DOCTORAL SCHOOL
UNIVERSITY OF MILANO-BICOCCA

Department of Informatics, Systems and Communication

PhD program in Computer Science

Cycle XXXV

Domain Adaptation in Fine-grained Entity Typing

Manuel VIMERCATI

Registration number: 793553

Supervisor: Prof. Matteo PALMONARI

Tutor: Prof. Gianluigi CIOCCA

PhD Program Director: Prof. Leonardo MARIANI

ACADEMIC YEAR 2021/22

Contents

Chapter 1

Introduction

Information extraction [CL96; McC05; Gri15; SGC09] is a complex task aimed at extracting structured information from text [CL96; Gri15]. This information is typically structured around *entities* and represented in statements of a formal language such as RDF. These statements represent facts about the entities referenced in the text, for example, *Barack Obama*. The facts can describe the entity's class membership, such as *Barack Obama* being a `Politician`, the relationships between entities, such as *Barack Obama* being born in *Honolulu*, and properties of these entities, such as *Barack Obama* being born in 1961.

A good Information Extraction System (IES) is beneficial for each downstream task in NLP, such as Question Answering, Entity Linking, and Ontology Population from Text. An IES is composed of different modules applied in cascade [CL96; McC05; Gri15]. Although the modules can be implemented independently, they are highly related because the output of one module is used as the input of the following module. According to [McC05], an IES can be decomposed into the following steps: Segmentation, Classification, Association, Normalization, and Coreference Resolution. The Segmentation step identifies the boundaries of a piece of information, such as a named entity, a domain string, a verb, or a simple word. The Classification step groups the segments into classes, which can be semantic, such as types/hypernyms of named entities (Person, Organization, Location, etc.), related to the text source (HTML tags), or syntactical, such as Part-of-Speech tags. The Association step determines relations between segments. The Normalization step reduces the variance of information by using standard formats depending on the segment's class, for example, normalizing dates using the same format or arranging person's names in a standard order. Finally, the Coreference step collapses redundant segments to ensure that the same information is represented by the same unique segment.

Recognizing and classifying entities in a given document is a fundamental capability of an IES. Annotated documents that specify the types of entities can be used for various downstream applications, such as semantic search in complex documents [BBH+16] and question answering [Zha+20d], even if more complex elaborations are not necessary. On the other hand, the extracted entities are typically used as input for more complex processes, such as relation extraction [Kua+20] or knowledge base population [Hos+19].

This PhD thesis focuses on a subtask of entity extraction, specifically the Classification step, which is called Entity Typing (ET). ET involves assigning types to an entity that has already been identified in a given sentence, selecting types from a given vocabulary. For instance, if the Segmentation step has identified the segment "Joe Biden" in the sentence "Joe Biden is the current president of the U.S.A.", a suitable solution for the task would be to assign types such as `Person`, `Politician`, and `President`.

Assigning types to entities found in a given sentence has proven to be beneficial in various NLP tasks, such as Question Answering [Han+17], Ontology Population [Hos+19], Entity Linking [Che+18], and Relation Extraction [Kua+20]. Moreover, different articles [Cho+18; OD20] demonstrate how the diversity and granularity of such types have an impact on those NLP tasks. Over the last decade, several ET benchmark datasets have been proposed to assess the ability of state-of-the-art models to tackle ET. The increasing size and granularity of the types' vocabulary have led to the task being named Fine-grained [LW12] and Ultra Fine-grained Entity Typing [Cho+18] (FET and UFET).

Like all IES, FET models must be applied in different contexts, from news articles to encyclopedic articles, from blogposts to product descriptions on e-stores. Type hierarchies are often domain-specific, and the annotation policies for building training datasets are specifically designed for each application domain. Moreover, automated annotation methods, often used to annotate large quantities of data, can be based on various NER and NEL systems to find entities in sentences, as well as different knowledge bases to extract and annotate the correct types. The various combinations of these factors in application domains can be seen as instances of a Variety Space [Pla16], where each application domain can be described as a region in this space. Note that regions that correspond to different domains can overlap. Given the usefulness of FET approaches in downstream tasks, it is expected that FET approaches should be able to recognize all types considered useful in a given domain. FET approaches are based on machine learning and deep neural networks and are trained on large datasets, ranging from hundreds of thousands to millions of examples. These models are data greedy and tend to overfit, which is why big datasets are needed to learn to classify between highly populated type ontologies. Although training domains often contain common types like `Person`, `Building`, `Location`, and `Animal`, a dedicated dataset is usually necessary to cover all types considered useful in a specific domain. However, building a dataset from scratch is known to be expensive and time-consuming.

Adapting a trained IES to a new domain is an activity that lies into the Transfer Learning research field, in particular, *transductive transfer learning* [PY10]. In *transductive transfer learning*, the source and target tasks remain the same, but the source and target domains differ in their underlying distribution [Rud+19]. The general application scenario of Domain Adaptation (DA) relies on cases where there is a rich source domain that can be used to perform an initial training of the IES, and also a low-resource target domain, where labeled data are absent or scarce [RP20]. Such rich source domains are costly to build and the initial training on them can be very heavy. Poor target domains instead are very common in real-case scenarios where very specific information needs to be extracted. Domain adaptation is largely explored in NER, where multiple techniques are proposed and challenging scenarios, for example class incremental [Mon+21], are defined. FET literature is focused instead on the in-domain scenario and domain adaptation is not covered yet. However, domain adaptation is particularly interesting in FET, since a foundational concept of this task is the helpfulness of fine-grained types. Following this concept, developing techniques to adapt a FET model to recognize new and possible *'finer'* types is necessary to transfer the abilities learned in rich source domain to more specific target domain, where new types and *'finer'* types can be central. For example, recognizing the typology of various products of an e-commerce store based on the description is a very specific but fundamental ability to scale on large database, as well as recognizing different jobs, drugs, organizations, and weapons in the juridical domain.

Adapting an already trained IES to a new domain is an application of Transfer Learning, specifically transductive transfer learning [PY10]. In transductive transfer learning, the source and target tasks remain the same, but the source and target domains differ in their underlying distribution [Rud+19]. Domain Adaptation (DA) is a general application scenario that relies on rich source domains for initial training of the IES and low-resource target domains where labeled data are absent or scarce [RP20]. Rich source domains are expensive to build and training on them can be time-consuming, while poor target domains are common in real-world scenarios where specific information needs to be extracted. DA is widely explored in NER, where various techniques and challenging scenarios, such as class-incremental NER, are defined [Mon+21]. However, in the FET literature, domain adaptation is not yet covered. Nonetheless, it is particularly interesting in FET since the concept of fine-grained types is fundamental to this task. Developing techniques to adapt a FET model to recognize new and possibly finer types is necessary to transfer the abilities learned in rich source domains to more specific target domains where new and finer types can be central. For instance, recognizing various products in an e-commerce store based on their descriptions is a specific yet crucial ability for scaling on large databases. Similarly, recognizing different jobs, drugs, organizations, and weapons in the legal domain is essential.

In my research, I am exploring the problem of domain adaptation in fine-grained entity typing. The main question that drives my thesis is whether it is possible to support domain adaptation for entity classification in the challenging scenario of fine-grained classification, which is a topic that has not been studied extensively. I am investigating this problem by leveraging novel adaptation techniques that integrate explicit knowledge representation and deep neural networks. Specifically, I am studying to what extent models trained on rich source domains can be adapted to new, typically resource-poor target domains, especially when the source and target domains partially overlap.

To answer this main question, I have first formalized the problem and defined three different domain adaptation scenarios. Based on this analysis, I have broken down the main question into three sub-questions, each of which relates to a specific scenario, as follows:

- **Model reuse.** To which extent can we reuse an existing FET model on a dataset that represents a novel data distribution without additional training? In the scenario of model reuse, we can observe the effects of domain differences on the behavior of a FET model. These differences can be expressed in terms of language distribution (e.g., scientific vs. informal) or the presence of new entities or words, as well as annotation policies (e.g., pronouns being considered entity mentions in the target domain but not in the source domain, indicating that the model was not trained to classify entities denoted by pronouns). This scenario is relevant because it may be that a user is only interested in a subset of the entity types considered in an existing model, and can select those that match their interests in their own textual documents. Additionally, this scenario represents a more extreme case of distribution shift [Shi+20], providing valuable insights into the difficulty of the problem.

- **Model specialization.** In the model specialization scenario, a FET model that has been trained on a source domain needs to be specialized to recognize new types in a target domain. The peculiarity of domain specialization is that we assume each target type is a specialization of one source type, for example, `Athlete` is a specialization of `Person`. The relationships between the known types and the new specialized types are a central notion to exploit in order to transfer the classification capabilities from the source to the target domain. This scenario is relevant because subtypes are often essential information for a given domain. For example, to obtain a juridical FET model useful for distinguishing between `Advocate`, `Judge`, and `Part`, a FET model that is already trained to distinguish `Person` is a more suitable starting point than a randomly initialized network. However, exploiting explicit hierarchical relations during training is a challenging task.

- **Full-fledged domain adaptation.** In the full-fledged domain adaptation scenario, the objective is to adapt a FET model trained on a source domain to recognize new types in a target domain where equivalence, specialization, generalization, and disjunction relations can exist between source and target types. To achieve this, it is crucial to consider the relationships between the source and target types and leverage them to transfer and adapt the knowledge from the source to the target domain.

The original contributions of this Ph.D. thesis are:

- the formal definition of domain adaptation in FET.
- the description of three different scenarios for domain adaptation in FET, with the analysis of their relevance in real cases.
- the analysis of a model reuse and the analysis of the related issues.
- the extension of a denoising approach proposed in FET literature to alleviate the problem of *noisy annotations* and its evaluation in the model reuse scenario.
- the definition and evaluation of novel techniques to perform specialization low-resource FET scenarios. Such techniques exploit different hierarchy encoding strategies with the neuro-symbolic approach Knolwedge Enhanced Neural Network (KENN) [DS19].
- the definition and the evaluation of techniques to perform the adaptation to a target domain in low-resource FET scenarios. Such techniques exploit different relations between source and target types through the usage of KENN [DS19].

With the above contributions, this thesis presents a comprehensive study of a novel task: domain adaptation in FET. To the best of my knowledge, the problem has not been addressed before in FET, since the literature is focused on proposing sophisticated encoders, classifiers, and denoising methods applied to *in-domain* FET. Domain adaptation is generally underinvestigated in information extraction, even if the described domain adaptation scenarios are common in the design of real world information extraction systems. Moreover, the approaches proposed to face domain adaptation in FET reflect the first attempt to use a neuro-symbolic integration technique in FET.

The thesis is organized as follows. The state of the art of FET and domain adaptation in information extraction is discussed in detail in Chapter 2. The focus of the thesis then shifts to domain adaptation in FET in Chapter 3, where a formal definition of tasks and a categorization of domain adaptation in FET is presented. Chapter 4 explores the concept of model reuse in partially overlapping domains, investigating the performance of FET models on target domains. The impact of noisy annotations on model reuse is also examined. Finally, Chapter 5 and 6 respectively present a study of FET model specialization and full-fledged domain adaptation in FET, both using Neuro-symbolic models. The results and conclusions of the study are presented in each chapter, providing valuable insights for future research in the field of domain adaptation FET. Conclusions and Future work are discussed in Chapter 7.

Chapter 2

State of the Art

In this chapter, the state-of-the-art*s* of Fine-grained Entity Typing (FET) and Domain Adaptation (DA) in information extraction are described. Regarding FET, firstly, the problem itself and its usefulness in downstream tasks are illustrated, afterwards, a categorization of approaches and techniques to build benchmark datasets are reported. Regarding domain adaptation in information extraction, the dedicated section opens with the different definitions of *domain* and with the definition of *domain adaptation*. Thereafter, its application in Named Entity Recognition and on language models is described.

2.1 Preliminaries

In this section, the notions of *Named Entity Recognition*, *entity mention*, *type & type hierarchy*, and *encoder/encoding* are described. These components are crucial to understand the Fine-grained Entity Typing task (FET), its literature and its usefulness.

2.1.1 Definitions

Named Entity Recognition (NER). Named entity recognition is the task of finding entity mentions in a given plain text. NER algorithms return the span, i.e., two indexes that indicates the start and the end of the entity mention, and often a categorization with one or more types. These types are often coarse-grained, such as `Person, Organization, Location, Date,` and `Miscellaneous`.

Entity Mention. In Natural Language Processing sentences are the fundamental input from which information is extracted. Information can be linguistic, i.e.

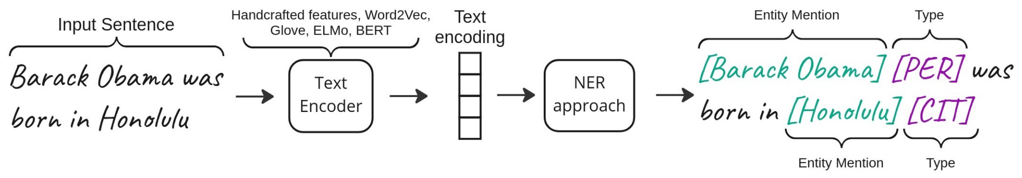

Input Sentence
Barack Obama was born in Honolulu
Handcrafted features, Word2Vec, Glove, ELMo, BERT
Text Encoder
Text encoding
NER approach
Entity Mention
Type
[Barack Obama] [PER] was born in [Honolulu] [CIT]
Entity Mention
Type

Figure 2.1: Schematization of the Named Entity Recognition (NER) task: given a sentence, an encoder is used to produce a dense vector representation of it, then an automatic NER approach finds entities and coarse types.

regarding the composition of the sentence and its correctness, or semantical, i.e. linked to the meaning of the sentence as a whole or to the meaning of a sentence part. A crucial step in the information extraction concerns the finding and the contextualization of named entities, i.e., part of a text that identifies real-world object, such as a person, a location, an organization, a product, etc., that can be denoted with a proper name. For example, in the sentence "*Barack Obama was born in Honolulu*" the word sequences "*Barack Obama*" and "*Honolulu*" are entity mentions, since they refer to a `Person` and to a `City`.

Type & Type Hierarchies. A type represents a concept and is used to classify and organize information found in text, or to enrich information while processing text. Referring to the previous example, the entity mention "*Barack Obama*" can be classified as a `Person`, in the same way "*Honolulu*" can be classified as a `City`; information about both these types is present directly in the sentence. With additional information more types can be also used to characterize each entity mention, e.g. `Politician`, `Lawyer` for "*Barack Obama*" and `Capital` for "*Honolulu*". Types are often organized in hierarchies that model the relations of superclass and subclass.

Text Encoders and Text Encodings. Different encoding techniques have been proposed to digest natural language sentences with automatic approaches. The goal of these techniques is to represent syntactical and semantical similarities and differences between sentences. In general, NLP techniques are based on a *text encoder* that maps the problem input (i.e. a sentence, a set of sentences) to an *encoding*, a feature-based representation used as proxy to represent the input during the information extraction process. The features can be handcrafted, e.g., based on human defined rules, categorization of the words, and statistics on word occurrences, or can be sub-symbolic, based on methods that represent words and sentences with dense vector representations. Sub-symbolic methods are based on the distributional semantics theory [Har81] and in the last decade impressive goals have been reached, first with word embeddings techniques (e.g., Word2Vec [Mik+13a; Mik+13b],

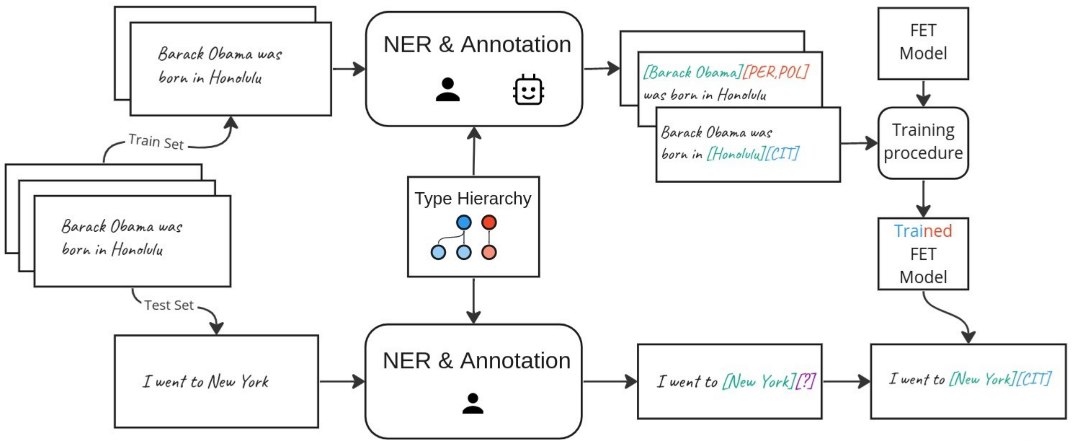

Barack Obama was born in Honolulu
NER & Annotation
[Barack Obama][PER,POL] was born in Honolulu
Barack Obama was born in [Honolulu][CIT]
FET Model
Training procedure
Trained FET Model
Train Set
Barack Obama was born in Honolulu
Type Hierarchy
Test Set
I went to New York
NER & Annotation
I went to [New York][?]
I went to [New York][CIT]

Figure 2.2: Schema of the Fine-grained Entity Typing task and the FET model training: given a set of documents and a type hierarchy, a training dataset is obtained by applying NER and the found entities are annotated (both manually or automatically). Then a FET model is trained on the obtained dataset and is tested on manually annotated examples from the same document collection.

GloVe [PSM14]) that accounts for global information about words and documents, then with recurrent networks (RNNs, LSTMs, ELMo [Pet+18]), and architectures based on attention [Dev+19], designed to extract contextual information exploiting both word positions and meanings.

2.2 Fine-grained Entity Typing

2.2.1 Problem definition

Fine-grained Entity Typing (FET) [Sun95; CR98; Fle01; LW12] is the task of assigning to each entity mention (a continuous span of text that refers to a real-world entity) in a document or a collection of documents a semantic type from a predefined taxonomy. This taxonomy can be represented with a set of types and relationships between them, composing a tree or a poset. An example of the FET task and of the standard setup to train a FET model is shown in Figure 2.2. From the same figure can be seen that NER is a requisite of FET, since finding the span of the entity mentions is not considered part of the typing task. In general, NER is applied to individuate the span of entity mentions and assigning coarse-grained types like `Location, Person`, and, `Organization`. These types are often removed from FET benchmarks and not considered since their semantics can shift when the type set increases its depth and size.

FET can be divided into two sub-tasks: mention-level FET and entity-level

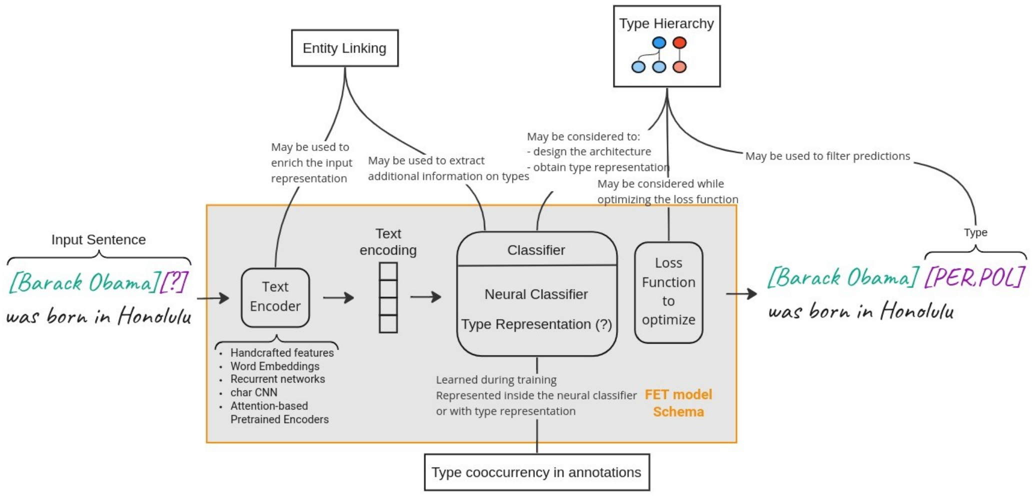

Entity Linking
Type Hierarchy
May be used to enrich the input representation
May be used to extract additional information on types
May be considered to:
- design the architecture
- obtain type representation
May be considered while optimizing the loss function
May be used to filter predictions
Input Sentence
[Barack Obama][?] was born in Honolulu
Text Encoder
Text encoding
Classifier
Neural Classifier
Type Representation (?)
Loss Function to optimize
Type
[Barack Obama] [PER,POL] was born in Honolulu
Handcrafted features
Word Embeddings
Recurrent networks
char CNN
Attention-based Pretrained Encoders
Learned during training
Represented inside the neural classifier or with type representation
FET model Schema
Type cooccurrency in annotations

Figure 2.3: Inside the orange box an abstraction of FET models is shown, derived from the most recent FET survey [Wan+22b]. Outside the orange box, input and output are represented, additional resources (Entity Linking, Type Hierarchy, and Type Cooccurrence in annotation) and their usage in the Literature is reported.

FET. Mention-level FET involves assigning a semantic type to an entity mention within a specific context, while entity-level FET involves finding all the possible semantic types for an entity across all contexts. Entity-level FET is often used for knowledge base construction and completion.

The main problems related to FET are the high multiplicity of the types (from dozens to hundreds, often thousands with Ultrafine-grained Entity Typing [Cho+18]), their hierarchical relationships, and the noise injected by automatic dataset construction techniques.

A particularity that distinguishes FET from other classification problems is that it is a multilabel problem, with dozens to hundreds of types organized in a hierarchy. This is a crucial aspect of the problem, since hierarchy is both an explicit information source, that can be used to correct automatic predictions with a post hoc procedure, e.g., by applying transitive closure (i.e. adding to the prediction all ancestors of the predicted types), and an implicit information source, since it affects manual and automatic annotations of the input corpus, and during learning the model can approximate type hierarchy from type cooccurrence.

In 2022, a survey from [Wan+22b] described FET task, FET approaches, proposed FET benchmark datasets to validate FET approaches, the problem of noise in input sentences and annotations, and the open problems in FET. An abstraction of a FET approach can be derived from the survey, shown in Figure 2.3. The survey classifies FET approaches by describing and categorizing:

- Feature representation models applied to FET, including:
  - *hand-crafted features*, to represent macro properties of the input
  - *mention representation with average pretrained word embeddings*, to account for commonsense meaning of single words
  - *recurrent networks*, to account for the order of the input
  - *char CNN*, to have a morphological representation of words
  - *context representation with recurrent networks*, to represent the context of the mention accounting for word order
  - *contextualized representation extracted from pretrained language models based on recurrent neural networks or attention* to exploit knowledge extracted by large language models
- Usage of type embeddings:
  - *neural classifiers not based on type embeddings* learn the link between input and labels and implicitly learn labels similarities from their c0ooccurrences
  - *type embeddings* learned during training or imported from other setups account for label similarity, and are used to create a joint space between input and labels used to compose the prediction
- Loss functions optimized to train models
- Classification paradigm adopted: unsupervised, semi-supervised, supervised with flat, local, local per parent note, global classifier, and supervised with collective or joint models

The articles previously cited consider FET in an in-domain setting [Wan+22b] (example in Figure 3.1), where the target and source domain are equivalent; thus, training documents and training types are used to approximate the annotation function, obtaining a model that is effective when applied on documents from the same *document source* to recognize already learned types. Therefore, it is not clear how previous work can handle use cases described in Section 3.2.1.

In this thesis, an uncovered aspect of FET is explored: the explicit usage of types relations (equivalence, inclusion, and disjunction) to drive the adaptation of FET models from their training domain to a new target domain through the use of a Neuro-Symbolic Integration approach. The adaptation of FET models trained in a source domain to a real-world target domain with a partially overlapped taxonomy is a future direction also pointed out by the literature [Wan+22b; Qia+21]. A fundamental problem to solve in order to handle the differences between domains, is to manage the differences between types used in different domains; thus, a categorization of FET approaches, complementary to the one presented by [Wan+22b], is proposed in this chapter.

In the categorization described in this chapter, FET approaches that proposed solutions to exploit dependencies between types are organized and described. Type dependencies can be partitioned in three categories: **hierarchy-aware**, that use explicit information like taxonomies or rules, **cooccurrence-aware**, where type cooccurrence in annotation is implicitly learned and represented during training, and **link-aware**, where additional information are inherited from external resources, for example from Knowledge Bases through entity linking. Table 2.1 offers an overview of the approaches described in this section. The usage of these dependencies is exemplified also in Figure 2.3.

Hierarchy-aware

The hierarchy of target types in a FET task can be used in different ways:

- **Classifier design:** Multiple approaches [RN10; Yos+12; LHS19; Ren20] train different classifiers for different layers (or branches), e.g., train a distinct classifier for each branch of the hierarchy and a classifier to choose the branch (obtaining 29 classifiers) [RN10], and propose mechanisms to combine them. Another example can be found in [LHS19; Ren20], where distinct classifiers are trained to classify for each level of the hierarchy, and each classifier is based on the output of the previous one. A similar idea is proposed by [CCV20]: a top-down learning-to-rank approach is used to train a model with inference thresholds that depend on type depth in the hierarchy. The limitation of these approaches is that they impose constraints on the composition of prediction, i.e. only single branch predictions [RN10; CCV20], or always predictions at each level [LHS19; Ren20]. A different usage of hierarchy to design the network is proposed by [Li+21]: type representations are computed by combining each type vector with the vector of its ancestors, in this way the representations are linked together during the optimization of the model, and the change in the representation is propagated through its family tree.

- **Prediction filtering:** Different approaches use the hierarchy to eventually filter the set of predicted types [YGL15; MCG16; Ren+16a; Ren20], or to recursively compute the prediction following the hierarchy [RN10; Yos+12; AAA17], often under the assumption that all the predicted types should belong to the same branch. This assumption could exclude some annotation in the test set and it is not respected in most of the training sets. The single-branch assumptions allow the author to redefine FET as a single-

	Hierarchy-Aware	Co-occurrence-Aware	Link-Aware
[RN10]	Classifier design	-	-
[Yos+12]	Classifier design	-	-
[YGL15]	Prediction filtering	Vector optimization	-
[MCG16]	Prediction filtering	-	-
[Ren+16a]	Hierarchy-driven loss	Vector optimization	Type similarity
[Ren+16b]	Inherited from external resources	Vector optmization	Type similarity
[AAA17]	Prediction filtering	Inherited from [Ren+16a]	-
[XB18]	Hierarchy-driven loss	-	-
[Mur+18]	Inherited from external resources	-	KGE
[LHS19]	Classifier design	Box optimization	-
[LJ19]	-	Vector optimization	-
[Dai+19]	-	Vector optimization	Encoder design
[Shi+20]	-	Vector optimization	-
[Ren20]	Classifier design - Inference Filtering	Vector optimization	-
[CCV20]	Prediction Filtering	Vector optimization	-
[DSL20]	-	[Dai+19] is used as backbone FET model	Encoder design
[Li+21]	Classifier & Encoder design	Vector optmization	-
[DSW21a]	-	[Dai+19] is used as backbone FET model	Encoder design
[Qia+21]	-	[LJ19] is used as backbone FET model	-

Table 2.1: Overview of entity-typing approaches described in Section 2.2 organized by the use of type hierarchy, type co-occurrence, and entity linking

label classification task (i.e., classify with models only the leaves and infer ancestors); this can lead to confirmation bias, as stated in [Ren20].

- **Hierarchy into encoder or hierarchy-driven losses:** Some approaches combine information from the type hierarchy with additional signals. Many of these methods also propose learning type representations: [Li+21] adds sentence-level cross-mention type interaction that includes hierarchical relations; [Ren+16a; Ren+16b] complements information from the hierarchy with information extracted from a KB to learn type similarities and eventually weight errors in the training process; loss weighting is also used by [XB18].
- **Hierarchy inherited from external resources:** Some articles refer to an external hierarchy instead of the given one: [Ren+16b] uses Wikipedia hierarchy to express similarity between types. [Mur+18] uses type representations derived with knowledge graph embeddings.

Co-occurrence-aware

The multilabel fashion of the FET problem allows researchers to exploit co-occurrence between labels in the training set. Some methods exploit them by learning type representations that are expected to represent types that co-occur more frequently with similar representations. Co-occurrence is useful since it can give similarity information about different branches in the hierarchy; for example in the BBN dataset [WB05; Ren+16a], a large amount of entities belongs to both types `BBN.LOCATION` and `BBN.GPE` (Geo-Political Entity), thus these types are often used in the same annotation.

The basic idea [YGL15] to represent types as embeddings in FET is to introduce a layer in order to learn type representations by optimizing their representation during training, and by replacing the classification module with threshold-based inference. Based on this idea, different approaches [Ren+16a; Ren+16b; AAA17; LJ19; Shi+20; Ren20; Qia+21] proposed more sophisticated methods to learn type representations. [Ren20] uses softmax to predict types from a single branch achieving best performance on FIGER datasets among this family of approaches that only consider type dependencies.

Link-aware

Entity linking can be used [Ren+16a; Dai+19; DSL20; DSW21a] to add information about types at the cost of defining a mapping between the FET type hierarchy and the ontology used in the linking phase. Additionally, an Entity Linking algorithm is required and as shown in the literature, the performance of the proposed link-aware models are highly dependent on the Entity Linking algorithm. Moreover, the inclusion of such external information requires more complex encoders and training procedures, however these approaches showed to be the state-of-the-art [Dai+19; DSL20; DSW21a] for very noisy datasets, such as OntoNotes [Gil+14; Shi+17a]

Entity Linking can be used in different ways:

- **Type Similarity:** [Ren+16a; Ren+16b] uses a linking process to extract the KB type similarity and use it to weight errors during the training.
- **KGE:** [Mur+18] represents each type with a vector learned using knowledge graph embedding, after mapping types to concepts in a knowledge base.
- **Encoder design:** The current state-of-the-art approach [DSL20] in different benchmark datasets is based on using additional information in the encoder. The principal idea of [Dai+19], the first article to propose entity linking, is to use an independent entity linking algorithm that uses only the mention (disregarding the context) and returns a list of types of the entity in a KB, together with an entity linking score. Types from the KB are then represented via one-hot encoding and concatenated to the contextualized entity representation from the encoder, together with the entity linking score. This architecture is used as backbone network in [DSL20; DSW21a]. Entity Linking is central in adding out-of-the-box information, independent on the corpus, on its annotation policies, and on type vocabulary. This is useful both as denoising, since the entity linking process gives an additional information able to correct errors in the dataset, and as augmentation, since a KB usually have a various type vocabulary and can be considered reliable.

In this thesis entity linking is not used since the necessity of a knowledge graph precludes the usage of such techniques in very specific domains, like the biomedical one, where new and rare entities are fundamental.

2.2.2 Collecting and cleaning training data

Most existing FET methods are trained on weak labels that are automatically generated by Distant Supervision (DS) [DSW21b], a technique that automatically produces annotated data mimicking a manual annotation procedure. With DS, the semantic tags in the KB are mapped to the type taxonomy; this inevitably induces label noise in the training data [HWZ21]. Although label noise is the main problem with DS, there are other issues. One problem related to automatically generated training data is that on average there are fewer than two labels for each sample; on the other hand, a human annotated sample has on average 5.4 labels [DSW21b]. Secondly, there is no approach that can create a large number of training samples for pronouns mentions. Lastly, it is difficult to obtain types that are highly dependent on the context.

Approach	**Assumptions on Noise**		**Countermeasures**	
	Heuristic	Mathematical	Cleaning	Noise-aware training
[Gil+14]	Rules	-	Rules	-
[Ren+16a]	Single-path	-	-	Loss
[Ren+16b]	Single-path	-	-	Loss
[AAA17]	Single-path	-	-	Loss
[XB18]	Single-path	-	-	Loss
[Zha+18]	-	Hierarchy	-	Attention
[Che+19]	Single-path	-	-	Clustering
[OD19]	Missing labels	-	Filter and relabel	-
[Wu+19]	Single-path	-	-	Clustering
[Ren20]	Single-path	-	-	Loss
[Shi+20]	Single-path	-	-	VAT
[Zha+20a]	Single-path	-	-	Statistical
[Ali+20]	-	Hierarchy	-	Spatial
[Ali+21]	-	Hierarchy	-	Spatial

Table 2.2: Denoising methods categorization

Denoising

Label noise in FET can be classified in two types: (i) Out-of-Context (OOC) noise, and (ii) Overly Specific (OS) noise [XB18]. The former occurs when an entity is annotated with correct types, but unrelated to the context; the latter occurs when the assigned types are not only unrelated to the context, but finer than all types related to the context.

While approaches trained on noisy datasets built using DS achieve good performances on manually annotated test sets [LW12; Gil+14; Cho+18; Wei+11; Shi+17b], different authors [Ren+16a; Ren+16b; AAA17; XB18; Zha+18; Zha+20a; Che+19; Wu+19; Shi+20; Ali+20; Ali+21; Gil+14; OD19] identified the noise in the training data as the main factor to overcome to achieve better performance in both benchmark and downstream tasks.

Over years, several authors proposed denoising approaches in FET. Each approach can be categorized (see Table 2.2) by specifying two attributes [ZW04; FV14; FK14; Qui86; NOF10]: (i) the presence of assumptions on the nature of noise or of a mathematical noise model in the design of the denoising approach; (ii) countermeasures adopted, i.e., cleaning the dataset and producing a denoised dataset or noise-aware training.

Assumptions on noise

- **Heuristic based assumptions:** These type of assumptions made on nature of noise can be further classified in: hierarchy-based assumptions and assumptions based on a contextual definition of noise. Hierarchy-based assumptions rely on the idea that examples annotated with types that lie on a single path in the hierarchy are always correct and can be considered as clean examples; this assumption is also called *single-path assumption.* Other assumptions can consider the definition of Out-of-Context (OOC) or Overly Specific (OS) labels, already mentioned in the previous paragraph.

The **single-path assumption** can be found in [Ren+16a; Ren+16b; Ren20; AAA17; Zha+20a; XB18; Che+19; Wu+19; Shi+20], in particular: [XB18] not only assumes the correctness of single-path annotations, but designs ad hoc loss functions to face OS and OOC. [Zha+20a] assumes that the correct types of an entity mention have to lie on a single path but may not be present in the noisy annotation from DS; furthermore, they represent the labels with a pseudo-truth label distribution. A slightly different set of heuristics can be found by inspecting the **rules** in [Gil+14], where an entity mention cannot be annotated (i) with sibling types or (ii) with a type that is not equivalent or a sub-type of the type selected by a coarse-grained NER approach; [OD19] assumes that **missing labels** in annotations are a central problem in FET and designs a training technique to discard too noisy examples, and relabel the others.

- **Mathematical:** Some authors do not make explicit assumptions on the nature of noise, instead they rely on **hierarchical knowledge** [Zha+18; Ali+20; Ali+21] and represent it with mathematical models to define noise-tolerant approaches assuming that hierarchical information implicitly reduces the impact of label noise. [Zha+18] exploits hierarchy using path-based type representation with attention to compute the likelihood of each type given a sentence; [Ali+20; Ali+21] exploit hierarchy using the representation of hierarchy in euclidean ([Ali+20]) or hyperbolic ([Ali+21]) space to refine type representations. [Zha+20a] uses a **statistical** model (pseudolabel distribution) to represent label distribution and obviate to the noise introduced by deviant examples.

Countermeasures

- **Cleaning:** Instead of defining noise-tolerant FET models and train them on noisy data, [Gil+14; OD19] propose approaches to obtain a denoised dataset, which will then be used to train a generic FET model. [Gil+14] defines three rules to detect and remove irrelevant labels from training examples, while [OD19] proposes a method that produces a clean dataset, by training two models: a **filtering** model to discard noisy examples and a **relabeling** model to produce a new set of labels given an annotated example.

