Hierarchical Entity Typing via Multi-level Learning to Rank

Tongfei Chen **Yunmo Chen** **Benjamin Van Durme**
Johns Hopkins University
{tongfei, ychen, vandurme}@cs.jhu.edu

Abstract

We propose a novel method for hierarchical entity classification that embraces ontological structure at both training and during prediction. At training, our novel multi-level learning-to-rank loss compares positive types against negative siblings according to the type tree. During prediction, we define a coarse-to-fine decoder that restricts viable candidates at each level of the ontology based on already predicted parent type(s). We achieve state-of-the-art across multiple datasets, particularly with respect to strict accuracy.[1]

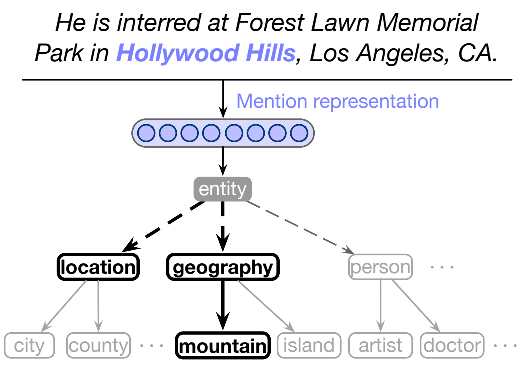

Figure 1: An example mention classified using the FIGER ontology. Positive types are highlighted.

1 Introduction

Entity typing is the assignment of a semantic label to a span of text, where that span is usually a *mention* of some entity in the real world. Named entity recognition (NER) is a canonical information extraction task, commonly considered a form of entity typing that assigns spans to one of a handful of types, such as `PER`, `ORG`, `GPE`, and so on. Fine-grained entity typing (FET) seeks to classify spans into types according to more diverse, semantically richer ontologies (Ling and Weld, 2012; Yosef et al., 2012; Gillick et al., 2014; Del Corro et al., 2015; Choi et al., 2018), and has begun to be used in downstream models for entity linking (Gupta et al., 2017; Raiman and Raiman, 2018).

Consider the example in Figure 1 from the FET dataset, FIGER (Ling and Weld, 2012). The mention of interest, *Hollywood Hills*, will be typed with the single label `LOC` in traditional NER, but may be typed with a *set* of types {`/location`, `/geography`, `/geography/mountain`} under a fine-grained typing scheme. In these finer-grained typing schemes, types usually form a hierarchy: there are a set of coarse types that lies on the top level—these are similar to traditional NER types, e.g. `/person`; additionally, there are finer types that are *subtypes* of these top-level types, e.g. `/person/artist` or `/person/doctor`.

Most prior work concerning fine-grained entity typing has approached the problem as a *multi-label classification* problem: given an entity mention together with its context, the classifier seeks to output a set of types, where each type is a node in the hierarchy. Approaches to FET include hand-crafted sparse features to various neural architectures (Ren et al., 2016a; Shimaoka et al., 2017; Lin and Ji, 2019, *inter alia*, see section 2).

Perhaps owing to the historical transition from "flat" NER types, there has been relatively little work in FET that exploits ontological *tree structure*, where type labels satisfy the *hierarchical property*: ***a subtype is valid only if its parent supertype is also valid.*** We propose a novel method that takes the explicit ontology structure into account, by a *multi-level learning to rank* approach that ranks the candidate types conditioned on the given entity mention. Intuitively, coarser types are easier whereas finer types are harder to classify: we capture this intuition by allowing distinct margins at each level of the ranking model.

[1] Code can be found at https://github.com/ctongfei/hierarchical-typing.

