statement can cover its corresponding negative example space, where a random type is chosen at each epoch of the training process.

4.5 Entity Typing with Natural Language Inference in Evaluation

The general idea of ET with NLI is to evaluate each possible type in its own NLI pair, where a ranking can be produced from the entailment score of each type. By doing so, we can easily incorporate new types and adapt the model to different typing schemes rather than having a dedicated classification layer. When dealing with an ET task where taxonomy is defined, only the prediction of the finest types is necessary. For example, in Few-NERD, each fine-grained type has only one coarse-grained type associated, which means that predicting a fine-grained type directly gives us the corresponding coarse type. For example, we know from the taxonomy that the fine type **athlete** has always **person** as a coarse type. The final predicted label assigned to the named entity is the fine type with the highest entailment score. From all the NLI statements developed, the taxonomic and the example statements can each give us a ranking. The dependency and ontological statements are disregarded because their goal is to capture the hierarchical type structure while knowing the correct fine type as a premise, which cannot be achieved during evaluation.

In terms of architecture for computing the entailment between a premise and a hypothesis, the problem is simply framed as a sentence classification task, where we only need to take the score of the entailment label produced by the model. For an encoder-only model such as RoBERTa, the output of the first token is dedicated to classification as shown in Figure 4.1.

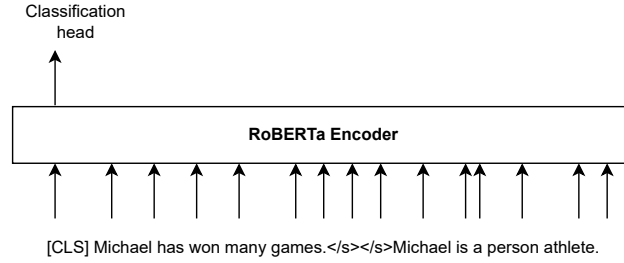


Figure 4.1 NLI with RoBERTa.

For an encoder-decoder model like BART, sentence classification uses both the encoder and the decoder to perform the classification, as shown in Figure 4.2. Instead of having the linear classification at the start as in RoBERTa, the last hidden state of the decoder model is given the entire input as a state. It is then forwarded to a linear classification head containing the entailment label.

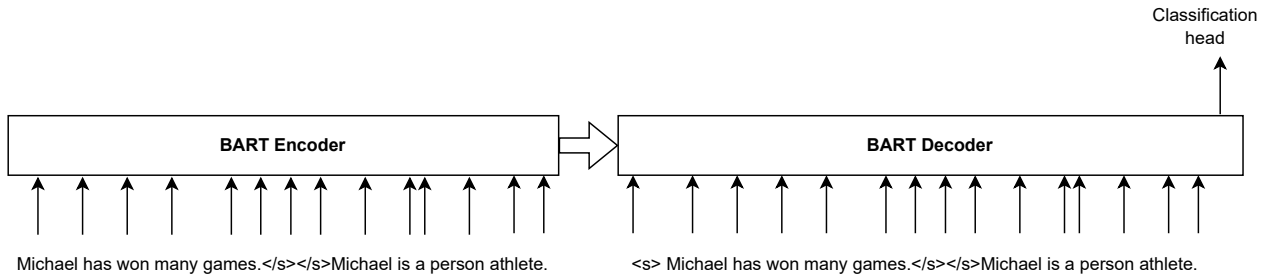


Figure 4.2 NLI with BART.

4.6 Taxonomic and Example Statements Ensemble Ranking

The training of a model with the taxonomic and example statements is somehow different in terms of deciding the best model based on the validation dataset. First, this model produces taxonomic and example statement rankings independently. At each checkpoint, the maximum loose MaF1 score between those two rankings over the validation dataset is kept as the validation performance. A new checkpoint can use this resulting value and

compare it to the best one obtained from previous checkpoints. The final ET model is the checkpoint with the highest maximum loose MaF1 score. At the end of the training, the weight for combining the taxonomic and example rankings is tuned with the strict MiF1 performance on the validation ET dataset. The ET model can now give a single ranking, which can be used by the rest of our FgNER system.

To increase the exactitude of our predictions, we can combine those rankings by following a similar approach to prompt ensembling with weight averaging. Prompt ensembling can enable a model to leverage the different unique capabilities of each specific prompt while also stabilizing the performance like any other ensemble method. Weight averaging associates a weight for each prompt to maximize the probability of obtaining the correct target over the training dataset. Normally, this weight is either predefined from individual training performance obtained for each statement or learned during the training process. However, we do not generate a direct probability distribution over all types in our case, since we are using a contrastive approach for training, where only the score for a positive statement and the score for a negative statement are produced. Our strategy employs the validation dataset to fine-tune this weight. After the training process, our model generates all type-rankings for all statements for all examples of the validation dataset. We can then evaluate by combining the taxonomic and example statement rankings what value gives us the best fine-matching MiF1 score for this weight, as shown in Figure 4.3. This is done as such:

$$\text{combined-ranking}(t) = \lambda_{\text{ranking}} \cdot \text{taxonomic-ranking}(t) + (1 - \lambda_{\text{ranking}}) \cdot \text{example-ranking}(t),$$

where t is a type and $\text{taxonomic-ranking}(t)$ and $\text{example-ranking}(t)$ are confidences for t over the taxonomic type ranking and the example ranking, respectively. The case where the best value is 0 for λ_{ranking} would mean that all the importance is attributed to the example ranking, while a value of 1 would mean the opposite. The case where the best value is 0.5 would signify that the taxonomic and example statement rankings are equivalently important, as shown with the example in Table 4.5.

Type	taxo		ex		$\lambda = 0.5$	
person-athlete	0.90	#1	0.75	#2	$0.5 \cdot 0.90 + (1 - 0.5) \cdot 0.75 = 0.825$	#2
person-politician	0.80	#2	0.95	#1	$0.5 \cdot 0.80 + (1 - 0.5) \cdot 0.95 = 0.875$	#1
...	...	...	...	...	...	...
product-food	0.10	#66	0.11	#66	$0.5 \cdot 0.10 + (1 - 0.5) \cdot 0.11 = 0.105$	#66

Table 4.5 Ensemble Reranking Example.

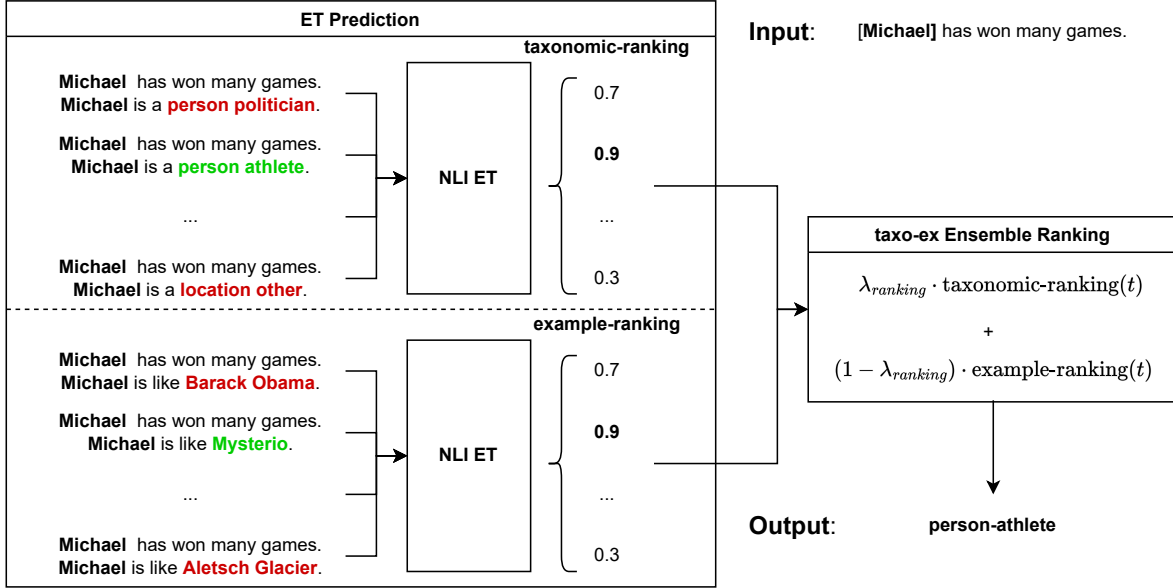


Figure 4.3 Taxonomic and Example Ensemble.

4.7 Experimental Results

Our objectives in ET are to assess the potential of NLI-based ET for large fully-supervised datasets and to develop new NLI strategies that would be beneficial in terms of performance. Since NLI-based strategies for ET have already been used for a similar but different setting, we want to re-evaluate how well each one of them translates from an ultra-fine low-resource typing setting to our more fully-supervised fine-grained setting. Doing such experiments will provide us with a path to the best ET system we can achieve given previous and proposed strategies. This ET system’s performance and unique abilities are compared to other strong approaches previously evaluated over our dataset.