- **Noise-aware training:**

  The majority of denoising methods in FET are based on model-level noise management, i.e., they are designed to be noise-tolerant during training [Ren+16a; Ren+16b; Ren20; AAA17; XB18; Che+19; Wu+19; Shi+20; Zha+20a]. [Ren+16a; Ren+16b; AAA17; XB18] train a FET model using different **losses** for single-path and multi-path examples, by encouraging the prediction of one single correct type in both cases. Similarly, [Ren20] incorporates hierarchy into the loss and tries to mimic human top-down inference of type labels. Leveraging the hierarchical structure of types, FETHI automatically denoises the training data, and infers entity type layer by layer in a top-down manner. [Zha+18] exploits hierarchy by defining path-based **attention** to propagate hierarchic dependencies. [Che+19] defines a **clustering**-based two-step inference function where (i) given a batch of examples, a FET model produces a likelihood for each type and (ii) single-path examples propagate their likelihoods with a Markov model built with spatial clustering to modify predictions of multi-path examples. Similarly, [Wu+19] uses a graph-based **clustering** to predict the most likely type. [Shi+20] splits training examples into clean and noisy ones and uses Virtual Adversarial Training (**VAT**) to regularize the ET model. [Zha+20a] defines a **statistical method** that relies both on DS annotations and sentence-based pseudo-truth label distribution to magnify the similarity between sentences instead of sentence-type relations. [Ali+20; Ali+21] imposes hierarchy-based **spatial** similarity while representing the labels of an example using Euclidean or Hyperbolic space.

2.2.3 Open Problems in FET

The last sections of the 2022 FET survey [Wan+22b] point to *tail types* and *new entities* as two of the main problems for FET approaches.

Tail type problem relates to the difficulty of correctly predicting rare types, i.e. types with a low quantity of annotated examples. New entities problem instead, relates to the tendency of the encoders to memorize entities during training and their difficulty to obtain good performance on unseen entities.

As discussed in many FET articles, performance of FET models for a specific type is highly dependent on the presence of examples of that type in the training set [NC15]. Moreover, [Cho+18] observed that their model often prefer coarse types instead of fine or ultrafine ones since coarse types have more examples. Methods to improve performance on tail types include the usage of pre-trained language models such as GloVe [PSM14] in [YD18] and in zero-shot approaches [MCG16; Xio+19; Zha+20c]. Representing types by exploiting annotation cooccurrence with spatial representation (vectors [YGL15; Ren+16a] or box embeddings [Ono+21]) is a data-driven way to deal with tail types; obtaining an independent representation from Wikipedia [Obe+19] also helps with tail types. Entity Linking [Dai+19; DSL20; DSW21a] is another method for dealing with tail types, since additional information from KB provides data augmentation.

Regarding new entities, the main affected approaches are entity-level ones [Obe+19; Mur+18; YS17], this is expected since those methods gather information about each entity from the entire dataset. The sensibility of other approaches on new entities depends heavily on the encoder and on the usage of contextualized word embeddings, since it is assumed that the context of new entities for a given type is similar to the context of already seen ones. Prompting is also a valid way to enrich the information for new types or new entities [Din+21].

As future directions, the most recent FET survey [Wan+22b] points at joint models for multitasking where FET is learned together with Entity Linking, NER, or Relation Extraction to increase the generalization of the information, avoiding task overfitting and noise overfitting. The addition of document-level information as done in [DSL20] increases the source of evidence and improves input representation; more sophisticated encoders are also hypothesized to be useful in this direction. Another important direction is the research for the adaptation of models to real case scenarios, as also argued in [Qia+21; Mon+21].

2.3 Domain Adaptation in Information Extraction

In this section *Domain Adaptation in Information Extraction* literature is reviewed to understand the problem, its implications and techniques to face it.

In this section, first different definitions of *domain* are described and discussed, then the domain adaptation problem is defined. Then techniques to perform domain adaptation in NER and domain adaptation of language models are reviewed.

2.3.1 What is a domain?

As reported by [Pla16], in NLP there is no common ground on what constitutes a domain. Three definitions for what a domain is can be extracted from the literature [Pla16]: Canonical vs Non-canonical, Dataset-Domain and Variety Space.

The **Canonical vs Non-canonical** definition partitions all possible domains in two categories: a *canonical domain* is typically defined by a well-edited (English) newswire corpus, a *non-canonical* domain is typically defined by a corpus which comes from social media and or is non-well edited. This definition is highly contrasted by [Pla16] that reports different studies which underline how different communities can be seen as different domains independently if their documents come from *canonical* or *non-canonical* source, and so the classical domain notion is ineffective.

Dataset-Domain is not explicit pointed as a definition, it is a suggestion proposed by [RP20]; since *canonical vs non-canonical* definition is ineffective, the authors ask themselves if each dataset defines its own domain, since each dataset has its sources and its document collection approaches. The authors leave this suggestion as a question, but they point as Variety Space as a possible answer.

The **Variety Space** is a theoretical notion introduced by [Pla16]. In the variety space a corpus is seen as a subregion or a sample of the variety space. A corpus is a set of instances drawn from the underlying unknown high-dimensional variety space, whose dimensions (or latent factors) are fuzzy language and annotation aspects. These latent factors can be related to different attributes, such as genre (e.g., scientific, newswire, informal), sub-domain (e.g., finance, immunology, politics, environmental law, molecular biology) and socio-demographic aspects (e.g., gender), among other unknown factors, as well as stylistic or data sampling impacts (e.g., sentence length, annotator bias).

In this thesis, following [RP20] variety space is used to answer to the Dataset-Domain question: since each dataset can be seen as a sample of the variety space, each dataset defines its own domain, but this sample is not exclusive and different datasets can share some regions of the variety space.

2.3.2 Domain Adaptation

After defining what a domain is, we now analyze how domain adaptation is defined and why it is needed. Domain Adaptation is an instance of a case of Transfer Learning [PY10] called *transductive transfer learning*. In transductive transfer learning, the source and target tasks remain the same, but the source and target domains differ in their underlying probability distribution [Rud19]. Domain Adaptation was initially defined [PY10] as a setting in which no labeled data in the target domain are available, while a lot of labeled data in the source domain are available. [Rud19] defined *supervised domain adaptation*, in which a small amount of annotated examples for target labels is available. Recalling the user stories in Section 3.2.1, the *model reuse* lies on the original Domain Adaptation Definition [PY10], since a partial mapping between source and target types and no target labeled data are given. *Model specialization* and *full-fledged model adaptation* instead lie on Supervised Domain Adaptation definition [Rud19], since a small amount of target labeled data is available.

Domain adaptation is deemed important for practical reasons and sustainability: the initial train (or pre-train) of the model can be costly in terms of required resources, data, electricity, and time; all these computational and natural resources cannot be expected to be always available to train a model for a new domain. In particular, [RP20] points to domain adaptation as a solution for labeled data lack in a domain; [Hed+21] discusses about general NLP approaches in Low-Resource Scenarios defining a categorization for them: (i) scenarios in which task-specific labels availability is scarce, (ii) scenarios in which unlabeled language or domain-specific text availability is scarce, (iii) scenarios in which auxiliary data availability is scarce

2.3.3 Domain Adaptation in Name Entity Recognition

Domain Adaptation techniques applied in Named Entity Recognition (NER) are fundamental to understand since NER and FET are highly related task: both refers to entity mentions contained in a given text and both uses a given taxonomy to classify the entity. Due to the reduced size of common NER systems, a manual mapping between source and target type vocabulary can be built and used directly, without the necessity of training a model on the target domain; a policy is often required to avoid subjective mappings. Building direct mapping is a useful technique when one-to-one or many-to-one mappings can be defined while preserving the semantics of source types after mapping them to target types. Clearly, the performance of manual mapping approaches on the target domain is strictly dependent on the performance on source domain, since there is no training on the target domain. A slightly different solution is presented in [ANC08], where the authors build *ad hoc* syntactical and/or semantic rules to extract types of interest and show how these rules can be adapted to target domain. In this case rules are manually defined instead of mappings. However, rule-based approaches are time-consuming due to the necessity of fine-tuning specific rules for each domain.

A specular idea regards the representation of the input, approaches based on this idea enrich input representation of target domain with examples from other domains, in order to avoid overfitting on small dataset and often to obtain *domain-agnostic* representations. [Liu+20] borrows examples from source dataset to enrich target dataset (that is assumed to be small) based on examples similarity. Some articles use co-training to jointly learn NER models for multiple domains: [Li+13] for english and chinese NER and [Mun+12] for common and bio NER A simple yet effective general approach that follows this idea is proposed in [Dau07]: the authors propose to train different models on different domains and to refine their encoded input representations using a common representation space obtained by the concatenation of individual encodings. The common representation is then used to enhance to train a model on a target domain (that can be part or not of the domains used in the first phase). This approach is applied not only on NER, but also on POS-tagging. Similarly, [ANC08] trains an encoder based on hierarchical feature tree using multiple domains at the same time to obtain a domain-independent representation. [KMC16] exploits linguistics differences between source and target domains to define an active learning algorithm that is used to substitute word representation from source domain with domain specific representation. This kind of approaches is very useful in those cases where a word drastically changes its meaning when used in different domains (e.g., *goal* in *sport* and *finance*).

Different approaches refine a model trained on a source domain to adapt it to the target domain: [LDS18] trains a neural network on a source domain and use the learned parameters or a portion of them to initialize the network to be trained on the target domain. Similar idea is used in [GB18] for biomedical NER. [Qu+16] trains a chain-CRF on the source domain and uses its parameters joint with correlations between source types to enhance training on small target domain. [LL18] adapts a NER model based on LSTM and CRF trained on a source domain by training an additional word embeddings encoder (paired with frozen char-based LSTM trained on source domain) and an additional classification layer before the CRF. [CM19] trains a network on the target domain and injects in it the last layer activations generated by the model trained on the source domain. Similarly, [Noz+21] uses a trained NER model paired with a trainable BERT instance to adapt from a source to a target domain.

In this thesis the experimented approaches cover different ideas presented in this section: type mappings are used to partially cover a new target domain with overlapping types without an additional training, when additional training is used to cover all the target types and/or to match the target input distribution, the neurosymbolic integration framework KENN [DS19] is used to express relations between source and target types. Moreover, domain adaptation is inspected under the light of ontology specialization, where the new types that have to be learned are subtypes of already known types.

Class-Incremental NER

Class-Incremental NER [Mon+21] is defined as a particular instance of Continual Learning [Hu+19] where the task is fixed (NER) and the classes to recognize changes over time. Since the task is fixed, the NER model has to be able to recognize all classes learned during each step of the Class-Incremental NER process. The main difficulties of this task are: avoiding forgetting, since the ability of classes seen in the first training steps has to be kept, and avoiding overfitting on a specific training step, since general knowledge could be useful in next steps. [Mon+21] defines Class-Incremental NER by highlighting its differences with Domain Adaptation in NER faced by [CM19]: in Domain Adaptation NER a rich domain can be used as target domain and its labels can be overlapped with the source ones. In contrast, [Mon+21] defines Class-Incremental NER as a problem where a NER model has to be trained to recognize new classes using annotated data for the new classes and maintaining its behavior on already known classes. Moreover, the annotated examples used to learn known classes cannot be used in successive steps.

Class-Incremental NER was firstly faced by [Mon+21] using Knowledge Distillation paradigm: to learn new types for NER, a new *student* NER model is initialized by copying the encoder and the linear classifier from an already trained *teacher* NER model. Then the teacher predicts on new examples and the student has to both mimic the teacher predictions while minimizing the loss on new classes. Similarly, [Wan+22a] uses Knowledge Distillation to train a CRF-classifier (state-of-the-art architecture in NER) under Class-Incremental setup, imposing the availability of only few shots for the new classes.

As described in Chapter 3, Domain Adaptation in FET, faced in this thesis, is also related to Class Incremental NER and to the techniques used to face this task. The idea of copying the parameters of a teacher model is crucial for retaining knowledge during the incremental phase. In addition, because Fine-grained Entity Types have a high coverage, exploiting class relationships becomes important in FET because it is more likely that a new class is a specialization of a previously known class rather than a completely separate new class. However, in this thesis both the specialization and the disjunction case are covered by the adoption of KENN [DS19].

2.3.4 Domain Adaptation with Language Models

Domain adaptation can regards the adaptation of NLP encoders to the language distribution of the target domain. In this case ad hoc approaches are defined depending on the encoder, on the task and on the target domain. In this section, the adaptation of Pretrained Language Models [Dev+19] is described, since these models reached the state-of-the-art in each NLP task [SLL22].

Fine-tuning on downstream tasks

A straightforward way to use pretrained language models in downstream tasks is to attach a task-specific module (i.e., classification module for text classification or language understanding) as described in the original article [Dev+19; RKR20]. This setup is called *fine tuning* [LKB20; RKR20] and transformer layer's weights can be frozen or not. The main advantage of this approach regards the massive size of pretraining corpus and the quantity and variety of language information and general knowledge in it; this information are generally not contained in a task specific dataset, where language and information in examples are biased by the task. As shown in [Hao+20] during fine-tuning of transformers layer, the attention in last layer is more involved in the optimization, while after the fine-tuning hidden representation is more different in middle and last layers. Authors also shown that first layers are resilient to catastrophic forgetting.

Alternative fine-tuning methods are proposed in literature: in [SC19; KS19] weighted sum is used to combine different hidden layer representations. [YZ19] combines hidden and output representation using a dedicated architecture.

Tokenizer Adaptation and Supplementary Language Modeling

Pretrained language models based on transformers often use subword tokenizer as first module of the encoder architecture, for example `bert-base` [Dev+19] has a tokenizer composed of the 32k most common words and subwords in English. This imposed tokenizer can be limited if a word very important for a domain is represented by its subwords [RKR20; Hon+21], for example, in the biology domain there are very long words that are split into multiple subwords (e.g., *corticosterone* is split in `co ##rti ##cos ##ter ##one` tokens [Hon+21]). Moreover, for specific domains, like the science domain, the language distribution can be very different from the one extracted during the original pretraining [Dev+19; Gu+21; BLC19; Lee+20].

Several articles focus on the adaptation of the Tokenizer: [ZDV18; PWS20] shows how adding tokens that are not present in the tokenizer but are central for a specific domain/task largely improves performance. [Zha+20b] adds to the vocabulary a token for each out-of-vocabulary word with frequency above a threshold. Similarly, [Tai+20] adds 12k wordpieces to a vocabulary and train them with additional training. [PWS20] doubles the vocabulary of `bert-base-cased` (from 29k to 60k) and initialize the new tokens using word2vec [Mik+13a; Mik+13b] vectors projected to the tokenizer space, no further training is used. [Hon+21] generalizes this idea by identifying which tokens must be added using *fragment score* that ranks tokens based on their coverage when used to tokenize the domain corpus (number of tokenized words over number of words in the corpus). Then tokens with higher fragment scores are added to the vocabulary, initialized as the average of their subtokens, and trained to refine their representation to deal with both domain and original meaning using a regularized loss.

The idea of further training a language model using corpora specific to a particular domain or task is discussed in [Als+19; Gur+20; Lee+20; CHM19; HR18; PFB18; Sun+19; Log+19; HE19], in particular [Gur+20] defines *domain-adaptive pretraining (DAPT)* [Als+19; CHM19; Lee+20] as the training of the MLM on the a specific domain and *task-adaptive pretraining (TAPT)* [Log+19; HE19; HR18; PFB18; Sun+19] as the training of the MLM on an unlabeled corpus extracted from a task datatset; the authors show how matching the domain used in DAPT and the task to face leads to performance improvement and how do not match the domains lead to performance decrease, moreover show the differences between DAPT and TAPT and the evidence that the two pretraining are complementary.

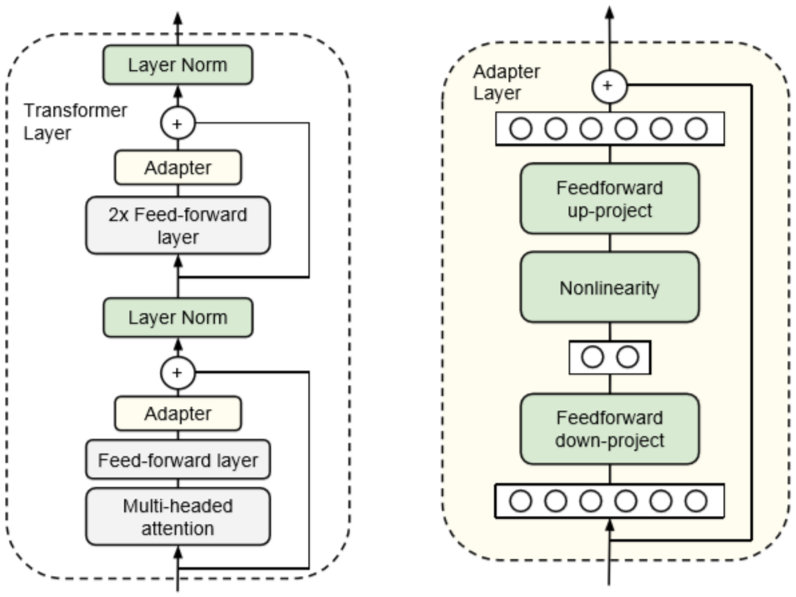


Figure 2.4: Schema of Adapters and their position inside transformer layer. Image from the origin article [Hou+19]

Adapters

Adapters [Hou+19] offer an alternative way to adapt pretrained large language model based on transformers to a specific task: one [Hou+19] or two [Pfe+20b] bootleneck adapters are inserted inside each transformer layer, these adapters compress the hidden representation using a `reduction factor` and expand back the representation to the original size. The fundamental idea is that adapters maintain only the fraction of information useful for the task. A key feature of adapters is that they are parameter-efficient since only their parameters are trained and the original pretrained parameters are kept frozen. Another key feature of adapters is that only their parameters have to be stored, reducing the size of trained models. In addition, [Pfe+21] shows how multiple adapters can be trained and their output *fused*.

Adapter are initialized with $0s$, in this way in the first steps of the training the skip-connection (see Figure 2.4) ensure that the original encodings are used. Adapters' parameters are then optimized on the specific task.

Adapters have been used in NER [He+21; Liu+21b; Che+22] to adapt language models and in low resource scenarios. In this thesis, Adapters are used to adapt a pretrained BERT encoder to FET domain, preliminary experiments showed how adapters match the performance of heavier fine-tuning training routine also in FET dataset, moreover their adoption is useful to face limits in GPU size (fine-tuning multiple transformers layers has high GPU requisites due to the high number of gradients to store during the backward) and limits in storage (persisting only adapters parameters reduced hard disk usage by a factor of 30 in the experiments of this thesis).

2.4 Domain Adaptation in FET: an open problem

Domain adaptation is explored in different NLP scenarios, but FET literature only cover in-domain scenario, where the FET model is used on data reflecting language and type distributions learned during training. This preference of the authors may be related to the difficulty of the FET task, as can be seen in Section 2.2, a large fraction of proposed approaches are based on the introduction of innovative encoders that mix different information; another aspect that is largely covered is denoising, a central problem in this research field due to the use of distant supervision algorithms to obtain large annotated datasets. Enhanced encoders and sophisticated denoising techniques are necessary to develop robust models, capable of recognizing the correct type for a given entity mention by choosing from a large type hierarchy. However, sophisticate encoders and in general sophisticated FET models are often trained on large datasets, with hundreds of thousands or millions annotated examples. Large annotated datasets are not expected to be present in all application domains, and the more specific the domain, the more complex and expensive the annotation process to produce annotated datasets. An interesting outlier approach in FET is [Obe+19], where a zero-shot FET model is designed without knowing the type hierarchy but by exploiting the type description in Wikipedia. However, with only the type description is difficult to model the differences between language distribution in different domains, or the differences in annotation policies, or also the handling of new entities, unseen during the initial encoder training.

Adapting a trained FET model to a specific application scenario is a necessity highlighted in [Qia+21] and in the most recent FET survey [Wan+22b]. Adapting a FET model by adding new types or expanding its ability on various documents and annotation policies is congruent with the idea that more information about fine-grained types is central to enrich the input of downstream tasks, as argued in Section 2.2. Moreover, FET can be also used in knowledge base population pipelines, thus recognizing new entities of known types and recognizing entities of new types is a central necessity. Moreover, the problem is challenging due to the common richness of fine-grained hierarchies and the subtle differences between very fine-grained entities (for example, discerning between *advocates* and *judges* requires a very sensible method). The problem of domain adaptation in FET can be faced by designing domain-agnostic FET models, suitable for the usage in disparate domains, or by designing domain adaptation techniques that exploit model capabilities on the source domain to favor the learning of new types useful in the target domain. Exploiting model capabilities on the source domain is central in low-resource scenarios, where the data scarcity makes training from scratch of state-of-the-art techniques a difficult task. The richness of types in FET domains enable the definition of equivalence, specialization, generalization, and disjunction relations between source an target types. In the next chapters, these relations are used to characterize different scenarios in domain adaptation in FET and are injected during the adaptation process with a neuro-symbolic integration approach.

Chapter 3

Domain Adaptation in Fine-grained Entity Typing

Domain Adaptation in Fine-grained Entity Typing is a central problem when models trained on benchmark datasets have to be used in real-case FET scenarios or downstream tasks with specific types. Moreover, [Qia+21] argued that FET approaches trained on benchmark datasets must be adapted for real-world applications of FET, since each application scenario relies on specific fine-grained types, and typically there is no one-to-one mapping from the application ontology to the benchmark ontology,

The combination of NER, FET and Domain Adaptation in FET is particularly useful in the Knowledge Base population use case, as stated in [Mon+21] where is argued that real-world NER systems must be continuously adapted to new classes. Given an Information Extraction system able to recognize entities as instances of types from a given vocabulary, if there is the necessity to recognize new classes (both regarding already represented entities or new ones), developing techniques to adapt the entire system both keeping its behavior on known classes and extending its capabilities on new classes by exploiting knowledge about already known classes is a central way to avoid costly training from scratch and annotation of big portion of data for each class.

In this chapter, the tasks of FET, the concept of Domain in FET, and FET model are formalized; additionally, some major problems of DA in FET are depicted. Then, Domain Adaptation in FET is categorized four categories, one trivial and three explored in chapters 4, 5, and 6. These categories are also described with use case scenarios. DA in FET is then formalized and the usefulness of relations between types in source and target domain is discussed and their exploitation is formally described.

3.1 Formal Definitions: FET and Domain Adaptation in FET

In this section, Fine-grained entity typing and domain adaptation in fine-grained entity typing, the two tasks faced in this thesis, are defined in a formal way. This formalization is useful to highlight the differences between domains in FET and how type dependencies between source and target domain can be exploited to favor the adaptation. This formalization is used in following chapters to describe techniques adopted in the experiments. The formalization is inspired from Variety Space theory from [Pla16], Entity Typing literature, and Domain Adaptation in NER, explained in chapter 2.

3.1.1 Fine-grained Entity Typing

Fine-grained Entity Typing (FET) is the task of classifying an entity mention found in a sentence by assigning types from a given vocabulary composed of dozens to hundreds types, often organized as a hierarchy. For example, given the sentence "*The former president* ***Barack Obama*** *was born in Honolulu*" and knowing that the word sequence "*Barack Obama*" identifies an entity, an expected result from a FET model is the assignment of types like `Person`, `Politician`, and `President`. A way to obtain this classification is to train a model on the entire vocabulary, another is to predict only the most fine type (e.g., `President`) and infer its supertypes using the given hierarchy (e.g., `Politician` is a supertype of `President` and `Person` is a supertype of `Politician`). In the following formalization it is assumed that a FET dataset always has a hierarchy (tree or poset), i.e. a general type *thing* is always present in the given vocabulary and can be considered as an ancestor of all other types. This assumption allows to include the *borderline case* where each type in the vocabulary identifies a root of a hierarchic structure and there are no hierarchic relations between types. In other words, forests of trees are considered as linked to the same common ancestor *thing*.

3.1.2 Formalization of Domain in Fine-grained Entity Typing

Recalling the Variety Space theory described in Section 2.3.2, this section contains a formalization of what a domain is in Fine-grained Entity Typing and how it is related with FET models and datasets proposed to train them.

A domain $\Delta^T = (D, G^T, \alpha)$ can be described by three components:

- A collection of documents D that is a set of sentences drawn from a *source*. The source can be described as an instance of a variety space that is characterized with attributes like: the *language*, the *vocabulary* (i.e. the set of words in the corpus), the *publishing year*, the *words distribution*, general information about *authors* and *class of document* (news article, scientific paper, novel, blogpost and so on). Each sentence x of D can be described as a tuple $x = \{x| < e, c >\}$ where e is an entity and c is its context. Recalling Figures 2.1 and 2.3, e is *Barack Obama* and c is the rest of the sentence.

- A type graph $G^T : (T, E), E = < t_i, t_j >: t_i \prec t_j, \{t_i, t_j\} \subseteq T$ that represents a set of types T and their relations E. In particular the relations E are order relations that describe hierarchical relations between t_i and t_j. For example, if t_i is `Politician` and t_j is `Person`, the relation $t_i \prec t_j$ states that all *politicians* are also *persons*.

- An annotation function α that applies a set of annotations policies to a sentence $x \in D$ to identify and classify with T the mention e. Formally, $\alpha(D, G^T) = X : \{x| < e, c >\}$, thus the annotation function is fundamental to obtain the input of a FET problem, i.e., a sentence with a mention.

A dataset designed for supervised training of FET models can be described as $X^T = \{< x, Y > |x \in X, Y \subseteq T\}$, that is a set of examples x annotated with a subset of the types in T. X^T is often composed of X^T_{train} and X^T_{test} used respectively to train and test a FET model; X^T_{train} represents the union between the training set and the validation set. The partitioning between train and test set is often performed on X and not on X^T; in fact, the train partition can be annotated using automatic as well as manual annotation strategies, while the test partition is generally annotated manually.

Hierarchy and Co-occurrence in benchmark datasets: in the most common benchmark datasets proposed to validate FET approaches, annotations Y respect the hierarchy, i.e., if $t_i \prec t_j$ and $t_i \in Y$, then also $t_j \in Y$. Another important aspect is that unrelated types t_i, t_j can co-occur in the same annotation Y, formally it can happen that $t_i \in Y$, $t_j \in Y$, and $< t_i, t_j > \notin E$, that is $t_i \not\prec t_j \wedge t_j \not\prec t_i$.

The annotation function is dependent on types: the extraction of the entity mention usually strongly depends on the types, since the entities are found basing to the ontology of interest; the presence or the absence or a type in T can in fact pose a bias on the annotators. For example, in the benchmark dataset BBN [WB05; Ren+16a] a type for *Body Part* is missing, while the benchmark datasets FIGER [LW12] and OntoNotes [Gil+14] respectively have types `/other/body_part` and `/body_part`. Since in BBN the type is missing it is expected that *body parts* present in the BBN corpus are not annotated. This fact justifies the presence of G^T as argument of the function *alpha*.

3.1.3 Formalization of Entity Typing Model

An entity typing model $\mathcal{M}$ can be interpreted as a function $\mathcal{M} : X \rightarrow \mathcal{P}(T)$ that predicts a set of types $\hat{Y}$ extracted from T when an example $x :< e, c >, x \in X$ is given in input to it.

A model is parametrized by Θ, parameters Θ are estimated with training the model on the dataset X^T_{train}, thus the optimized parameters Θ^{Δ^T} are dependent on the domain Δ^T represented by the annotated dataset X^T_{train}.

Since types T are fundamental to characterize FET domains Δ^T and annotated datasets for FET X^T, from now trained models $\mathcal{M}$ with parameters optimized for domain Δ^T will be indicated by the symbol Θ^T to simply the notation, thus $\mathcal{M}^T \approx \Theta^T$.

Given an annotated dataset $X^T : \{(x, Y)|x \in X, Y \subseteq T\}$ composed of a training dataset X^T_{train} and a test dataset X^T_{test}, a trained model with parameters Θ^T is considered optimized when for the majority of cases it is true that $\Theta^T(x) = \hat{Y}, \hat{Y} \approx Y$ iff $(x, Y) \in X^T_{train}$. A trained model with parameters Θ^T is then validated with a test set X^T_{test} by verifying that for the majority of cases $\Theta^T(x) = \hat{Y}, \hat{Y} \approx Y$ iff $(x, Y) \in X^T_{test}$. Given $\Theta^T(x) = \hat{Y}$, the likelihood for each type $t_i \in T$ is represented by $\Theta^T(x)_i = \hat{Y}_i$, i.e., the i-th output value, thus $p(t_i|\Theta^T, x) = \Theta^T(x)_i = \hat{Y}_i$.

3.2 Categorization of Domain Adaptation in FET

In this section, different instances of Domain Adaptation in FET are described in terms of expected differences in domains, their usefulness in real-case scenarios, their main problems and possible ways to tackle them.

Domain Adaptation in FET is needed if given a source domain $\Delta^S = (D^S, G^S, \alpha^S)$ and target domain $\Delta^T = (D^T, G^T, \alpha^T)$ the differences between (D^S, D^T), (G^S, G^T), and (α^S, α^T) precludes the usage of a model with parameters Θ^S, on the target domain Δ^T. These differences are also known as domain shift (or dataset shift [Shi+20]) and are very common in machine learning downstream applications. In fact, supervised machine learning paradigm often uses different *sources* or uses automatic annotation and human annotation respectively for *train* and *test* datasets, thus the annotation function α is different between *train* and *test.* Differences in the source type vocabulary G^S and target type vocabulary G^T instead are present only in some instances of transfer learning tasks like Class-incremental Learning or Curriculum Learning.

Adapting a model $\mathcal{M}$ from a known source domain Δ^S to a new target domain Δ^T implies adapting Θ^S to Θ^T to approximate: the annotation technique α^T, the language distribution in D^T and the types in G^T, instead of α^S, the language distribution in D^S and types in G^S approximated during the initial training on Δ^S.

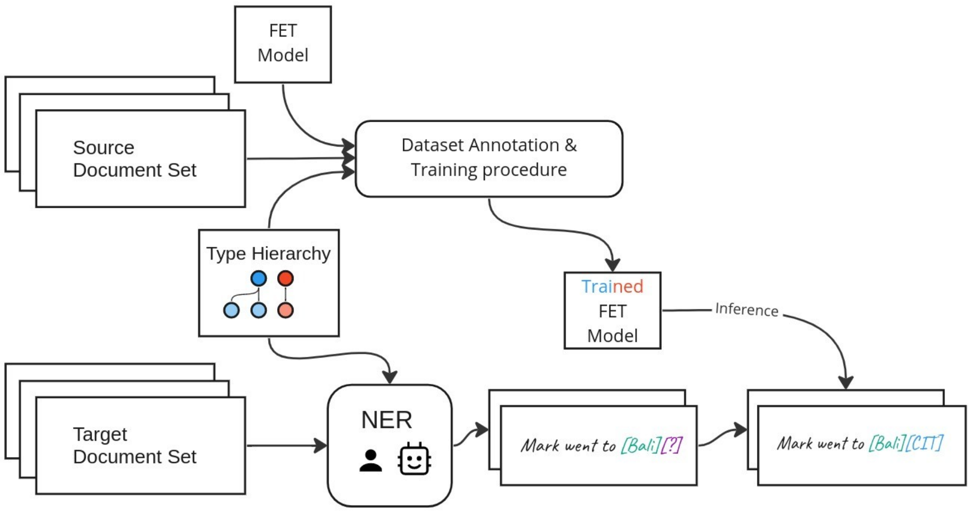


Figure 3.1: Schema of in-domain Fine-grained entity typing: a FET model is trained on a set of documents (as shown in Figure 2.2) to classify entities in a given hierarchy. The trained model is used on a new set of document to classify entities in it.

3.2.1 Application Use Cases

In-domain Fine-grained Entity Typing

A user wants to use a fine-grained entity typing algorithm to classify entities found in sentences using a Named Entity Recognition (NER) approach with types from a given vocabulary. The user expects that its vocabulary will not change, its aim may be to enrich a knowledge base with new entities from new or unseen documents. The user wants to classify entities present in the documents with coarse-grained types like `person`, `location`, or `organization` and with fine-grained types like `politician`, `city`, or `political_party`. This user story is schematized in Figure 3.1.

FET Model Reuse

A user wants to use a fine-grained entity typing algorithm to classify entities found in sentences using a Named Entity Recognition (NER) approach with types from a given target type hierarchy. The user trusts a given FET model trained on a source type hierarchy. The source type hierarchy overlaps with its target type hierarchy, so that all target types are already known by the FET model. The user notices these relations between all target types and types known by the trained FET model, thus it constructs mappings to traduce source types into target types. The user

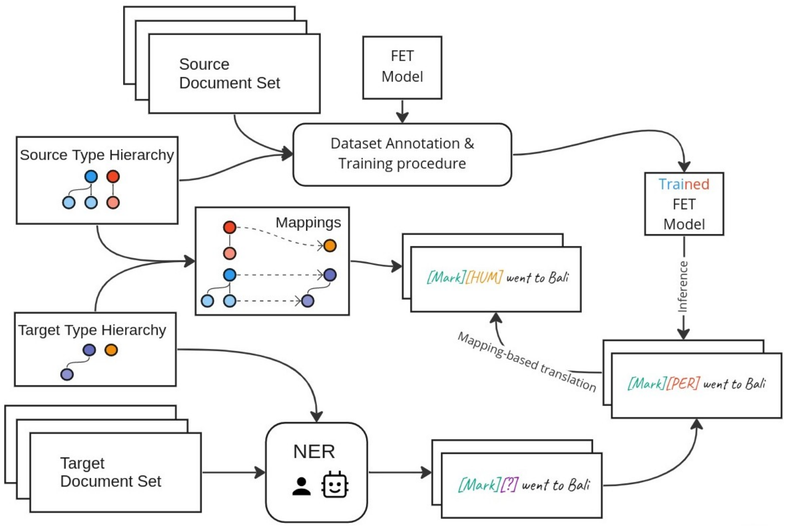


Figure 3.2: Schema of model reuse in FET: given a FET model trained on a source domain (documents + type hierarchy + annotation functions), a mapping is needed to translate source types use into equivalent target types and use the model in the target domain (in figure, `[PER]` stays for `Person`) into target types (in figure, `[HUM]` stays for `Human`)

feeds the trained model with target documents and traduces the predictions on source types, obtaining predictions on target types. This user story is schematized in Figure 3.2.

FET Model Specialization

A user has a trained FET model that classifies entities in some type. The user is interested in classifying job mentions, but the model only knows `Person`. The user needs to *specialize* the FET model by adding subtypes such as `Athlete` and `Musician`. The user has to teach the model the new types, avoiding forgetting of already known types and exploiting its ability on `Person`. This user story is schematized in Figure 3.3.

Full-fledged FET Domain Adaptation

A user has a trained FET model that classifies entities in some type. The user is interested in classifying a set of overlapping types that comprises new types, but also equivalent, generalization, or specialization of the already known types. The user needs to *adapt* the FET model by teaching the new types, by avoiding forgetting of already known types and by exploiting its ability on known types related to new ones. This user story is schematized in Figure 3.3.