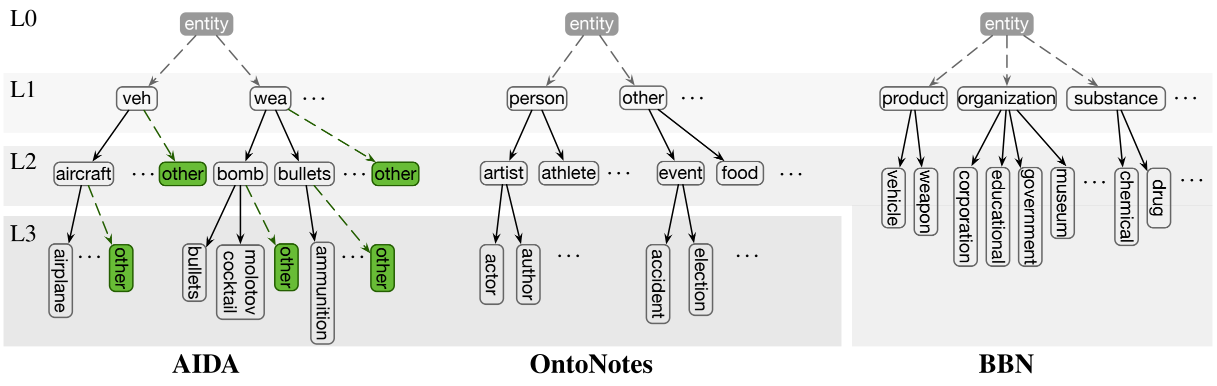

Figure 2: Various type ontologies. Different levels of the types are shown in different shades, from L0 to L3. The ENTITY and OTHER special nodes are discussed in section 3.

Coupled with a novel coarse-to-fine decoder that searches on the type hierarchy, our approach guarantees that predictions do not violate the hierarchical property, and achieves state-of-the-art results according to multiple measures across various commonly used datasets.

2 Related Work

FET is usually studied as allowing for sentence-level context in making predictions, notably starting with Ling and Weld (2012) and Gillick et al. (2014), where they created the commonly used FIGER and OntoNotes datasets for FET. While researchers have considered the benefits of document-level (Zhang et al., 2018), and corpus-level (Yaghoobzadeh and Schütze, 2015) context, here we focus on the sentence-level variant for best contrast to prior work.

Progress in FET has focused primarily on:

- **Better mention representations:** Starting from sparse hand-crafted binary features (Ling and Weld, 2012; Gillick et al., 2014), the community has moved to distributed representations (Yogatama et al., 2015), to pre-trained word embeddings with LSTMs (Ren et al., 2016a,b; Shimaoka et al., 2016; Abhishek et al., 2017; Shimaoka et al., 2017) or CNNs (Murty et al., 2018), with mention-to-context attention (Zhang et al., 2018), then to employing pre-trained language models like ELMo (Peters et al., 2018) to generate ever better representations (Lin and Ji, 2019). Our approach builds upon these developments and uses state-of-the-art mention encoders.

- **Incorporating the hierarchy:** Most prior works approach the hierarchical typing problem as *multi-label classification*, without using information in the hierarchical structure, but there are a few exceptions. Ren et al. (2016a) proposed an adaptive margin for learning-to-rank so that similar types have a smaller margin; Xu and Barbosa (2018) proposed hierarchical loss normalization that penalizes output that violates the hierarchical property; and Murty et al. (2018) proposed to learn a *subtyping* relation to constrain the type embeddings in the type space. In contrast to these approaches, our coarse-to-fine decoding approach strictly guarantees that the output does not violate the hierarchical property, leading to better performance. HYENA (Yosef et al., 2012) applied ranking to sibling types in a type hierarchy, but the number of predicted positive types are trained separately with a meta-model, hence does not support neural end-to-end training.

Researchers have proposed alternative FET formulations whose types are not formed in a type hierarchy, in particular Ultra-fine entity typing (Choi et al., 2018; Xiong et al., 2019; Onoe and Durrett, 2019), with a very large set of types derived from phrases mined from a corpus. FET in KB (Jin et al., 2019) labels mentions to types in a knowledge base with multiple relations, forming a type graph. Dai et al. (2019) augments the task with entity linking to KBs.