4.7.1 Comparative Results of Entity Typing with Natural Language Inference Model

As mentioned previously, it is in our main interest to validate the importance of each strategy that can affect the design choices of our final NLI-based ET system. As the main paradigm behind much NLP-related progress in recent years, choosing the right PLM for the desired objective is essential, both in terms of general pretraining task achieved or fine-tuning, which is why we first compare the results between **roberta-large-mnli-taxo-dep** and **bart-large-mnli-taxo-dep**. We also assess the necessity of preliminary supervision over MNLi when dealing with a fully-supervised setting, which involves a comparison be-

tween `bart-large-mnli-taxo-dep` and `bart-large-taxo-dep`. To make sure that comparisons are made properly, any common parameters involved in the training procedure are kept the same for all experiences. This first includes the random seed value of 42 used for generating negative examples during the whole training process of 70 epochs on the ET-Few-NERD-Small dataset. An evaluation of the development dataset is made two times per epoch, which is at every 3233 steps. We reproduce similar behaviours to what has been observed in the previous studies for NLI-based training procedures for ET. This includes the use of AdamW with an epsilon $\epsilon = 1 \times 10^{-8}$ and a learning rate $lr = 1 \times 10^{-6}$. The training batch size is 16 for all these experiments. More precisely, there are 4 named entity examples, where each example encompasses the taxonomic and dependency statements, which both contain a positive NLI statement and a negative NLI statement, for a total of 16.

Focussing on NLI statements themselves, different combinations of them are done to understand their own impact. We first verify the hypothesis emitted in LITE, which is that capturing the hierarchical structure of type using the dependency statement is relevant for an ET task with defined taxonomies. The validation of this finding is also taken into account with the ontological statement, which is a variation of the dependency strategy to capture the taxonomy’s structure. The study concerns the importance of the example statement alone and combined with the taxonomic statement. This also includes an analysis of how combining templates in evaluation can benefit the final performances.

As done for the previous comparison, 4 named entity examples are passed at the same time with a maximum of 4 NLI statements in the case where `taxo`, `dep`, `onto` and `ex` are all in use. As mentioned previously, the example statement is analogous to the taxonomic statement and shares an equivalent hyperparameter value of 1 for its importance in the training loss. A value of 0.05 is used for the dependency and taxonomic statement since they are tuning the model toward the same objective.

For all the comparisons study specified above, the resulting performances are shown in table 4.6. Overall, we can see that the default pretraining for BART followed by a fine-tuning with taxonomic and example statements seems to work best for the large fully-supervised ET-Few-NERD-Small with a fine-grained MiF1 score of 78.99.

Choice of Pretrained Language Model

The previous approach [33] for performing NLI-based ET used RoBERTa as a base architecture. Since one of our objectives is to validate their work for a larger dataset and fully-supervised ET, our starting point for performing ET is their best-tested architecture, which is `roberta-large-mnli`. RoBERTa and other larger models like BART have shown good re-

Base	Model				Coarse Matching			Fine Matching		
	taxo	dep	onto	ex	P	R	MiF1	P	R	MiF1
roberta-large-mnli	✓	✓			90.75	90.72	90.73	77.77	77.71	77.74
bart-large-mnli	✓	✓			91.01	91.00	91.01	77.83	77.79	77.81
bart-large	✓				91.37	91.35	91.36	78.53	78.48	78.51
				✓	91.45	91.45	91.45	78.45	78.40	78.42
	✓	✓			91.15	91.14	91.14	78.03	77.96	78.00
	✓		✓		90.94	90.93	90.94	77.56	77.52	77.50
	✓			✓	91.88	91.88	91.88	79.01	78.97	78.99
	✓	✓	✓	✓	91.31	91.30	91.30	78.24	78.18	78.21

Table 4.6 Entity Typing performances on ET-Few-NERD-Small.

sults over the NLI task on the MNLI dataset, which is a great indicator that they also should perform well for our objective of using NLI for ET. As described in the methodology, BART is the default generative architecture used in LightNER to perform NER as part of one of our FgNER strategies. Another strategy is to use transfer learning from ET to NER, while also being able to compare it to what the default LightNER model can achieve on fully supervised FgNER. With the available models of **roberta-large-mnli** and **bart-large-mnli** on HuggingFace, a comparison of performance is done. The results indicate that **bart-large-mnli** has a slight improvement of 0.07% from 77.74% to 77.81%. We can also observe this small difference over the course of the training process, shown in Figure 4.4. As stated in previous studies, BART also demonstrates good classifying performances while having good generative capabilities, which means that even this small improvement is enough to choose **bart-large-mnli** over **roberta-large-mnli** as the preferred PLM.

Importance of Supervision from MNLI

Our goal is to adapt what has been done with NLI-based ET. Hence, a validation of the necessity of training PLMs on MNLI to achieve fully-supervised fine-grained ET is valuable. Since we previously observed that BART is more suitable than RoBERTa, we can now compare **bart-large** and **bart-large-mnli**. We see that for the same ET baseline with taxonomic and dependency statements, **bart-large** performed with 78.00% while **bart-large-mnli** 77.81%. While training on MNLI remains valuable for zero-shot or few-shot classification,

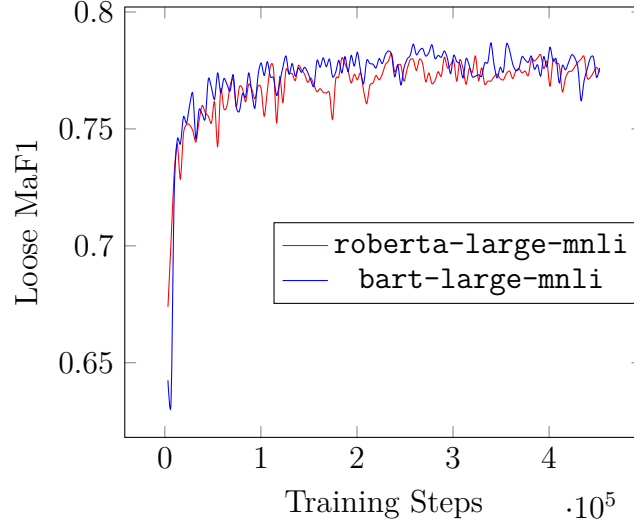


Figure 4.4 Performance of different Pretrained Language Models during Entity Typing training. They both use **taxo** and **dep** as NLI statements.

our results indicate that the additional training is not useful when a large fully-supervised dataset is available for the task to achieve. This confirms the finding of [38], where the use of textual entailment with NLI is said to mostly matter when there are not a lot of annotations available. Shown in Figure 4.5, we see that the benefits of pretraining on MNLI are rapidly becoming insignificant after some small number of training steps.

As mentioned in the methodology, the MNLI dataset has more entailment labels for longer sentences [34], which can be considered a form of bias. This would mean that using a model trained on MNLI should classify longer sequences as entailment. For NLI-based ET on FewNERD, the only length difference when comparing our premise-hypothesis pairs for ranking comes from the length of the verbalizers used for representing the type. For example, the longer hypothesis "Michael is a person artist or author" should be predicted more often than "Michael is a person athlete". This can happen for zero-shot ET since the same architecture and classification head are used during the training over MNLI. This is shown in Figure 4.6 for the fine types with the longest verbalizers composed of around 6 or 7 words. However, we can also observe that the fine-tuned model with FewNERD is not affected anymore by the number of words in verbalizers. As expected from a complete fine-tuning, its false positive frequencies seem to be more correlated to the support, which is the number of examples provided for each corresponding fine type.

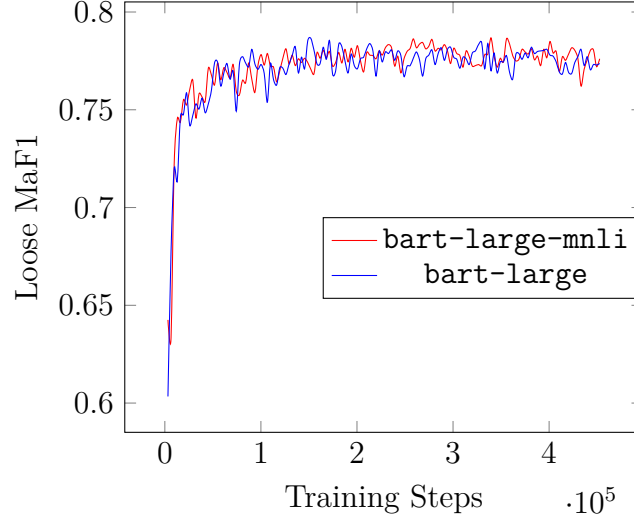


Figure 4.5 Performance difference of using MNLI during ET training. They both use **taxo** and **dep** as NLI statements.

Importance of Training with Dependency and Ontological Statements

One key idea from LITE was to use a second objective to capture the label dependency. This has been helpful for low-resource datasets with ultra-fine types since a lot of different positive examples can be generated as a source of additional information for the NLI task to perform. The use of the taxonomic and dependency statements has also been used over the larger dataset Ontonotes, which deals with a simple type structure since each fine type always has the same coarse type associated, like in Few-NERD. However, an ablation analysis of the dependency has not been done, which is why we need to validate how useful it is over a large ET dataset. To this end, we compare **bart-large-taxo** to **bart-large-taxo-dep**. As mentioned in the methodology, the training weights of 1 and 0.05 assigned respectively to the taxonomic and dependency statements follow the same hyperparameter values used in LITE.