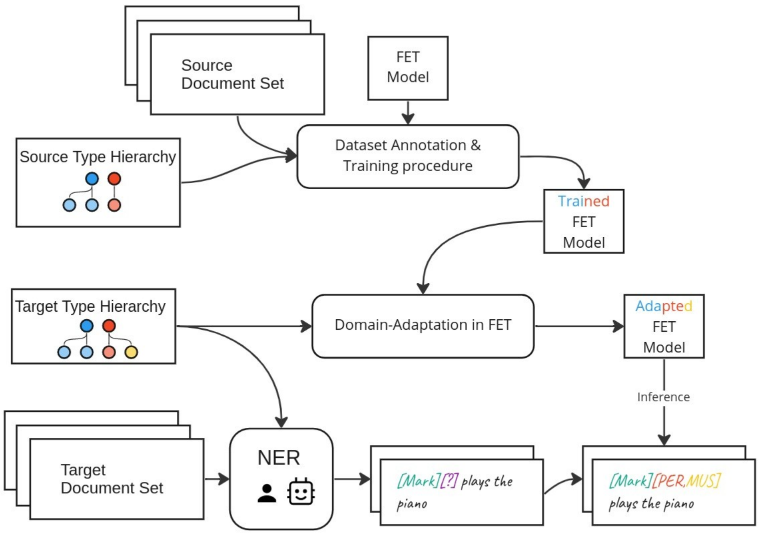


Figure 3.3: Schema of model specialization in FET: given a FET model trained on a source domain (documents + type hierarchy + annotation functions) to use it in a target domain with more specific types, an adaptation technique is needed to teach the new types to the model (in figure, `[PER]` stays for `Person`, `[MUS]` stays for `Musician`.

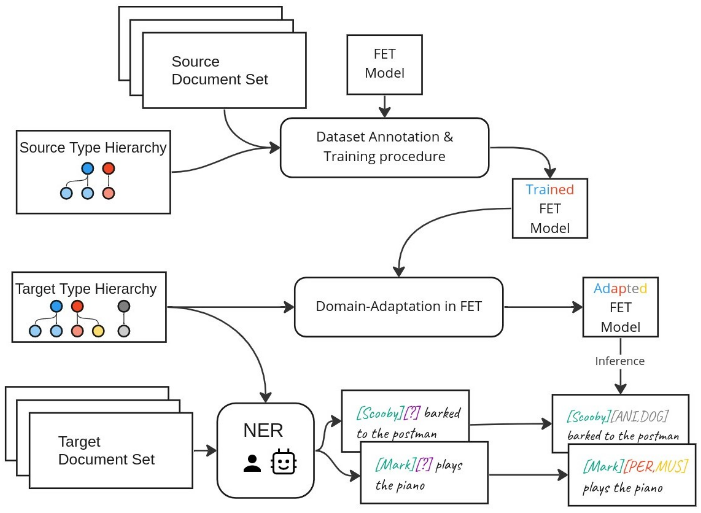

Source Document Set
FET Model
Source Type Hierarchy
Dataset Annotation & Training procedure
Trained FET Model
Target Type Hierarchy
Domain-Adaptation in FET
Adapted FET Model
Inference
Target Document Set
NER
[Scooby][?] barked to the postman
[Mark][?] plays the piano
[Scooby][ANI,DOG] barked to the postman
[Mark][PER,MUS] plays the piano

Figure 3.4: Schema of full-fledged domain adaptation of a FET model: given a FET model trained on a source domain (documents + type hierarchy + annotation functions) to use it in a target domain with overlapping and additional types, an adaptation technique is needed to teach the new types to the model (in figure, `[PER]` stays for `Person`, `[MUS]` stays for `Musician`, `[ANI]` stays for `Animal`, and `[DOG]` stays for `Dog`.

3.2.2 Domain Adaptation in FET Formalization

Recalling the formalization of a model in Section 3.1.3, a model with parameters Θ is considered trained on a source domain Δ^S when $\Theta^S(x) = \hat{Y}, \hat{Y} \approx Y$ if and only if $(x, Y) \in X^S_{train}$, thus the training of the model consists in the estimation of Θ^S given Θ and S.

Given a source domain Δ^S, a target domain Δ^T and a model with parameters Θ^S, the domain adaptation problem consists in the optimization of Θ^T by exploiting the similarities between Δ^S and Δ^T, thus estimating a $\delta_{S,T}$ such that: $\Theta^T = \Theta^S + \delta_{S,T}$ and verifying that $\Theta^T(x) = \hat{Y}, \hat{Y} \approx Y$ iff $(x, Y) \in X^T_{test}$, where X^T_{test} is the test partition of the target domain, often the only partition available. The estimation of $\delta_{S,T}$ is a key factor in domain adaptation and it is highly dependent on the similarities or dissimilarities between Δ^S and Δ^T.

As discussed in Sections 3.1.2 and 3.1.3, the presence and the absence of types in G^T with respect to G^S highly characterizes the similarities of domains in FET. In the following sections the different instances of domain adaptation in FET are formally defined and analyzed in term of the estimation of $\delta_{S,T}$ given some assumptions on relations between types in the type vocabularies in Δ^S and Δ^T.

Given source type vocabulary S and target type vocabulary T, and given two types $s \in S$ and $t \in T$ four relations can be found between them:

- Specialization $\prec$: if all entities that belong to t also belongs to s it follows that $t \prec s$. For example, `artist` $\prec$ `person`.
- Generalization $\succ$: the inverse of $\prec$. For example, `person` $\succ$ `artist`
- Equivalence $\equiv$: if $s \prec t \land t \prec s$, it follows that $s \equiv t$. For example, `person` $\equiv$ `human`.
- Disjunction $\not\parallel$: if no entities that belong to t also belongs to s and vice versa it follows that $t \not\parallel s$. For example, `person` $\not\parallel$ `location`

Partial relations, i.e., the partial share of entities between types, (some *artists* are also *athletes*) are not included in this list, since if a partial relations holds between s, t, a more abstract type can be found or created to establish generalization or specialization relations.

In the rest of the thesis the relations will be descried from source types S to target types T, The set of relations R is defined as $R = \{(s, t, r) : s \in S, t \in T, r \in \{\equiv, \prec, \succ, \not\parallel\}\}$.

3.2.3 Completely new domain

The most general case where Domain Adaptation is needed is when a model trained on a *source domain* Δ^S has to be used in a *target domain* Δ^T with a disjoint target type vocabulary $S \cap T = \emptyset$, i.e., $\forall s \in S, t \in T \; s \not\cap t$. This case is interesting to highlight some aspects of Domain Adaptation in FET and the relations between sources, annotation technique and type vocabulary in a domain. Since the type vocabularies G^S and G^T are disjoint, it is expected that the annotation techniques are also very dissimilar (as said in Sections 3.1.2 and 3.1.3 annotation techniques depend on types), the differences in *sources* of the source corpus D^S and the target corpus D^T regard the presence of entities of types G^T (expected to be present in each sentence of D^T, but only sporadically in D^S) and their context terms.

The real-case scenario that lies in this category can be the adaptation of a model to a specific disjoint domain, for example, the adaptation of a general NER approach that uses only the most common types `Person`, `Location`, and `Organization` to a specific task such as biomedical NER [ML18]. However, in FET this scenario is unlikely to happen, since the basic idea of FET is to have a high number of types, covering at least a coarse classification for each entity [LW12].

A main problem of this scenario can be the difficulty of recognizing completely new entities, for example, if the model is heavily anchored to an entity representation independent from its context. If also the sources of the corpora D^S and D^T are very different, their word distributions are probably divergent; thus, the encoder needs to be heavily adapted to the new language distribution.

Even with many differences between the source domain D^S and the destination domain D^T, some transfer learning techniques remain valid: using an already trained encoder is still a good starting point since if the language matches, general words maintain their meaning. An interesting opportunity regards exploiting the notion of "disjunction" between source types G^S and target types G^T: since they are assumed to be disjoint sets, their entities are also likely to be disjoint (with some exceptions). Thus, knowing that a portion of text belongs to a type in G^S can decrease the likelihoods of belonging to types in G^T. This intuition is used in this thesis by defining disjunction clauses using KENN in Chapter 5.

3.2.4 Exploiting explicit type relations across domains

Fine-grained Entity Typing models are designed to recognize an elevated quantity of types, and the purpose of the task is to cover a high variety of entities [LW12]. It follows that if a benchmark dataset proposed to validate a FET model is used as source dataset, its type vocabulary G^S will probably be overlapped with the target type vocabulary G^T. Even if a common NER system [GS96; Tjo02; NS07] is used as source Entity Typing model, its coarse-grained types `Person`, `Location`, and

`Organization` are so general that it is expected that some target types will share semantics with them.

Classic transfer learning approaches account for implicit overlap of domains based on the assumption that sub-symbolic representation of the input can be used to represent general information considered useful in multiple tasks. This assumption is validated by the countless transfer learning approaches used in very different computer science domains, from image recognition to text generation. For example, pretrained Word2Vec [Mik+13a; Mik+13b] or GloVe [PSM14] word vectors that contains a general representation of terms meaning are used as input of LSTMs to solve downstream tasks, or the pretrained VGG16 [SZ14] is used to extract image sub-symbolic features used to image segmentation. In general, methods not based on mappings and on explicit modeling of source and destination types relations are based on classic transfer learning techniques like feature extraction, fine-tuning or knowledge distillation, and can be used to obtain a model adjusted for a destination domain starting from a source model.

One idea developed in this thesis is to accounting for explicit type vocabulary overlap between source and destination domain by explicitly exploiting the knowledge on the source domain to favor the expected behavior on the destination domain. Knowing that two types s, t, respectively from G^S and G^T, share semantic meaning can motivate one to avoid training of t while adapting parameters Θ^S on the target domain. However, as discussed in Section 3.1.2 domain differences not only refer to types since the same semantic type (i.e. `Person`) can be annotated in the text in different ways (e.g., admitting or not the pronouns as valid entities), moreover, similar types (e.g., `FIGER.building` and `BBN.FACILITY/BUILDING`) can express slightly different semantics (`FIGER.building` is used to annotate general buildings, while `BBN.FACILITY/BUILDING` is more similar to the concept of infrastructure).

These observations can be used as basement to define different approaches to express the semantical equivalence, generalization, specialization, or disjunction of types in source and target type vocabulary and to design different solutions to tackle domain adaptation in FET.

Specialization scenario

A first interesting scenario, described in the user stories in Section 3.2.1, is the *specialization scenario*, where the target domain Δ^T contains more types than Δ^S, but each new target type $t \in G^T$ is a specialization of a type $s \in G^S$, i.e., $t \prec s$. For simplicity, in this scenario, the source domain is identified by S, while the target domain, specialization of S, is identified by S^+, such that $S^+ \supset S$ and $\forall s_i \in S^+ \; \exists s_j \in S : s_i \prec s_j$. Formally, given a model $\mathcal{M}^S$, parametrized with Θ^S, the domain adaptation problem in the specialization scenario asks to find parameters $\Theta^{S^+} = f_{S \to S^+}(\Theta^S)$ that parametrize a model $\mathcal{M}^{S^+}$. The model $\mathcal{M}^{S^+}$

can be seen as a function $\mathcal{M}^{S^+} : X \to \mathcal{P}(S^+)$.

If the specialization relation $s_i \prec s_j$ with $s_i \in S^+, s_j \in S$ is known before the estimation of $f_{S\to S^+}$, it can be used to drive the adaptation process by conditioning the probability of type $s_i \in S^+$ on both likelihoods $\Theta^{S^+}(x)_i$, $\Theta^S(x)_j$. Formally $p(s_i|\Theta^{S^+}, x, \Theta^S(x)_j) = \Theta^{S^+}(x)_i$. This intuition is used in Chapter 5 to design and apply specialization techniques to specialize a given FET model, and in Chapter 6 to design and apply domain adaptation techniques. For example, given the source type `Person` and the target types `Person/Athlete` and `Person/Politician`, the probability of `Person/Athlete` has to be conditioned by the probability of `Person`, i.e., the higher the probability of the father, the higher the probability of the child and vice versa.

Full-fledged domain adaptation

Another scenario described in the user stories in Section 3.2.1 is the *cross-domain scenario with overlapping domains* where the relations R between source and a target domain Δ^S and Δ^T, with type vocabularies S and T, are in the form $(s, t, \rho) \in R$ with $s \in S, t \in T, \rho \in \{\equiv, \prec, \succ\}$. These relations establish connections between the source and target domains by explicitly linking pairs of types. Formally, given a model $\mathcal{M}^S$, parametrized with Θ^S, the domain adaptation problem in the cross-domain with overlapping domain scenario asks to find parameters $\Theta^T = f_{S\to T}(\Theta^S)$ that parametrize a model $\mathcal{M}^T$. The model $\mathcal{M}^T$ can be seen as a function $\mathcal{M}^T : X \to \mathcal{P}(T)$.

If the relation (s_i, t_j, ρ) with $s_i \in S, t_j \in T, \rho \in \{\equiv, \prec, \succ\}$ is known before the estimation of $f_{S\to T}$, it can be used to drive the adaptation process by conditioning the probability of type $t \in T$ on both likelihoods $\Theta^T(x)_j$, $\Theta^S(x)_i$. Formally $p(t_j|\Theta^T, x, \Theta^S(x)_i) = \Theta^T(x)_j$. This intuition is used in Chapter 6.

Model Reuse Scenario

The last scenario described in user stories in Section 3.2.1 is the *model reuse scenario* where the target domain Δ^T contains less types than Δ^S, but each new target type $t \in G^T$ is a generalization of or equivalent to a type $s \in G^S$, i.e., $t \succ s$ or $t \equiv s$. For simplicity, in this scenario, the source domain is identified by S, while the target domain is identified by S^-, such that $S^- \subseteq S$ and $\forall s_i \in S^- \; \exists s_j \in S : s_i \succ s_j \vee s_i \equiv s_j$. Formally, given a model $\mathcal{M}^S$, parametrized with Θ^S, the domain adaptation problem in the model reuse scenario asks to find parameters $\Theta^{S^-} = f_{S\to S^-}(\Theta^S)$ that parametrize a model $\mathcal{M}^{S^-}$. The model $\mathcal{M}^{S^-}$ can be seen as a function $\mathcal{M}^{S^-} : X \to \mathcal{P}(S^-)$.

If the differences between documents D^S and D^{S^-} and between annotation functions α^S and α^{S^-} are ignored, $f_{S\to S^-}$ can be used to directly map elements of

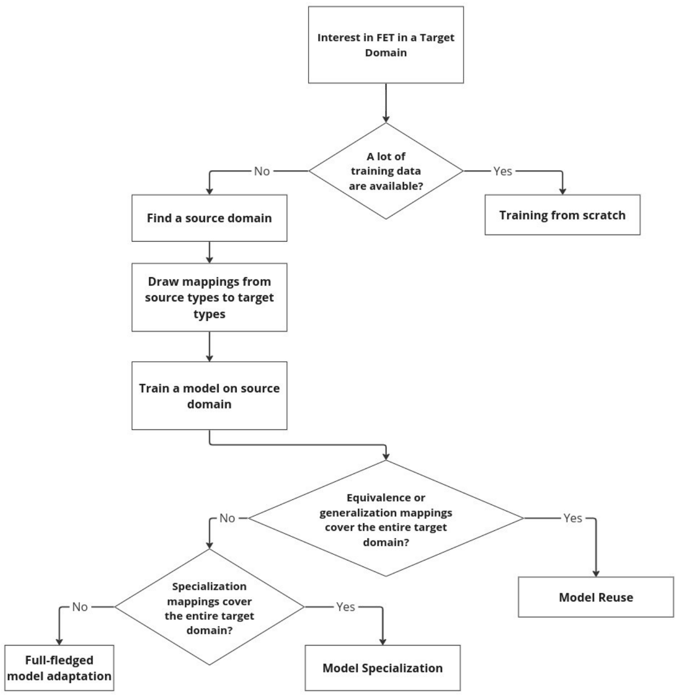

Interest in FET in a Target Domain
A lot of training data are available?
No
Yes
Find a source domain
Training from scratch
Draw mappings from source types to target types
Train a model on source domain
Equivalence or generalization mappings cover the entire target domain?
No
Yes
Model Reuse
Specialization mappings cover the entire target domain?
No
Yes
Full-fledged model adaptation
Model Specialization

Figure 3.5: Workflow showing the different adaptation techniques for a trained FET model depending on mappings availability

S to elements of S^-; this scenario is explored in Chapter 4. If $f_{S \to S^-}$ can be used instead to modify the whole parameters Θ^S, thus accounting for domain differences between documents D^S and D^{S^-} and between annotation functions α^S and α^{S^-}, but also to allow to represent unseen entities. The latter scenario is a special version of the *cross domain with overlapping domains* scenario, which is explored in Chapter 6.

The described adaptation techniques are schematized in the workflow in Figure 3.5

Chapter 4

Model Reuse in Partially Overlapping Domains

In Domain Adaptation, similarities between source and target domains allow the total or partial reuse in the target domain of a model trained on source domain. This is a basic intuition for designing a transfer learning system, with pretrained models used to generate representations considered useful as starting point in downstream tasks. As described in Section 3.2.4, common examples of static transfer learning are the usage of Word2Vec [Mik+13a; Mik+13b] or GloVe [PSM14] vectors as input to train LSTMs on downstream tasks, or using VGG16 [SZ14] to extract image sub-symbolic features used to image segmentation.

In Fine-grained Entity Typing (FET) similarities between source and target type vocabularies may motivate the reuse of a trained model, in order to exploit its ability on shared types, avoiding the necessity of training on those types (Section 3.2.4). To design a system with model reuse, mapping between types are needed, in order to establish semantical correspondence between source and target types, as argued in Section 3.2.4 the adoption of *equivalence* mappings to avoid training on already learned types its promising when annotation policies and corpora characteristics are similar between domains, but may brings problems if there are differences in these domain components or *generalization* and *specialization* mappings are exploited disregarding the representation of types.

In this chapter, model reuse without training solutions are designed to be applied in FET Domain Adaptation Scenario. The main focuses of this chapter are: (1) understanding the similarities between the FET domains (under the variety space definition [Pla16], see Sections 2.3.1, 2.3.2,and 3.2.2) and their effects in the scenario of model reuse without retraining, and (2) underline critical issues of model without retraining in FET.

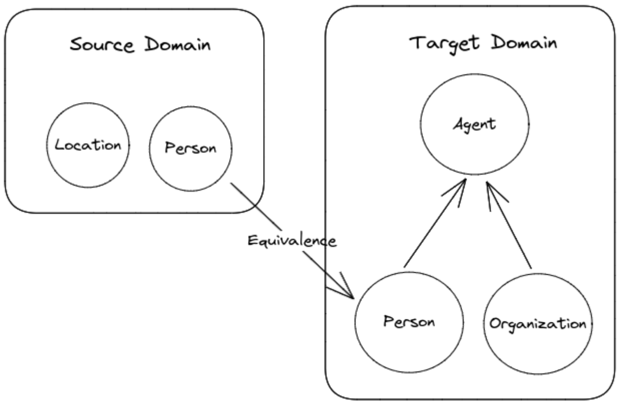


Figure 4.1: If transitive closure is used in target domain only person can be recognized as *agents*. Without a training, the model will wrongly classify each `Organization`, propagating the error also on `Agent`.

4.1 Research Questions - Model Reuse

Question 1: to what extent do FET models transfer their performance between partially overlapping domains without additional training data?

Question 2: how approximating a source domain (and may overfitting on it) impact the reuse of models across target partially overlapping domains?

Question 3: equivalent types from different FET domains are represented by entities and sentences with different linguistic or semantical characteristics?

4.2 Critical Issues of Model Reuse without Training

Reuse a model in a target domain trusting the training on source domain and the equivalent semantics between types mapped with equivalence and generalization mappings can lead to multiple unexpected behaviors depending on differences in the domains:

Different representation of an equivalent type. annotation policy differences between source and target domains can produce annotations with different characteristics, for example in OntoNotes [Gil+14] entity mentions are longer than the ones in FIGER [LW12] and BBN [WB05; Ren+16a]. The fact that an encoder

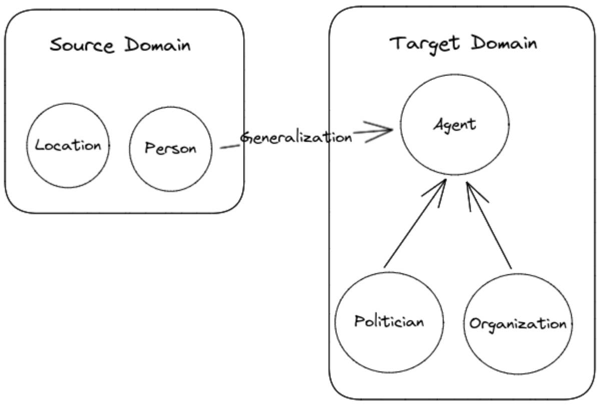


Figure 4.2: An example of mapping that can lead to paradoxical correct classification: the model can recognize an *organization entity* as a *person* making an error but correctly predicting `Agent` in the target domain hiding the error behind good performance.

is able to produce similar representations for entities of equivalent types that come from different domains highly depends on the robustness of the encoder and on the absence of overfitting on the source domain. In a setup without training, the similarity in encoded representation is a requisite to maintain performance between original source type and its equivalent target type.

Transitive closure in the target domain. the usage of equivalent mappings to predict in the target domain without additional training may include a transitive closure step where the target predicted types are expanded by adding their ancestors in the target hierarchy. In particular cases, reusing a trained model in a target domain to predict types that are more abstract than types in the training source domain may lead to an under-representation of entities and misconception of model capabilities. Consider the example in Figure 4.1, one can claim that using this equivalence embedding and transitive closure in the target domain, a model trained on the source domain can predict the type `Agent`. The expected behavior of this "*adapted*" entity typing model instead is to recognize *persons* as *agents* but making wrong predictions on *organizations*.

Paradoxical Correct Classification. consider the scenario in Figure 4.2, given a sentence that contains an entity of type `Organization`, the model can erroneously assign the source type `Person`, that is translated to the target type `Agent`. Thus, the model produces a correct prediction based on a wrong evidence (i.e. an *organization* recognized as a *person*). This concatenation of event hides behind good performance a fallacy of the model.

4.3 Datasets, Mappings and FET model

To experiment with model reuse in FET, three common FET datasets (BBN [WB05; Ren+16a], FIGER [LW12] and OntoNotes [Gil+14; Shi+17a]) are paired, obtaining six source-target pairs. Equivalence, generalization and specialization mappings between their type vocabularies are manually drawn. A FET model based on recent encoding approaches (BERT [Dev+19] with Adapters [Hou+19; Pfe+20b]) and flat classifier is used. More sophisticated models are ignored since in this Chapter the focus is to answer to research questions, thus the trained model and its behavior can be seen as a synthesis of types representations in the source domain. Evidences on results are showed and discussed. Finally a solution for improved adaptation is discussed and evaluated.

4.3.1 Datasets

Three common benchmark FET datasets are used to experiment the mapping-based model reuse on target domain scenario. These datasets are used in the majority of the FET articles and share common characteristics: english language, news or encyclopedic source documents, various and hierarchic type vocabulary, distant supervision for training/dev set and manual annotation for test set. The three datasets are:

FIGER: proposed in [LW12] while explicitly defining Fine-grained Entity Typing as a task, has a taxonomy originally composed of 112 types extracted from Freebase [Wan+22b]. Entity mentions are extracted by applying a NER approach and refined using Wikipedia and Freebase links. Types are extracted from wikipedia links an filtered based on the taxonomy. [Ren+16a] reconstructed the dataset by re-applying and refining the annotation method from [LW12] in 2016, obtaining a version with 128 types. [Ren+16a] version of FIGER dataset is used in this thesis due to its large use in literature.

OntoNotes: proposed in [Gil+14], this dataset is based on the OntoNotes corpus, largely used in NER. Its type taxonomy is based on the FIGER taxonomy, with type merging and corpus based types additions [Wan+22b], obtaining a three level taxonomy with 89 types. POS tagging and linking to Wikipedia are used to automatically find and annotate entities in the corpus. Filtering rule-based approaches are used to deal with overly-specific noise, out-of-context noise and underrepresented types. [Shi+17a] modified the dataset by collapsing types with the same semantics (`location/geography` and `location/geograpy`) and deleting from the test set the examples annotated with types not present in the training set. [Shi+17a] version is used in this thesis due to its large use in literature.

Dataset	#types	Hierarchy depth	#training examples	#dev examples	#test examples
BBN	47	2	84,492	1,039	12,349
FIGER	128	2	2,676,846	1,094	563
OntoNotes	87	3	202,689	2,192	8,963

Table 4.1: Statistics of benchmark datasets

BBN: proposed in [WB05], this dataset is extracted from Wall Street journal articles and annotated by [Ren+16a] using 47 of the original propose 93 types, since only 47 could be mapped to Freebase types. In this thesis [Ren+16a] version is used due to its large use in literature.

Statistics about the benchmark datasets are shown in Table 4.1.

4.3.2 Define relations between type hierarchies

To answer the research questions, datasets are paired to define six source-target scenarios, then equivalence and generalization relations are manually drawn. In this chapter, the equivalence and generalization relations are grouped under the term *mapping*. Mappings are drawn from source to target type vocabulary following two heuristics:

- Heuristic 1: For each type t_s in the source domain, draw an equivalence mapping if there exists a unique target type t_t that preserves the semantics in the source domain. Some example mappings are: (`BBN.EVENT`, `FIGER.event`, $\equiv$), (`BBN.GPE/CITY`, `FIGER.location/city`, $\equiv$), and (`BBN.ANIMAL`, `OntoNotes.other/living_thing/`, $\equiv$).

- Heuristic 2: For each type in the source domain t_s that is not mapped to any type in target domain, draw a generalization mapping if there exists a unique target type t_t that preserves the semantics in the source domain. Examples of mappings obtained by the combination of heuristic 1 and 2 are: (`BBN.PLANT`, `OntoNotes.other/living_thing`, $\prec$), and (`BBN.LOCATION/RIVER`, `FIGER.location/body_of_water`, $\prec$).

In both cases fork mappings are excluded from the process, as in [Noz+21]. In the case that fork mappings can be drawn, no mapping is drawn instead. Note that this heuristics does not impact the manual defined mappings between the used datasets. In OntoNotes Shimaoka partition is used, thus only 87 out of 89 types are included in mapping design, `OntoNotes.person/business` and `OntoNotes.other/product/camera` are excluded.

Scenario	Source	Target	#source types	#target types	#mappings	Covered Target Types
In-Domain	BBN	BBN	47	47	-	-
Cross-Domain	Figer	BBN	127	47	93	30
Cross-Domain	OntoNotes	BBN	87	47	76	29
In-Domain	Figer	Figer	127	127	-	-
Cross-Domain	BBN	Figer	47	127	41	34
Cross-Domain	OntoNotes	Figer	87	127	81	72
In-Domain	OntoNotes	OntoNotes	87	87	-	-
Cross-Domain	BBN	OntoNotes	47	87	40	33
Cross-Domain	Figer	OntoNotes	127	87	111	77

Table 4.2: Statistics of manually defined mappings for the domain adaptation through model reuse with no training. For each source type, one mapping can be drawn (fork mappings are heuristically excluded), the same target type can be involved in multiple mappings, thus the columns `#mappings` and `Covered target types` respectively describe: how much source types are used to predict in the target domain, and how much target types are covered by these mapping in the "no training" setup.

4.3.3 Model Reuse: Domain Adaptation without Additional Training

Given a source domain Δ^s and a model trained on it $\mathcal{M}^s$, as described and discussed in Section 3.2.4, a straightforward way to use $\mathcal{M}^s$ on a target domain Δ^t is to draw equivalence $\equiv$ and generalization $\succ$ mappings from source type vocabulary S to target type vocabulary T and use them to translate predictions. For example, given the instance x: "*Britain , **France** and Italy pull out of a proposal to build new NATO frigates ; the U.S. and West Germany have each withdrawn from missile projects .*", the model $\mathcal{M}^{BBN}$ can classify it by predicting $\mathcal{M}^{BBN}(x) =$`[BBN.GPE, BBN.GPE/COUNTRY]`, if x has to be classified using `FIGER` vocabulary instead, using equivalence and generalization mappings, the prediction can be obtained by applying the mappings as a function $f_{BBN \rightarrow FIGER}(\mathcal{M}^{BBN}(x))$, obtaining `[FIGER.location/country]`, then transitive closure can be applied on the target domain, obtaining the prediction `[FIGER.location/country, FIGER.location]`.

The capabilities of model reuse with mappings not only depends on the model trained on source domain and on the similarity between source and target domain, but also on the choices adopted during the usage of mappings to traduce the predictions from the source to the target vocabulary. A straightforward choice can be to translate the types and then apply the transitive closure to add abstracter and unmapped types. Transitive closure is not always a good choice, since false positives in the source domains may be amplified in the target domain, instead it can be useful to predict general types in the target domain, decreasing the false negatives. This choice has to be taken by accounting the precision of the source model and the desired precision/recall in the target domain.

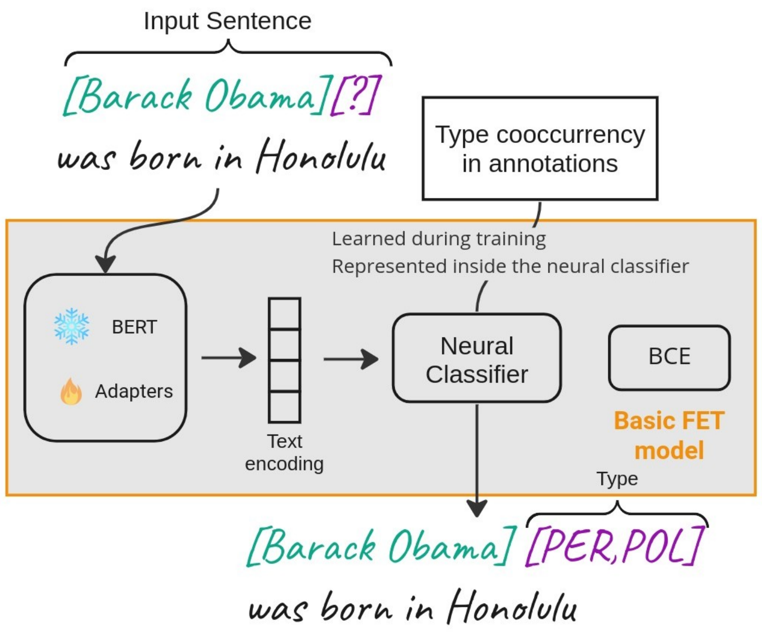


Figure 4.3: Architecture of the basic FET model used as classifier in model reuse (Section 4.3.3), denoising (Section 4.5.1) and as baseline classifier in specialization and full-fledged domain adaptation (Section 5)

When applying model reuse with partial mappings paradigm, void type set can be predicted in target domain, since the model may predict source types that are not mapped to the target domain. This particularity decreases the recall of model reuse approaches, but increases their precision, since false positives are avoided. Recall instead depends on the type coverage offered by mappings. Statistics of manual-defined mappings for model reuse are reported in Table 4.2. The table reports the scenario (`in-domain` or `cross-domain`) to compare the differences in the available types for predictions: in an `in-domain` scenario, the source and target domains are identical, thus the model is trained on the target type vocabulary and is able to predict each type in it; in a `cross-domain` scenario without training on the target domain, a model trained on the source domain is able to predict only those target types that are involved in at least one mapping drawn from source to target domain. This information is expressed in `Covered target types` column.

4.3.4 Fine-grained Entity Typing Model

A simple yet competitive classification model is used to understand the impact of training domain distribution on performance on target domain with different distribution. The architecture is shown in Figure 4.3. The version of BERT [Dev+19] is

`BERT-Large-cased` from `huggingface`[1] and it is used as a pre-trained backbone language model. Adapters [Hou+19; Pfe+20a] are inserted inside each transformer layer of `BERT-Large-Cased` and trained to adapt the encoder to the source domain. Then the neural classifier is fed with the encoded representation, the classifier is composed of two fully connected layers, with the last layer matching the vocabulary size. The model is optimized by minimizing binary cross-entropy (BCE), standard loss function in multilabel classification problems and in FET.

With this simple setup, it is expected that the encoder will approximate the language distribution and the classifier will account for the label distribution (in the sense of similarity between inputs with different labels and similarity between labels) in the source domain.

4.3.5 Hyperparameters

Encoder:

- Pretrained Language Model: `BERT-large-cased`
- Input Format: `[CLS] mention [SEP] sentence` as in [Ono+21]
- Input Encoding: Encoding of the token `#0` (`[CLS]` after 24 transformer layers, commonest choice in literature)
- Hidden Representation Size: 1024, default hidden size of `BERT-large-cased`
- Adapter Architecture: Pfeiffer [Pfe+20a], 2 adapters in each transformer layer
- Adapter Reduction Factor: `16`, each adapter represent information with $\frac{1024}{16} = 64$ dimensions

Classifier

- Fully Connected Layers: 2
- Fully connected units per layer: 1024 in the first layer, #type in vocabulary for the second layer
- Activation Function: `relu` in the first layer, `sigmoid` in the second layer

[1]https://huggingface.co/

Training Hyperparameters

- Batch Size: `64`
- Batch per epoch: `160` (each epoch 10k circa examples are seen), this parameter is necessary to deal with huge training size of datasets (80k, 220k and 2.6M), during preliminary experiment an overfitting during the steps of first epoch was observed
- Stopping Criteria: `early stopping` on validation loss, with `patience = 5`
- Loss function: `Binary Cross Entropy`
- Learning Rate: `5e-4`, preliminary experiments
- Initialization & random seed for batch sample: `0`, `1`, or `2`; three instances are trained to compute variance

For each experimental setup, three instances are trained with `Initialization & random seed` that ranges from 0 to 2. This choice is a tradeoff between the necessity of running multiple instances of the same experiment to evaluate the robustness of the drawn evidences and to the time necessary to train each instance (to run all experiments 3 months circa were needed).

4.3.6 Metrics

The metrics used to evaluate the trained method are the standard ones in literature, plus some ad-hoc metrics used and described in Chapter 5.

Fine-grained Entity Typing is a multiclass multilabel problem, thus precision, recall and f1-score are often used to evaluate and compare trained models. In a multilabel problem multiple correct classes/types can be used to correctly annotate each example, thus three kinds of evaluation metrics can be computed:

Metrics macro averaged on examples: Given an example with n correct types, a FET model can predict m types. For each example, example-level metric can be computed: **example-level precision** expresses the fraction of *correctly predicted types* on the total predicted types m for a given example, **example-level recall** expresses the fraction of *correctly predicted types* on the total types to predict n for a given example. Having these metrics for each example in the test set allows to compute the *Macro Averaged on Examples Precision* and *Recall*, the *Macro Averaged on Examples f1-score* is simply the harmonic mean between the macro averaged on examples precision and recall.

Metrics macro averaged on types: Given a dataset with t types, the performance of a FET model on each type can be computed by computing precision and recall on the single types. Type-level metrics are computed by gathering all predictions of a given type predicted by a trained model on a test dataset (assuming m predictions for a given type) and by counting the number of test examples annotated with that type (assuming n). The **type-level** precision expresses the fraction of correct predictions over all predictions m for a given type. The **type-level** recall expresses the fraction of correct predictions over all annotation n for a given type. Having these metrics for each type in the test set allows to compute the *Macro Averaged on Types Precision* and *Recall*, the *Macro Averaged on Types f1-score* is simply the harmonic mean between the macro averaged on types precision and recall. In this thesis metrics macro average on types are computed considering only those types with at least 3 annotated examples in the test set, thus avoiding the inclusion of types of the vocabulary that are totally absent from the test set, as well as the presence of extreme values (0 and 1) obtained by computing metrics only on one or two examples.