3 Problem Formulation

We denote a mention as a tuple $x = (w, l, r)$, where $w = (w_1, \cdots, w_n)$ is the sentential context and the span $[l : r]$ marks a mention of interest in sentence w. That is, the mention of interest is $(w_l, \cdots, w_r)$. Given x, a hierarchical entity typing model outputs a set of types Y in the type ontology $\mathcal{Y}$, i.e. $Y \subseteq \mathcal{Y}$.

Type hierarchies take the form of a forest, where each tree is rooted by a top-level supertype (e.g. `/person`, `/location`, etc.). We add a dummy parent node `ENTITY` = "/", the supertype of all entity types, to all the top-level types, effectively transforming a type forest to a type tree. In Figure 2, we show 3 type ontologies associated with 3 different datasets (see subsection 5.1), with the dummy `ENTITY` node augmented.

We now introduce some notation for referring to aspects of a type tree. The binary relation "type z is a subtype of y" is denoted as $z <: y$.[2] The unique parent of a type y in the type tree is denoted $\bar{y} \in \mathcal{Y}$, where $\bar{y}$ is undefined for $y = \texttt{ENTITY}$. The immediate subtypes of y (children nodes) are denoted $\mathrm{Ch}(y) \subseteq \mathcal{Y}$. Siblings of y, those sharing the same immediate parent, are denoted $\mathrm{Sb}(y) \subseteq \mathcal{Y}$, where $y \notin \mathrm{Sb}(y)$.

In the AIDA FET ontology (see Figure 2), the maximum depth of the tree is $L = 3$, and each mention can only be typed with at most 1 type from each level. We term this scenario ***single-path*** typing, since there can be only 1 path starting from the root (`ENTITY`) of the type tree. This is in contrast ***multi-path*** typing, such as in the BBN dataset, where mentions may be labeled with multiple types on the same level of the tree.

Additionally, in AIDA, there are mentions labeled such as as `/per/police/<unspecified>`. In FIGER, we find instances with labeled type `/person` but not any further subtype. What does it mean when a mention x is labeled with a ***partial type path***, i.e., a type y but none of the subtypes $z <: y$? We consider two interpretations:

- **Exclusive:** x is of type y, but x is not of any type $z <: y$.
- **Undefined:** x is of type y, but whether it is an instance of some $z <: y$ is unknown.

We devise different strategies to deal with these two conditions. Under the ***exclusive*** case, we add a dummy `OTHER` node to every intermediate branch node in the type tree. For any mention x labeled with type y but none of the subtypes $z <: y$, we add this additional label "y/`OTHER`" to the labels of x (see Figure 2: AIDA). For example, if we interpret a partial type path `/person` in FIGER as *exclusive*, we add another type `/person/OTHER` to that instance. Under the ***undefined*** case, we do not modify the labels in the dataset. We will see this can make a significant difference depending on the way a specific dataset is annotated.

[2] Per programming language literature, e.g. the type system $F_{<:}$ that supports subtyping.

4 Model

4.1 Mention Representation

Hidden representations for entity mentions in sentence w are generated by leveraging recent advances in language model pre-training, e.g. ELMo (Peters et al., 2018).[3] The ELMo representation for each token w_i is denoted as $\mathbf{w}_i \in \mathbb{R}^{d_w}$. Dropout is applied with probability p_{D} to the ELMo vectors.