We can see that the effect of adding the dependency statements is not beneficial, since we observe a drop of 0.51% from 78.51% to 78.00% in terms of MiF1 score for fine matching. This drop is significant enough to consider that adding the objective of capturing the hierarchical semantics with the dependency objective can hinder the objective achieved by the taxonomic statement. Moreover, from **bart-large-taxo** to **bart-large-taxo-dep**, the training time goes from 53 hours to 65 hours.

Validating the usefulness to capture the hierarchical type dependency from the taxonomy can also be done with the ontological statement. Hence, we also trained **bart-large-taxo-onto**

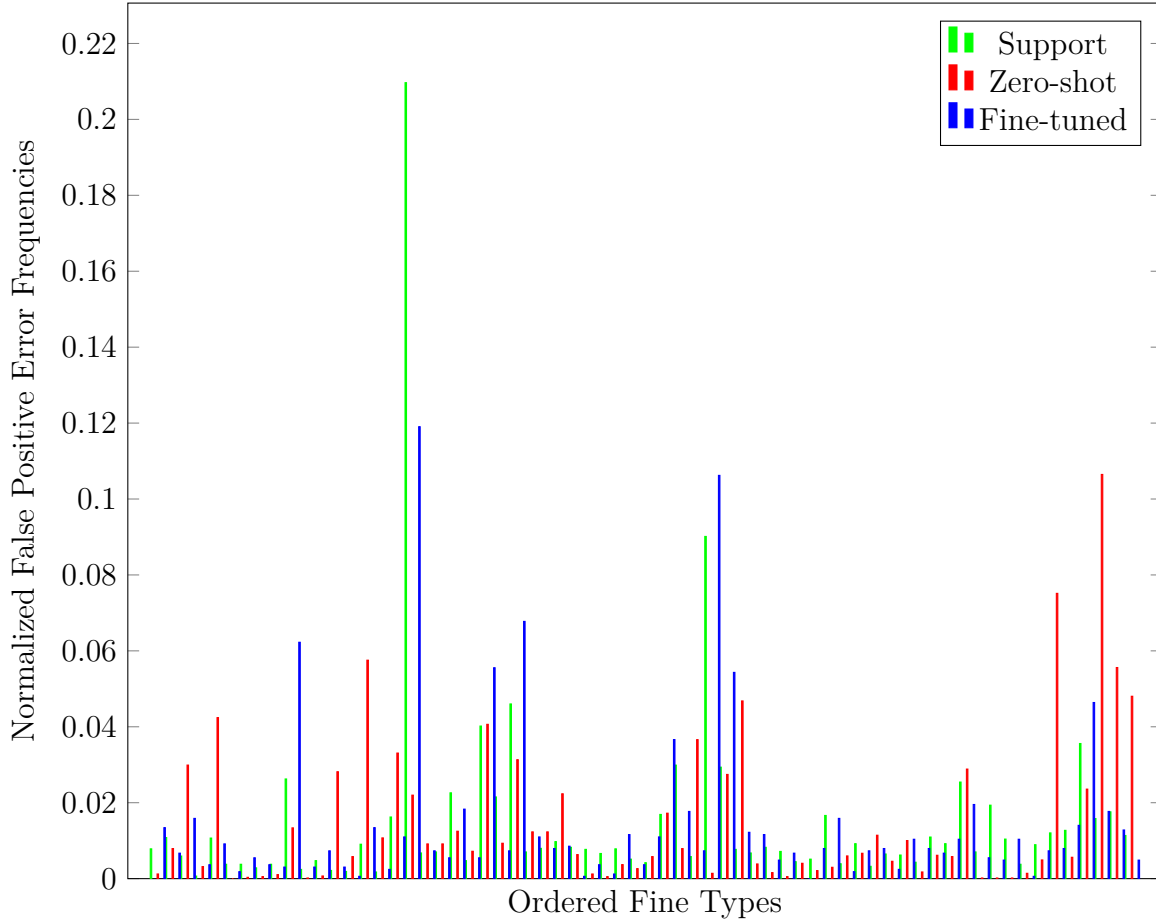


Figure 4.6 False positives before and after training. Fine types are sorted in ascending order of their verbalizer’s word count. The frequencies are normalized by the total amount of false positives produced by the corresponding model. The base model is `bart-large-mnli`.

and compare it to `bart-large-taxo`. We also use the weight of 0.05 for the training loss of the ontological statement. We observe the same behaviour as before, which is that adding an objective to capture the hierarchy deters performance. More precisely, a drop of 0.97% in value occurs between from 78.51% for `bart-large-taxo` to 77.54% for `bart-large-taxo-onto` in MiF1 in fine matching.

We showed here that training with dependency or ontological statement reduces performances when large full supervision is done.

Training with Example Statements

As explained in the methodology, it has been shown that ensemble prompting helps achieve better performance by taking advantage of different prompts. In our case, we experiment with

the addition of the example statement in the training loss and making an ensemble of rankings produced during the evaluation by both the taxonomic and example statements. With the same hyperparameter weight of 1 given to the training loss of both of them, we can understand the value of the ensemble by comparing **bart-large-taxo** to **bart-large-taxo-ex**. We observe an increase of 0.48% from 78.51% to 78.99%. The weight learned on the validation dataset encapsulates how much weight is given to the rankings produced by the taxonomic statement versus what is given to the example statement. The final value obtained after the training is 0.6395, meaning that the model gives more importance to the taxonomic statement rankings. This follows the performance difference between training a model only with the example statement versus training only with the taxonomic statement. More precisely, **bart-large-taxo** has a 78.51% fine matching MiF1 score while **bart-large-ex** scores 78.42%. This shows that training with the taxonomic statement is valuable for the example statement.

Training with All Statements

Our final comparison involves training with all statements. To this end, we use the weight hyperparameter of 1, 0.05, 0.05 and 1 respectfully for the taxonomic, dependency, ontological and example statements when back propagating their loss. As observed previously, the performance of 78.21% for **bart-large-taxo-dep-onto-ex** indicates that the dependency and ontological statement are not increasing the performance.

Overall, we observe that our best model **bart-large-taxo-ex** has gained an improvement of 1.25% from 77.74% to 78.99% compared to the baseline **roberta-large-mnli-taxo-dep** on the ET-Few-NERD-Small dataset. This difference is also shown through the training process presented in Figure 4.7. Moving forward, we will use **bart-large-taxo-ex** as our best ET model, both for the training on the large ET dataset and as part of our FgNER systems. However, it is important to consider that each evaluation over the validation dataset during the course of the training process takes double the time to achieve since we make a ranking evaluation for the taxonomic statement ranking and for the example statement ranking. This results in an increase in training time from around 53 hours to 100 hours.

4.7.2 SotA Entity Typing Comparison

The goal of this section is to compare our best ET model **bart-large-taxo-ex** to other benchmarked approaches on the ET-Few-NERD-SUP dataset in terms of performances and designs. The two ET baselines previously tested are Vanilla Fine-Tuning and PLET.

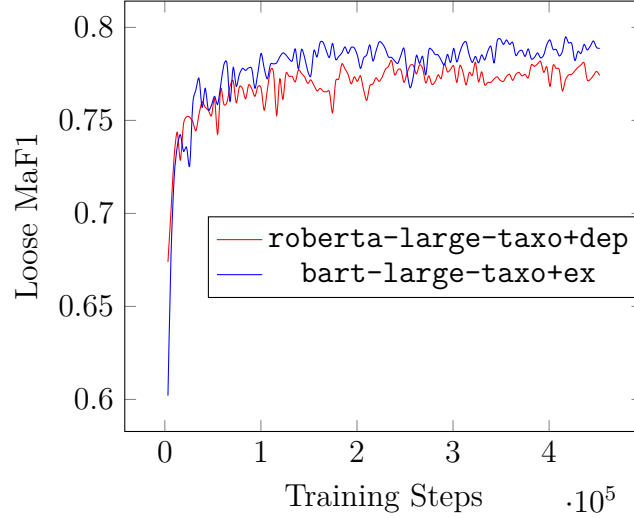


Figure 4.7 Performance of different Natural Language Inference statements during Entity Typing training.

Baselines

The vanilla Fine-Tuning (Vanilla FT) [2] for ET involves the BERT PLM, followed by a classification layer where a softmax is performed to find the most probable fine type. The input is simply the sentence with the marked named entity. It uses the embedding $\mathbf{h}_1$ produced by the first token, "[CLS]", as input for a label-specific classification layer. As a PLM, this approach uses the **bert-base-cased** optimized with a learning rate $lr = 5 \times 10^{-5}$. The training batch size is 16 and an evaluation is made for every 2000 steps during a total of 10 epochs.

PLET follows the same training settings of the fine-tuning baseline for the learning rate, batch size and evaluation. Two methods for PLET have been used, which are the soft and hard encoding methods. The soft encoding method is used with a length of 2 for the number of special tokens as such: "[Ent] [P1] [P2] [MASK]". For the hard encoding, the template "In this sentence, [Ent] is a [MASK]".