Micro averaged metrics: micro average metrics do not consider type-level or example-level, but consider dataset level. Given all predicted types m and all annotations m, the micro-average precision expresses the fraction of correct predictions on all predictions m, the micro-average recall expresses the fraction of correct predictions on all annotations m. The micro-average f1-score is the harmonic mean between micro-average precision and micro-average recall.

Metrics usage and sensibility

Macro-averaged on Examples and micro average metrics are used in each article of FET literature, since the former expresses the ability of recognizing all types for a given examples and to predict only the correct ones, the latter expresses the ability of recognizing all types for a given datasets and predicting only the correct ones for each example. Since these metrics depend on examples and dataset they are biased by the type distribution used to compute them, i.e., to maximize them the model has to be able to correctly predicting the most common types. For example, in Shimaoka's version of OntoNotes [Shi+17a; Gil+14] (described in Section 4.3.1), but this consideration is also valid for the original version [Gil+14], the 65.8% of examples in test set is annotated with the type `OntoNotes.other`, thus a dummy model that always predicts `OntoNotes.other` independently on the input example obtains a score of .61 both in micro average and macro averaged over examples f1-score (third decimal place is different). Even if Macro-averaged on Examples and micro average metrics are biased by type distribution, these metrics are still useful to evaluate the ability of a model in a real scenario, if the assumption that dataset type distribution and real-scenario type distribution are similar holds, i.e., if the most present types in the dataset are also most present and important types in the real case scenario.

Due to this sensibility to label distribution in the dataset, in this thesis also *metrics averaged over types* are used to evaluate the performance of models, under the assumption that all types in a vocabulary have similar importance, thus have to affect the evaluation in the same way.

4.4 Model reuse - Results

In this section, evidences are extracted by comparing the behavior of the FET model introduced in Section 4.3.4 when it is used in in-domain setup (classic setup in literature) or in cross-domain setup without additional training (described in Section 3.2.4 and in the introduction of this Chapter) exploiting the mappings introduced in Section 4.3.2.

4.4.1 Performance in Source Domain

The performance analyses of the ET model introduced in Section 4.3.4 when it is used under in-domain setup, where the source and the target domains are the same domain, i.e., the same benchmark dataset from the literature, are reported in this subsection. This performance is important to evaluate the general quality of the model and to have a reference for the following analyses. The results are obtained by using all data available for each dataset and with BERT tokenizer's parameter search on input for each dataset. The trained models are expected to approximate the input and label distribution in order to solve the FET problem. The same trained model will be used as starting point for cross-dataset experiments, both without (section 4.4.2) and with training (Section 6).

In this section, two results are reported: the average results of three instances of the ET model trained on the entire benchmark datasets (Tables 4.3, 4.4, and 4.5) and the results of the model initialized with seed 0 and trained on the entire benchmark datasets (Table 4.6).

From Tables 4.3, 4.4, and 4.5 can be seen that the model adopted as a backbone-model in this thesis suffers from the absence of additional information and by the absence of denoising technique, thus its performance are good but lower than the state-of-the-art. This is an expected result, since the model is designed to reflect the domain language and its label distribution, in order to understand how these characteristics affect the domain adaptation process. Adding additional information or denoising results in an increase of the variables to analyze in the domain adaptation experiments. Moreover, the technique proposed in FET

	Macro Averaged over examples			
Approaches	**F1**	**P**	**R**	**Notes**
[Ono+21]	.783	-	-	Hierarchical & co-occurrence dependencies
[CCV20]	.797	-	-	Hierarchical dependencies
[Shi+20]	.830	-	-	Denoising
[Liu+21a]	.876	-	-	Document and type co-occurrence & hierarchy
[Wu+22]	.819	-	-	Denoising + Hierarchy
[DSL20]	.914	-	-	Link-aware (best approach overall)
BERT + Adapters + Classifier	.781 ± .004	.729 ± .016	.842 ± .016	No additional external information

Table 4.3: Performance from literature and from three instances of the basic model introduced in Section 4.3.4 on the benchmark dataset BBN [WB05; Ren+16a].

	Macro Averaged over examples			
Approaches	**F1**	**P**	**R**	**Notes**
[XB18]	.819	-	-	Hierarchical dependencies
[CCV20]	.826	-	-	Hierarchical dependencies
[Ono+21]	.816	-	-	Hierarchical & co-occurrence dependencies
[Li+21]	.877	-	-	Document and type co-occurrence & hierarchy
[Pan+22]	.822	-	-	Denoising
[Wu+22]	.826	-	-	Denoising + Hierarchy
[DSL20]	.891	-	-	Link-aware (best approach overall)
BERT + Adapters + Classifier	.836 ± .017	.784 ± .030	.897 ± .013	No additional external information

Table 4.4: Performance from literature and from three instances of the basic model introduced in Section 4.3.4 on the benchmark dataset FIGER [LW12].

	Macro Averaged over examples			
Approaches	**F1**	**P**	**R**	**Notes**
[XB18]	.764	-	-	Hierarchical dependencies
[CCV20]	.73	-	-	Hierarchical dependencies
[Ono+21]	.773	-	-	Hierarchical & co-occurrence dependencies
[Liu+21a]	.845	-	-	Document and type co-occurrence & hierarchy
[Wu+22]	.732	-	-	Denoising + Hierarchy
[Shi+20]	.891	-	-	Denoising (best approach overall)
BERT + Adapters + Classifier	.697 ± .020	.729 ± .016	.669 ± .026	No additional external information

Table 4.5: Performance from literature and from three instances of the basic model introduced in Section 4.3.4 on the benchmark dataset OntoNotes [Gil+14; Shi+17a].

		Dataset		
		FIGER	**BBN**	**OntoNotes**
Macro averaged on Examples	**precision**	.785	.743	.692
	recall	.834	.804	.710
	f1	.809	.772	.701
Micro averaged	**precision**	.713	.723	.630
	recall	.820	.808	.660
	f1	.763	.763	.645
Macro averaged on types	**precision**	.231	.444	.199
	recall	.262	.481	.225
	f1	.246	.462	.211

Table 4.6: Performance of the Entity Typing Network introduced in section 4.3.4 trained and tested under in-domain setup, seed 0 is shown for comparison with cross-domain experiments.

literature are often dataset-specific, thus their analysis and their adoption in a domain adaptation setup is complex and may be based on wrong premises, since a model with architecture and optimization overfitted on a specific dataset (and consequently on its domain) can be more difficult to adapt to another domain. However in this thesis also a denoising method is used to understand its effect on domain adaptation without training setup (Section 4.5). This technique is used only as case study and it is not included by default in the backbone model.

4.4.2 Performance on the Target Domain

Table 4.7 shows the behavior of models trained on common benchmark FET dataset and used under in-domain setup and under cross-domain with no additional training setup; metrics on test sets are reported. Figures 4.4, 4.5, and 4.6 show a Precision/Recall plot where, for each target dataset, performance of in-domain and cross-domain models on types covered by mappings are reported.

The first evidence is that cross-domain models perform worst than in-domain models if evaluated with classic metrics (i.e., macro metrics averaged over examples and micro averaged metrics). The reason of this behavior is already explained in Section 4.3.6: the value of these metrics is biased by the dataset composition. Thus, cross-domain models without training on target domain are penalized by the impossibility to predict the whole target type vocabulary (see Table 4.2 for reference on coverage lack offered by mappings), as already explained in Chapter 4 and in Section 3.2.4. However, Table 4.7 also offers interesting information: macro metrics averaged over types show that a model trained on FIGER and used to classify examples from OntoNotes test set with OntoNotes type vocabulary reach better performance than the correspective in-domain model (i.e. trained and used on OntoNotes). This can be due to the high label imbalance in OntoNotes, where the type `OntoNotes.other` dominates train, validation and test set. This evidence

Source Domain	Target Domain	Macro averaged on Examples			Micro Average			Macro average over Types		
		Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
BBN (in-domain)	BBN	.74	.80	.77	.72	.81	.76	.53	.57	.55
OntoNotes	BBN	.69	.69	.69	.69	.67	.68	.37	.46	.41
FIGER	BBN	.73	.73	.73	.67	.72	.70	.39	.39	.39
OntoNotes (in-domain)	OntoNotes	.69	.71	.70	.63	.66	.64	.33	.34	.33
BBN	OntoNotes	.32	.24	.28	.36	.29	.32	.21	.22	.21
FIGER	OntoNotes	.33	.27	.30	.40	.34	.37	.37	.39	.38
FIGER (in-domain)	FIGER	.78	.83	.81	.71	.82	.76	.65	.65	.65
BBN	FIGER	.68	.66	.67	.63	.62	.63	.26	.27	.26
OntoNotes	FIGER	.68	.73	.70	.62	.70	.66	.27	.32	.29

Table 4.7: Mapping-based Cross-Domain performance computed on original test sets. Note that some types (statistics in Table 4.2) cannot be predicted in this setup since they are not covered by mappings and the models are not trained on target domains.

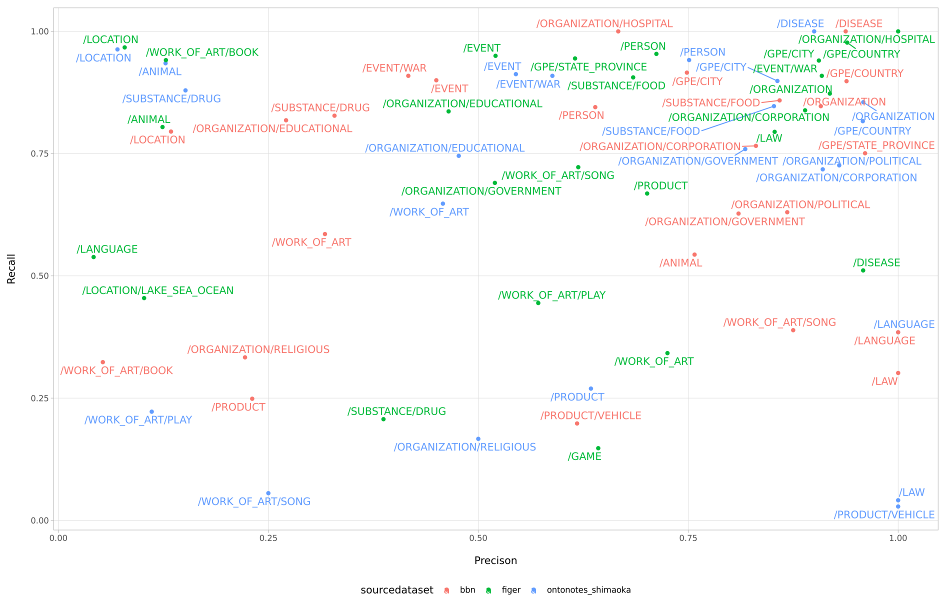


Figure 4.4: Precision/Recall plot of models trained on BBN, FIGER or OntoNotes and used to predict examples from the test set of BBN, with BBN type vocabulary. Only types covered by mappings are reported (see Table 4.2). Red dots represent performance of in-domain model.

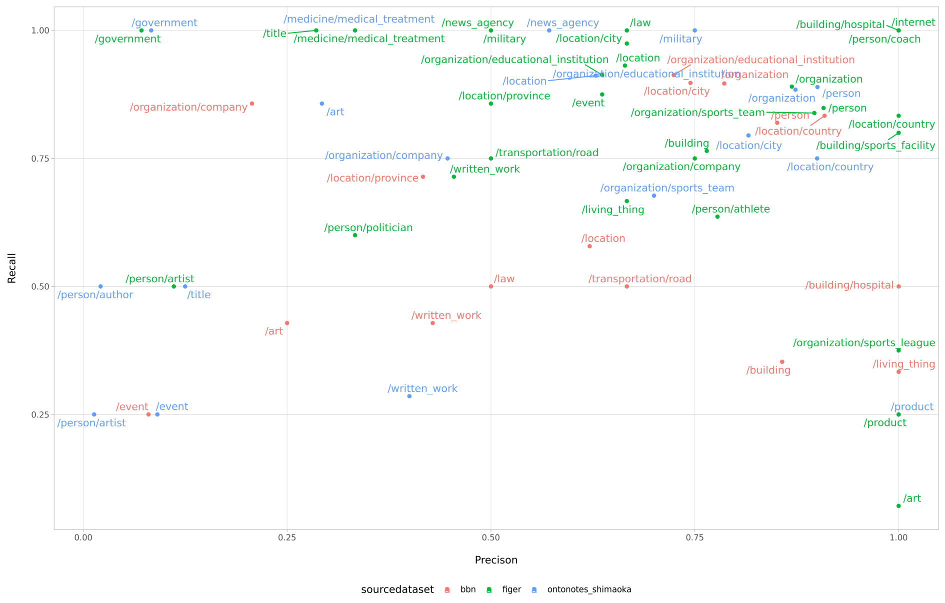


Figure 4.5: Precision/Recall plot of models trained on BBN, FIGER or OntoNotes and used to predict examples from the test set of FIGER, with FIGER type vocabulary. Only types covered by mappings are reported (see Table 4.2). Green dots represent performance of in-domain model.

can be extracted by crossing the performance of the two models when evaluated with macro-averaged over examples and micro-averaged metrics; it can be seen that cross-domain model trained on FIGER does not perform well on these metrics, due to the fact that there is no mapping that reaches type `OntoNotes.other` since its semantics (*miscellaneous*) is not present in FIGER and BBN. Another evidence is that when BBN is the target domain, the precision of the in-domain and cross-domain models is very similar (rows 1-3 in Table 4.7), while the recall drops due to the uncovered types (Table 4.2. This fact may indicates that covered types are represented very similarly in FIGER, OntoNotes and BBN.

Figure 4.4 reports the precision and recall of BBN types covered by at least one mapping from FIGER or OntoNotes. Figure 4.5 reports the precision and recall of FIGER types covered by at least one mapping from BBN or OntoNotes. Figure 4.6 reports the precision and recall of OntoNotes types covered by at least one mapping from BBN or FIGER. From these figures different interesting evidences can be highlighted. There are types that benefits from cross-domain, for example the in-domain model performs worst on `BBN.PRODUCT` and `BBN.WORK_OF_ART` than both cross-domain models in Figure 1: in-domain approach (red dots) obtain respectively a Precision/Recall measure of .23/.25 on `BBN.PRODUCT` and .31/.58 on `BBN.WORK_OF_ART`, while cross-domain models obtain respectively .70/.66 and

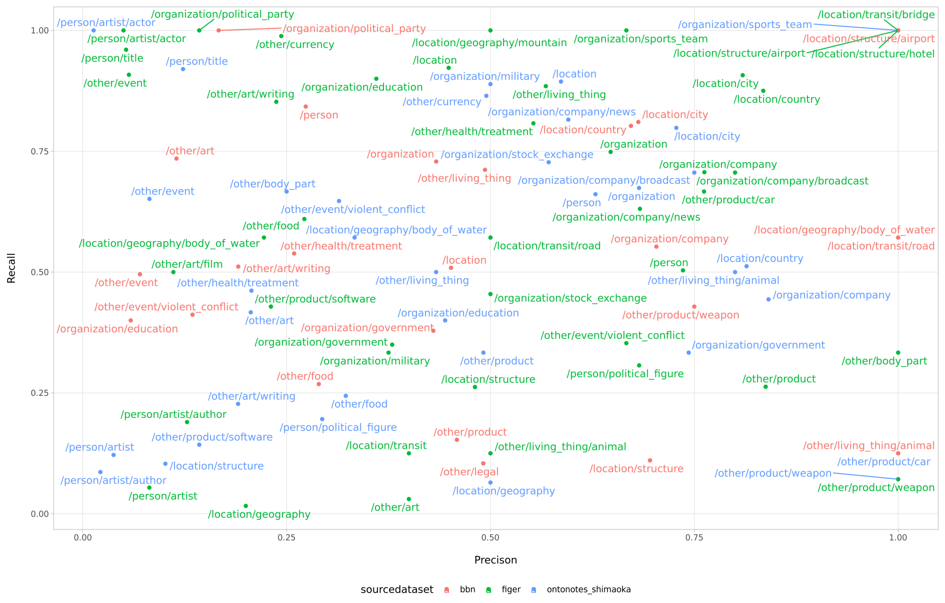


Figure 4.6: Precision/Recall plot of models trained on BBN, FIGER or OntoNotes and used to predict examples from the test set of OntoNotes, with OntoNotes type vocabulary. Only types covered by mappings are reported (see Table 4.2). Blue dots represent performance of in-domain model.

.72/.34 when trained on FIGER, and .63/.26 and .45/.64 when trained on OntoNotes. This is probably due to the presence and to the variety of these types in the training set: in BNN there are 6, 530 examples for `BBN.WORK_OF_ART`, while in FIGER there are 41, 876 examples for `FIGER.art`, and in Ontonotes there are 14, 471 examples for `OntoNotes.other/art`. Similarly, for `BBN.PRODUCT` there are 1, 396 examples, while in FIGER there are 41, 078 examples for the type `FIGER.product` and in OntoNotes there are 7, 826 examples of the type `OntoNotes.other/product`. The quantity of examples seen during training is not the only important aspect, since similar observations do not hold for `FIGER.written_work` (Figure 4.5), where the in-domain model performs worst than cross-domain ones even if in BBN there are less examples of `BBN.WORK_OF_ART/BOOK`.

4.5 Noisy Annotation and Model Reuse

A FET model may overfit on language and label distribution in the training domain and evidences in Section 4.4.2 underlined how the domain affect performance on types that theoretically have the same semantics. In this section, a denoising method is used to reduce the sensibility of the model on label distribution. The intuition is to apply the denoising technique to obtain denoised version of the dataset before the training and train on the denoised verison instead of the noisy one. This approach is largely based on the model proposed in [OD19], since the same denoising paradigm is used and extended, also the original code was reused and extended. The main idea is to rely directly on the noisy domain to obtain two denoising models to apply in cascade. This may sound counterintuitive, but the performance obtained on manually annotated datasets by literature models trained on noisy datasets may denotes that the noise is not a major factor in the dataset and examples that reflect real case language and label distributions are present in the training set.

4.5.1 AutoDenoise

This section describes the AutoDenoise method and its differences (depicted in Figure 4.7) with OnoeD [OD19]. In addition a collection of training procedures is added to OnoeD to face different kinds of noise that are often found in datasets. Preliminary studies suggested that some of the denoising approaches for FET are tuned to face a specific noise typology within a certain dataset; thus, a description of the noise classification is provided, followed by an explanation of which of the proposed techniques applied on the architecture explained in Section 4.3.4 can face a specific category of noise.

OnoeD

[OD19]'s denoising approach (OnoeD) was first proposed in 2019 in the UFET domain [Cho+18]. Their technique aims to denoise distantly supervised data through a two-step data cleansing framework, which includes a *filtering model* trained to discard all excessively noisy examples and a *relabeling model* trained to relabel the remaining examples with a more appropriate set of labels (if needed). Both models share the same ELMo-based encoder architecture and are trained to recognize the synthetic noise that has been previously injected into a portion of ground truth data. This method was designed to denoise those datasets that mainly contain noise caused by *missing labels*, thus the relabeling model will often add labels to the examples rather than removing them.

OnoeD's denoising method was proved to allow ET models to achieve state-of-the-art performance [OD19]. However, it presents two major drawbacks that may limit its portability: it needs an additional portion of manually labeled examples, and it is best suited for denoising those datasets that suffer most from noise caused by missing labels. Unfortunately, ground truth examples are difficult to obtain (their generation requires expensive collection processes like crowdsourcing) and cannot be used to denoise other datasets, as they are annotated using dataset-

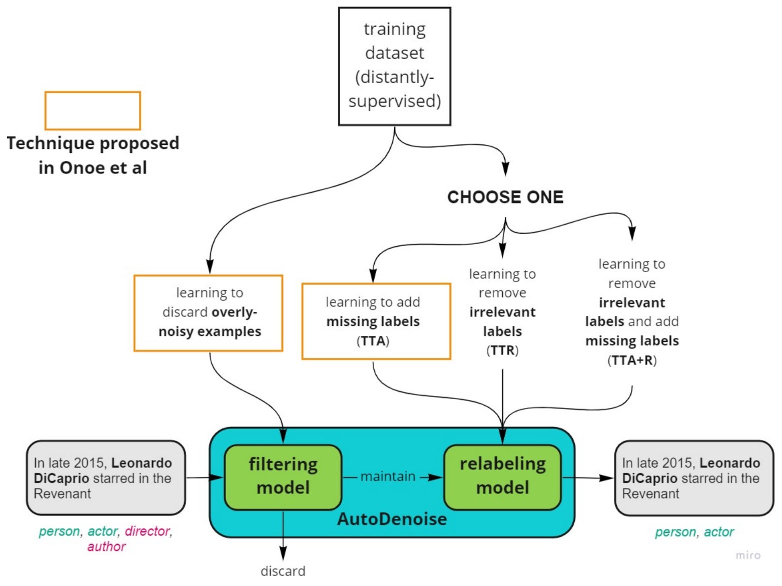

training dataset (distantly-supervised)
Technique proposed in Onoe et al
CHOOSE ONE
learning to discard overly-noisy examples
learning to add missing labels (TTA)
learning to remove irrelevant labels (TTR)
learning to remove irrelevant labels and add missing labels (TTA+R)
In late 2015, Leonardo DiCaprio starred in the Revenant
person, actor, director, author
filtering model
maintain
relabeling model
AutoDenoise
discard
In late 2015, Leonardo DiCaprio starred in the Revenant
person, actor
miro

Figure 4.7: AutoDenoise's submodules (filtering and relabeling); the modules' proposed training routines are suited for the noise category faced.

specific type vocabularies. The authors tried to adapt their approach to another dataset [Gil+14] by training both filtering and relabeling models with an augmented dataset (UFET [Cho+18]) representing the ground data; their predictions were then adjusted on the new dataset (OntoNotes [Gil+14]) by manually building a mapping between labels. Although this solution performs well and is easily adaptable to fit other datasets, it will always rely on the dataset which the models have been trained on, and requires the existence of the mapping between labels. Moreover, considering that UFET datasets use wider type vocabularies than FET ones, the construction of such mapping becomes challenging. Finally, once again it is worth noticing that the learned model is suited only for denoising those datasets that suffer most from noise caused by missing labels.

Training the Filtering Model The goal of the filtering model is to detect and discard excessively noisy examples. It consists of a binary classifier that decides whether each example (m, c, Y) should be discarded or not. The examples that need to be discarded are not known a priori; thus, the training procedure involves the creation of a new dataset by injecting synthetic noise into the ground truth examples. This corruption procedure swaps the ground truth labels of a percentage q of the clean examples with a set of non-overlapping labels taken from other examples. More formally, the ground truth labels Y of each selected example (m, c, Y) are replaced with the labels Y' of another randomly chosen example

(m', c', Y') such that $Y \cap Y' = \emptyset$. Then, the selected examples are grouped to form the positive examples $((m, c, Y'), 1)$ that the model should discard, while the remaining examples are grouped to form the negative examples $((m, c, Y), 0)$ that the model should keep. The model is trained on the examples of the corrupted dataset using a binary cross-entropy loss.

Training the Relabeling Model The relabeling model repairs those examples that make it through the filtering model but which still have errors in their labels. It consists of a function that maps each example (m, c, Y) to a new set of labels $\tilde{Y}$ such that $\tilde{Y} \subseteq Y$. This module is used to create the denoised examples $(m, c, \tilde{Y})$ that are the output of the denoising procedure. At inference time, an example is relabeled with labels $\tilde{Y}$ only if the new set contains at least two labels, otherwise it is discarded.

To train this module, a training set with synthetic noise is created by corrupting the distant supervision dataset. For each example (m, c, Y) the new set of labels $\hat{Y} \subseteq Y$ of a synthetic example $(m, c, \hat{Y})$ is built by dropping each element $y \in Y$ with a probability of p. Then the relabeling module is trained using the examples $((m, c, \hat{Y}), Y)$ and binary cross-entropy loss as optimization function.

AutoDenoise

AutoDenoise is a denoising approach proposed in this thesis to overcome the limitations of OnoeD; the main innovations of AutoDenoise are:

- AutoDenoise trains both filtering and relabeling models on distantly-supervised data, avoiding the need for ground truth data.

- AutoDenoise provides two new alternative training procedures for the relabeling model, allowing the end-user to produce a model that best fits the noise distribution of the dataset that needs to be denoised.

The first extension assumes that most of the distantly-supervised examples are clean; this assumption is motivated by the performance achieved by previous approaches trained on distantly-supervised examples and tested on manually annotated test datasets. Thus, it is assumed that noisy data can still be used to properly train both filtering and relabeling models. Clearly, noisy examples may interfere with training procedures, but assuming that clean data are the majority, this interference is expected to be negligible. As shown in Figure 4.7, the approach provides a total of three different procedures that can be used alternatively to produce three different relabeling models:

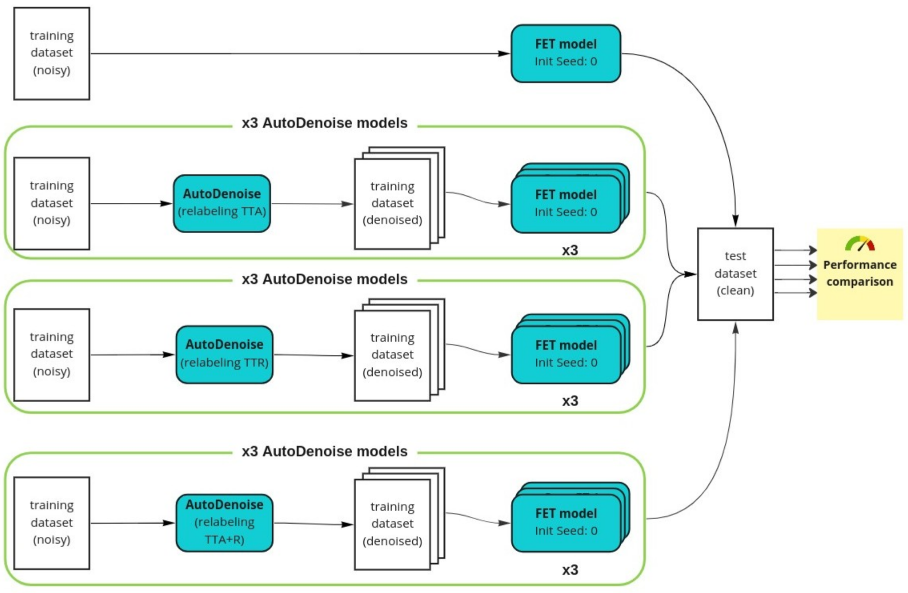

training dataset (noisy)
FET model Init Seed: 0
x3 AutoDenoise models
AutoDenoise (relabeling TTA)
training dataset (denoised)
x3
AutoDenoise (relabeling TTR)
AutoDenoise (relabeling TTA+R)
test dataset (clean)
Performance comparison

Figure 4.8: Schema of the AutoDenoise in-domain validation experiment: starting from the same noisy FET dataset, a FET model is trained. Different instances of AutoDenoise are trained using `TTA`/`TTR`/`TTA+R` relabeling technique. The nine AutoDenoise models are then used to obtain nine denoised datasets on which the same model, initialized with the same seed, is trained. The whole 10 models are compared using the original manually annotated test set

Trained-To-Add (TTA): originally developed by [OD19], this relabeling model can be useful for denoising those datasets that suffer mainly from noise caused by missing labels.

Trained-To-Remove (TTR): this relabeling model can be useful for denoising those datasets that suffer mainly from noise caused by irrelevant labels.

Trained-To-Add-and-Remove (TTA+R): this relabeling model can be useful for denoising those datasets that suffer from equal amounts of noise caused by irrelevant labels and noise caused by missing labels.

4.5.2 Research Question - AutoDenoise and Model reuse

Question 1: Does a denoising approach based only on a noisy dataset and on additional synthetic noise injection is useful to create a denoised version of the dataset?

		Macro averaged on Examples			Micro averaged			Macro averaged on types		
		P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Original FIGER		.785	.834	.809	.713	.820	.763	**.648**	.654	**.651**
	TTA	.770	**.851**	.808	.691	**.828**	.753	.627	**.664**	.645
Denoised FIGER	TTR	**.820**	**.851**	**.835**	**.755**	.824	**.788**	.614	.635	.624
	TTA+R	.798	.846	.821	.734	.820	.775	.639	.653	.646
Original BBN		.743	.804	.772	.723	.808	.763	.532	.569	.550
	TTA	.755	**.835**	.793	.728	**.837**	.779	.528	.560	.543
Denoised BBN	TTR	.759	.829	.793	.728	.833	.777	.514	.554	.533
	TTA+R	**.762**	.833	**.796**	**.737**	.835	**.783**	**.539**	**.575**	**.556**
Original OntoNotes		**.692**	.710	.701	.630	.660	.645	**.325**	.341	.333
	TTA	.681	.703	.689	.614	.640	.622	.212	.267	.236
Denoised OntoNotes	TTR	.687	**.751**	**.717**	**.631**	**.698**	**.663**	.323	**.353**	**.337**
	TTA+R	.657	.718	.686	.602	.669	634	.279	.330	.303

Table 4.8: In-domain performance of models trained on original datasets or on denoised datasets. The model trained on original dataset is initialized with seed 0. The models trained on denoised datasets are initialized with seed 0, three instances are trained on different datasets denoised with the same relabeling procedure (`TTA/TTR/TTA+R`)

Question 2: Removing annotation noise from the source domain with a dedicated denoising method brings benefit to the model reuse scenario in FET?

4.5.3 Experimental Setup

Starting from the original code, the original ELMo-based filtering model is trained maintaining the same hyperparameters of [OD19]. Relabeling models instead are trained using the original TTA procedure, or the proposed TTR or TTA+R procedures. The training phase is showed in Figure 4.7.

Figure 4.8 shows the steps to obtain models trained on denoised datasets and compare them with a model trained on the original noisy version. Three seeds (`10`, `20`, and `30`) are used to drive random initialization and example batching, obtaining 9 trained models (3 seeds times 3 procedures). These 9 trained AutoDenoise models are used to obtain 9 denoised datasets, that are used to train the FET model explained in section 4.3.4.

The nine FET models trained on denoised datasets are then used under in-domain and cross-domain with model reuse setup, exactly like the ones trained for previous experiments.

4.5.4 Performance in Source Domain

Table 4.8 reports the comparison of the same FET model trained on the original dataset or on the denoised ones. Only the model initialized with seed 0 is reported since the models trained on denoised datasets are always initialized with seed 0. In this experiment, robustness is evaluated using multiple denoised datasets obtained by applying the same relabeling procedure (`TTA`/`TTR`/`TTA+R`) three times. Since the batch sampling routine and the initialization of both filtering and relabeling models is driven by the same `seed`, three different values (`10`, `20`, and `30`) are used to obtain 3 versions of the same AutoDenoise model and consequently nine different denoised datasets.

Table 4.8, shows the performance of models trained on original datasets (original FIGER, original BBN, original OntoNotes) or on denoised datasets (Denoised FIGER, Denoised BBN, Denoised OntoNotes). The performance is measured by precision (P), recall (R) and F1-score (F1). The results are presented in terms of macro averaged on examples, micro averaged, and macro averaged on types.

Using **macro averaged on examples** and **micro averaged** metrics it can be seen that the models trained on denoised datasets have better performance in terms of recall and F1-score, especially for the Denoised FIGER and Denoised BBN datasets. For example, the model trained on the denoised FIGER obtained with `TTA` relabeling procedure has a recall of .851, which is higher than the recall of the model trained on the original FIGER model (.834). Similarly, the model trained on denoised BBN obtained using `TTA+R` relabeling procedure has an F1-score of .796, which is higher than the F1-score of the model trained on the original BBN (.772).

The results also show that the relabeling procedure used to denoise the datasets (`TTA`, `TTR` or `TTA+R`) generally has a positive impact on the performance of the models. The best results are achieved when using `TTA+R` on the BBN dataset and using `TTR` on FIGER and on OntoNotes.

Using **macro** metrics **averaged on types** it can be seen that on FIGER the performance increase do not depends on better performance over all types, but on better performance over types most present in the test set. On BBN and OntoNotes instead, a slightly increase of the performance obtained by models trained on datasets denoised respectively using `TTA+R` and `TTR` can be observed, which indicates that the general performance improvement is reflected by a general improvement on the single types.

Overall, the Table suggests that denoising the datasets can improve the performance of models when use in in-domain setting, although the specific relabeling procedure used also plays a role in the final results.

4.5.5 Performance in Target Domain

Tables 4.9, 4.10, and 4.11, show the performance under *model reuse with mappings* scenario of models trained on original or denoised datasets. The tables report the performance of in-domain models compared with performance obtained by models under model reuse with mappings setup. Differently from the in-domain evidences,

Training Dataset	Relabeling procedure	Macro averaged on examples P	R	F1	Micro averaged P	R	F1	Macro averaged on types P	R	F1
FIGER (in-domain)	-	**.785**	**.834**	**.809**	**.713**	**.820**	**.763**	**.648**	**.654**	**.651**
BBN	-	.683	.658	.670	**.635**	**.623**	.629	**.259**	**.267**	**.263**
BBN	TTA	.670	.653	.661	.621	.609	.615	.248	.247	.247
BBN	TTR	.676	.662	.669	.627	.619	.623	.251	.255	.253
BBN	TTA+R	**.688**	**.669**	**.679**	**.635**	**.623**	.629	.238	.249	.243
OntoNotes	-	**.677**	.734	.704	**.617**	.699	**.655**	.273	.317	.294
OntoNotes	TTA	.549	.670	.602	.478	.612	.535	.210	.254	.229
OntoNotes	TTR	.673	**.747**	**.708**	.611	**.703**	.653	**.289**	**.337**	**.311**
OntoNotes	TTA+R	.619	.741	.674	.550	.692	.613	.259	.322	.286

Table 4.9: Model reuse of FET models when used on test set of FIGER. The first row reports the performance of the in-domain model. Rows 2-5 reports the performance of the model reuse for the FET model trained on original BBN (row 2) or on its denoised version (rows 3-5). Rows 6-9 reports the performance of the model reuse for the FET model trained on original OntoNotes (row 6) or on its denoised version (rows 7-9). **Bold** indicates the best performance for a specific training domain for model-reuse, **underlined bold** indicates best performance overall.

Training Dataset	Relabeling procedure	Macro averaged on examples P	R	F1	Micro averaged P	R	F1	Macro averaged on types P	R	F1
BBN (in-domain)	-	.743	**.804**	**.772**	**.723**	**.808**	**.763**	**.532**	**.569**	**.550**
FIGER	-	.726	.735	.730	.675	.719	.696	.369	.457	.408
FIGER	TTA	.739	**.749**	**.744**	.689	**.734**	**.711**	.388	.462	.422
FIGER	TTR	**.746**	.733	.739	**.702**	.713	.708	.390	**.481**	**.431**
FIGER	TTA+R	.742	.738	.740	.699	.721	.710	**.396**	.459	.425
OntoNotes	-	.693	.687	.690	.689	.665	.677	**.391**	**.387**	**.389**
OntoNotes	TTA	.656	.611	.630	.651	.582	.609	.252	.288	.269
OntoNotes	TTR	**.722**	**.724**	**.723**	**.713**	**.710**	**.711**	.340	.383	.360
OntoNotes	TTA+R	.708	.709	.709	.699	.690	.695	.329	.370	.349

Table 4.10: Model reuse of FET models when used on test set of BBN. The first row reports the performance of the in-domain model. Rows 2-5 reports the performance of the model reuse for the FET model trained on original FIGER (row 2) or on its denoised version (rows 3-5). Rows 6-9 reports the performance of the model reuse for the FET model trained on original OntoNotes (row 6) or on its denoised version (rows 7-9). **Bold** indicates the best performance for a specific training domain for model-reuse, **underlined bold** indicates best performance overall.