Our mention encoder largely follows Lin and Ji (2019). First a mention representation is derived using the representations of the words in the mention. We apply a max pooling layer atop the mention after a linear transformation:[4]

$$\mathbf{m} = \mathrm{MaxPool}(\mathbf{T}\mathbf{w}_l, \cdots, \mathbf{T}\mathbf{w}_r) \in \mathbb{R}^{d_w} . \quad (1)$$

Then we employ mention-to-context attention first described in Zhang et al. (2018) and later employed by Lin and Ji (2019): a context vector $\mathbf{c}$ is generated by attending the sentence with a query vector derived from the mention vector $\mathbf{m}$. We use the multiplicative attention of Luong et al. (2015):

$$a_i \propto \exp(\mathbf{m}^{\mathrm{T}}\mathbf{Q}\mathbf{w}_i) \quad (2)$$

$$\mathbf{c} = \sum_{i=1}^{N} a_i \mathbf{w}_i \in \mathbb{R}^{d_w} \quad (3)$$

The final representation for an entity mention is generated via concatenation of the mention and context vector: $[\mathbf{m}\ ;\ \mathbf{c}] \in \mathbb{R}^{2d_w}$.

4.2 Type Scorer

We learn a type embedding $\mathbf{y} \in \mathbb{R}^{d_t}$ for each type $y \in \mathcal{Y}$. To score an instance with representation $[\mathbf{m}\ ;\ \mathbf{c}]$, we pass it through a 2-layer feed-forward network that maps into the same space as the type space $\mathbb{R}^{d_t}$, with tanh as the nonlinearity. The final

[3] Lin and Ji (2019) found that ELMo performs better than BERT (Devlin et al., 2019) for FET. Our internal experiments also confirm this finding. We hypothesize that this is due to the richer character-level information contained in lower-level ELMo representations that are useful for FET.

[4] Lin and Ji (2019) proposed an attentive pooler with a learned global query vector. We found out that a simple max pooling layer achieves similar performance.

score is an inner product between the transformed feature vector and the type embedding:

$$F(x,y) = \mathrm{FFNN}([\mathbf{m}\,;\,\mathbf{c}]) \cdot \mathbf{y}\,. \tag{4}$$

4.3 Hierarchical Learning-to-Rank

We introduce our novel hierarchical learning-to-rank loss that (1) allows for natural multi-label classification and (2) takes the hierarchical ontology into account.

We start with a multi-class hinge loss that ranks positive types above negative types (Weston and Watkins, 1999):

$$J_{\text{flat}}(x,Y) = \sum_{y\in Y}\sum_{y'\notin Y}[\xi - F(x,y) + F(x,y')]_+ \tag{5}$$

where $[x]_+ = \max\{0,x\}$. This is actually learning-to-rank with a ranking SVM (Joachims, 2002): the model learns to rank the positive types $y \in Y$ higher than those negative types $y' \notin Y$, by imposing a margin ξ between y and y': type y should rank higher than y' by ξ. Note that in Equation 5, since it is a linear SVM, the margin hyperparameter ξ could be just set as 1 (the type embeddings are linearly scalable), and we rely on L_2 regularization to constrain the type embeddings.

Multi-level Margins However, this method considers all candidate types to be *flat* instead of hierarchical — all types are given the same treatment without any prior on their relative position in the type hierarchy. Intuitively, coarser types (higher in the hierarchy) should be easier to determine (e.g. `/person` vs `/location` should be fairly easy for the model), but fine-grained types (e.g. `/person/artist/singer`) are harder.

We encode this intuition by **(i)** learning to rank types *only* on the same level in the type tree; **(ii)** setting different margin parameters for the ranking model with respect to different levels:

$$\sum_{y\in Y}\sum_{y'\in \mathrm{Sb}(y)\setminus Y}[\xi_{\mathrm{lev}(y)} - F(x,y) + F(x,y')]_+ \tag{6}$$

Here $\mathrm{lev}(y)$ is the level of the type y: for example, $\mathrm{lev}(\texttt{/location}) = 1$, and $\mathrm{lev}(\texttt{/person/artist/singer}) = 3$. In Equation 6, each positive type y is only compared against its negative siblings $\mathrm{Sb}(y)\setminus Y$, and the margin hyperparameter is set to be $\xi_{\mathrm{lev}(y)}$, i.e., a margin dependent on which level y is in the tree. Intuitively, we should set $\xi_1 > \xi_2 > \xi_3$ since our model should be able to learn a larger margin between easier pairs: we show that this is superior than using a single margin in our experiments.