Entity Typing with Taxonomic and Example Statements

We train our best ET model **bart-large-taxo-ex** to see how our approach performs when even more data is provided. Most of the training settings remain the same from our previous experiments on ET-Few-NERD-Small. This includes the learning rate $lr = 1 \times 10^{-6}$ and the epsilon $\epsilon = 1 \times 10^{-8}$ for AdamW. Also, the batch size is 8, since 4 examples are also used here for each forward pass, where the taxonomic and example statements are generated. Because

of the size of Few-NERD, we limit the training to 59 epochs, where an evaluation of the development dataset is made two times per epoch or in other words each time after 42548 steps.

Fully-supervised Results

As shown in Table 4.7, our final performances for the `bart-large-taxo-ex` are 80.01% in fine matching and 85.98% for the loose MiF1 score.

With enough epochs, we see that our best NLI-based ET system can achieve slightly better performance than both prompt-learning approaches done with PLET and the common ET training with Vanilla FT.

Similar to what has been shown with the study done in PLET, we validated that using a prompt-based approach can obtain SotA results compared to common fine-tuning approaches. Because of the smaller number of epochs used to achieve SotA performances, PLET with hard encoding remains the best-performing strategy for ET in a fully-supervised setting.

Model	Coarse Matching			Fine Matching			Loose MiF1
	P	R	MiF1	P	R	MiF1	
Vanilla FT [2]	-	-	-	-	-	79.75	85.74
Hard Encoding PLET [2]	-	-	-	-	-	79.90	85.84
Soft Encoding PLET [2]	-	-	-	-	-	79.86	85.76
<code>bart-large-taxo-ex</code>	91.90	91.87	91.89	80.03	79.99	80.01	85.98

Table 4.7 ET performances on ET-Few-NERD-SUP. For ET, fine-grained matching is equivalent to accuracy.

Zero-shot Results

The zero-shot setting is hard over ET-Few-NERD-SUP since it does not involve any form of training of the data, while the model still needs to be tested over the same large amount of test examples. Hence, the performances of models tend to be relatively low compared to the full-supervised setting. The zero-shot results are presented in Table 4.8. We did not evaluate the Vanilla FT baseline over the zero-shot setting. Either for zero-shot or few-shot learning, this model cannot perform properly since the classification layer has random weights assigned when no training has been done [2]. As for PLET, this model performs well with their self-learning strategy. The best-developed strategy for zero-shot learning however needs additional self-supervised training. This is not directly usable as a PLM trained on a large

NLI dataset, like when directly using `bart-large-mnli-taxo` or `roberta-large-mnli-taxo`. It is important to note that `taxo` in the zero-shot means that we evaluate test examples with the use of the taxonomic statement, but no training on Few-NERD has been done with this statement.

Model	Coarse Matching			Fine Matching			Loose MiF1
	P	R	MiF1	P	R	MiF1	
PLET	-	-	-	-	-	17.55	28.39
PLET(S)	-	-	-	-	-	23.99	47.98
bart-large-mnli-taxo	44.84	44.73	44.78	22.64	22.58	22.61	33.80
roberta-large-mnli-taxo	44.63	44.55	44.59	23.02	22.97	22.99	33.89

Table 4.8 Zero-shot performances on ET-Few-NERD-SUP. "S" denotes the use of self-supervised training.

Overall, we validated and tested different NLI-based ET models over Few-NERD. Since the best model `bart-large-taxo-ex` has comparable performance to other baselines, we can start to apply our different strategies to integrate this model inside FgNER systems.

CHAPTER 5 INTEGRATING ENTITY TYPING FOR FINE-GRAINED NAMED ENTITY RECOGNITION

We experiment with two named entity detection architectures with our ET for FgNER.

The first architecture is simply to add a NED model and feed the predictions to a separate ET model. The NED model forwards each named entity span detected as a single example since the ET model is designed to type one named entity at a time. This approach is advantageous since the best independent approaches in NED and ET can be utilized.

The second architecture is to merge ET with a NER model. This is because one hypothesis is that advancement in ET might produce better typing predictions than what a NER system generates. Hence, leveraging the benefits of training and combining each of these models might improve overall performance.

5.1 Named Entity Detection with Entity Typing

5.1.1 Adapting the Dataset for Named Entity Detection

As described in the overview of our approach (see Section 3.2), the initial dataset follows the simple IO tagging scheme. We also explained earlier that we follow the IOB2 tagging scheme when training our NED model, where the possible labels for each token are either **B-entity**, **I-entity** or **O**. This is because the information given as input to the ET system needs one named entity at a time, which means that our model needs to clearly identify the start and end of the predicted named entities. The process to modify the labels of our dataset is simple. Because of the low probability of this happening, Few-NERD has been developed with the assumption that two consecutive named entities with the same fine type are nonexistent in the dataset. From this assumption, we can transform the IO labels to follow the IOB2 tagging scheme. We then simplify the annotations by relabelling each fine type tag to the general **entity** type. This enables us to fine-tune our NED model BERT-CRF-NED, presented in Section 3.1.1, with the IOB2 span-strict matching MiF1 score. No early stopping is done by our training process. We instead use the strategy of choosing the best validation checkpoint as our best model.

5.1.2 Named Entity Detection with Entity Typing During Evaluation

After training our NED model with the IOB2 tagging scheme, we are now able at inference time to take all the predicted entity spans from a single sentence and divide them into different examples to feed to our trained NLI-based ET system. With each of these examples forwarded to our ET system, rankings for all fine-grained types are generated. As done for the evaluation of our NLI-based ET system, we take the type with the highest ranking score as the most probable type to assign to this predicted named entity span. This gives us the simplest integration between these two systems, which we call BERT-CRF-NED+ET.

5.1.3 Pruning Named Entities from Type Rankings

When forwarding the predicted named entity spans, errors can be propagated to the ET model. More precisely, entity span errors can come from complete or partial false positive and negative predictions. For example, if our named entity detection predicts "games" as a named entity in "Michael Jordan has won many games.", our ET system will assign the highest scoring type even if the score is low and not actually suitable for this named entity. Hence, we can leverage our ET system to re-validate the possibility that the span predicted as a named entity by our NED model is not actually one. This can be achieved when our ET model predicts low entailment scores for all types. It is thus possible with the validation dataset to tune an additional threshold value $\epsilon \in [0, 1]$ to help us integrate these models into a coherent and more robust FgNER system. Doing this will classify the named entity as a non-entity with 0 when all ranking confidences are under ϵ :

$$\text{Label} = \begin{cases} t & \text{if } \max_{t \in T} \text{conf}(t) \geq \epsilon \\ O & \text{otherwise} \end{cases}$$

, where t is one of the type to evaluate.

This threshold serves as our Named Entity Pruning (NEP) module. As presented in the general overview with figure 3.2, adding NEP gives us the BERT-CRF-NED+ET+NEP model. This module is also present in the following integration.

5.2 Fine-grained Named Entity Recognition with Entity Typing

In this section, we describe how we integrate our NLI-based ET model with NER predictions, whether the integration is done independently or directly inside the LightNER model.

5.2.1 Adapting LightNER to Few-NERD

For the FgNER task, we use a similar process as done in ET to manually generate verbalizers over the typeset. Some examples of unique verbalization are shown in Table 5.1. From this verbalizer, each type-verbalization is decomposed as tokens taken from the model’s vocabulary. As stated previously, the word "other" is used throughout the Few-NERD taxonomy. In the case of the semantic answer space, this word would be shared amongst types that are not related in any way to each other. For example, the fine types **event-other** and **location-other** should not share a word.

Since each type containing "other" is solely defined by its siblings from the same coarse type, this is removed from the answer space. As shown in Table 5.1, the answer space for **event-other** is "event" instead of "event other". Moreover, the word embeddings of "other" add no real semantic value to the representation of the corresponding fine type. In contrast to the ET verbalizers, the semantic answer space does not take into account the ordering of the words, since a set of words is constructed for each type. However, producing this set implies that words appearing twice in the type label are now only there once. This alleviates a bias of starting with more weight for the redundant word, like "art" in **art-writtenart**, which could have led the model to over-predict this specific type instead of another fine type within the same coarse-grained category "art".

Type Label	Semantic-aware Answer Space
art-broadcastprogram	art broadcast program
art-writtenart	art, written
location-bodiesofwater	location, bodies, of, water
other-livingthing	living, thing
person-artist/author	person, artist, or author

Table 5.1 Semantic-aware answer space.

5.2.2 Independent Named Entity Recognition and Entity Typing

As visualized in Figure 5.1, this strategy proposes to use the LightNER, or in reality, any NER model to produce named entity span information for ET.