Training Dataset	Relabeling	Macro averaged on examples			Micro averaged			Macro averaged on types		
	procedure	P	R	F1	P	R	F1	P	R	F1
OntoNotes (in-domain)	-	**.692**	**.710**	**.701**	**.630**	**.660**	**.645**	.325	.341	.333
BBN	-	**.315**	.244	.275	.357	.293	.322	**.213**	.215	**.214**
BBN	TTA	.308	.248	.275	.358	.298	.325	.202	.205	.203
BBN	TTR	.309	.248	.275	.355	.298	.324	.184	.208	.195
BBN	TTA+R	.314	**.250**	**.278**	**.360**	**.300**	**.327**	.192	**.222**	.205
FIGER	-	**.333**	**.272**	**.299**	.403	**.344**	**.371**	.368	**.388**	.378
FIGER	TTA	.321	.269	.293	.401	.343	.370	.391	.373	**.382**
FIGER	TTR	.318	.250	.280	.391	.321	.352	.350	.370	.360
FIGER	TTA+R	.331	.264	.294	**.404**	.337	.367	**.396**	.368	.381

Table 4.11: Model reuse of FET models when used on test set of OntoNotes. The first row reports the performance of the in-domain model. Rows 2-5 reports the performance of the model reuse for the FET model trained on original BBN (row 2) or on its denoised version (rows 3-5). Rows 6-9 reports the performance of the model reuse for the FET model trained on original FIGER (row 6) or on its denoised version (rows 7-9). **Bold** indicates the best performance for a specific training domain for model-reuse, **underlined bold** indicates best performance overall.

where AutoDenoise shown to be effective in raising in-domain performance, models trained on datasets denoised with AutoDenoise and used in *model reuse with mappings* setup do not robustly overperform the same models trained on original datasets. This negative evidence can be explained by focusing the macro metrics averaged over types in Table 4.8. From these metrics can be seen that performance over single types do not increase by using denoised datasets. When a FET model is used in *model reuse with mappings* setup, its performance of single types are directly transferred on the target domain, i.e., the more the model performs well on single types, the more the model can recognize the same types in the target domain.

Some interesting evidences can still be drawn by inspecting table 4.11. In this table it can be seen that single types in sentences from OntoNotes are better recognized from models trained on FIGER than from model trained on OntoNotes. This may be related with the already discussed vast presence of the type `OntoNotes.other` in the train set of OntoNotes, that leads to models biased in recognizing that type. This evidence was also present in Table 4.7, where the same behavior was observed.

4.6 Conclusions on Model Reuse

In Fine-grained Entity Typing model reuse between domains is motivated by the high coverage of general purpose types that is an intrinsic property of FET models.

Experiments on model reuse between common benchmark datasets proposed in literature to validate FET models proposed in this chapter show how the coverage of types in the target domain is a critical requisite to apply this paradigm. However the precision of models trained on a different source domain is often comparable or higher than precision of models trained on the target domain, showing that some types are better represented in some domains and that domain differences in terms of language distribution and annotation routines have not a major impact on this scenario.

Answering to research questions depicted in Section 4.1, we can state that models trained following literature strategies may overfit on source type distributions and tend to overperform on most represented types, it follows that the expected behavior on the target domain is linked to the similarity between domains in term of type distribution. This is not the only evidence observed with these experiments, heavy performance drop on the same types across source and target domain was observed on some types, suggesting that the language distribution is also crucial in this scenario.

A denoising technique found in literature is extended and used to alleviate label noise in the source domain, a problem that may affect model reuse. Experiments on the effectiveness of the different denoising procedures in in-domain scenario show that the denoising techniques can be used to improve performance when measured with literature standard metrics, this evidence let us to positively answer to the first research question depicted in Section 4.5.2. However, answering to the seconf research question about AutoDenoise and Model Reuse, the usage of models trained on denoised datasets on target different domain (model reuse paradigm) show that the performance enhancement only regards the capabilities on recognizing more correct types on the single example, but not in a general ability on all types.

In the next Chapter, the problem of partial type coverage is faced by exploring two different scenarios, already described in user stories in Section 3.2.1. In the specialization scenario a model trained on a source domain has to be specialized to be used in a target domain, this is experimented by setup family-based experiments in which starting from the knowledge of a type, new subtypes have to be learned. In the full-fledged adaptation scenario instead, a model trained on a source domain has to be adapted to be applied in a target domain, with some equivalent or specialized types, but also with new types. In both these scenarios additional training is necessary.

Chapter 5

Neuro-symbolic Fine-grained Entity Typing for Domain Specialization

In this Chapter, Domain Adaptation in Fine-grained Entity Typing is faced with techniques to exploit knowledge on source domain while extending the type coverage in target domain (see Section 3.2, in particular subsection 3.2.4). The problem was already described in user stories (Section 3.2.1) and formalized in Chapter 3. The main intuition is to exploit relations between type vocabularies from source to target domain, taking advantage of a FET model trained on a source domain to extract information on source domain and transfer them on the target using explicit knowledge expressed through mappings and implemented in different ways. This intuition differs from the subsymbolic transfer learning approach that transfers a pretrained encoder between tasks, trusting its ability of extracting important information across tasks without considering the interpretation of this information, that are directly used as subsymbolic representations.

In this chapter, first a NeuroSymbolic Integration framework, KENN [DS19], is introduced and described, then its adoption as explicit knowledge enhancer in a FET model is detailed. KENN is used to exploit type vocabularies relations in order to influence the predictions of target types with evidences on source types, as described in subsection 3.2.4.

5.1 Knowledge Enhanced Neural Network

Knowledge Enhanced Neural Networks (KENN) [DS19], is a NeuroSymbolic Integration framework that modifies the predictions of a neural network using fuzzy logic clauses. The main intuition is to define logical rules that involve the classes of a classification task, relying on external or on human-curated knowledge. KENN

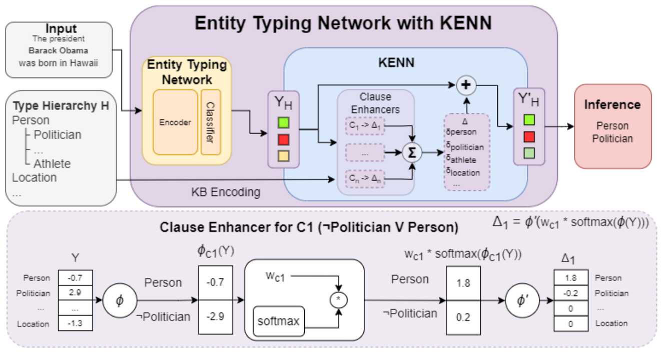


Figure 5.1: (top) The knowledge enhancement module is placed on top of the Entity Typing Network, KENN exploits each clause C_i producing a variation vector Δ_i, Δs are summed obtaining the vector Δ that expresses a variation for each type. Δ is then summed to predictions Y_H, obtaining the enhanced predictions Y'_H. (bottom) Each clause is exploited by a Clause Enhancer that: selects the preactivations of types involved in the clause and invert the sign of preactivations for types negated in the clause using the function ϕ_{c_i}, computes the enhancement using `softmax` to approximate Gödel t-conorm, invert the enhancement value according to the negation in the clause and adds 0 for the other types using ϕ'_{c_i}

evaluates the satisfaction of these rules during training and modifies the network predictions to increase the satisfaction of them. This process affects backpropagation, since *knowledge enhanced predictions* are involved in the loss computation. An example of logical rule can be *instances that belong to class A, also belong to class B*, that can be traduced with the logical proposition $A \to B$, or the equivalent $\neg A \lor B$. This proposition is useful to express hierarchical relations between classes in multilabel problems (e.g., "all *actors* are *persons*"), thus can be used in in-domain FET to include the hierarchy in the training process (e.g., "all instances of class `FIGER.person/actor` are also of class `FIGER.person`") and cross-domain FET to express equivalence, generalization, specialization or disjunction mappings (e.g., "all instances of class `FIGER.person` are also of class `BBN.person`"). A more accurate description is given in Section 5.3.

While NeuroSymbolic approaches [Bad+22; Mar+20; DGS17; DS19; Seo+21] (including KENN) have been used in similar multilabel classification tasks, they have not been applied to named entity classification. To apply KENN to FET, different ways to encode hierarchical structures into clauses are explored, and a new multi-loss function to improve training is designed.

5.1.1 How does KENN work?

KENN injects external knowledge expressed as clauses: universally quantified disjunctions of literals (atomic formulas or their negations). In this thesis, only clauses composed by unary predicates, that shares one variable are considered. For brevity, $\bigvee L$ denotes the clause $\bigvee L(x)$. Formally, the Gödel t-conorm `max` is used to represent the observance of a clause; this is approximated by the differentiable $softmax$ function to allow learning.

In KENN, logical rules are expressed using the equivalence $A(x) \Rightarrow B(x) \equiv \neg A(x) \vee B(x)$. These disjunctions are interpreted via a Gödel t-conorm which corresponds to the `max` operator. The logical knowledge is injected into the architecture of a neural model as shown in Figure 5.1.

KENN as a composition of Clause enhancers

KENN promotes the adherence to clauses through a set of *Clause Enhancers* (CE), one per clause, which are later combined. Figure 5.1 shows an example for the clause $\neg$`Politician`$\vee$`Person`. For each clause c_i, the i-th CE produces a variation Δ_{c_i} of the confidence scores Y defined by the equation

$$\Delta_{c_i} = \phi'_{c_i}(w_{c_i} \cdot softmax(\phi_{c_i}(Y))). \tag{5.1}$$

The function ϕ_{c_i} filters out the predicates not involved in c_i and changes the signs of scores of predicates negated in c_i; w_{c_i} is the clause weight expressing the impact of the CE on the enhanced prediction (a higher weight yields a larger variation). The value of each w_{c_i} is initialized before training and can optionally be set as learnable. The function ϕ'_{c_i} changes the signs of the variations according to negated predicates in c_i and expands them back to the original dimension of Y_H filling with 0s the variations for the predicates not involved in c_i. Each clause-specific variation vector Δ_{c_i} contributes to the update of the final prediction vector $Y'_H := Y_H + \sum_{c_i \in C} \Delta_{c_i}$. Hence, the scores for types in the clause influence each other; this influence works in both directions (see Figure 5.1).

5.2 Encoding a hierarchy for FET

The type hierarchy of a FET datasets is often a forest-shaped set of types. $S \prec T$ expresses that S is *direct* subclass or subtype of T (instances of S are also instances of T). The most natural encoding of the relationship $S \prec T$ as a predicate logic formula is $\forall x(S(x) \rightarrow T(x))$, which corresponds to the clause $\neg S(x) \vee T(x)$ (from now on simplified as $\neg S \vee T$).

To model the hierarchical relations between types, three encoding techniques (exemplified in Figure 5.2) are proposed:

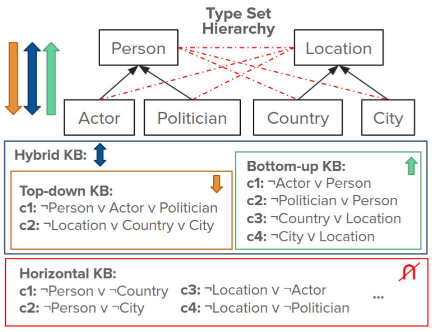


Figure 5.2: Bottom-up, Top-down, Hybrid and Horizontal KBs.

- The **Bottom-up strategy** is defined based on the open world assumption and consists in transforming each pairwise relationship $S \prec T$ into a *bottom-up clause* $\neg S \vee T$. For example, $\neg Politician \vee Person$ express that every instance of `Politician` is an instance of `Person`).

- The **Top-down strategy** is defined based on the close world assumption. Given a type T (e.g., `Person`) with n subtypes S_i, (e.g., `Politician`, `Actor`), the *top-down clause* $\neg T \vee S_1 \vee \ldots \vee S_n$ specifies that an instance of T (`Person`) must be an instance of at least one of its subtypes S_i (`Politician`, `Actor`, etc.).

- The **Hybrid strategy** is defined as the collection of all the top-down and bottom-up clauses. Even if it seems redundant, it is interesting assessing whether combining the signals from both strategies can bring additional benefits to the learning process.

- The **Horizontal strategy** is defined by exploiting negative dependencies in the form $S \to \neg T$, this is traduced in the clause $\neg S \vee \neg T$ that is not satisfied only if both S and T are predicted. This strategy is useful to avoid joint prediction of disjoint types e.g., avoiding the joint prediction of `Person` and `Country` for the same entity.

Although the top-down and the bottom-up KBs represent the same hierarchical structure, the latter is more precise than the former. If the model is confident about a given type, *bottom-up clauses* propagate this certainty towards the root of the hierarchy, while *top-down clauses* only guarantee that one of the subtypes is appropriate, but do not express which one. The reason, of course, is the tree structure, where each type has only one parent, but potentially many descendants.

Note that in all KBs (bottom-up, top-down, hybrid and horizontal) the first term of a clause (i.e. the antecedent in the logic formula) is always negated, thus *bottom-up clauses* will always lead to negative δ on subtypes, while *top-down clauses* will always lead to negative δ on supertypes. The remaining terms always receive a positive δ. In the horizontal KB, both terms receive a negative δ, this is not a problem since if types preactivations are unbalanced (i.e., the model *prefers* a type instead of the other), one of the δs became 0, due to the usage of the `softmax`.

Horizontal KB is particularly useful in domain adaptation scenarios, since horizontal clauses can inject in the target domains the preactivation of disjoint source type that are not present in the target domain; this can be helpful since this is an external information to the target domain. For brevity, each strategy is identified with the KB it produces, and call them **Bottom-up KB, Top-down KB**, **Hybrid KB**, and **Horizontal KB**.

5.3 KENN for FET (in-domain)

Incorporating the hierarchy in the information used to train a FET model is an intuition present in FET literature (Section 2.2). Most of the proposed solution employs an hard-encoding of the hierarchy, with dedicated architecture that predict on different hierarchical levels or by filtering the predictions according to the hierarchy and single-path assumptions (a prediction is correct only if all predicted types lies in the same branch of the hierarchy). With a dedicated architecture, hierarchical information are often represented with subsymbolic embeddings that are produced by the model, with a post-hoc filtering the model does not have a *perception* of the hierarchy, since it is free to produce wrong prediction that will be fixed. Methods that do not incorporate hierarchy often approximate the hierarchical relations between types using cooccurrence in training data as a proxy, since in most of FET dataset the annotation of a type always implies the annotation of its ancestors (e.g. an entity annotated with `FIGER.person/artist` is always annotated also with `FIGER.person`).

The intuition of using KENN to explicitly represent the hierarchical relations between types is to avoid the necessity by the model of learning these relations during training, having this information from the first epoch can speed up the training and let the model to extract more sophisticated information useful to solve the FET task. Moreover, the differentiability of KENN let the encoder to adapt its behavior to the presence of a hierarchical enhancement, possibly adding a degree of freedom since the prediction does not rely anymore only on a neural classifier.

5.3.1 Research Questions

The usage of KENN in in-domain FET raise some general questions about the injection of hierarchy-related information in the FET model:

Question 1: Does the usage of KENN to encode the hierarchy speed up the training?

Question 2: Does the usage of KENN to encode the hierarchy increase the performance?

Question 3: How does the different parametrization of KENN (kb encoding, initial weight, learnable weight, loss function) impact training and the behavior of the FET model?

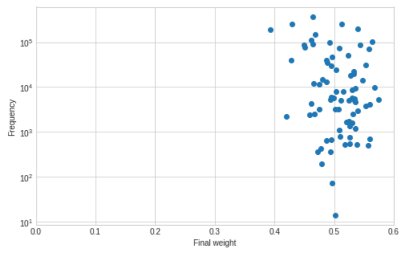
5.3.2 Experimental Setup

The entire network (Figure 5.1) consists in a KENN module that implements a KB on-top of the ET Network described in Section 4.3.4. The encodings exposed in Section 5.2 are experimented in a classic FET setup where a network is random initialized and trained with supervised paradigm. The hyperparameters are the same as in Section 4.3.5, plus the KENN hyperparameters:

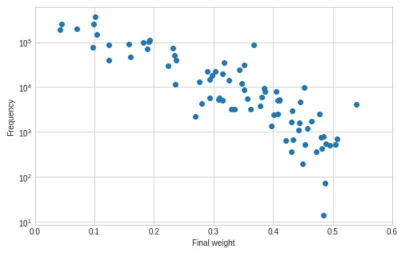
- KB encoding: `bottom-up`, `top-down`, or `hybrid`, all encodings are experimented.
- Clause Weight initialization: 0.5, 1, 2, different initial weights are used in order to measure the impact of a different initialization values and the evolution of the weights during training.
- Learnable Weight: `True` or `False`, clause weight is both fixed or learnable during training.
- Loss Function: Binary Cross-Entropy or Multiloss setup with Binary Cross-Entropy, loss evaluated only on the enhanced predictions or both before and after the enhancement.

5.3.3 In-domain FET with KENN - Results

Preliminary experiments on the usage of KENN in in-domain FET were conducted to inspect the differences between a FET model without KENN that has to learn the hierarchical dependencies between types and a FET model with KENN that knows hierarchical dependencies from the first epoch. For these initial experiments the horizontal KB was ignored, since it is designed to transfer the disjunction between source types and target types.



(a) Distribution of clause weights at the end of the training with BCE on enhanced prediction Y'_H



(b) Distribution of clause weights at the end of the training with BCE on prekenn predictions Y and BCE on enhanced prediction Y'_H

Figure 5.3: Comparison of final clause weights optimized by minimizing BCE only on enhanced predictions (left) or both on pre-KENN predictions and on enhanced predictions (right) with the other hyperparameters set with the same values. The optimization of both predictions encourage the usage of KENN clauses only if they are needed to correct a prediction.

To train the ET model with KENN, **BCE multiloss** is introduced: BCE multiloss is a variation of the BCE to simultaneously optimize Y_H and Y'_H when clause weights are *learnable* parameters. During preliminary experiments, the adoption of this loss favors the optimization of clause weights (see Figure 5.3). This is because a type t that is correctly predicted in Y_H, do not need to be enhanced, thus the weights of the clauses referring to t will decrease. Conversely, if KENN is needed to correctly predict t, the weights will not be lowered. The final loss is the average of BCE on Y_H and Y'_H.

Preliminary experiments result in a negative evidence, summarized by Figure 5.4: FET models with KENN have an initial boost in macro F1-score averaged over examples with respect to models without KENN. This boost is independent on the KB and is positively correlated with the initial clause weight (in the experiment it ranges values in $\{.5, 1, 2\}$, i.e., the higher the initial weight, the greater the boost. However, this boost vanishes over epochs, leading to models with the same performance. This is probably due to the fact that hierarchical dependencies between types are intrinsically represented in the training dataset, since all annotations respect the hierarchy, i.e., if a type is present in an annotation, its ancestors in the type hierarchy are always present in the annotation; thus during the training even a FET model without KENN learns these dependencies.

Answering to research questions, this preliminary experiment suggests that KENN can be useful in scenarios with a limited amount of data, where the observed initial boost may help the model and speed up the training. Moreover, both the initial boost and the differences in final clauses' weights show that KENN is able

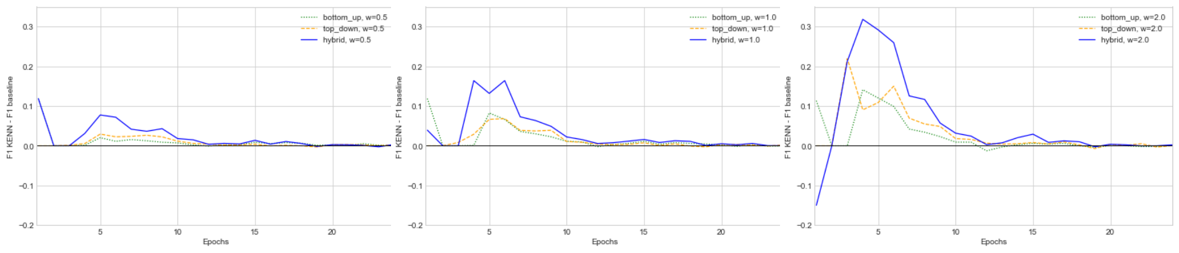


Figure 5.4: Differences in F1 during the training of a FET model with KENN and the same FET model without KENN. KENN is used with *bottom up* (green), *top down* (yellow) or *hybrid* KB. Clauses weight is initialized with values .5 (left), 1 (center) and 2 (right). Independently by the KB, KENN based network have an initial performance boost that depends on the initial clause weight, however the boost disappear over epochs.

to inject explicit type dependencies in the network. These evidences, found for each parametrization of KENN with BCE multiloss, are a key feature to handle the domain differences in a domain adaptation scenario, as explained in Chapter 3.

5.4 Specialization of a FET model

In this section, techniques for model specialization are proposed and compared. Model specialization is a use case presented in the user stories in Section 3.2.1 where a trained FET model needs to be specialized to recognize subtypes of already known types and only a limited quantity of annotated data is available to adapt the FET model. The key intuition is to exploit knowledge about supertypes to favor the learning of new target subtypes. For example, given a network that can recognizes *locations*, how can this ability be used to improve the recognition of *cities* and *countries*?

5.4.1 Research Questions - Specialization of a FET Model

Question 1: How does a FET model can be specialized to recognize subtypes of already known types by exploiting the hierarchical relations in a low-resource scenario?

Question 2: Does the injection of explicit hierarchy representation during training is more effective than hierarchy-driven initialization of a specialized neural classifier?

5.4.2 Base Specialization Network

Additional classification layer: Given a trained FET model with the architecture shown in Figure 2.3, new classification neurons are needed to learn the new target types. These neurons are gathered in a single classification layer.

Additional classification layer smart initialization: In the specialization scenario, the classification layer of the trained FET model can be used to smartly initialize neurons in the additional classification layer (i.e., the layer used to classify specialized types). Given a source type s and n specialized types $t_1, ..., t_n$ such that $\forall i \in [1, n]$ $t_1 \prec s$, the parameters of the t_i's neuron are initialized with the neuron's parameters of s, such that in the first step the predictions on s will match the predictions on $t_1, .., t_n$. This is an intuitive behavior and shown to be a useful setup both for positive and negative predictions.

Batch composition: Since the added classification layer must be trained to recognize the presence/absence of each specialization type, each training batch must be heterogeneous. Batches exclusively composed of examples annotated with specialization types may lead the model to always predict at least a specialization type. In addition, each training batch contains an equal number of examples for each specialization type.

Inference: In the specialization scenario, it is fundamental that predictions about types in H remain the same before and after the specialization. *ThresholdOrMax* is first applied to Y'_H then specialization types are inferred using a conditioned *ThresholdOrMax*: the type with the highest confidence score in Y'_s is inferred only if it is also the highest in Y'.

5.4.3 Experimental Setup

Source domain and trained FET model: To experiment with specialization scenario it is necessary to define a source and a target domain. Starting from the FET benchmark datasets presented in Section 4.3.1, types with `maximum depth` (2 for FIGER and BBN, 3 for OntoNotes) are collected by their father forming families (9 for BBN, 22 for FIGER, and 11 for OntoNotes). All types with depth < `maximum depth` are considered part of the source domain, so the FET model trained on source domain will be able to predict them.

Source domain dataset creation: Starting from the source domain types (explained in the previous paragraph), each example of the original training set is filtered and only the source domain types are kept.

Transplanted test specialization dataset: Due to the imbalance of types in the original test sets, new test sets are created by transplanting examples from the training set in which each type has at least 30 examples.

Few-shot FET model specialization: A common real case is to have low quantity of available data for very specific types. To mimic this scenario, 5-shots, 10-shot and 20-shot datasets are created for each family: for each type in the family, k examples are randomly selected from the original training set and are used in the specialization training set, other randomly selected k examples are used in the specialization dev set.

5.4.4 Experimented specialization techniques

Different techniques are experimented in this scenario:

Classifier: the base specialization network explained in Section 5.4.2 is implemented with standard classification neurons both for the training in source domain and for the training in target specialization domain.

Box embedding classifier: Box Embeddings [Das+20; Ono+21] were proposed in FET literature due to their expected ability to exploit implicitly model hierarchical relations between coarse and fine grained type. A classifier with box embeddings optimized during training is trained on the source domain. The initialization heuristics explained in Section 5.4.2 are used to initialize the additional specialization box embedding based classification layer.

KENN classifier: relying on strategies defined in Section 5.2, hierarchical relations between new specialized types and their father in the source domain are exploited creating `bottom up, top down,` and `hybrid` KBs. `Horizontal` KB instead is used to link the new subtypes to all disjoint in the hierarchy types.

KB creation:

- `bottom up, top down,` and `hybrid` KBs are automatically created from the given hierarchy: for each specialization type $t_i \in t_1, ..., t_n$ such that $\forall i \in [1, n]\ t_i \prec s$, a `bottom up` clause $\neg t_i \vee s$ is created; for each family a single top down clause $\neg s \vee t_1 \vee ... \vee t_n$ is created, the `hybrid` KB is the union of `bottom up` and `top down`.
- `horizontal` KB instead is critical to create, since the disjunction is not always implied by the hierarchy. For example, in BBN there are the types

Dataset	Trained Instances	#Families	#Techniques	K-Shots	#Seeds	Specialization Techniques
BBN	486	9	6	5/10/20	3	Bottom Up, Top Down, Hybrid, Horizontal, Classifier, Box
FIGER	1188	22	6	5/10/20	3	Bottom Up, Top Down, Hybrid, Horizontal, Classifier, Box
OntoNotes	495	11	5	5/10/20	3	Bottom Up, Top Down, Hybrid, Classifier, Box

Table 5.1: Details on trained instances for the experiments on specialization scenario. Due to the high quantity of experiments, results and plots are partitioned in Section 5.4.5, where general evidences are described, and in Section 5.4.5, where punctual results on Datasets × K-shots combinations are described.

`BBN.LOCATION` and `BBN.GPE` (geo-political entity); due the slight semantic difference between these types, they often share entities. For this reason, disjunction is computed on the train set and only those types which never appear in the same annotation are considered disjoint. Note that in OntoNotes no types satisfy this condition, thus `Horizontal` KB is not created with this dataset.

5.4.5 Results

In this sections results of the experiments on techniques to specialize FET models are discussed. Due to the massively quantity of trained instances (details in Table 5.1) this section contains general evidences and observed patterns, while Section 5.4.5 contains the punctual comments on Datasets × K-Shots combinations.

Plots

Three kinds of plots are used to visualize the behavior of the different experimented specialization techniques: barplots (Figures 5.5, 5.6, and 5.7), histograms (Figures 5.8, 5.11, and 5.12), and stacked barplots (Figures 5.9, 5.13, 5.10, and 5.14).

Barplots: are used to compactly compare average performance of all techniques. Each barplot shows the F1-score grouped by K-shot value (5/10/20), family (9 in BBN, 22 in FIGER, or 11 in OntoNotes) and colored by specialization technique. This plot is useful to **compare the specialization techniques and shows trends across families and K-shots values**.

Stacked Barplots: are used to count on how many types for each family a KENN-based technique performed better than a non-KENN based technique. Each barplot shows the count grouped by K-shot value (5/10/20), family (9 in BBN, 22 in FIGER, or 11 in OntoNotes) and colored by specialization technique. An additional grey bar show the total number of specialized subtypes for a given family. This plot is useful to **visualize the type coverage of the enhancement obtained by a KENN-based technique**. Each bar contains also the average improvement/worsening percentage by the KENN-based technique with respect to the non-KENN-based technique.

Histograms: show the distributions of F1-score difference between KENN-based instances (`Bottom up`, `Top Down`, `Hybrid`, and `Horizontal`) and non-KENN based instances (`Classifier` and `Box`). Histograms are grouped by K-Shots and KENN strategy. Each bar in the histogram represents the count of instances for which a KENN-based instance has a difference of F1-score on a specialization type in the range expressed by x-axis. While barplots and stacked barplots group types by family, this type of plot is useful to **visualize differences between models' performance on types level**.

General Results

Table 5.2 summarizes performance of the different specialization approaches experimented. The F1-scores computed by averaging performance on specialization types for each dataset and specialization approach are reported in the table, providing a general view of the entire specialization experimentation. From the table it can be seen that independently on the metrics, on the dataset, and on the value of k, all `KENN`-based approaches outperform the `Box`-based specialization approach. In the same way, except one case (`Top Down` in 20-shot OntoNotes), all KENN-based appraches outperform also the `Classifier`-based specialization approach. This indicates that KENN effectively propagate the information from type to subtype, thus being helpful for the specialization process. The high value of STD is related to the large performance difference between families (observable in the barplots, Figures 5.5, 5.6, and 5.7).

Barplots (Figures 5.5, 5.6, and 5.7) show that independently of KENN encoding strategy, the usage of KENN to propagate information from known source types to new target types results in a general performance increment with respect to simple `classifier` or `box`; moreover, the less the examples the more the increment. This evidence can be seen focusing on the first column of Figures 5.5, 5.6, and 5.7. Besides for 4 families out of 41, in the first column of the images can be seen the increment of at least one KENN-based technique with respect to `classifier` and `box`, the only families in which this is

	5-shot		10-shot		20-shot	
	F1	STD	F1	STD	F1	STD
BBN						
`Box`	.337	.213	.388	.243	.413	.258
`Classifier`	.356	.144	.418	.179	.507	.239
`Bottom up`	.588	.194	.622	.188	**.654**	.173
`Top down`	.5	.199	.534	.21	.565	.179
`Hybrid`	**.591**	.195	.595	.172	.645	.189
`horizontal`	.581	.183	**.64**	.173	.64	.166
FIGER						
`Box`	.246	.212	.259	.233	.43	.32
`Classifier`	.303	.234	.343	.25	.519	.307
`Bottom up`	**.616**	.286	**.666**	.288	**.688**	.286
`Top down`	.481	.261	.577	.297	.615	.294
`Hybrid`	.572	.285	.65	.29	.672	.284
`horizontal`	.605	.289	.658	.284	.673	.269
OntoNotes						
`Box`	.263	.185	.33	.242	.419	.27
`Classifier`	.329	.197	.437	.249	.62	.223
`Bottom up`	**.575**	.233	.598	.217	**.633**	.205
`Top down`	.49	.231	.531	.219	.565	.216
`Hybrid`	.565	.248	**.6**	.231	.626	.222

Table 5.2: Macro-F1 averaged over all types involved in the specialization experiment. For each dataset, the metric computed for 5/10/20-shot sampling is given. Performance of each specialization technique are reported with standard deviation. **Bold** highlights the best specialization technique for a given dataset and a given k-shot dataset.

not true are: `FIGER.art`, `FIGER.medicine`, `FIGER.metropolitan_transit`, and `OntoNotes.location/transit`, note that performance for these types are very low for all techniques, `FIGER.medicine` excepted. From histograms it can be observed that `Box` is often the worst model, with elevated distances from other models. This may indicates that a complex model like a classifier based on box embeddings may need more data than the quantity available in a few shot-setup. A last emergent pattern is that the `Top down` KB is often the worst KENN-based technique, this may indicate that a positive delta divided by all types is not useful since the prediction of all new types receives a boost and the model still need to learn how to discriminate between them.

Histograms (Figures 5.8, 5.11, and 5.12) show that the already discussed general performance increment over families highlighted with barplots is spread between new types and is not instead due to single types in families on which models overperforms. Moreover, the average magnitude of the improvement is higher in 5-shot than in 20-shot (Figure 5.8).

Lastly, stacked barplots (Figures 5.9, 5.13, 5.10, and 5.14) organize and quantify the general improvement on types observed with histograms, from them it can be seen how for the majority of families, most of subtypes are better recognized by applying a KENN-based technique than by smartly initializing a neural classifier or a box embedding classifier.