FgNER with a Token Classifier Model

As mentioned previously, the start and end of each named entity need to be clearly defined in order to leverage the ET training procedure of typing named entities indepen-

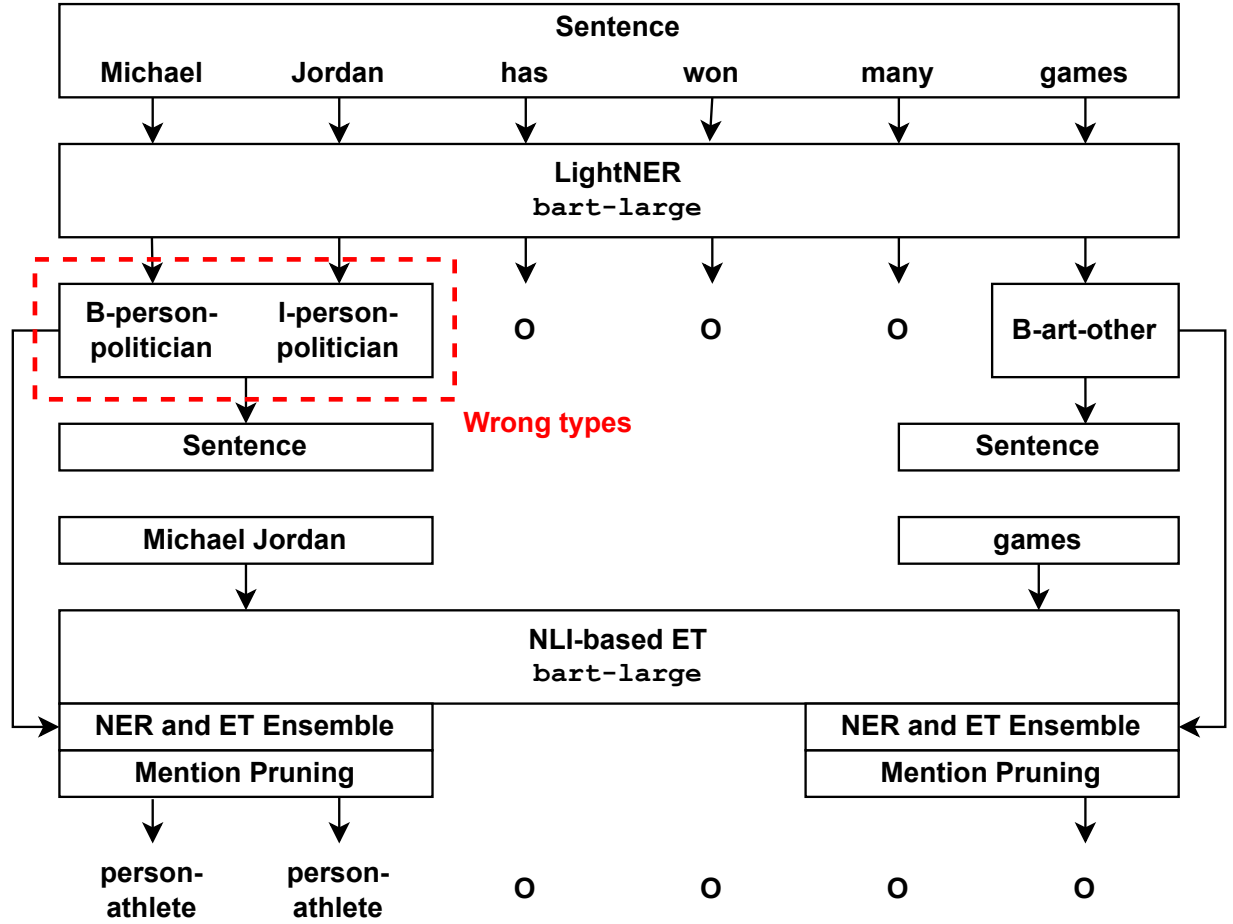


Figure 5.1 Architecture for independent Named Entity Recognition and Entity Typing

dently. We could assume that having the predicted fine types would give enough information to separate the beginning and end, as done in our final evaluation with the IO tagging scheme. However, this is not valid because consecutive predictions of the same fine types could be produced by the model when they are actually different fine types. This can be seen in the following example: "Many games have been won by the former NBA Chicago Bulls star Michael Jordan.", where the consecutive words "NBA", "Chicago", and "Bulls" could have been tagged with `organization-sportsteam`. However, the real label for "NBA" is `organization-sportsleague`. "NBA Chicago Bulls" will thus be forwarded as one single named entity to our ET model because the IO scheme did not provide any information about the start and end of detected named entities. This error cannot be currently corrected by our ET system. However, with the IOB2 scheme, the NER model can produce the labels [B-organization-sportsteam, B-organization-sportsteam, I-organization-sportsteam] for the words ["NBA", "Chicago", "Bulls"]. The tag for "NBA"

can now be corrected by our ET model to the fine type **organization-sportsleague**. Hence, our model is trained with the IOB2 tagging scheme, for a total of $1 + 66 * 2 = 133$ labels, where the 1 corresponds to the outside label **O**, and the multiplication is 2 since we need to account for the beginning label **B**-[fine-type] and the inside label **I**-[fine-type].

FgNER with LightNER

When using LightNER as the NER submodule of our independent NER and ET architecture for FgNER, its predictions are easily transformable to the IOB2 tagging scheme. This is because LightNER directly generates named entity span start and end indexes with the corresponding suggested type.

Combining Named Entity Recognition and Entity Typing Predictions

Whether we use a token classifier model or LightNER as the NER submodule, our aforementioned strategy uniformizes their predictions to the same final predictions, which we can use with our best ET model. The process for the joint prediction of the NER and an NLI-based ET model is similar to the strategy achieved for combining a span model and an ET model. A single type is generated by the NER model for a specific span. Afterwards, each entity span is forwarded to the ET model to generate a ranking over all types. We are now left with the type prediction from our NER model and the rankings from our ET model. Our goal is to make use of NER to adjust the position of this specific type in the ET type rankings. Hence, we learn a weight $\lambda_{ner} \in [0, 1]$ on the validation dataset that is used to re-rank confidences, which we call NER and ET Ensemble (NEE), as described below:

$$\text{conf}(t) = \begin{cases} \lambda_{ner} \cdot \text{conf}(t) & \text{if } t = \text{NER-type} \\ (1 - \lambda_{ner}) \cdot \text{conf}(t) & \text{otherwise} \end{cases},$$

, where t is one of the type to evaluate.

As shown in the example presented in Table 5.2, we see that when λ_{ner} takes a value above 0.5, the resulting type rankings can bring up the score of the predicted label by our NER system to the most probable fine type. If the final best-resulting value learned is 0.5, there would be no effect whatsoever in the ordering of ranks, but only in terms of their score, which would be scaled down by a factor of 2.

After fine-tuning the NEE weight λ_{ner} , we can also fine-tune the Named Entity Pruning (NEP) module. These tunings are done sequentially because at inference time we first apply NEE and then the NEP module. Understanding the effects of these two modules individually

NER prediction	ET Ranking			$\lambda_{\text{ner}} = 0.52$	
person-politician	person-athlete	0.90	#1	$(1 - 0.52) \cdot 0.90 = 0.43$	#2
	person-politician	0.88	#2	$0.52 \cdot 0.88 = 0.46$	#1
	...	...	...	...	...
	product-food	0.10	#66	$(1 - 0.52) \cdot 0.10 = 0.05$	#66

Table 5.2 Named Entity Recognition and Entity Typing ensemble.

and together with respect to the final performance will be necessary. Hence, an analysis of their resulting best weight values is important.

5.2.3 Transfer Learning of Entity Typing through the Pretrained Language Model

Our final strategy involves transfer learning. The core of transfer learning from ET to NER consists of pretraining a BART model with an ET task and using it as a frozen encoder-decoder for LightNER. The reasoning for doing this sequential transfer learning is that the NLI-based training where types are naturally integrated can bake most of the typing knowledge directly inside BART’s encoder’s and decoder’s weight, which can be used as a pre-trained model for an entity-related task like NER.

5.3 Experimental Results

In this section, we evaluate the impact of our proposed ET integrations to achieve fully supervised FgNER. First, we apply our integrations over the Few-NERD-Small dataset. Over this dataset, the main focus is to understand each component of these integrations. For Few-NERD-SUP, we can compare our integrations with our strong baselines and SotA approaches. Following our methodology, we use our best ET model **bart-large-taxo-ex** for all the ET integrations.

5.3.1 Models with ET Integrations

We presented earlier three major integration strategies of ET to form our different FgNER models. The training parameters for models involved in these three strategies are presented in Table 5.3.

The first integration consists of separating the FgNER objective into the detection and typing to reduce the complexity of our approach while training models adaptable to new taxonomies.

This is done by using the predictions of a NED model as input to our ET model, where Named Entity Pruning (NEP) is applied to remove some probable false positives. Since the ET training is already done from previous experiments, we only need to train our NED model, which is BERT-CRF-NED with the weights of **bert-based-cased**. The resulting model is BERT-CRF-NED+ET+NEP.

Our second integration is composed of independent NER and ET, where we apply NER and ET Ensemble (NEE) followed by Named Entity Pruning (NEP). In other words, we leverage NER predictions to perform a reranking of our ET predictions, which we can then use to perform pruning for these predicted spans. We use LightNER because its semantic answer space makes the model rapidly adaptable to new taxonomies [1]. LightNER is also able to achieve reasonable performance on fully-supervised datasets without updating the weights of the encoder and the decoder. Hence, using it as a NER model while integrating ET is a valid consideration. Our NER model is LightNER with **bart-large** as the pre-trained weights. The resulting model is LightNER(**bart-large**)+ET+NEE+NEP.

As for our third integration, we use transfer learning from ET to NER through the encoder and decoder’s weights of our NER model. Setting aside the use of pre-trained weights of **bart-large-taxo-ex**, our approach entirely follows the architecture and training process of LightNER. In other words, there is no generation of predictions from our ET model, which is a relative advantage since it reduces the time complexity. The FgNER representing this integration is LightNER(**bart-large-taxo-ex**).

Dataset	Learning Rate	AdamW ϵ	Batch Size	Maximum Sentence Length	Epochs Small	Epochs SUP
BERT-CRF-NED	5×10^{-5}	1×10^{-8}	16	512	30	9
LightNER	2×10^{-5}	1×10^{-8}	16	512	100	60

Table 5.3 Parameters of models involved in FgNER integrations. Small is for Few-NERD-Small and SUP is for Few-NERD-SUP. BERT-CRF-NER follows the same hyperparameters as BERT-CRF-NED. Additional hyper-parameters for the LightNER architecture follow the study [1].

5.3.2 Results of ET Integrations

We aim to understand how much our ET integrations for FgNER can improve the final performance of our model. Hence, we identify the benefit of each part of our integrations. We present our results in an incremental manner, where we start with a base model and then add some of our modules and evaluate the impact on performance as shown in Table 5.4.

The first row of each integration strategy starts with the baseline to which we compare our results.

Model	Span Match MiF1	Coarse Matching			Fine Matching		
		P	R	MiF1	P	R	MiF1
Integration 1: NED and ET							
BERT-CRF-NER	82.22	74.89	74.36	74.63	59.43	62.41	60.88
BERT-CRF-NED	82.72	-	-	-	-	-	-
BERT-CRF-NED+ET	82.72	77.02	75.95	76.48	65.44	65.32	65.38
BERT-CRF-NED+ET+NEP	82.61	77.55	75.64	76.58	65.95	65.13	65.54
Integration 2: NER and ET							
LightNER(bart-large)	82.46	73.72	77.37	75.50	62.57	65.47	63.99
LightNER(bart-large)+ET	82.46	74.36	78.05	76.16	64.19	67.18	65.65
LightNER(bart-large)+ET+NEE	82.46	74.44	78.13	76.24	64.33	67.33	65.79
LightNER(bart-large)+ET+NEE+NEP	82.40	75.53	77.57	76.54	65.45	67.03	66.23
Integration 3: Transfer Learning from ET to NER							
LightNER(bart-large)	82.46	73.72	77.37	75.50	62.57	65.47	63.99
LightNER(ET)	82.41	73.46	77.45	75.40	62.94	66.23	64.54

Table 5.4 Results of different ET integrations on Few-NERD-Small. ET is used as a shorter name for our best ET model **bart-large-taxo-ex**, while NEE stands for NER and ET Ensemble and NEP for Named Entity Pruning.

Integration 1: NED and ET. To evaluate how effective is our first integration of NED with ET, we use the baseline BERT-CRF-NER as a comparison. This is a simple fine-tuned NER model, which is also based on **bert-base-cased** like our BERT-CRF-NED and follows the same training procedure as BERT-CRF-NED. However, it is also composed of a CRF layer used to label each token with fine-grained types instead of only indicating the presence of a named entity. In Table 5.4, the overall MiF1 score of BERT-CRF-NED+ET+NER is 65.54%, which is better than the label-specific NER baseline BERT-CRF-NER with 60.88%.

Named Entity Pruning from ET rankings has slightly increased the fine-matching MiF1 score. We can see that removing the pruning drops the performance from 65.54% to 65.38%. However, it has the effect of reducing the span-matching MiF1 score, which goes from 82.72% to 82.61% in this integration. This is because removing potential false positive named entities reduces the span-matching recall of our model. However, an increase in performance happens for the MiF1 score of span-matching, coarse-matching, and fine-matching.

Integration 2: NER and ET. Overall, the performance of this ET integration is 66.23% in MiF1 score for fine-matching, while the default LightNER model for FgNER achieves 63.99%. This indicates the value that an additional ET can add in terms of performance for the FgNER system.

The major increase in performance comes from adding solely ET predictions to LightNER(**bart-large**) for a gain of 1.66% from 63.99% to 65.65%.

Adding NER and ET Ensemble has provided an increase from 65.65% to 65.79% in fine-matching MiF1 score. We also observed that the fine-tuning of the corresponding hyperparameter has produced an importance weight higher than 0.50 for the NER label. This indicates that NER predictions have provided some value to the rankings produced by ET, even though the improvements might not be considerable.

As for the previous integration, the drop in span-matching also happens when adding NEP, but still comes with an increase in fine-matching MiF1 score from 65.79% to 66.23%.

Integration 3: Transfer Learning from ET to NER. Our third integration relates to the effectiveness of transfer learning from ET to a FgNER model. Following the same training procedure and hyperparameters, we investigate how swapping the **bart-large** weights of the encoder and decoder for the weights of **bart-large-taxo-ex** can benefit the final performance of our LightNER model. Results show that this method can slightly increase the performance from 63.99% to 64.54% in fine matching MiF1 score.

5.3.3 SotA in Fine-grained Named Entity Recognition

In this section, we compare our best approaches to other SotA approaches and representative baselines evaluated on the large Few-NERD-SUP. It is also important to reevaluate each integration, even if they were not the best in terms of performance. This is because some characteristics remain different for our two Few-NERD datasets since our approach to generating Few-NERD-Small was greedy and limited. More precisely, the average length of sentences in Few-NERD-SUP is higher and the ratio of entities per sentence also differs. Table 5.6 shows our results.

Baselines. Different baselines are presented to understand where our label-agnostic FgNER model constructed from our different integrations stands. Hence, we need a baseline adaptable to new types and other baselines that have these label-specific layers that achieve SotA performances.

As our type-adaptable FgNER baseline, we use LightNER(**bart-large**). The training follows what has been said in Table 5.3.

The best SotA model with label-specific layers tested on Few-NERD-SUP is PL-Marker [23]. As stated in the literature review, this model produces good results because of markers dedicated to span representation separated from the textual representation of named entities, which enables better use of the attention mechanism.

Learning Rate	AdamW ϵ	Training Batch Size	Epochs	Group size	Maximum Span Length	Maximum Sentence Length
1×10^{-5}	1×10^{-8}	8	3	128	16	512

Table 5.5 Training parameters of PL-Marker. For a single sentence, all possible spans are split into different groups. The group size corresponds to the number of named entity span markers inside a single group. The sentence is forwarded with a single group at a time for all the existing groups, covering all named entity spans.

Because it follows a similar setting as our first integration BERT-CRF-NED+ET+NEP, we added BERT-CRF-NER as a second baseline with label-specific layers.

Model	Span Match	Coarse Matching			Fine Matching		
	MiF1	P	R	MiF1	P	R	MiF1
Baselines with label-specific layers							
BERT-CRF-NER	85.17	77.89	78.54	78.21	66.81	67.89	67.35
PL-Marker	86.86	81.15	80.61	80.88	71.32	70.49	70.90
Baselines without label-specific layers							
LightNER(bart-large)	84.63	76.36	79.07	77.69	65.98	68.25	67.10
Our Models without label-specific layers							
BERT-CRF-NED+ET+NEP	85.98	79.26	80.15	79.70	69.01	69.80	69.40
LightNER(bart-large) +ET+NEE+NEP	83.10	78.04	77.06	77.55	69.03	68.09	68.56
LightNER(ET)	84.59	76.38	79.29	77.81	66.32	68.79	67.53

Table 5.6 Fine-grained Named Entity Recognition performances on Few-NERD-SUP of our SotA models and our Entity Typing integration of **bart-large-taxo-ex**. ET is used as a shorter name for our best ET model **bart-large-taxo-ex**, while NEE stands for NER and ET Ensemble and NEP for Named Entity Pruning.

Integration 1: NED and ET. In Table 5.6, we can see that using NED with all ET integrations gives us a fine-matching MiF1 score of 69.40%, while our token classifier baseline

BERT-CRF-NER achieves 67.35%. Even if the gain is smaller compared to the gains observed over Few-NERD-Small, this integration provides the best results amongst models with type-agnostic parameters.

Moreover, we notice that the span-matching MiF1 performance is better when only doing NED with BERT-CRF-NED than doing FgNER with BERT-CRF-NER, which goes from 85.17% to 85.98%. This might be due to the larger number of labels to choose from while the most prevalent label is outside 0. Since BERT-CRF-NED captures named entities with only **B-entity** or **I-entity** which covers all types, it is easier to overcome the prevalence of 0. Another hypothesis is similar to the findings that separating each part of the label, for example from **B-person-athlete** to **B, person, athlete** is beneficial for the model [18]. More precisely, the label **B** can capture in its embedding common knowledge shared across types, that would not have been fully captured by each single fine type label with BERT-CRF-NER. We also presented in Appendix A at Table A.1 that BERT-CRF-NER with ET+NEE+NEP cannot provide better results than BERT-CRF-NED+ET+NEP.