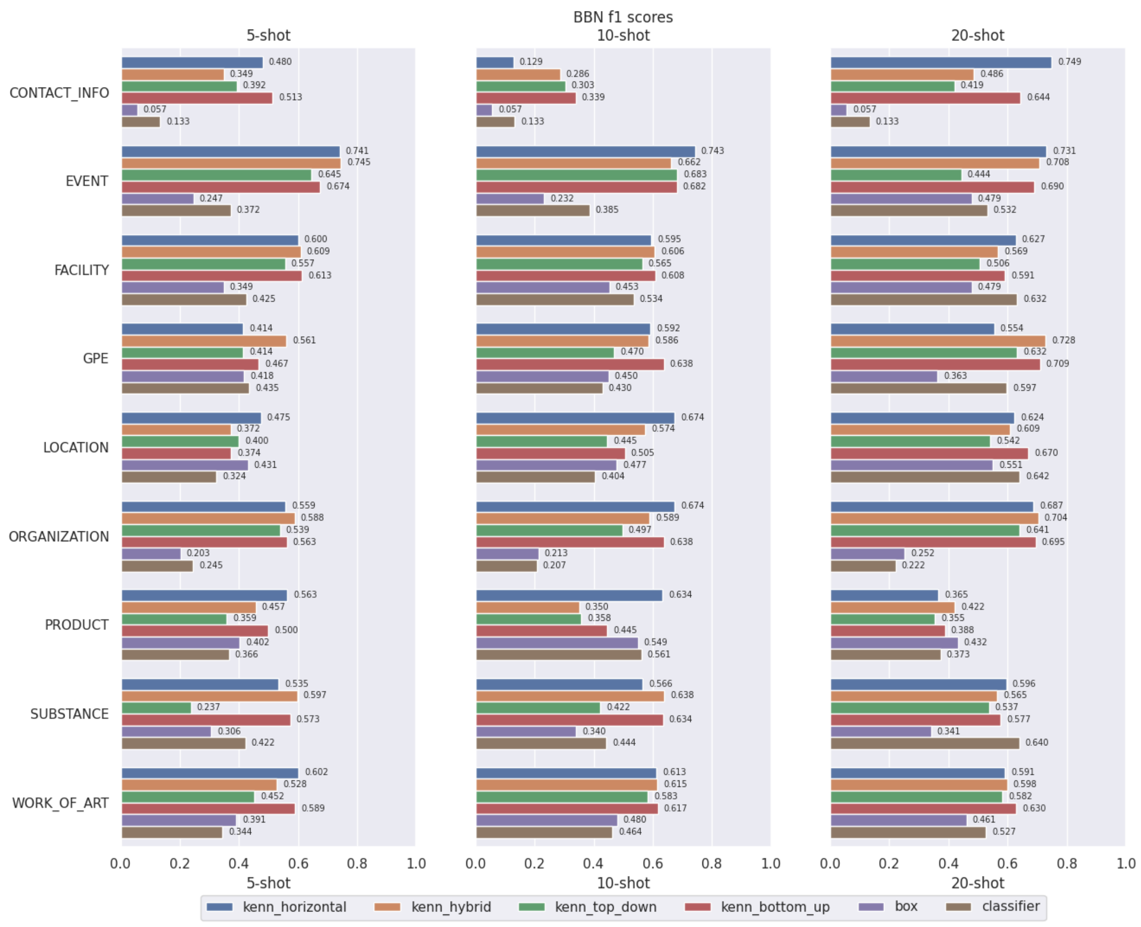


Figure 5.5: Barplot showing the F1-scores grouped by K-shot value (5/10/20) and BBN family, colored by specialization technique. In the majority of the cases, a `KENN`-based technique is effective to increase the F1 with respect to `Classifier` and `Box`-based techniques

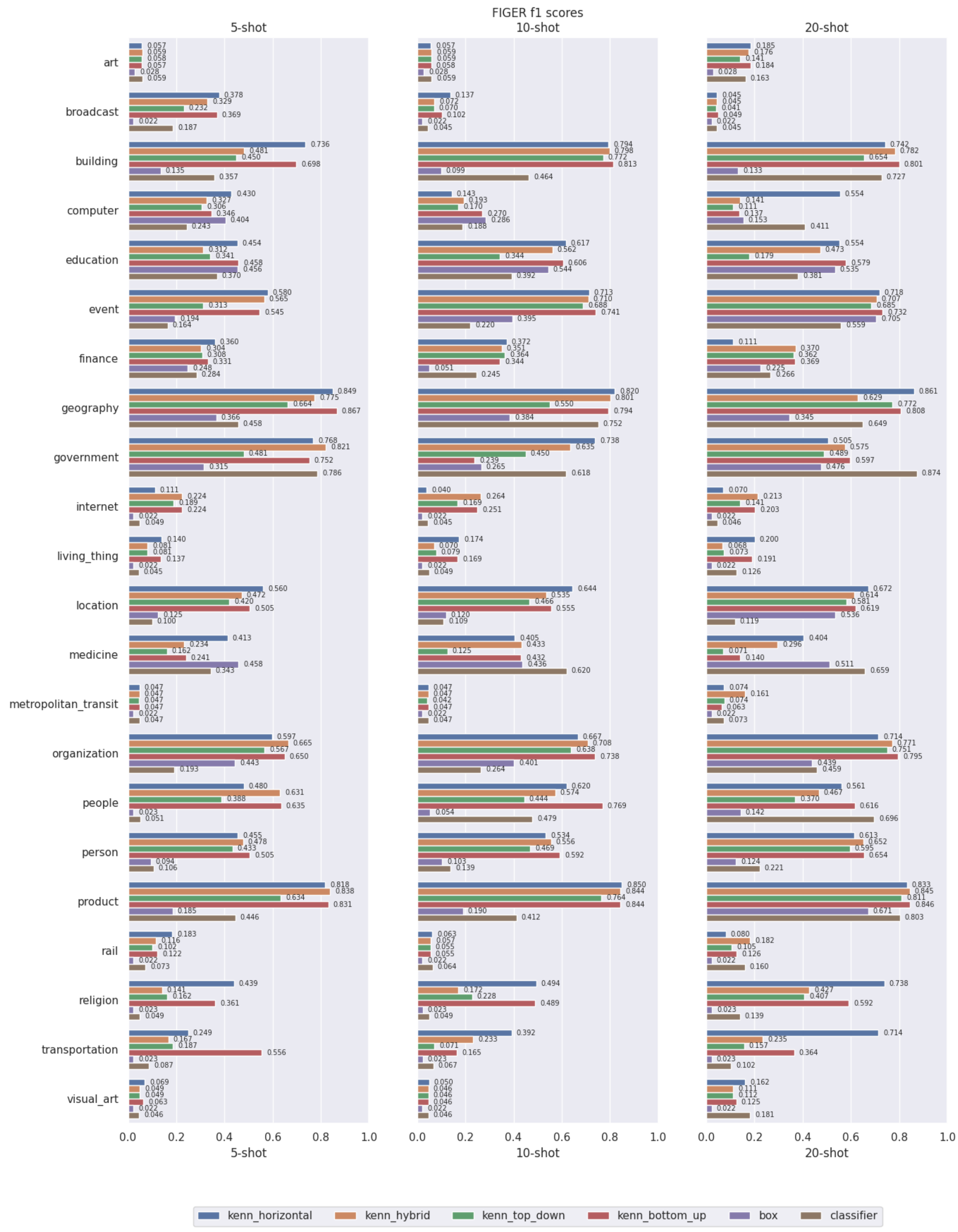


Figure 5.6: Barplot showing the F1-scores grouped by K-shot value (5/10/20) and FIGER family, colored by specialization technique. In the majority of the cases, a `KENN`-based technique is effective to increase the F1 with respect to `Classifier` and `Box`-based techniques

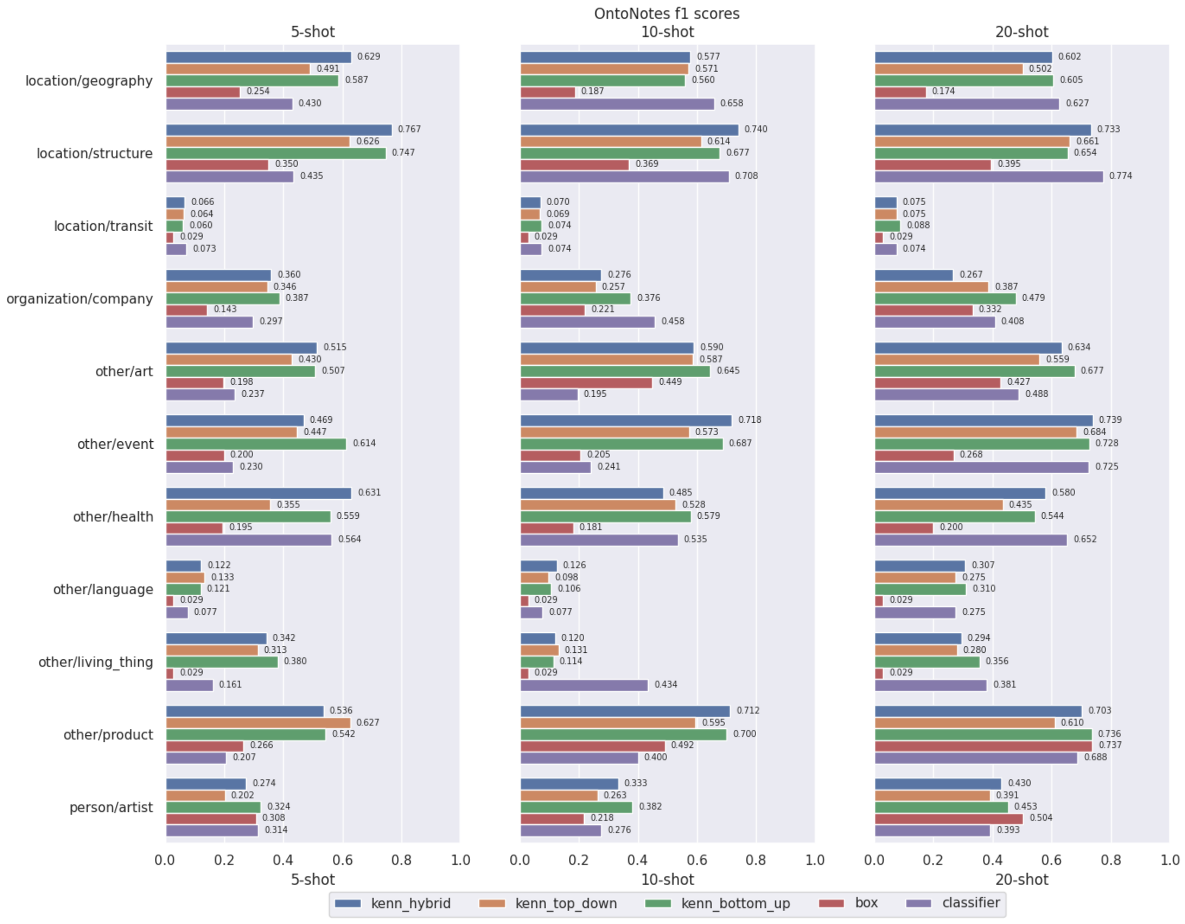


Figure 5.7: Barplot showing the F1-scores grouped by K-shot value (5/10/20) and OntoNotes family, colored by specialization technique.

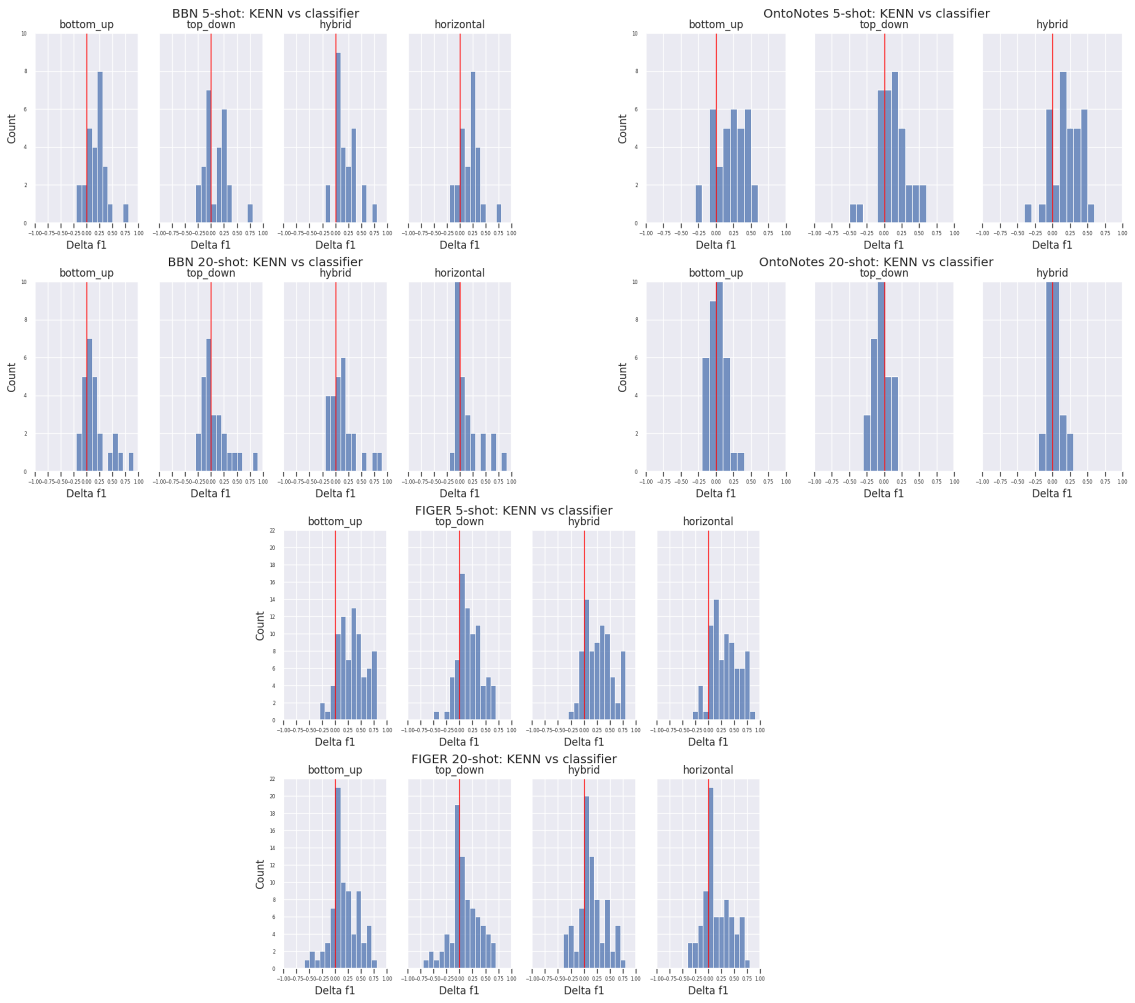


Figure 5.8: Histogram showing the distribution of the differences between F1 of KENN-based techniques and Classifier-based technique for each type involved in the specialization averaged by sampling seed used to create k-shot dataset. $k \in \{5, 20\}$. All datasets are reported: BBN (top-left), OntoNotes (top-right), and FIGER (Bottom). Difference is mostly positive, showing that KENN-based techniques perform better than Classifier-based technique in most of the types. The less the data used ($k = 5$), the more the increment caused by KENN-based techniques.

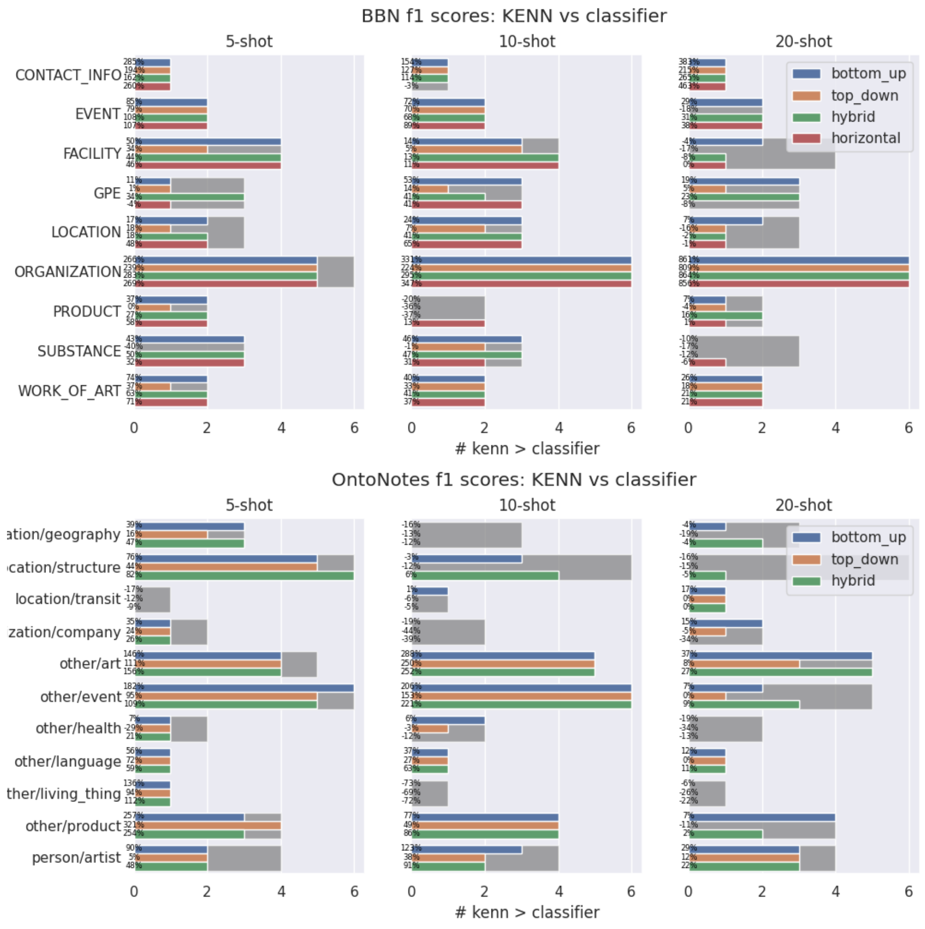


Figure 5.9: Stacked barplot showing the count of subtypes in each family for which a `KENN`-based technique (colored) perform better than the `Classifier`-based technique. The grey box represent the total number of subtypes for a given family. BBN is showed above, OntoNotes is showed below. In BBN, `KENN`-based techniques perform better than `Classifier`-based technique on the majority of types in each family.

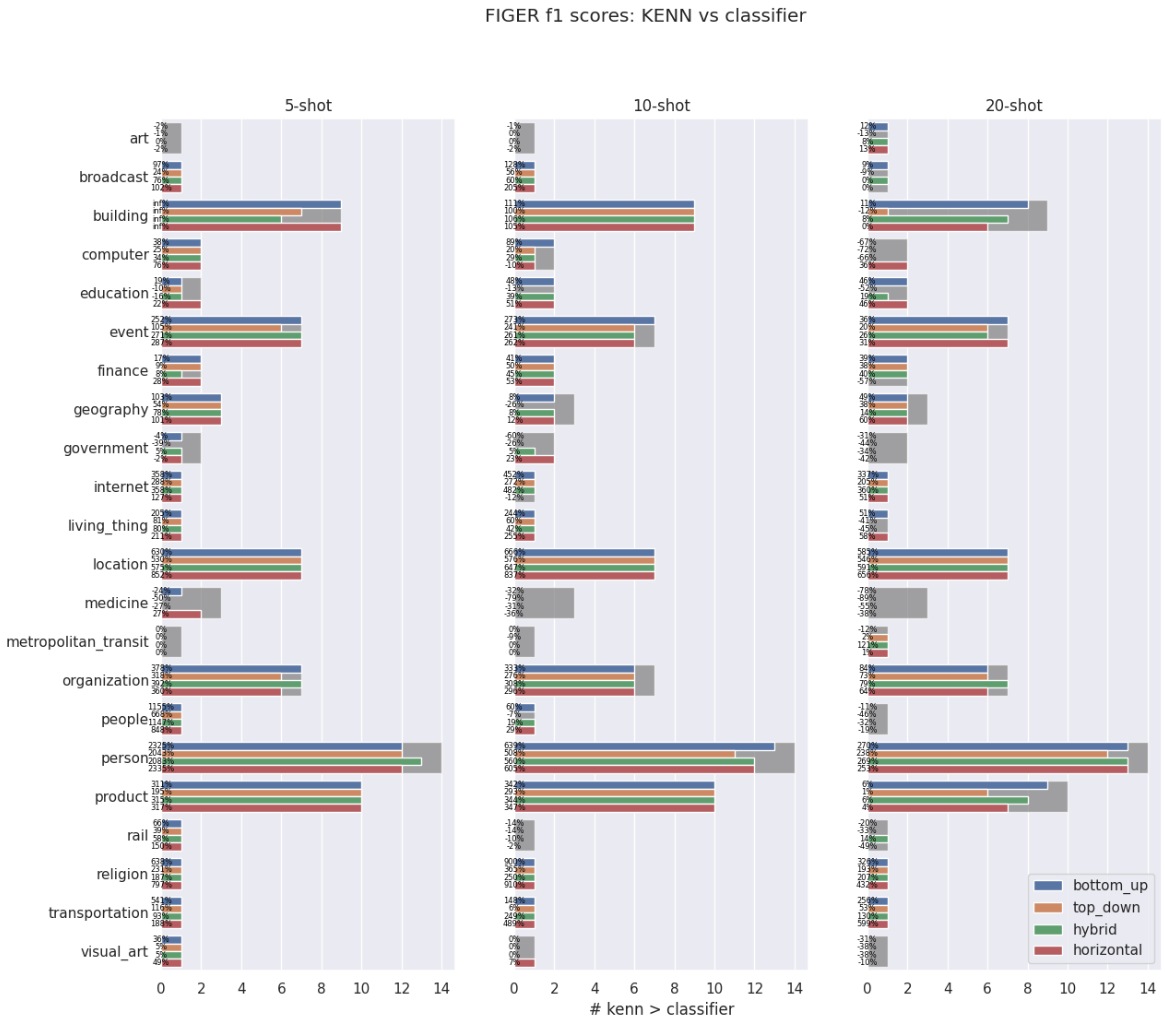


Figure 5.10: Stacked barplot showing the count of subtypes in each FIGER family for which a `KENN`-based technique (colored) perform better than the `Classifier`-based technique. The grey box represent the total number of subtypes for a given family. In FIGER, `KENN`-based techniques perform better than `Classifier`-based technique on the majority of types in each family.

Specific Results

In this subsection, the general evidences described in the previous subsection are analyzed by datasets.

Specialization in BBN. Figures 5.5, 5.8, 5.11, 5.9, and 5.13 describe the specialization experiment on BBN. In particular:

- from the barplots (Figure 5.5) it can be seen that for all families, `KENN`-based specialization approaches outperform `Classifier` and `Box` based specialization approaches in 5-shot experiments; in 10-shot and 20-shot experiments the best specialization approach is always based on `KENN`, except for the family of `BBN.SUBSTANCE` in 20-shot, where the best specialization approach is the `Classifier`-based. Lastly, `Top-down` KB is often the worst `KENN`-based specialization approach.
- from the histograms (Figures 5.8, and 5.11) it can be seen how the F1-score on single types obtained with `KENN`-based specialization approaches tends to be positive in the majority of the cases, and it is less evident with more data (distributions for 20-shot are more concentrated around the 0). This phenomenon is less visible in BBN than in other datasets.
- from the stacked barplots (Figure 5.9 and 5.13) it can be seen how `KENN`-based specialization techniques outperform `Box`-based specialization technique on almost all types involved in the specialization experiments.

Specialization in FIGER. Figures 5.6, 5.8, 5.11, 5.10, and 5.14 describe the specialization experiment on FIGER. In particular:

- from the barplots (Figure 5.6) it can be seen that in the majority of the cases `KENN`-based specialization approaches favor the specialization, in particular `Bottom up` or `Horizontal` are often the best KBs to inject. Some families shows to be more difficult, in particular `FIGER.art`, `FIGER.metropolitan_transit`, and `FIGER.visual_art`. A particular case is given by `FIGER.medicine`, where the `classifier` and the `box`-based specialization approaches shown to be better than `KENN`-based ones independently of the values of k; this is the only family on which this happened in the entire specialization experiment. Lastly, `Top-down` KB is often the worst `KENN`-based specialization approaches.
- from the histograms (Figures 5.8, and 5.11 it can be seen how the F1-score on single types obtained with `KENN`-based specialization approaches tends to be positive in the majority of the cases, and it is less evident with more

data (distributions for 20-shot are more concentrated around the 0). This phenomenon is more visible in FIGER than in other datasets and it is in accord with results in Table 5.2, where FIGER was the dataset with the highest difference between `KENN`-based and non-`KENN`-based specialization approaches.

- from the stacked barplots (Figure 5.10 and 5.14) it can be seen how `KENN`-based specialization techniques outperform `Classifier` and `Box`-based specialization approaches on almost all types involved in the specialization experiments. Again, it can be seen how in `FIGER.medicine` and `FIGER.metropolitan_transit` `KENN`-based specialization techniques were not effective.

Specialization in OntoNotes. Figures 5.7, 5.8, 5.12, 5.9, and 5.13 describe the specialization experiment on OntoNotes. In particular:

- from the barplots (Figure 5.7) it can be seen that in the majority of the cases `KENN`-based specialization approaches favor the specialization.
- from the histograms (Figures 5.8, and 5.12 it can be seen how the F1-score on single types obtained with `KENN`-based specialization approaches tends to be positive in the majority of the cases, and it is less evident with more data (distributions for 20-shot are more concentrated around the 0).
- from the stacked barplots (Figure 5.9 and 5.13) it can be seen that in many cases only a portion of the subtypes in a family benefits of the usage of `KENN`-based specialization techniques. However when a family benefits from the application of KENN, its improvement is very high, as can be seen in Table 5.2 and in barplots.

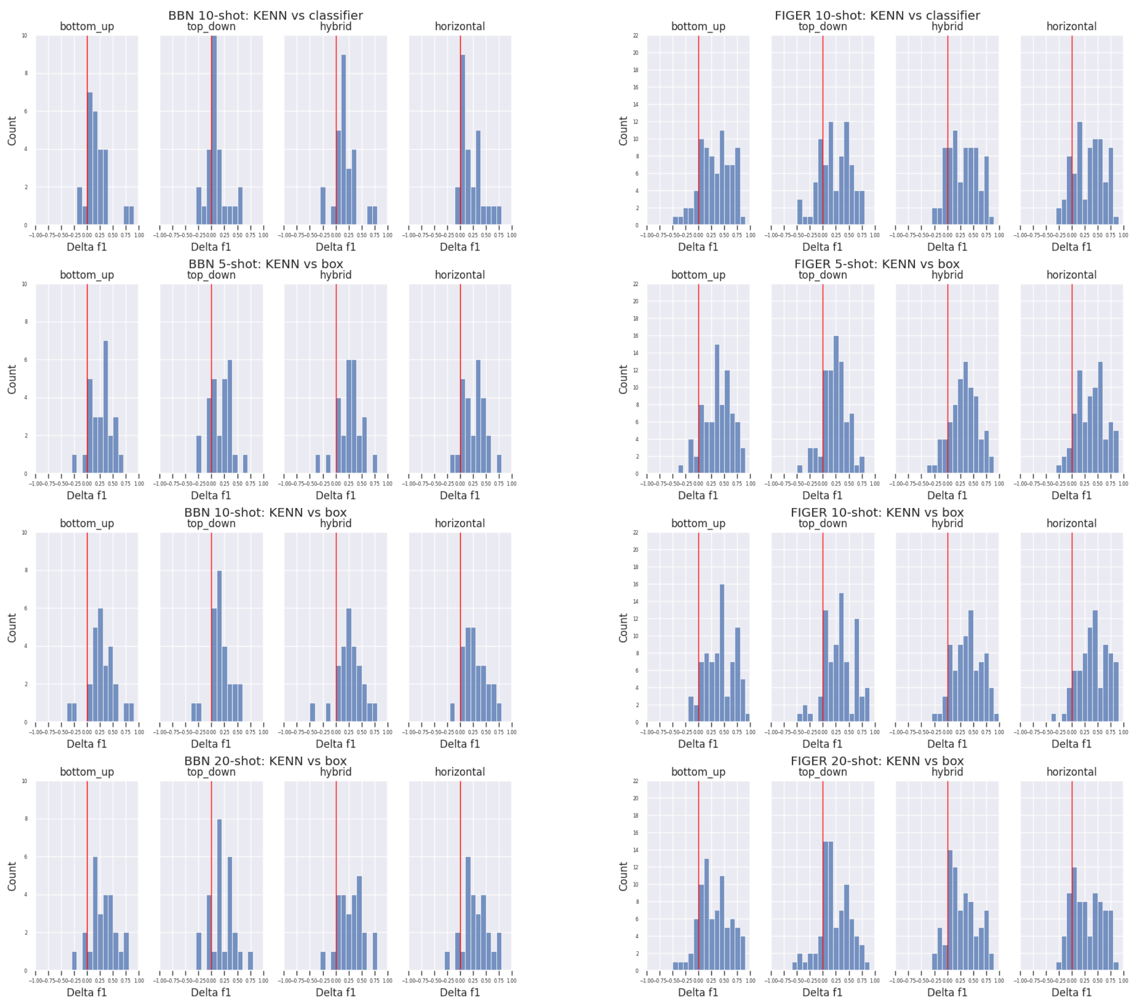


Figure 5.11: Histogram showing the distribution of the differences between F1 of `KENN`-based techniques and `Classifier`-based technique (first row) or `Box`-based technique (rows 2-4) for each type involved in the specialization averaged by sampling seed used to create k-shot dataset. $k = 10$ for comparison with `Classifier`, $k \in \{5, 10, 20\}$ for comparison with `Box`. BBN (left), and FIGER (right) are reported. Difference is mostly positive, showing that `KENN`-based techniques perform better than `Classifier` and `Box`-based technique in most of the types. The less the data used, the more the increment caused by `KENN`-based techniques.

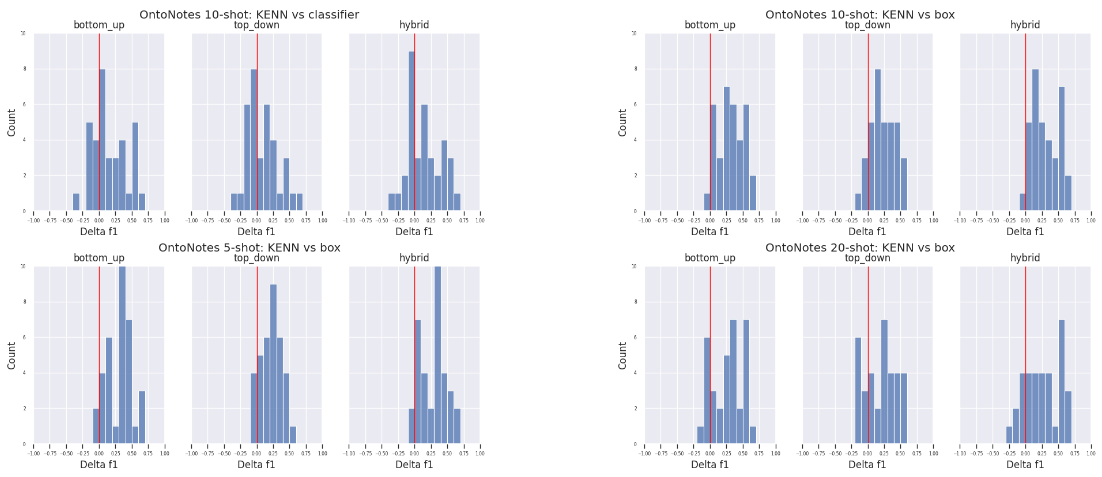


Figure 5.12: Histogram showing the distribution of the differences in OntoNotes between F1 of `KENN`-based techniques and `Classifier`-based technique (top left) or `Box`-based technique (top right, bottom) for each type involved in the specialization averaged by sampling seed used to create k-shot dataset. $k = 10$ for comparison with `Classifier`, $k \in \{5, 10, 20\}$ for comparison with `Box`. Difference is mostly positive, showing that `KENN`-based techniques perform better than `Classifier` and `Box`-based technique in most of the types. The less the data used, the more the increment caused by `KENN`-based techniques.

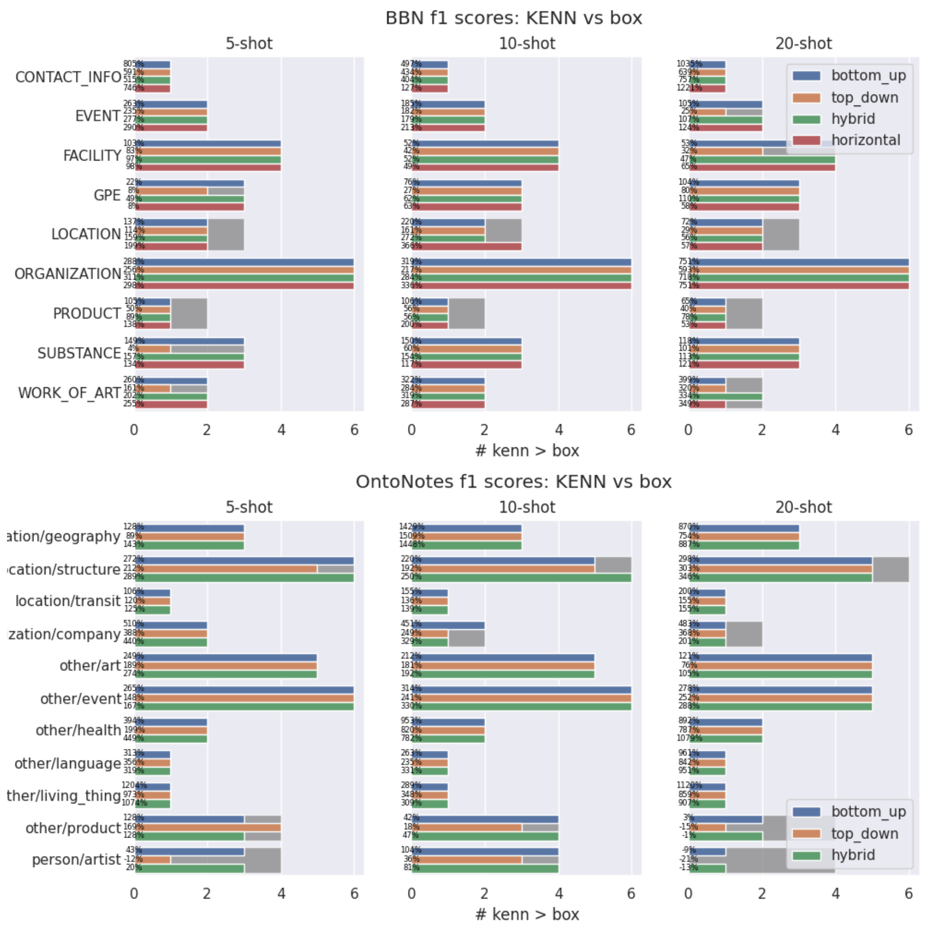


Figure 5.13: Stacked barplot showing the count of children in each family for which a `KENN`-based technique (colored) perform better than the `Box`-based technique. The grey box represent the total number of children for a given family. BBN is showed above, OntoNotes is showed below. Both in BBN and OntoNotes, `KENN`-based techniques perform better than `Box`-based technique on the majority of types in each family.

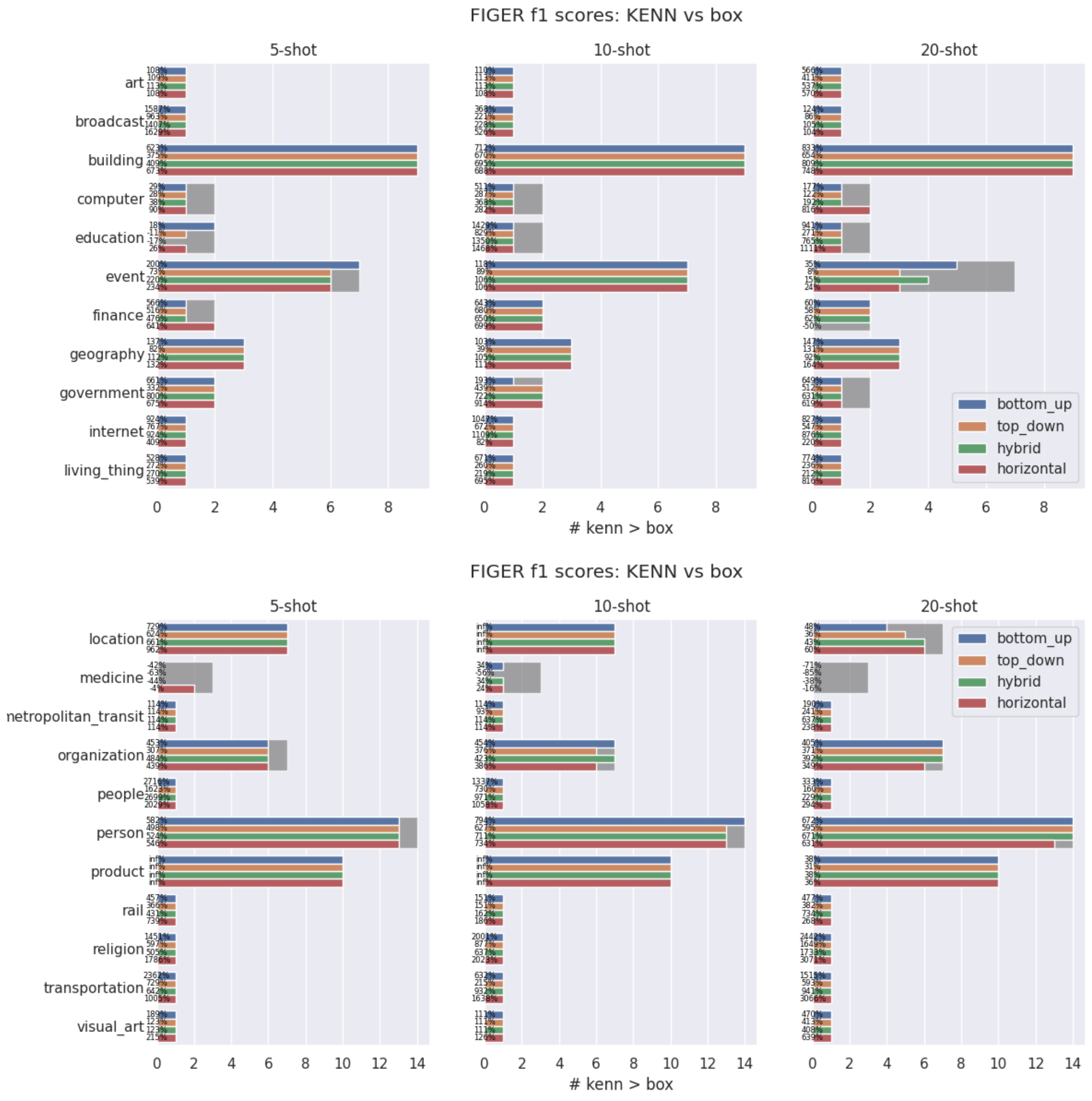


Figure 5.14: Stacked barplot showing the count of children in each FIGER family for which a `KENN`-based technique (colored) perform better than the `Box`-based technique. The grey box represent the total number of children for a given family. `KENN`-based techniques perform better than `Box`-based technique on the majority of types in each family.