Even if using BERT-CRF-NED+ET+NEP achieves a slight improvement over our label-agnostic baseline LightNER, the approach still lacks behind the SotA baseline with type-specific layers PL-Marker with 70.90% in fine-matching MiF1 score. It is important to note that the best final model developed with PL-Marker for Few-NERD is a two-stage model where a coarse-grained NER model reduces the number of potential spans fed to the second model which is PL-Marker. This is achieved to reduce the inference time by an important factor, which indicates that dividing a FgNER model into more specialized models is common and can lower the time complexity of the developed approaches.

Integration 2: NER and ET. When using our second integration, which means the model LightNER(**bart-large**)+ET+NEE+NEP, we see an increase from 67.10% to 68.56% compared with LightNER(**bart-large**). It cannot, however, as our other integrations, surpass the performance of the SotA PL-Marker. As a side note, these integrations have also been added to PL-Marker as presented in appendix Table A.1, where we obtain a slight improvement from 70.90% to 70.98%.

Integration 3: Transfer Learning from Entity Typing to Named Entity Recognition. Transfer learning from ET can boost performance from 67.10% to 67.53% in fine-matching MiF1 score. However, as demonstrated previously in our second integration, our best-resulting model with LightNER benefits more from other ET integrations (ET+NEE+NEP) than transfer learning with the weights of **bart-large-taxo-ex** instead of **bart-large** for

LightNER.

CHAPTER 6 DISCUSSION

In this chapter, we first present a synthesis of our experimental results that are related to the objective of developing a fully-supervised NLI-based ET model. Then, we discuss the results of our ET integrations in the task of fully-supervised FgNER.

Entity Typing. Performing ET with NLI over a fully-supervised ET dataset with a defined taxonomy has revealed that some architectural designs and choices of objectives are important in these settings, while others are not. We have shown in our results that using a model pre-trained on MNLI does not provide any additional value when performing fully-supervised ET. Nonetheless, pretraining on MNLI remains valuable for performing zero-shot classification. More precisely, it provides good performances without the need for an additional self-supervised training procedure, as done in other studies. However, the zero-shot model is biased since it seems to overpredict an entailment towards longer verbalizers, which comes from the prevalence of entailment for longer sequences during the MNLI training. We also demonstrated that the choice of the PLM is significant for NLI-based ET, where BART is better than RoBERTa. The major advantage of BART is its encoder-decoder architecture, while RoBERTa is an encoder-only model. More precisely, the denoising pretraining of BART covers tasks that require generative capabilities, which is more general than the dynamic masking task performed for RoBERTa. Hence, BART benefits from the use of its encoder-decoder architecture since NLI-based ET task is composed of many sequence classifications, which use both the encoder and the decoder to correctly perform the task.

Concerning the usefulness of each tested NLI statement, we have shown that capturing the hierarchical structure through the dependency or the ontological statements is not valuable in a fully-supervised setting. Also, removing this objective from our training process can reduce the overall training time. Validating the use of these types of statements on a fully supervised dataset was important since it is assumed to be beneficial. The proposed example statement has been valuable when combined with the taxonomic statement. Capturing the NLI-framed ET task through unique representative examples, training and evaluating types with the example statement helps the overall performance of the model. One major component of these improvements comes from the difference between the taxonomic and the example statements. As a similar procedure to prompt ensembling with weight averaging, the weight learned on the validation dataset gives appropriate importance to each of these statements.

Over the ET-Few-NERD-SUP where other methodologies have also been tested, we show that

our model **bart-large-taxo-ex** achieves comparable performance to other SotA prompt learning-based approaches.

Fine-grained NER. A simple integration of our best ET model to achieve FgNER is to use a simple NED model to feed the correct named entities to our ET model. One key strategy to enhance the integration between those two models was to prune named entities based on typing confidence of our ET model. The resulting FgNER model, called BERT-CRF-NED+ET+NEP achieved 69.20% in fine-matching MiF1 score. These results are valuable compared to what can be obtained from a similar model that achieves the detection and typing simultaneously. Moreover, using BERT-CRF-NED+ET+NEP instead of BERT-CRF-NER removes the need for a label-specific classifier.

Our second ET integration is composed of the named entity forwarding from the NER model to ET, the NER and ET type prediction ensemble and the pruning of named entities. This has provided some modest improvements over the fine-matching MiF1 score of LightNER from 67.10% to 68.56% for LightNER(**bart-large**) when adding ET predictions, NER and ET ensembling and named entity pruning. This increase is mainly coming from forwarding the correct named entity spans as input to the ET model for typing, while NER and ET ensembling and named entity pruning can provide some increase. However, even if LightNER facilitates the adaptation to new types, models like PL-Marker with label-specific classifiers are still the ones to achieve SotA performance in a fully-supervised FgNER setting.

As for our last integration that specifically applies to LightNER, we performed sequential transfer learning of the BART model trained on the NLI-based ET task as a PLM to be trained on the FgNER task. When analyzed separately from other integrations, transfer learning has some small positive effects on the results, which can be beneficial considering that the resulting model has no increase in inference time and does not require any label-specific layer.

Hence, both our two major FgNER models LightNER(**bart-large**)+ET+NEE+NEP and BERT-CRF-NED+ET+NEP achieve better results than some traditional approaches while not directly modelling type with labels. Normally, a traditional model fine-tuned with a specific dataset and taxonomy would need to be retrained from scratch if the model needs to be used for a new taxonomy. Not directly modelling these types through the fixed classifier solves this problem automatically. We also have shown that integrating our best ET model, **bart-large-taxo-ex**, into SotA models with label-specific layers can help. Nonetheless, prompt-based approaches for ET are more successful in few-shot settings than in fully supervised settings.

CHAPTER 7 CONCLUSION

We present in this conclusion a summary of how we achieved our objectives with the different proposed strategies while showing some of their major limitations. Finally, we present some improvements that can potentially solve some of the mentioned limitations and increase performance.

7.1 Summary of our work

Our first general objective was to use NLI to achieve fully-supervised fine-grained ET. We were able to accomplish this by achieving the auxiliary goals of adapting low-resource ultra-fine ET to maximize performances for fully-supervised ET and developing new NLI strategies to improve performance. We changed the architecture from RoBERTa to BART and removed the training on MNLI, which has provided better results for our setting. In terms of NLI statements, the impact of the dependency statement was also reevaluated and we observed that neither this statement nor our own similar ontological statement can bring any value in a fully-supervised setting with a strict taxonomy of types. However, prompt ensembling with taxonomic and example statements has provided some gains and formed our best ET model **bart-large-taxo-ex**.

The final objective was to propose different strategies to integrate our best ET system in the broader task of FgNER. Overall, integrating our best ET model **bart-large-taxo-ex** to form a fully-supervised FgNER model has been beneficial. More precisely, we demonstrated that our combination of the NED model with **bart-large-taxo-ex** produces better results than what the NED model with an additional type classifier layer can obtain. Whether the FgNER model is using a label classifier or not, adding different ET integrations has led to slight increases in fine-matching MiF1 scores.

7.2 Limitations

Even though we can achieve our objectives, many pitfalls come with our different strategies, which mostly come down to time complexity.

As stated above, the major problem with our best ET model is the time complexity, both for training and inference. In training, many epochs are needed before achieving complete fine-tuning. This comes from the contrastive learning objective, where a single negative example is compared to the positive example when there are many more negative examples. Hence,

it takes a lot of epochs to cover most of the negative examples available for each example, which makes the learning process a lot slower than Vanilla fine-tuning where a cross-entropy objective is used over all types.

While some benefits come from encoding both the sentence, the named entity and one type at the same time, it comes with an increase in complexity. More precisely, a single forward pass is needed for each type, which becomes really expensive when dealing with a large number of types like in FgNER. In comparison, a simple approach which uses a classifier like Vanilla Fine-tuning would only need a single forward pass to compute the score of all types. Moreover, our best NLI-based model generates a ranking with the taxonomic statement and the example statement, which doubles the time to perform the overall evaluation. This is one major issue when using prompt ensembling techniques, even though this technique can provide better results.

LightNER is memory efficient in training because only a small fraction of parameters are tuned since the encoder and decoder remain frozen throughout the process. However, the complete fine-tuning process over a fully-supervised dataset still needs around 61 hours of training compared to a traditional NER model like BERT-CRF-NER which achieve its maximum validation performance in 9 hours.

With regards to our different FgNER integrations, the NER and ET ensemble is a simple technique applicable to any NER system regardless of how they generate their predictions. On the other hand, this general ensembling strategy does not take complete advantage of NER predictions of some specific models. For instance, some NER models produce probabilities over all labels for each token. Hence, an ensemble that takes into account these probabilities and the rankings might have generated better performances.

7.3 Future Research

The aforementioned limitations have shown that there is a need to reduce the overall time complexity of our strategies.

Our NLI-based ET approach encodes the context, the named entity and a natural representation of one type at a time. Since the pretraining on MNLI did not affect the performance of our approach, we think that the major benefit of the ET strategy in a fully-supervised setting comes down to encoding types through the model.

Exploring the different combinations for encoding these elements that would reduce the computing time and might provide some gain in terms of performance. For example, encoding many types with the context and the named entity could be done at the same time, where

the score for each of these encoded types can be generated from their associated hidden representations. This would result in a smaller inference time since fewer forward passes would be needed.