5.5 Conclusions on the Specialization of FET Models

The specialization of a FET model is often needed in specific domains, where there is a special interest in recognizing subtle differences between types (for example recognizing different jobs or specific kinds of products). This specialization scenario is particularly tricky when low quantity of annotated data is available, but this it is also very common situation, since annotations on specific types require more sophisticated annotation policies and often more qualified annotators. Different model specialization techniques were proposed and experimented. Answering to research questions, experiments show that injecting the hierarchy only to smartly initialize the classification layer of the new types is not as effective as injecting the hierarchy during training with a neuro-symbolic approach. KENN shown to be useful in increasing the performance of model on all datasets and on the majority of the families, showing that exploiting hierarchical type dependencies is crucial for model specialization.

Chapter 6

NeuroSymbolic Fine-grained Entity Typing for Full-fledged Domain Adaptation

In this Chapter, techniques for Full-fledged adaptation of a FET model are proposed and compared. Full-fledged adaptation of a FET model is a use case presented in the user stories in Section 3.2.1 where a trained FET model has to be used in a new domain, with some equivalent types, some specializations and some new types. Like in the specialization experiments, the key intuition is to exploit model capabilities on source domain Δ^S to favor the training in the target domain Δ^T. The expected behaviors based on relations between types in source and target domain are the following:

- given two equivalent types $s \in S$ and $t \in T$ such that $s \equiv t$ the prediction for t matches the prediction for s
- the prediction on $\tau \prec t$, subtype of $t \equiv s$, is conditioned by the prediction on s. The higher the prediction on s, the higher the prediction on τ, the lower the prediction on s, the lower the prediction on τ,

6.1 Research Questions - Full-fledged Adaptation of a FET model

Question 1: How does relations between source types and target types can be used to favor the adaptation of a FET model to a low-resource target scenario?

Question 2: How does hierarchical relations between target types can be used to favor the adaptation of a FET model to a low-resource target scenario?

Question 3: How does the quantity and the variety of types in the source domain impact the adaptation of a FET model to a low-resource target scenario?

6.2 Adapt a FET model

The adaptation of a FET model in a target domain is conditioned by the initialization of the neurons used for the target types and by the techniques adopted to propagate the information from the source to the target domain during training.

6.2.1 Smart initialization

Like the procedure adopted in the specialization scenario, a smart initialization can be applied to some types. In particular, given an equivalence relation $s \equiv t$, the neuron of t is initialized with parameters from neuron of s. This choice is more conservative than the one adopted in the specialization scenario, where $\prec$ relations were exploited. This difference is motivated by experimental results, a more conservative choice is better in full-fledged domain adaptation scenario. If linear layers are present between the encoder and the classification layer, these linear layers are frozen and used also for the additional classification layer. In this way, the two classifiers will classify from the same embedding space.

6.2.2 KENN for full-fledged domain adaptation

If smart initialization is used, during the first steps of training in the target domain, the expected behavior of KENN implementing `Bottom up`, `Top Down`, or `Hybrid` KB is to act in the same way as shown in the specialization scenario, that is, modifying the prediction by injecting the hierarchy. Since only equivalence relations are used in the smart initialization, the effect of hierarchy injection is, in the first steps, to propagate the knowledge from father types in source domain to their specialization in target domain (due to the initialization); during the training the hierarchy injection maintains linked the predictions of hierarchy-related types, helping their joint prediction.

`Cross` KB instead is used to persist the equivalence from the source domain to the target domain *during training*, equivalence relations are used to link source types to target types, formally $s \equiv t$.

6.3 Experimental Setup

The same FET benchmark datasets (Section 4.3.1) used in the previous experiments are paired and used as source-target pairs, obtaining six pairs. For each source-target pair, few-shot experiments are carried on, to mimic the necessity of a trained model in a low-resource target domain.

Few-shot dataset creation: Few-shot datasets are built from the original training datasets by selecting k examples for each type at `max depth`, $k \in [5, 10, 20]$. For each selected example its annotation is filtered by keeping only one `max depth` type and its ancestors. In this way, exactly k examples for each `max depth` type are selected, for each non-`max depth` type exactly $k \times \#descendant$ are selected. The validation set is composed in the same way, the test set instead is the original test set.

Training on source domain: an instance of the FET model explained in Section 4.3.4 is trained on each dataset. These three models are used as starting point to perform the adaptation process.

6.3.1 Experimented domain adaptation approaches

In this section, the trained nine adaptation techniques are proposed to adapt a trained FET model to a target domain.

Additional classifier: an additional classifier initialized with smart initialization (explained in Section 6.2), is trained on the k-shot target datasets. This setup useful to understand the benefits of the source domain both in terms of trained encoder and smart initialization.

Literature baseline: L2AWE [Noz+21] has shown to be useful in NER domain adaptation with a limited quantity of types (a dozen circa), this approach was re-implemented using the FET model trained on source domain. Its encoding based on BERT and Adapters is paired with predictions on source domain, this concatenated input is used to fed a linear classifier trained on the k-shot target datasets.

KENN in target domain: starting from the additional classifier, `Bottom up`, `Top Down`, or `Hybrid` KB are injected using KENN, as explained in Section 6.2. These models are useful to understand the effect of using KENN with a smart initialized classifier.

Cross KENN: starting from the additional classifier, `Cross` KB is injected using KENN, as explained in Section 6.2. This model is useful to understand the effect of the propagation of predictions from source domain to target domain.

Cross+ KENN: the KBs of the previous setups (*KENN in target domain* and *Cross KENN*) are merged and used while training on the target domain. This model is useful to understand the effect of combining both KB.

The proposed domain adaptation approaches start from the same FET model trained on a source domain. During the training on the target domain the encoder is kept frozen to (i) avoid forgetting of the source domain knowledge, (ii) understand if the encoder is able to produce encodings that reflect the differences between entity mentions of types that may be unseen (i.e., types not present in the source domain), and (iii) maintaining the availability of predictions on source domain types, necessary for L2AWE, Cross-domain KENN and Cross-vertical KENN. Moreover, a baseline approach is proposed: a FET model (Section 4.3.4) directly trained on the k-shot dataset. This baseline is useful to understand the performance obtainable by the model using only the limited target data.

Summary: for each source-target pair, one in-domain model is trained (case (i) of *Baseline models*) and nine cross-domain models (case (i) of *Baseline models*, L2AWE, three *KENN in target domain*, one *cross-domain KENN*, and three *Cross-domain KENN*) are trained. Each model is trained using three different k-shot sampling, obtaining a total of $(1 + 9) * 3 = 30$ trained models. Since source-target pairs are 6, the total trained models are $[(9 * 6) * (1 * 3)] * 3 * 3 = 513$, i.e. nine cross domain models for each pair $(9 * 6)$, one in-domain model for each target domain $(1 * 3)$, three k values $(*3)$, and three k-shot sampling seeds $(*3)$.

6.4 Results

Tables 6.1, 6.2,6.3, 6.4, 6.5, 6.6, 6.7 6.8, and 6.9 show the results of k-shot full-fledged domain adaptation experimentation. Each table resumes the performance of 18 domain adaptation techniques applied to adapt a FET model to a target domain using k examples for each target type as training dataset and a different set of k examples for each target type as validation dataset.

6.4.1 Results with BBN as target domain

Tables 6.1, 6.2,6.3 resume the performance of model trained respectively on the 5-shot, 10-shot, and 20-shot datasets built from the training set of BBN and tested on the original test set of BBN.

Source Domain	Adaptation Technique	Macro averaged on Examples			Micro averaged			Macro averaged on Types		
		P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
BBN	-	.295	.162	.209	.296	.179	.223	.143	.088	.109
FIGER	Additional classifier	.698	.579	.633	.687	.58	.629	.513	.547	.53
FIGER	L2AWE	.567	.379	.454	.593	.415	.488	.474	.397	.432
FIGER	KENN bottom up	.699	.593	.641	.689	.596	.639	**.538**	.551	.544
FIGER	KENN top down	.704	.611	.654	.707	.612	.656	.487	.601	.538
FIGER	KENN hybrid	.728	.64	.681	**.722**	.637	.676	.519	.598	.556
FIGER	KENN cross	.716	.649	.681	.689	.647	.667	.526	.6	.56
FIGER	KENN cross bottom up	.716	.649	.681	.687	.648	.667	**.538**	.598	.567
FIGER	KENN cross top down	.714	.654	.683	.695	.651	.673	.493	.623	.55
FIGER	KENN cross hybrid	**.734**	**.672**	**.702**	.71	**.668**	**.688**	.518	**.628**	**.568**
OntoNotes	Additional classifier	.684	.545	.607	.72	.571	.637	.496	.51	.503
OntoNotes	L2AWE	.615	.44	.513	.658	.484	.558	.505	.453	.478
OntoNotes	KENN bottom up	.685	.553	.612	.72	.576	.64	**.526**	.506	.515
OntoNotes	KENN top down	.677	.583	.626	.719	.607	.658	.477	.562	.516
OntoNotes	KENN hybrid	.699	.604	.648	.733	.618	.67	.501	.54	.52
OntoNotes	KENN cross	.69	.595	.639	.72	.619	.665	.497	.537	.517
OntoNotes	KENN cross bottom up	.69	.594	.638	.72	.615	.663	.519	.533	**.526**
OntoNotes	KENN cross top down	.688	.606	.645	.722	.626	.671	.473	**.568**	.516
OntoNotes	KENN cross hybrid	**.707**	**.623**	**.662**	.733	**.636**	**.681**	.499	.554	.525

Table 6.1: Results of Full-fledged Domain Adaptation with BBN $5 - shot$ as target domain. First row reports the in-domain FET model trained directly on the 5-shot dataset. Rows 2-10 reports different adaptation techniques starting from a model trained on FIGER. Rows 11-19 reports different adaptation techniques starting from a model trained on OntoNotes. **Bold** highlights best adaptation model for a given source domain, **underlined bold** highlights the best performance overall.

Source	Adaptation	Macro averaged on Examples			Micro averaged			Macro averaged on Types		
Domain	Technique	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
BBN	-	.445	.267	.334	.452	.291	.354	.303	.249	.273
FIGER	Additional classifier	.701	.58	.635	.706	.586	.641	.526	.583	.553
FIGER	L2AWE	.577	.429	.491	.615	.463	.528	.481	.473	.477
FIGER	KENN bottom up	.692	.582	.632	.7	.59	.64	**.564**	.58	.572
FIGER	KENN top down	.709	.616	.659	.719	.617	.664	.491	.628	.551
FIGER	KENN hybrid	.704	.615	.656	.713	.614	.66	.513	.621	.562
FIGER	KENN cross	.722	.648	.683	.711	.649	.678	.527	.617	.569
FIGER	KENN cross bottom up	.719	.646	.68	.707	.646	.675	.549	.615	**.58**
FIGER	KENN cross top down	.725	.653	.687	.717	.65	.682	.484	**.648**	.554
FIGER	KENN cross hybrid	**.733**	**.661**	**.695**	**.722**	**.658**	**.688**	.519	.646	.576
OntoNotes	Additional classifier	.689	.576	.627	.73	.603	.66	.517	.549	.532
OntoNotes	L2AWE	.634	.508	.563	.684	.551	.61	.5	.514	.507
OntoNotes	KENN bottom up	.685	.575	.625	.724	.598	.655	**.535**	.54	.537
OntoNotes	KENN top down	.68	.603	.639	.723	.626	.671	.47	.592	.524
OntoNotes	KENN hybrid	.702	.617	.656	.736	.632	.68	.482	.574	.524
OntoNotes	KENN cross	.697	.619	.655	.732	.644	.685	.508	.576	.54
OntoNotes	KENN cross bottom up	.697	.616	.654	.732	.637	.681	.526	.563	**.544**
OntoNotes	KENN cross top down	.696	.628	.66	.729	.647	.685	.468	**.602**	.527
OntoNotes	KENN cross hybrid	**.713**	**.635**	**.672**	**.741**	**.648**	**.691**	.482	.587	.529

Table 6.2: Results of Full-fledged Domain Adaptation with BBN $10-shot$ as target domain. First row reports the in-domain FET model trained directly on the 10-shot dataset. Rows 2-10 reports different adaptation techniques starting from a model trained on FIGER. Rows 11-19 reports different adaptation techniques starting from a model trained on OntoNotes. **Bold** highlights best adaptation model for a given source domain, **underlined bold** highlights the best performance overall.

Source	Adaptation	Macro averaged on Examples			Micro averaged			Macro averaged on Types		
Domain	Technique	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
BBN	-	.523	.372	.431	.552	.4	.462	.26	.269	.264
FIGER	Additional classifier	.728	.632	.677	.744	.641	.688	.539	.622	.577
FIGER	L2AWE	.614	.503	.552	.661	.542	.595	.478	.535	.505
FIGER	KENN bottom up	.726	.637	.678	.743	.644	.69	**.566**	.623	.593
FIGER	KENN top down	.734	.663	.697	.748	.667	.705	.49	.663	.563
FIGER	KENN hybrid	.743	.672	.706	.756	.673	.712	.518	.663	.582
FIGER	KENN cross	.741	.689	.714	.743	.695	.718	.531	.649	.584
FIGER	KENN cross bottom up	.737	.679	.707	.741	.683	.711	.551	.646	**.595**
FIGER	KENN cross top down	.745	.695	.719	.748	.698	.722	.492	**.68**	.571
FIGER	KENN cross hybrid	**.755**	**.702**	**.727**	**.757**	**.702**	**.728**	.526	**.68**	.593
OntoNotes	Additional classifier	.722	.638	.677	.752	.649	.697	.503	.584	.54
OntoNotes	L2AWE	.689	.614	.649	.729	.644	.684	.493	.567	.527
OntoNotes	KENN bottom up	.718	.635	.674	.751	.647	.695	**.522**	.573	.546
OntoNotes	KENN top down	.724	.672	.696	.752	.679	.714	.475	**.619**	.537
OntoNotes	KENN hybrid	.74	.683	.71	**.764**	.687	**.723**	.483	.605	.537
OntoNotes	KENN cross	.728	.679	.703	.749	.69	.718	.49	.608	.543
OntoNotes	KENN cross bottom up	.726	.671	.697	.75	.681	.714	.51	.594	**.549**
OntoNotes	KENN cross top down	.732	.685	.708	.753	.689	.719	.469	.618	.534
OntoNotes	KENN cross hybrid	**.742**	**.695**	**.718**	.763	**.699**	.73	.479	.612	.537

Table 6.3: Results of Full-fledged Domain Adaptation with BBN $20 - shot$ as target domain. First row reports the in-domain FET model trained directly on the 20-shot dataset. Rows 2-10 reports different adaptation techniques starting from a model trained on FIGER. Rows 11-19 reports different adaptation techniques starting from a model trained on OntoNotes. **Bold** highlights best adaptation model for a given source domain, **underlined bold** highlights the best performance overall.

The comparison of the in-domain model (row 1) with all cross-domain models (rows 2 to 19) shows that adapting a FET model trained in different source domain (FIGER or OntoNotes) to this domain brings a big improvement in all performance, when only k examples are available to learn each type. Across all k values, the best adaptation technique is using KENN to jointly inject `Cross` and `Hybrid` KBs (rows 10 and 19), injecting only the `Hybrid` KB (row 15) also seem a good choice. The `additional classifier` adaptation technique (rows 2 and 11) show that the trained encoder is able to produce encodings that let to distinguish between the target types. Furthermore, the addition of KENN in addition to the additional classifier brings an improvement that ranges from .01 to .06, depending on the injected KB. Lastly, the performance obtained by `L2AWE` (rows 3 and 12), compared to the performance obtained by all KENN-based models, shows that the usage of all predictions on the types of the source domain does not favor adaptation in the same way as exploiting explicit relationships between types.

6.4.2 Results with FIGER as target domain

Tables 6.4, 6.5,6.6 resume the performance of the model trained, respectively, on the 5-shot, 10-shot, and 20-shot datasets built from the FIGER training set and tested on the original FIGER test set.

The comparison of the in-domain model (row 1) with domain adaptation techniques to adapt a model trained on BBN (rows 2 to 9) shows that in this domain the training of the encoder (available only for the in-domain model) may be more important than exploiting predictions of a model trained on a source domain. In fact, the in-domain model often has better performance than all domain adaptation techniques applied to a model trained on BBN. This may be related to the quantity of unseen types present in FIGER, that is the dataset with more types (127). With a high number of unseen types, an encoder trained on BBN, that has only 47 types, may ignore the differences between their entities or collapse the representation of specialized types, since these differences were not useful in the source domain (for example, BBN only has the type `BBN.person`, FIGER instead has 15 subtypes of `FIGER.person`. The encoder trained on OntoNotes instead (row 10) seems sensible enough to match the performance obtained by the in-domain model. The best adaptation model technique when FIGER is the target domain depends on the metric and on the value of k, in general the adaptation techniques based on `Cross` KB brings more benefits than others, L2AWE instead is often the best adaptation model in terms of *Precision*.

Source Domain	Adaptation Technique	Macro averaged on Examples			Micro averaged			Macro averaged on Types		
		P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
FIGER	-	.688	.588	.634	.683	.512	.585	.434	.406	.419
BBN	Additional classifier	.644	.537	.586	.631	.468	.537	.455	.361	.403
BBN	L2AWE	**.681**	**.56**	**.615**	**.672**	.485	**.563**	.414	.382	.397
BBN	KENN bottom up	.655	.544	.594	.641	.473	.545	.403	.343	.37
BBN	KENN top down	.593	.497	.541	.58	.437	.498	.482	.393	.433
BBN	KENN hybrid	.623	.525	.569	.606	.456	.52	.433	.362	.394
BBN	KENN cross	.644	.557	.597	.62	**.49**	.548	.476	.388	.427
BBN	KENN cross bottom up	.652	.552	.598	.628	.485	.547	.449	.369	.405
BBN	KENN cross top down	.606	.524	.562	.588	.464	.519	**.483**	**.402**	**.439**
BBN	KENN cross hybrid	.646	.556	.598	.622	.488	.547	.47	.382	.422
OntoNotes	Additional classifier	.699	.597	.644	.69	.516	.591	.45	.39	.418
OntoNotes	L2AWE	**.731**	.603	.661	**.717**	.513	.598	.38	.364	.372
OntoNotes	KENN bottom up	.676	.582	.625	.666	.5	.571	.424	.391	.407
OntoNotes	KENN top down	.661	.587	.622	.647	.512	.571	.481	.428	.453
OntoNotes	KENN hybrid	.671	.596	.631	.653	.518	.578	.443	.423	.432
OntoNotes	KENN cross	.714	**.626**	**.667**	.698	**.554**	**.618**	.503	.419	.457
OntoNotes	KENN cross bottom up	.71	.618	.661	.694	.543	.609	.499	.417	.454
OntoNotes	KENN cross top down	.687	.615	.649	.664	.544	.598	**.506**	**.439**	**.47**
OntoNotes	KENN cross hybrid	.692	.613	.65	.67	.538	.597	.497	.426	.459

Table 6.4: Results of Full-fledged Domain Adaptation with FIGER 5-shot as target domain. First row reports the in-domain FET model trained directly on the 5-shot dataset. Rows 2-10 reports different adaptation techniques starting from a model trained on BBN. Rows 11-19 reports different adaptation techniques starting from a model trained on OntoNotes. **Bold** highlights best adaptation model for a given source domain, **underlined bold** highlights the best performance overall.

Source Domain	Adaptation Technique	Macro averaged on Examples			Micro averaged			Macro averaged on Types		
		P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
FIGER	-	.694	.622	.656	.677	.554	.609	**.543**	**.506**	**.524**
BBN	Additional classifier	.647	.546	.592	.637	.481	.548	.499	.424	.458
BBN	L2AWE	**.675**	**.572**	**.619**	**.665**	.501	**.571**	.444	.418	.431
BBN	KENN bottom up	.648	.551	.596	.637	.485	.55	.451	.401	.425
BBN	KENN top down	.594	.52	.555	.59	.465	.52	.477	.446	.461
BBN	KENN hybrid	.62	.546	.581	.606	.483	.537	.462	.43	.446
BBN	KENN cross	.627	.565	.595	.608	**.508**	.554	**.503**	.455	**.478**
BBN	KENN cross bottom up	.641	.568	.602	.618	.506	.556	.48	.435	.456
BBN	KENN cross top down	.591	.543	.566	.578	.49	.53	.484	**.46**	.472
BBN	KENN cross hybrid	.631	.565	.596	.609	.502	.55	.496	.447	.47
OntoNotes	Additional classifier	.702	.608	.652	.692	.536	.604	.465	.44	.452
OntoNotes	L2AWE	**.724**	.614	.664	**.711**	.54	.614	.458	.446	.452
OntoNotes	KENN bottom up	.714	.618	.663	.699	.546	.613	.458	.442	.45
OntoNotes	KENN top down	.628	.575	.6	.617	.512	.56	.496	.461	.478
OntoNotes	KENN hybrid	.672	.606	.637	.655	.538	.591	.471	.469	.47
OntoNotes	KENN cross	.699	.624	.659	.682	.559	.614	.491	.468	.479
OntoNotes	KENN cross bottom up	.716	**.633**	**.672**	.698	**.565**	**.625**	**.528**	.457	**.49**
OntoNotes	KENN cross top down	.652	.604	.627	.632	.539	.582	.505	**.476**	**.49**
OntoNotes	KENN cross hybrid	.687	.625	.654	.664	.559	.607	.504	.472	.487

Table 6.5: Results of Full-fledged Domain Adaptation with FIGER 10-shot as target domain. First row reports the in-domain FET model trained directly on the 10-shot dataset. Rows 2-10 reports different adaptation techniques starting from a model trained on BBN. Rows 11-19 reports different adaptation techniques starting from a model trained on OntoNotes. **Bold** highlights best adaptation model for a given source domain, **underlined bold** highlights the best performance overall.

Source Domain	Adaptation Technique	Macro averaged on Examples			Micro averaged			Macro averaged on Types		
		P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
FIGER	-	**.684**	**.637**	**.659**	.653	**.565**	.606	.517	**.557**	**.536**
BBN	Additional classifier	.636	.556	.593	.623	.5	.555	**.523**	.462	.49
BBN	L2AWE	**.662**	.575	**.615**	**.652**	.513	**.574**	.481	.462	.471
BBN	KENN bottom up	.639	.562	.598	.623	.505	.558	.489	.445	.466
BBN	KENN top down	.583	.535	.558	.576	.487	.528	.513	**.487**	**.5**
BBN	KENN hybrid	.6	.55	.574	.587	.5	.54	.509	.479	.493
BBN	KENN cross	.617	.576	.596	.593	.523	.556	.517	.481	.498
BBN	KENN cross bottom up	.629	**.579**	.603	.606	**.525**	.562	**.523**	.467	.493
BBN	KENN cross top down	.581	.548	.564	.566	.501	.532	.501	.485	.493
BBN	KENN cross hybrid	.614	.577	.595	.589	**.525**	.555	.508	.486	.497
OntoNotes	Additional classifier	.663	.591	.625	.655	.531	.587	.479	.488	.484
OntoNotes	L2AWE	.677	.604	.638	**.669**	.539	.597	.493	.487	.49
OntoNotes	KENN bottom up	.68	.609	.642	.668	.544	.599	.527	.482	.504
OntoNotes	KENN top down	.601	.564	.582	.598	.511	.551	.509	.505	.507
OntoNotes	KENN hybrid	.649	.601	.624	.636	.54	.584	.507	.505	.506
OntoNotes	KENN cross	.667	.618	.642	.653	.562	.604	**.539**	.514	**.526**
OntoNotes	KENN cross bottom up	**.684**	**.626**	**.654**	.666	**.564**	**.611**	.53	.501	.515
OntoNotes	KENN cross top down	.611	.587	.599	.602	.538	.568	.531	**.518**	.524
OntoNotes	KENN cross hybrid	.654	.612	.632	.637	.553	.592	**.539**	.513	**.526**

Table 6.6: Results of Full-fledged Domain Adaptation with FIGER 20-shot as target domain. First row reports the in-domain FET model trained directly on the 20-shot dataset. Rows 2-10 reports different adaptation techniques starting from a model trained on BBN. Rows 11-19 reports different adaptation techniques starting from a model trained on OntoNotes. **Bold** highlights best adaptation model for a given source domain, **underlined bold** highlights the best performance overall.

Source	Adaptation	Macro averaged on Examples			Micro averaged			Macro averaged on Types		
Domain	Technique	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
OntoNotes	-	.626	.521	.569	.619	.43	.508	.067	.043	.053
BBN	Additional classifier	.689	.622	.654	.666	.545	.599	.338	.217	.264
BBN	L2AWE	**.708**	.616	**.659**	**.688**	.53	.599	.334	.163	.219
BBN	KENN bottom up	.68	.633	.656	.653	.556	.601	.343	.219	.268
BBN	KENN top down	.675	.613	.642	.653	.54	.591	.346	.252	.292
BBN	KENN hybrid	.673	.624	.647	.649	.55	.596	.358	.241	.288
BBN	KENN cross	.671	.634	.652	.639	.563	.598	.352	.255	.295
BBN	KENN cross bottom up	.675	**.643**	.658	.64	**.57**	**.603**	**.363**	.248	.295
BBN	KENN cross top down	.652	.607	.629	.627	.542	.581	.344	**.272**	**.303**
BBN	KENN cross hybrid	.668	.629	.648	.638	.559	.596	.352	.265	.302
FIGER	Additional classifier	.733	.688	.71	.697	.611	.651	.442	.357	.395
FIGER	L2AWE	.711	.638	.672	.679	.545	.605	.38	.196	.259
FIGER	KENN bottom up	.734	.707	.721	.691	.625	.656	.459	.328	.383
FIGER	KENN top down	.718	.673	.695	.685	.6	.64	.423	.394	.408
FIGER	KENN hybrid	**.739**	.694	.716	**.702**	.613	.655	.45	.362	.401
FIGER	KENN cross	.721	.712	.716	.68	.645	.662	.466	.407	.434
FIGER	KENN cross bottom up	.729	**.724**	**.727**	.683	**.65**	**.666**	**.467**	.375	.416
FIGER	KENN cross top down	.716	.689	.702	.678	.624	.65	.446	**.426**	**.436**
FIGER	KENN cross hybrid	.732	.708	.72	.69	.636	.662	.459	.402	.428

Table 6.7: Results of Full-fledged Domain Adaptation with OntoNotes 5-shot as target domain. First row reports the in-domain FET model trained directly on the 5-shot dataset. Rows 2-10 reports different adaptation techniques starting from a model trained on BBN. Rows 11-19 reports different adaptation techniques starting from a model trained on FIGER. **Bold** highlights best adaptation model for a given source domain, **underlined bold** highlights the best performance overall.

6.4.3 Results with OntoNotes as target domain

Tables 6.7, 6.8,6.9 resume the performance of the model trained, respectively, on the 5-shot, 10-shot, and 20-shot datasets built from the OntoNotes training set and tested on the original OntoNotes test set.

This dataset is characterized by the predominance of the type `Ontonotes.other` in the test set (and generally also in training set, but with k-shot sampling this problem is avoided). For this reason, analysis of these experiment focus on Macro metrics averaged over types (last three columns of the table) to have an unbiased vision of models behavior.

Results on the 5-shot experiment (Table 6.7) clearly show how the combination of a pretrained encoder and predictions on source domain types are key feature reach good performance on multiple types in this domain. A big difference is also underlined by performance of models adapted from BBN and the models from FIGER: models adapted from FIGER have always better performance than models adapted from BBN, this may indicates that the encoder trained on FIGER is more

Source Domain	Adaptation Technique	Macro averaged on Examples			Micro averaged			Macro averaged on Types		
		P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
OntoNotes	-	.633	.592	.612	.597	.506	.548	.249	.173	.204
BBN	Additional classifier	.678	.618	.647	.649	.55	.596	**.376**	.27	.314
BBN	L2AWE	**.695**	.617	**.654**	**.668**	.543	**.599**	.373	.234	.288
BBN	KENN bottom up	.669	**.631**	.649	.635	.561	.596	.374	.274	.317
BBN	KENN top down	.654	.604	.628	.628	.542	.582	.326	.307	.316
BBN	KENN hybrid	.664	.619	.64	.632	.556	.591	.361	.294	.324
BBN	KENN cross	.654	.625	.639	.616	.563	.588	.353	.3	.324
BBN	KENN cross bottom up	.658	.629	.643	.621	**.568**	.593	.354	.292	.32
BBN	KENN cross top down	.634	.604	.619	.603	.548	.574	.331	**.321**	.326
BBN	KENN cross hybrid	.647	.619	.633	.612	.558	.584	.355	.305	**.328**
FIGER	Additional classifier	.738	.682	.709	**.704**	.608	.652	.452	.384	.415
FIGER	L2AWE	.702	.633	.666	.674	.551	.606	.367	.261	.305
FIGER	KENN bottom up	**.745**	.706	.725	.702	.626	.662	.46	.369	.409
FIGER	KENN top down	.72	.675	.697	.686	.603	.642	.431	.43	.43
FIGER	KENN hybrid	.738	.698	.718	.696	.622	.657	.442	.398	.419
FIGER	KENN cross	.73	.71	.72	.691	.646	.668	.463	.438	**.45**
FIGER	KENN cross bottom up	.74	**.725**	**.732**	.692	**.655**	**.673**	**.479**	.413	.443
FIGER	KENN cross top down	.717	.692	.704	.677	.629	.652	.43	**.46**	.444
FIGER	KENN cross hybrid	.734	.71	.722	.688	.639	.663	.455	.422	.438

Table 6.8: Results of Full-fledged Domain Adaptation with OntoNotes 10-shot as target domain. First row reports the in-domain FET model trained directly on the 10-shot dataset. Rows 2-10 reports different adaptation techniques starting from a model trained on BBN. Rows 11-19 reports different adaptation techniques starting from a model trained on FIGER. **Bold** highlights best adaptation model for a given source domain, **underlined bold** highlights the best performance overall.

Source Domain	Adaptation Technique	Macro averaged on Examples			Micro averaged			Macro averaged on Types		
		P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
OntoNotes	-	.643	.661	.652	.603	.587	.595	.323	.349	.336
BBN	Additional classifier	.666	.636	.651	.63	.566	.596	.386	.327	.354
BBN	L2AWE	**.68**	.632	.655	**.644**	.56	.599	**.408**	.293	.341
BBN	KENN bottom up	.669	.651	.66	.628	.58	**.603**	.397	.325	.357
BBN	KENN top down	.631	.62	.625	.598	.555	.576	.334	.362	.347
BBN	KENN hybrid	.657	.646	.651	.616	.576	.595	.354	.341	.347
BBN	KENN cross	.652	.649	.651	.607	.585	.596	.372	.36	**.366**
BBN	KENN cross bottom up	.659	**.657**	**.658**	.613	**.59**	.601	.384	.331	.356
BBN	KENN cross top down	.62	.625	.623	.585	.566	.575	.332	**.379**	.354
BBN	KENN cross hybrid	.646	.642	.644	.603	.578	.59	.354	.35	.352
FIGER	Additional classifier	.72	.684	.701	.68	.607	.641	.433	.388	.409
FIGER	L2AWE	.69	.642	.665	.66	.567	.61	.416	.346	.378
FIGER	KENN bottom up	**.725**	.699	.712	**.681**	.622	.65	**.455**	.38	.414
FIGER	KENN top down	.693	.672	.682	.656	.6	.627	.395	.428	.411
FIGER	KENN hybrid	.717	.695	.706	.674	.62	.646	.431	.409	.42
FIGER	KENN cross	.716	.71	.713	.672	.642	.657	.427	.423	.425
FIGER	KENN cross bottom up	.721	**.716**	**.719**	.674	**.645**	**.659**	.446	.409	.427
FIGER	KENN cross top down	.692	.685	.689	.653	.621	.637	.399	**.45**	.423
FIGER	KENN cross hybrid	.713	.704	.708	.667	.636	.651	.429	.431	**.43**

Table 6.9: Results of Full-fledged Domain Adaptation with OntoNotes 20-shot as target domain. First row reports the in-domain FET model trained directly on the 20-shot dataset. Rows 2-10 reports different adaptation techniques starting from a model trained on BBN. Rows 11-19 reports different adaptation techniques starting from a model trained on FIGER. **Bold** highlights best adaptation model for a given source domain, **underlined bold** highlights the best performance overall.

useful to distinguish between OntoNotes types. Like in the previous experiment, adaptation techniques that use KENN to inject the `Cross` KB are the most effective to adapt a FET model. Lastly, L2AWE is again the model that often obtains the best *Precision*, but only when the source domain is BBN.

6.5 Conclusions on the Full-fledged Domain Adaptation in FET Scenario

In Fine-grained Entity Typing full-fledged domain adaptation is often necessary to use a FET model trained on a rich source domain in a target domain, with only a limited quantity of data available to adapt the model. Different domain adaptation techniques based on the exploitation of predictions on source domain types have been proposed and experimented to adapt a trained FET model to target scenario using only a limited quantity of annotated data for the adaptation process. Experiments show that the variety of types in the source domain is a key factor, since the encoder can be trained to extract features useful for representing the entity mentions with a rich representation, necessary to correctly classify types from a broad type hierarchy. Moreover, the exploitation of predictions of source domain types shown to be useful to enhance the learning of new target types.

Answering to the research questions, to inject the explicit knowledge with a neuro-symbolic integration method (Knowledge Enhanced Neural Network, KENN [DS19]), different adaptation techniques to inject the explicit knowledge in a neural architecture were proposed and experimented. In particular, hierarchic dependencies in target type hierarchy and equivalence cross-domain relations between source and target types are used enrich the available information in a low resource scenario. Experiments show that cross-domain relations (injected through the `Cross` KB) are often a key information to enhance performance in the target domain.

As argued along this thesis (Chapters 3 and 4), types highly characterize different domains in FET, for this reason, this experimentation only covered the adaptation of the classifier of a FET model exploiting information related to types in source and target domain. The joint adaptation of the encoder and of the classifier is an interesting and challenging future direction, since the limited resources available in the source scenario may cause forgetting of source domain knowledge and also may be not sufficient to generalize the target language distribution. However, as shown by the experiment with FIGER as target domain, extracting the correct features from the input is still important in this task.

Chapter 7

Conclusions and Future Works

7.1 Conclusions

This Ph.D. thesis focuses on domain adaptation in fine-grained entity typing (FET): in the first place, the notion of domain in FET is formalized, thereafter the domain adaptation problem is described and different scenarios related to FET are introduced, formalized, and addressed by introducing experimental analyses (see Chapter 4) or novel solutions (see Chapters 5 and 6). In particular, the *model reuse* scenario refers to the problem of using a FET model that has been trained on a source domain, on a target domain, under the assumption that the target type hierarchy is a subset of the source type hierarchy; the *model specialization* scenario concerns the problem of specializing a FET model to classify entities with types that are more specific than the ones on which the model has been trained, i.e. extending its capabilities on a given *family* of types (i.e., specializing a FET model that simply recognizes *persons* to recognize different *jobs*); finally, the *full-fledged domain adaptation* scenario relates to the problem of a FET model that needs to be adapted in order for it to be used in a target domain partially overlapped with the source domain, such that relations between types can be defined.

Blending explicit knowledge and latent representation derived from data using neuro-symbolic integration techniques is becoming increasingly popular in several Artificial Intelligence fields; the solution proposed as a novel contribution in this thesis takes advantage of *Knowledge Enhanced Neural Network* (KENN) to explicitly represent equivalence, generalization, specialization and disjunction relations and inject these pieces of knowledge into a model to support its adaptation to a new domain.