A different strategy to avoid the complexity of the NLI-based approach would be to generate a type embedding that shares the same embedding space as the embedding generated from encoding the sentence with the named entity. This would mean that a single forward pass would be necessary to encode the sentence with its named entity. Then, a similarity metric with all the type embeddings could be computed to find the closest one.

One remaining problem which is disregarded by the setting in which ET takes place is to consider the dependency between the type prediction of two named entities that are located in the same sentence. Since each named entity is typed independently of other named entities, performances might be improved by jointly predicting the types of named entities.

REFERENCES

- [1] X. Chen *et al.*, “LightNER: A lightweight tuning paradigm for low-resource NER via pluggable prompting,” in *Proceedings of the 29th International Conference on Computational Linguistics*. Gyeongju, Republic of Korea: International Committee on Computational Linguistics, Oct. 2022, pp. 2374–2387. [Online]. Available: <https://aclanthology.org/2022.coling-1.209>
- [2] N. Ding *et al.*, “Prompt-learning for fine-grained entity typing,” in *Findings of the Association for Computational Linguistics: EMNLP 2022*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 6888–6901. [Online]. Available: <https://aclanthology.org/2022.findings-emnlp.512>
- [3] E. F. Tjong Kim Sang and F. De Meulder, “Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition,” in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, 2003, pp. 142–147. [Online]. Available: <https://aclanthology.org/W03-0419>
- [4] D. Gillick *et al.*, “Context-dependent fine-grained entity type tagging,” *ArXiv*, vol. abs/1412.1820, 2014.
- [5] X. Ling and D. Weld, “Fine-grained entity recognition,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26, no. 1, pp. 94–100, Sep. 2021. [Online]. Available: <https://ojs.aaai.org/index.php/AAAI/article/view/8122>
- [6] N. 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. [Online]. Available: <https://aclanthology.org/2021.acl-long.248>
- [7] A. Vaswani *et al.*, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
- [8] L. Zhuang *et al.*, “A robustly optimized BERT pre-training approach with post-training,” in *Proceedings of the 20th Chinese National Conference on Computational*

- Linguistics*. Huhhot, China: Chinese Information Processing Society of China, Aug. 2021, pp. 1218–1227. [Online]. Available: <https://aclanthology.org/2021.ccl-1.108>
- [9] M. Lewis *et al.*, “BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 7871–7880. [Online]. Available: <https://aclanthology.org/2020.acl-main.703>
- [10] R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. [Online]. Available: <https://aclanthology.org/P16-1162>
- [11] A. Radford *et al.*, “Improving language understanding by generative pre-training,” 2018.
- [12] J. 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 Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: <https://aclanthology.org/N19-1423>
- [13] Hugging Face, “Bert for token classification,” https://huggingface.co/docs/transformers/main/en/model_doc/bert#transformers.BertForTokenClassification, accessed: 2023-01-19.
- [14] J. Pennington, R. Socher, and C. Manning, “GloVe: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. [Online]. Available: <https://aclanthology.org/D14-1162>
- [15] Y. Nie *et al.*, “Improving named entity recognition with attentive ensemble of syntactic information,” in *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov. 2020, pp. 4231–4245. [Online]. Available: <https://aclanthology.org/2020.findings-emnlp.378>

- [16] S. Pradhan *et al.*, “Towards robust linguistic analysis using OntoNotes,” in *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 143–152. [Online]. Available: <https://aclanthology.org/W13-3516>
- [17] O. Agarwal *et al.*, “Interpretability analysis for named entity recognition to understand system predictions and how they can improve,” *Computational Linguistics*, vol. 47, no. 1, pp. 117–140, Mar. 2021. [Online]. Available: <https://aclanthology.org/2021.cl-1.5>
- [18] T. Kato *et al.*, “Embeddings of label components for sequence labeling: A case study of fine-grained named entity recognition,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. Online: Association for Computational Linguistics, Jul. 2020, pp. 222–229. [Online]. Available: <https://aclanthology.org/2020.acl-srw.30>
- [19] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML ’01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, p. 282–289.
- [20] H. Zhu *et al.*, “Fine grained named entity recognition via seq2seq framework,” *IEEE Access*, vol. 8, pp. 53 953–53 961, 2020.
- [21] J. Yu, B. Bohnet, and M. Poesio, “Named entity recognition as dependency parsing,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 6470–6476. [Online]. Available: <https://aclanthology.org/2020.acl-main.577>
- [22] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 11 1997. [Online]. Available: <https://doi.org/10.1162/neco.1997.9.8.1735>
- [23] D. Ye *et al.*, “Packed levitated marker for entity and relation extraction,” 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. 4904–4917. [Online]. Available: <https://aclanthology.org/2022.acl-long.337>
- [24] X. Li *et al.*, “A unified MRC framework for named entity recognition,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online:

- Association for Computational Linguistics, Jul. 2020, pp. 5849–5859. [Online]. Available: <https://aclanthology.org/2020.acl-main.519>
- [25] —, “Dice loss for data-imbalanced NLP tasks,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 465–476. [Online]. Available: <https://aclanthology.org/2020.acl-main.45>
- [26] L. Cui *et al.*, “Template-based named entity recognition using BART,” in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online: Association for Computational Linguistics, Aug. 2021, pp. 1835–1845. [Online]. Available: <https://aclanthology.org/2021.findings-acl.161>
- [27] L. Wang *et al.*, “Instructionner: A multi-task instruction-based generative framework for few-shot ner,” *ArXiv*, vol. abs/2203.03903, 2022.
- [28] C. Raffel *et al.*, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. [Online]. Available: <http://jmlr.org/papers/v21/20-074.html>
- [29] N. Houlsby *et al.*, “Parameter-efficient transfer learning for NLP,” in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 2790–2799. [Online]. Available: <https://proceedings.mlr.press/v97/houlsby19a.html>
- [30] E. Choi *et al.*, “Ultra-fine entity typing,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 87–96. [Online]. Available: <https://aclanthology.org/P18-1009>
- [31] H. Dai, Y. Song, and H. Wang, “Ultra-fine entity typing with weak supervision from a masked language model,” 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. 1790–1799. [Online]. Available: <https://aclanthology.org/2021.acl-long.141>
- [32] M. A. Hearst, “Automatic acquisition of hyponyms from large text corpora,” in *COLING 1992 Volume 2: The 14th International Conference on Computational Linguistics*, 1992. [Online]. Available: <https://aclanthology.org/C92-2082>

- [33] B. Li, W. Yin, and M. Chen, “Ultra-fine entity typing with indirect supervision from natural language inference,” *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 607–622, 2022. [Online]. Available: <https://aclanthology.org/2022.tacl-1.35>
- [34] A. Williams, N. Nangia, and S. Bowman, “A broad-coverage challenge corpus for sentence understanding through inference,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1112–1122. [Online]. Available: <https://aclanthology.org/N18-1101>
- [35] M. A. Yosef *et al.*, “HYENA: Hierarchical type classification for entity names,” in *Proceedings of COLING 2012: Posters*. Mumbai, India: The COLING 2012 Organizing Committee, Dec. 2012, pp. 1361–1370. [Online]. Available: <https://aclanthology.org/C12-2133>
- [36] A. Poliak *et al.*, “Collecting diverse natural language inference problems for sentence representation evaluation,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 67–81. [Online]. Available: <https://aclanthology.org/D18-1007>
- [37] R. Weischedel *et al.*, “Ontonotes release 5.0 ldc2013t19,” *Linguistic Data Consortium, Philadelphia, PA*, vol. 23, 2013.
- [38] W. Yin *et al.*, “Universal natural language processing with limited annotations: Try few-shot textual entailment as a start,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 8229–8239. [Online]. Available: <https://aclanthology.org/2020.emnlp-main.660>

APPENDIX A APPENDIX

Here we present other FgNER models that have been produced on the fully-supervised FgNER, where some use our integration strategies. The baseline trained with the development of the Few-NERD dataset, BERT-Tagger [6], follows the same architecture as the token classifier model BertForTokenClassification on HuggingFace [13]. Moreover, we present the results of combining both our second and third integrations with LightNER(ET)+ET+NEE+NEP since they are complementary. Since PL-Marker achieves the best results over Few-NERD, we also tested our second integration of NER with ET on PL-Marker, where we can see a small improvement in MiFi score from the 70.90% that was obtained on PL-Marker 5.6 versus PL-Marker+ET+NEE+NEP with 70.98%.

Table A.1 Performances of other models on Few-NERD-SUP. ET is used as a shorter name for our best ET model `bart-large-taxo-ex`, while NEE stands for NER and ET Ensemble and NEP for Named Entity Pruning.

Model	Span Match	Coarse Matching			Fine Matching		
	MiF1	P	R	MiF1	P	R	MiF1
BERT-Tagger	-	-	-	-	65.56	68.78	67.13
BERT-CRF-NER+ET+NEE+NEP	84.66	79.08	78.65	78.87	69.10	69.22	69.16
LightNER(ET)+ET+NEE+NEP	82.70	78.10	76.47	77.27	69.20	67.71	68.45
PL-Marker+ET+NEE+NEP	86.70	81.41	80.29	80.85	71.48	70.48	70.98