Regarding the *model reuse* scenario, the addressed research question concern how performance of a FET model on equivalent types transfer across domains. Experiments showed that the precision of models trained on a source domain tested on a target domain can be comparable or even better than the precision of models trained directly on the target domain, e.g., when the source domain has a rich type set and comes with clean training data. A denoising technique was also used to alleviate label noise in the source domain but results show that this kind of denoising is not effective in this scenario.

In regards to the *model specialization* scenario, addressed research questions concern how explicit hierarchical type relations can be exploited to favor the specialization of a FET model. During the experimentation, a classifier (neural or box embedding based) with a smart initialization (based on equivalence or specialization relations) is compared with KENN-based techniques showing that the usage of a neuro-symbolic approach (KENN) to exploit hierarchical type relations results in an improvement of performance in low-resource target domain.

Lastly, in the *full-fledged domain adaptation* scenario, research questions ask how does explicit type relations can be exploited to favor the domain adaptation of a FET model. The number of type in the source type hierarchy has shown to be a central factor, as well as the exploitation of predictions on source domain types, to enhance the learning of new target types. Injecting explicit knowledge about target type hierarchy and cross-domain equivalence relations through a neuro-symbolic integration method (KENN) has also shown to be useful to enhance performance in the low-resource target domain.

Finally, the problem of domain adaptation in FET, especially when declined to specialization and full-fledged adaptation, is a central problem to face in order to enable effective transfer learning approaches between different domains, favoring the performance on low-resource scenarios. The proposed techniques has proven to be effective; however, the joint optimization of the encoder and the classifier while adapting a FET model, maintaining its capabilities on the source domain while exploiting them on the target domain, remains a challenging problem.

7.2 Future Work

In this thesis, domain adaptation is faced by using a neuro-symbolic integration approach (Knowledge Enhanced Neural Network) to inject equivalence, specialization, generalization, and disjunction relations in order to favor the adaptation of a FET model to low-resource target domain. The proposed approaches are compared with models from the literature that seems to be suitable to handle hierarchical type similarity [Ono+21] and cross-domain types relation [Noz+21], even if these solutions were proposed respectively for in-domain FET [Ono+21] and for domain adaptation in NER [Noz+21]. The possibility of plugging KENN on top of a given network enables for future work focused on KENN used on-top of different models from literature, resulting in the joint exploit of different explicit and implicit evidences. For example, by combining L2AWE [Noz+21] and KENN, explicit and latent type relations can be handled during the adaptation.

Even if KENN-based adaptation techniques has proven to be quite effective in the low-resource model specialization and low-resource full-fledged domain adaptation scenarios, KENN relies on explicit knowledge and on the FET predictions. In NLP literature, a large number of approaches optimize type representation to face a specific task, or are focused on obtaining task-agnostic dense representations of types, known to be useful in zero-shot classification or in natural language inference and understanding tasks. The integration of hidden and type representation inside KENN will enable the usage of these dense representations during the computation of the knowledge enhancement; this interesting evolution of this thesis may be useful to obtain an additional way to mix sub-symbolic and symbolic knowledge.

The inclusion of hidden representation in KENN framework can enable for the joint usage of the proposed KENN-based techniques with other techniques based on hidden representation like Contrastive Learning [KAM22], where examples are paired and the encoder is optimized to minimize a loss function where examples pairs have to be close (positive pairs) or far (negative pairs). Contrastive Learning is often used jointly with classification models, since KENN can be placed on top of a classifier, that is treated as a black-box model, the joint usage of these approaches is an interesting future work.

In this thesis FET is faced, but the proposed approaches can be applied to other multilabel classification tasks where external knowledge can be explicitly modeled with clauses and injected with KENN to facilitate the adaptation problem. However, type hierarchies are not so present in general multilabel classification problems, thus to apply the presented techniques in different tasks, the source and the logical model of the knowledge is a crucial step.

Bibliography

[AAA17] Abhishek Abhishek, Ashish Anand, and Amit Awekar. “Fine-Grained Entity Type Classification by Jointly Learning Representations and Label Embeddings”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 797–807. URL: https://aclanthology.org/E17-1075.

[Ali+20] Muhammad Asif Ali et al. “Fine-Grained Named Entity Typing over Distantly Supervised Data Based on Refined Representations”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.05 (Apr. 2020), pp. 7391–7398. DOI: 10.1609/aaai.v34i05.6234. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6234.

[Ali+21] Muhammad Asif Ali et al. “Fine-Grained Named Entity Typing over Distantly Supervised Data via Refinement in Hyperbolic Space”. In: *arXiv preprint arXiv:2101.11212* (2021).

[Als+19] Emily Alsentzer et al. “Publicly Available Clinical BERT Embeddings”. In: *Proceedings of the 2nd Clinical Natural Language Processing Workshop*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, June 2019, pp. 72–78. DOI: 10.18653/v1/W19-1909. URL: https://aclanthology.org/W19-1909.

[ANC08] Andrew Arnold, Ramesh Nallapati, and William Cohen. “Exploiting feature hierarchy for transfer learning in named entity recognition”. In: *Proceedings of ACL-08: HLT*. 2008, pp. 245–253.

[Bad+22] Samy Badreddine et al. “Logic Tensor Networks”. In: *Artificial Intelligence* 303 (2022), p. 103649. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2021.103649. URL: https://www.sciencedirect.com/science/article/pii/S0004370221002009.

[BBH+16] Hannah Bast, Björn Buchhold, Elmar Haussmann, et al. “Semantic search on text and knowledge bases”. In: *Foundations and Trends® in Information Retrieval* 10.2-3 (2016), pp. 119–271.

[BLC19] Iz Beltagy, Kyle Lo, and Arman Cohan. “SciBERT: A Pretrained Language Model for Scientific Text”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3615–3620. DOI: 10.18653/v1/D19-1371. URL: https://aclanthology.org/D19-1371.

[CCV20] Tongfei Chen, Yunmo Chen, and Benjamin Van Durme. “Hierarchical Entity Typing via Multi-level Learning to Rank”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, 2020, pp. 8465–8475. DOI: 10.18653/v1/2020.acl-main.749. URL: https://aclanthology.org/2020.acl-main.749.

[Che+18] Lihan Chen et al. “Short Text Entity Linking with Fine-Grained Topics”. In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. CIKM ’18. Torino, Italy: Association for Computing Machinery, 2018, pp. 457–466. ISBN: 9781450360142. DOI: 10.1145/3269206.3271809. URL: https://doi.org/10.1145/3269206.3271809.

[Che+19] Bo Chen et al. “Improving Distantly-supervised Entity Typing with Compact Latent Space Clustering”. 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, June 2019, pp. 2862–2872. DOI: 10.18653/v1/N19-1294. URL: https://aclanthology.org/N19-1294.

[Che+22] Xiang 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. URL: https://aclanthology.org/2022.coling-1.209.

[CHM19] Tuhin Chakrabarty, Christopher Hidey, and Kathy McKeown. “IMHO Fine-Tuning Improves Claim Detection”. 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, June 2019, pp. 558–563. DOI: 10.18653/v1/N19-1054. URL: https://aclanthology.org/N19-1054.

[Cho+18] Eunsol 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, July 2018, pp. 87–96. DOI: 10.18653/v1/P18-1009. URL: https://aclanthology.org/P18-1009.

[CL96] Jim Cowie and Wendy Lehnert. "Information Extraction". In: *Commun. ACM* 39.1 (Jan. 1996), pp. 80–91. ISSN: 0001-0782. DOI: 10.1145/234173.234209. URL: https://doi.org/10.1145/234173.234209.

[CM19] Lingzhen Chen and Alessandro Moschitti. "Transfer learning for sequence labeling using source model and target data". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 6260–6267.

[CR98] N. Chinchor and P. Robinson. "Appendix E: MUC-7 Named Entity Task Definition (version 3.5)". In: *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*. 1998. URL: https://aclanthology.org/M98-1028.

[Dai+19] Hongliang Dai et al. "Improving Fine-grained Entity Typing with Entity Linking". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6210–6215. DOI: 10.18653/v1/D19-1643. URL: https://aclanthology.org/D19-1643.

[Das+20] Shib Dasgupta et al. "Improving local identifiability in probabilistic box embeddings". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 182–192.

[Dau07] Hal Daumé III. "Frustratingly Easy Domain Adaptation". In: *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Prague, Czech Republic: Association for Computational Linguistics, June 2007, pp. 256–263. URL: https://aclanthology.org/P07-1033.

[Dev+19] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[DGS17] Michelangelo Diligenti, Marco Gori, and Claudio Sacca. "Semantic-based regularization for learning and inference". In: *Artificial Intelligence* 244 (2017), pp. 143–165.

[Din+21] Ning Ding et al. "Prompt-learning for fine-grained entity typing". In: *arXiv preprint arXiv:2108.10604* (2021).

[DS19] Alessandro Daniele and Luciano Serafini. "Knowledge enhanced neural networks". In: *Pacific Rim International Conference on Artificial Intelligence*. Springer. 2019, pp. 542–554.

[DSL20] Hongliang Dai, Yangqiu Song, and Xin Li. "Exploiting Semantic Relations for Fine-grained Entity Typing". In: *Conference on Automated Knowledge Base Construction, AKBC 2020, Virtual, June 22-24, 2020*. Ed. by Dipanjan Das et al. 2020. DOI: 10.24432/C5J017. URL: https://doi.org/10.24432/C5J017.

[DSW21a] Hongliang Dai, Yangqiu Song, and Haixun 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, 2021, pp. 1790–1799. DOI: 10.18653/v1/2021.acl-long.141. URL: https://aclanthology.org/2021.acl-long.141.

[DSW21b] Hongliang Dai, Yangqiu Song, and Haixun 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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*. Ed. by Chengqing Zong et al. Association for Computational Linguistics, 2021, pp. 1790–1799. DOI: 10.18653/v1/2021.acl-long.141. URL: https://doi.org/10.18653/v1/2021.acl-long.141.

[FK14] Benoît Frénay and Ata Kaban. "A Comprehensive Introduction to Label Noise: Proceedings of the 2014 European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2014)". English. In: *Proceedings of the 2014 European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2014)*. i6doc.com.publ., 2014.

[Fle01] Michael Fleischman. "Automated Subcategorization of Named Entities". In: *ACL*. 2001.

[FV14] Benoit Frenay and Michel Verleysen. "Classification in the Presence of Label Noise: A Survey". In: *IEEE Transactions on Neural Networks and Learning Systems* 25.5 (2014), pp. 845–869. DOI: `10.1109/TNNLS.2013.2292894`.

[GB18] John M Giorgi and Gary D Bader. "Transfer learning for biomedical named entity recognition with neural networks". In: *Bioinformatics* 34.23 (2018), pp. 4087–4094.

[Gil+14] Dan Gillick et al. *Context-Dependent Fine-Grained Entity Type Tagging*. 2014. URL: `http://arxiv.org/abs/1412.1820`.

[Gri15] Ralph Grishman. "Information Extraction". In: *IEEE Intelligent Systems* 30.5 (2015), pp. 8–15. DOI: `10.1109/MIS.2015.68`.

[GS96] Ralph Grishman and Beth Sundheim. "Message Understanding Conference-6: A Brief History". In: *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*. 1996. URL: `https://aclanthology.org/C96-1079`.

[Gu+21] Yu Gu et al. "Domain-specific language model pretraining for biomedical natural language processing". In: *ACM Transactions on Computing for Healthcare (HEALTH)* 3.1 (2021), pp. 1–23.

[Gur+20] Suchin Gururangan et al. "Don't Stop Pretraining: Adapt Language Models to Domains and Tasks". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 8342–8360. DOI: `10.18653/v1/2020.acl-main.740`. URL: `https://aclanthology.org/2020.acl-main.740`.

[Han+17] Sangdo Han et al. "Answer Ranking Based on Named Entity Types for Question Answering". In: *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*. IMCOM '17. Beppu, Japan: Association for Computing Machinery, 2017. ISBN: 9781450348881. DOI: `10.1145/3022227.3022297`. URL: `https://doi.org/10.1145/3022227.3022297`.

[Hao+20] Yaru Hao et al. “Investigating Learning Dynamics of BERT Fine-Tuning”. In: *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing.* Suzhou, China: Association for Computational Linguistics, Dec. 2020, pp. 87–92. URL: https://aclanthology.org/2020.aacl-main.11.

[Har81] Zellig S. Harris. “Distributional Structure”. In: *Papers on Syntax.* Dordrecht: Springer Netherlands, 1981, pp. 3–22. ISBN: 978-94-009-8467-7. DOI: 10.1007/978-94-009-8467-7_1. URL: https://doi.org/10.1007/978-94-009-8467-7_1.

[He+21] Ruidan He et al. “On the Effectiveness of Adapter-based Tuning for Pretrained Language Model Adaptation”. 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. 2208–2222. DOI: 10.18653/v1/2021.acl-long.172. URL: https://aclanthology.org/2021.acl-long.172.

[HE19] Xiaochuang Han and Jacob Eisenstein. “Unsupervised Domain Adaptation of Contextualized Embeddings for Sequence Labeling”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).* Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4238–4248. DOI: 10.18653/v1/D19-1433. URL: https://aclanthology.org/D19-1433.

[Hed+21] Michael A Hedderich et al. “A Survey on Recent Approaches for Natural Language Processing in Low-Resource Scenarios”. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* 2021, pp. 2545–2568.

[Hon+21] Jimin Hong et al. “AVocaDo: Strategy for Adapting Vocabulary to Downstream Domain”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing.* 2021, pp. 4692–4700.

[Hos+19] Mohammad Javad Hosseini et al. “Duality of Link Prediction and Entailment Graph Induction”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.* Florence, Italy: Association for Computational Linguistics, July 2019, pp. 4736–

4746. DOI: 10.18653/v1/P19-1468. URL: https://aclanthology.org/P19-1468.

[Hou+19] Neil Houlsby et al. “Parameter-Efficient Transfer Learning for NLP”. In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 2790–2799. URL: http://proceedings.mlr.press/v97/houlsby19a.html.

[HR18] Jeremy Howard and Sebastian Ruder. “Universal Language Model Fine-tuning for Text Classification”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 328–339. DOI: 10.18653/v1/P18-1031. URL: https://aclanthology.org/P18-1031.

[Hu+19] Wenpeng Hu et al. “Overcoming catastrophic forgetting for continual learning via model adaptation”. In: *International conference on learning representations*. 2019.

[HWZ21] Feng Hou, Ruili Wang, and Yi Zhou. “Transfer learning for fine-grained entity typing”. In: *Knowledge and Information Systems* 63.4 (Apr. 2021), pp. 845–866. ISSN: 0219-3116. DOI: 10.1007/s10115-021-01549-5. URL: https://doi.org/10.1007/s10115-021-01549-5.

[KAM22] Adnan Khan, Sarah AlBarri, and Muhammad Arslan Manzoor. “Contrastive Self-Supervised Learning: A Survey on Different Architectures”. In: *2022 2nd International Conference on Artificial Intelligence (ICAI)*. 2022, pp. 1–6. DOI: 10.1109/ICAI55435.2022.9773725.

[KMC16] Vivek Kulkarni, Yashar Mehdad, and Troy Chevalier. “Domain adaptation for named entity recognition in online media with word embeddings”. In: *arXiv preprint arXiv:1612.00148* (2016).

[KS19] Dan Kondratyuk and Milan Straka. “75 Languages, 1 Model: Parsing Universal Dependencies Universally”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 2779–2795. DOI: 10.18653/v1/D19-1279. URL: https://aclanthology.org/D19-1279.

[Kua+20] Jun Kuang et al. “Improving neural relation extraction with implicit mutual relations”. In: *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE. 2020, pp. 1021–1032.

[LDS18] Ji Young Lee, Franck Dernoncourt, and Peter Szolovits. “Transfer Learning for Named-Entity Recognition with Neural Networks”. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. 2018.

[Lee+20] Jinhyuk Lee et al. “BioBERT: a pre-trained biomedical language representation model for biomedical text mining”. In: *Bioinformatics* 36.4 (2020), p. 1234.

[LHS19] Federico López, Benjamin Heinzerling, and Michael Strube. “Fine-Grained Entity Typing in Hyperbolic Space”. In: *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 169–180. DOI: `10.18653/v1/W19-4319`. URL: `https://aclanthology.org/W19-4319`.

[Li+13] Yegang Li et al. “Named entity recognition based on bilingual co-training”. In: *Workshop on Chinese Lexical Semantics*. Springer. 2013, pp. 480–489.

[Li+21] Jinqing Li et al. “Enhancing Label Representations with Relational Inductive Bias Constraint for Fine-Grained Entity Typing”. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. Ed. by Zhi-Hua Zhou. Main Track. International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 3843–3849. DOI: `10.24963/ijcai.2021/529`. URL: `https://doi.org/10.24963/ijcai.2021/529`.

[Liu+20] Chuanbo Liu et al. “An Instance Transfer-Based Approach Using Enhanced Recurrent Neural Network for Domain Named Entity Recognition”. In: *IEEE Access* 8 (2020), pp. 45263–45270. DOI: `10.1109/ACCESS.2020.2974022`.

[Liu+21a] Qing Liu et al. “Fine-grained Entity Typing via Label Reasoning”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 4611–4622. DOI: `10.18653/v1/2021.emnlp-main.378`. URL: `https://aclanthology.org/2021.emnlp-main.378`.

[Liu+21b] Wei Liu et al. "Lexicon Enhanced Chinese Sequence Labeling Using BERT Adapter". 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. 5847–5858. DOI: `10.18653/v1/2021.acl-long.454`. URL: `https://aclanthology.org/2021.acl-long.454`.

[LJ19] Ying Lin and Heng Ji. "An Attentive Fine-Grained Entity Typing Model with Latent Type Representation". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6197–6202. DOI: `10.18653/v1/D19-1641`. URL: `https://aclanthology.org/D19-1641`.

[LKB20] Qi Liu, Matt J Kusner, and Phil Blunsom. "A survey on contextual embeddings". In: *arXiv preprint arXiv:2003.07278* (2020).

[LL18] Bill Yuchen Lin and Wei Lu. "Neural Adaptation Layers for Cross-domain Named Entity Recognition". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 2012–2022. DOI: `10.18653/v1/D18-1226`. URL: `https://aclanthology.org/D18-1226`.

[Log+19] Lajanugen Logeswaran et al. "Zero-Shot Entity Linking by Reading Entity Descriptions". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 3449–3460. DOI: `10.18653/v1/P19-1335`. URL: `https://aclanthology.org/P19-1335`.

[LW12] Xiao Ling and Daniel S. Weld. "Fine-Grained Entity Recognition". In: *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*. Ed. by Jörg Hoffmann and Bart Selman. AAAI Press, 2012. URL: `http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5152`.

[Mar+20] Giuseppe Marra et al. "Relational Neural Machines". In: (2020).

[McC05] Andrew McCallum. "Information extraction: Distilling structured data from unstructured text". In: *Queue* 3.9 (2005), pp. 48–57.

[MCG16] Yukun Ma, Erik Cambria, and Sa Gao. "Label Embedding for Zero-shot Fine-grained Named Entity Typing". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 171–180. URL: https://aclanthology.org/C16-1017.

[Mik+13a] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems*. Ed. by C. J. C. Burges et al. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[Mik+13b] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. URL: http://arxiv.org/abs/1301.3781.

[ML18] Sunil Mohan and Donghui Li. "MedMentions: A Large Biomedical Corpus Annotated with UMLS Concepts". In: *Automated Knowledge Base Construction (AKBC)*. 2018.

[Mon+21] Natawut Monaikul et al. "Continual Learning for Named Entity Recognition". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.15 (May 2021), pp. 13570–13577. DOI: 10.1609/aaai.v35i15.17600. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17600.

[Mun+12] Tsendsuren Munkhdalai et al. "Bio Named Entity Recognition Based on Co-training Algorithm". In: *2012 26th International Conference on Advanced Information Networking and Applications Workshops* (2012), pp. 857–862.

[Mur+18] Shikhar Murty et al. "Hierarchical Losses and New Resources for Fine-grained Entity Typing and Linking". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 97–109. DOI: 10.18653/v1/P18-1010. URL: https://aclanthology.org/P18-1010.

[NC15] Arvind Neelakantan and Ming-Wei Chang. "Inferring Missing Entity Type Instances for Knowledge Base Completion: New Dataset and Methods". In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, May 2015, pp. 515–525. DOI: 10.3115/v1/N15-1054. URL: https://aclanthology.org/N15-1054.

[NOF10] David F. Nettleton, Albert Orriols-Puig, and Albert Fornells. "A study of the effect of different types of noise on the precision of supervised learning techniques". In: *Artificial Intelligence Review* 33.4 (Apr. 2010), pp. 275–306. ISSN: 1573-7462. DOI: 10.1007/s10462-010-9156-z. URL: https://doi.org/10.1007/s10462-010-9156-z.

[Noz+21] Debora Nozza et al. "LearningToAdapt with Word Embeddings: Domain Adaptation of Named Entity Recognition Systems". In: *Inf. Process. Manage.* 58.3 (May 2021). ISSN: 0306-4573. DOI: 10.1016/j.ipm.2021.102537. URL: https://doi.org/10.1016/j.ipm.2021.102537.

[NS07] David Nadeau and Satoshi Sekine. "A survey of named entity recognition and classification". In: *Lingvisticae Investigationes* 30.1 (2007), pp. 3–26.

[Obe+19] Rasha Obeidat et al. "Description-Based Zero-shot Fine-Grained Entity Typing". 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, June 2019, pp. 807–814. DOI: 10.18653/v1/N19-1087. URL: https://aclanthology.org/N19-1087.

[OD19] Yasumasa Onoe and Greg Durrett. "Learning to Denoise Distantly-Labeled Data for Entity Typing". 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, June 2019. DOI: 10.18653/v1/N19-1250. URL: https://aclanthology.org/N19-1250.

[OD20] Yasumasa Onoe and Greg Durrett. "Interpretable Entity Representations through Large-Scale Typing". In: *Findings of the Association for Computational Linguistics: EMNLP 2020.* Online: Association for Computational Linguistics, Nov. 2020, pp. 612–624. DOI: 10.18653/v1/2020.findings-emnlp.54. URL: https://aclanthology.org/2020.findings-emnlp.54.

[Ono+21] Yasumasa Onoe et al. "Modeling Fine-Grained Entity Types with Box Embeddings". 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. 2051–2064. DOI: 10.18653/v1/2021.acl-long.160. URL: https://aclanthology.org/2021.acl-long.160.

[Pan+22] Kunyuan Pang et al. "Divide and Denoise: Learning from Noisy Labels in Fine-Grained Entity Typing with Cluster-Wise Loss Correction". 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. 1997–2006. DOI: 10.18653/v1/2022.acl-long.141. URL: https://aclanthology.org/2022.acl-long.141.

[Pet+18] Matthew E. Peters et al. "Deep Contextualized Word Representations". 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, June 2018, pp. 2227–2237. DOI: 10.18653/v1/N18-1202. URL: https://aclanthology.org/N18-1202.

[PFB18] Jason Phang, Thibault Févry, and Samuel R. Bowman. "Sentence Encoders on STILTs: Supplementary Training on Intermediate Labeled-data Tasks". In: *CoRR* abs/1811.01088 (2018). arXiv: 1811.01088. URL: http://arxiv.org/abs/1811.01088.

[Pfe+20a] Jonas Pfeiffer et al. "AdapterHub: A Framework for Adapting Transformers". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 46–54. DOI: 10.18653/v1/2020.emnlp-demos.7. URL: https://aclanthology.org/2020.emnlp-demos.7.

[Pfe+20b] Jonas Pfeiffer et al. "MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 7654–7673. DOI: 10.18653/v1/2020.emnlp-main.617. URL: https://aclanthology.org/2020.emnlp-main.617.

[Pfe+21] Jonas Pfeiffer et al. "AdapterFusion: Non-Destructive Task Composition for Transfer Learning". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, Apr. 2021, pp. 487–503. DOI: 10.18653/v1/2021.eacl-main.39. URL: https://aclanthology.org/2021.eacl-main.39.

[Pla16] Barbara Plank. “What to do about non-standard (or non-canonical) language in NLP”. In: *arXiv preprint arXiv:1608.07836* (2016).

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher 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. DOI: `10.3115/v1/D14-1162`. URL: `https://aclanthology.org/D14-1162`.

[PWS20] Nina Poerner, Ulli Waltinger, and Hinrich Schütze. “Inexpensive Domain Adaptation of Pretrained Language Models: Case Studies on Biomedical NER and Covid-19 QA”. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov. 2020, pp. 1482–1490. DOI: `10.18653/v1/2020.findings-emnlp.134`. URL: `https://aclanthology.org/2020.findings-emnlp.134`.

[PY10] Sinno Jialin Pan and Qiang Yang. “A survey on transfer learning”. In: *IEEE Transactions on knowledge and data engineering* 22.10 (2010), pp. 1345–1359.

[Qia+21] Jing Qian et al. “Fine-grained Entity Typing without Knowledge Base”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 5309–5319. DOI: `10.18653/v1/2021.emnlp-main.431`. URL: `https://aclanthology.org/2021.emnlp-main.431`.

[Qu+16] Lizhen Qu et al. “Named Entity Recognition for Novel Types by Transfer Learning”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 899–905. DOI: `10.18653/v1/D16-1087`. URL: `https://aclanthology.org/D16-1087`.

[Qui86] J. R. Quinlan. “Induction of Decision Trees”. In: *Mach. Learn.* 1.1 (Mar. 1986), pp. 81–106. ISSN: 0885-6125. DOI: `10.1023/A:1022643204877`. URL: `https://doi.org/10.1023/A:1022643204877`.

[Ren+16a] Xiang Ren et al. “AFET: Automatic Fine-Grained Entity Typing by Hierarchical Partial-Label Embedding”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1369–1378. DOI: `10.18653/v1/D16-1144`. URL: `https://aclanthology.org/D16-1144`.

[Ren+16b] Xiang Ren et al. “Label Noise Reduction in Entity Typing by Heterogeneous Partial-Label Embedding”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 1825–1834. ISBN: 9781450342322. DOI: 10.1145/2939672.2939822. URL: https://doi.org/10.1145/2939672.2939822.

[Ren20] Quan Ren. “Fine-Grained Entity Typing with Hierarchical Inference”. In: *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. Vol. 1. 2020, pp. 2552–2558. DOI: 10.1109/ITNEC48623.2020.9085112.

[RKR20] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. “A Primer in BERTology: What We Know About How BERT Works”. In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 842–866. DOI: 10.1162/tacl_a_00349. URL: https://aclanthology.org/2020.tacl-1.54.

[RN10] Altaf Rahman and Vincent Ng. “Inducing Fine-Grained Semantic Classes via Hierarchical and Collective Classification”. In: *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Beijing, China: Coling 2010 Organizing Committee, Aug. 2010, pp. 931–939. URL: https://aclanthology.org/C10-1105.

[RP20] Alan Ramponi and Barbara Plank. “Neural Unsupervised Domain Adaptation in NLP—A Survey”. In: *Proceedings of the 28th International Conference on Computational Linguistics*. 2020, pp. 6838–6855.

[Rud+19] Sebastian Ruder et al. “Latent Multi-Task Architecture Learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019), pp. 4822–4829. DOI: 10.1609/aaai.v33i01.33014822. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4410.

[Rud19] Sebastian Ruder. “Neural transfer learning for natural language processing”. PhD thesis. NUI Galway, 2019.

[SC19] Ta-Chun Su and Hsiang-Chih Cheng. *SesameBERT: Attention for Anywhere*. 2019. DOI: 10.48550/ARXIV.1910.03176. URL: https://arxiv.org/abs/1910.03176.

[Seo+21] Sungyong Seo et al. “Controlling Neural Networks with Rule Representations”. In: *Advances in Neural Information Processing Systems* 34 (2021).

[SGC09] Gonçalo Simoes, Helena Galhardas, and Luısa Coheur. “Information Extraction tasks: a survey”. In: *Simpósio de Informática 2009* (2009), p. 540.

[Shi+17a] Sonse Shimaoka et al. “Neural Architectures for Fine-grained Entity Type Classification”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, 2017, pp. 1271–1280. URL: https://aclanthology.org/E17-1119.

[Shi+17b] Sonse Shimaoka et al. “Neural Architectures for Fine-grained Entity Type Classification”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 1271–1280. URL: https://aclanthology.org/E17-1119.

[Shi+20] Haochen Shi et al. “Alleviate Dataset Shift Problem in Fine-grained Entity Typing with Virtual Adversarial Training”. In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. Ed. by Christian Bessiere. Main track. International Joint Conferences on Artificial Intelligence Organization, July 2020, pp. 3898–3904. DOI: 10.24963/ijcai.2020/539. URL: https://doi.org/10.24963/ijcai.2020/539.

[SLL22] Kaili Sun, Xudong Luo, and Michael Y. Luo. “A Survey ofnbsp;Pretrained Language Models”. In: *Knowledge Science, Engineering and Management: 15th International Conference, KSEM 2022, Singapore, August 6–8, 2022, Proceedings, Part II*. Singapore, Singapore: Springer-Verlag, 2022, pp. 442–456. ISBN: 978-3-031-10985-0. DOI: 10.1007/978-3-031-10986-7_36. URL: https://doi.org/10.1007/978-3-031-10986-7_36.

[Sun+19] Chi Sun et al. “How to Fine-Tune BERT for Text Classification?” In: *CoRR* abs/1905.05583 (2019). arXiv: 1905.05583. URL: http://arxiv.org/abs/1905.05583.

[Sun95] Beth M. Sundheim. “Overview of Results of the MUC-6 Evaluation”. In: *Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995*. 1995. URL: https://aclanthology.org/M95-1002.

[SZ14] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[Tai+20] Wen Tai et al. "exBERT: Extending Pre-trained Models with Domain-specific Vocabulary Under Constrained Training Resources". In: *Findings of the Association for Computational Linguistics: EMNLP 2020.* Online: Association for Computational Linguistics, Nov. 2020, pp. 1433–1439. DOI: `10.18653/v1/2020.findings-emnlp.129`. URL: `https://aclanthology.org/2020.findings-emnlp.129`.

[Tjo02] Erik F. Tjong Kim Sang. "Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition". In: *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002).* 2002. URL: `https://aclanthology.org/W02-2024`.

[Wan+22a] Rui Wang et al. "Few-Shot Class-Incremental Learning for Named Entity Recognition". 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. 571–582. DOI: `10.18653/v1/2022.acl-long.43`. URL: `https://aclanthology.org/2022.acl-long.43`.

[Wan+22b] Ruili Wang et al. "Fine-Grained Entity Typing with a Type Taxonomy: a Systematic Review". In: *IEEE Transactions on Knowledge and Data Engineering* (2022), pp. 1–1. DOI: `10.1109/TKDE.2022.3148980`.

[WB05] Ralph Weischedel and Ada Brunstein. "BBN pronoun coreference and entity type corpus". In: *Linguistic Data Consortium, Philadelphia* 112 (2005).

[Wei+11] Ralph Weischedel et al. "OntoNotes: A large training corpus for enhanced processing". In: *Handbook of Natural Language Processing and Machine Translation. Springer* 3.3 (2011), pp. 3–4.

[Wu+19] Junshuang Wu et al. "Modeling Noisy Hierarchical Types in Fine-Grained Entity Typing: A Content-Based Weighting Approach". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019.* Ed. by Sarit Kraus. ijcai.org, 2019, pp. 5264–5270. DOI: `10.24963/ijcai.2019/731`. URL: `https://doi.org/10.24963/ijcai.2019/731`.

[Wu+22] Junshuang Wu et al. “Dealing With Hierarchical Types and Label Noise in Fine-Grained Entity Typing”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 30 (2022), pp. 1305–1318. DOI: 10.1109/TASLP.2022.3155281.

[XB18] Peng Xu and Denilson Barbosa. “Neural Fine-Grained Entity Type Classification with Hierarchy-Aware Loss”. 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, June 2018, pp. 16–25. DOI: 10.18653/v1/N18-1002. URL: https://aclanthology.org/N18-1002.

[Xio+19] Wenhan Xiong et al. “Imposing Label-Relational Inductive Bias for Extremely Fine-Grained Entity Typing”. 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, June 2019, pp. 773–784. DOI: 10.18653/v1/N19-1084. URL: https://aclanthology.org/N19-1084.

[YD18] Zheng Yuan and Doug Downey. “OTyper: A Neural Architecture for Open Named Entity Typing”. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI’18/IAAI’18/EAAI’18. New Orleans, Louisiana, USA: AAAI Press, 2018. ISBN: 978-1-57735-800-8.

[YGL15] Dani Yogatama, Daniel Gillick, and Nevena Lazic. “Embedding Methods for Fine Grained Entity Type Classification”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China: Association for Computational Linguistics, July 2015, pp. 291–296. DOI: 10.3115/v1/P15-2048. URL: https://aclanthology.org/P15-2048.

[Yos+12] Mohamed Amir 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. URL: https://www.aclweb.org/anthology/C12-2133.

[YS17] Yadollah Yaghoobzadeh and Hinrich Schütze. "Multi-level Representations for Fine-Grained Typing of Knowledge Base Entities". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 578–589. URL: `https://aclanthology.org/E17-1055`.

[YZ19] Junjie Yang and Hai Zhao. "Deepening Hidden Representations from Pre-trained Language Models". In: *arXiv preprint arXiv:1911.01940* (2019).

[ZDV18] Sheng Zhang, Kevin Duh, and Benjamin Van Durme. "Fine-grained Entity Typing through Increased Discourse Context and Adaptive Classification Thresholds". In: *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 173–179. DOI: `10.18653/v1/S18-2022`. URL: `https://aclanthology.org/S18-2022`.

[Zha+18] Denghui Zhang et al. "Path-Based Attention Neural Model for Fine-Grained Entity Typing". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, 2018, pp. 8179–8180. URL: `https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16544`.

[Zha+20a] Haoyu Zhang et al. "Learning with Noise: Improving Distantly-Supervised Fine-grained Entity Typing via Automatic Relabeling". In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. Ed. by Christian Bessiere. Main track. International Joint Conferences on Artificial Intelligence Organization, July 2020, pp. 3808–3815. DOI: `10.24963/ijcai.2020/527`. URL: `https://doi.org/10.24963/ijcai.2020/527`.

[Zha+20b] Rong Zhang et al. "Multi-Stage Pre-training for Low-Resource Domain Adaptation". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 5461–5468. DOI: `10.18653/v1/2020.emnlp-main.440`. URL: `https://aclanthology.org/2020.emnlp-main.440`.

[Zha+20c] Tao Zhang et al. "MZET: Memory Augmented Zero-Shot Fine-grained Named Entity Typing". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 77–87. DOI: `10.18653/v1/2020.coling-main.7`. URL: `https://aclanthology.org/2020.coling-main.7`.

[Zha+20d] Chen Zhao et al. "Complex Factoid Question Answering with a Free-Text Knowledge Graph". In: *Proceedings of The Web Conference 2020*. WWW '20. Taipei, Taiwan: Association for Computing Machinery, 2020, pp. 1205–1216. ISBN: 9781450370233. DOI: `10.1145/3366423.3380197`. URL: `https://doi.org/10.1145/3366423.3380197`.

[ZW04] Xingquan Zhu and Xindong Wu. "Class Noise vs. Attribute Noise: A Quantitative Study". In: *Artificial Intelligence Review* 22.3 (Nov. 2004), pp. 177–210. ISSN: 1573-7462. DOI: `10.1007/s10462-004-0751-8`. URL: `https://doi.org/10.1007/s10462-004-0751-8`.