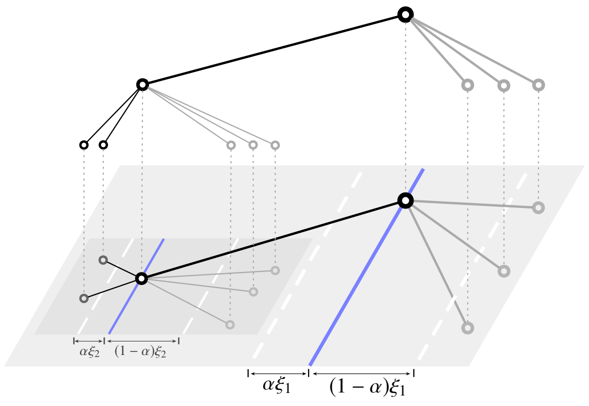

Figure 3: Hierarchical learning-to-rank. Positive type paths are colored black, negative type paths are colored gray. Each blue line corresponds to a threshold derived from a parent node. Positive types (on the left) are ranked above negative types (on the right).

Analogous to the reasoning that in Equation 5 the margin ξ can just be 1, only the relative ratios between ξ's are important. For simplicity,[5] if the ontology has L levels, we assign

$$\xi_l = L - l + 1\,. \tag{7}$$

For example, given an ontology with 3 levels, the margins per level are $(\xi_1, \xi_2, \xi_3) = (3, 2, 1)$.

Flexible Threshold Equation 6 only ranks positive types higher than negative types so that all children types given a parent type are ranked based on their relevance to the entity mention. What should be the threshold between positive and negative types? We could set the threshold to be 0 (approaching the multi-label classification problem as a set of binary classification problem, see Lin and Ji (2019)), or tune an adaptive, type-specific threshold for each parent type (Zhang et al., 2018). Here, we propose a simpler method.

We propose to directly use *the parent node as the threshold*. If a positive type is y, we learn the following ranking relation:

$$y \succ \bar{y} \succ y', \quad \forall y' \in \mathrm{Sb}(t) \tag{8}$$

where $\succ$ means "ranks higher than". For example, a mention has gold type `/person/artist/singer`. Since the

[5] We did hyperparameter search on these margin hyperparameters and found that Equation 7 generalized well.

parent type `/person/artist` can be considered as a kind of *prior* for all types of artists, the model should learn that the positive type "singer" should have a higher confidence than "artist", and in turn, higher than other types of artists like "author" or "actor". Hence the ranker should learn that "a positive subtype should rank higher than its parent, and its parent should rank higher than its negative children." Under this formulation, at decoding time, given parent type y, a child subtype $z <: y$ that scores higher than y should be output as a positive label.

We translate the ranking relation in Equation 8 into a ranking loss that extends Equation 6. In Equation 6, there is an expected margin ξ between positive types and negative types. Since we inserted the parent in the middle, we divide the margin ξ into $\alpha\xi$ and $(1-\alpha)\xi$: $\alpha\xi$ being the margin between positive types and the parent; and $(1-\alpha)\xi$ is the margin between the parent and the negative types. For a visualization see Figure 3.

The hyperparameter $\alpha \in [0, 1]$ can be used to tune the precision-recall tradeoff when outputting types: the smaller α, the smaller the expected margin there is between positive types and the parent. This intuitively increases precision but decreases recall (only very confident types can be output). Vice versa, increasing α decreases precision but increase recall.

Therefore we learn 3 sets of ranking relations from Equation 8: (i) positive types should be scored above parent by $\alpha\xi$; (ii) parent should be scored above any negative sibling types by $(1-\alpha)\xi$; (iii) positive types should be scored above negative sibling types by ξ. Our final hierarchical ranking loss is formulated as follows.

$$J_{y \succ \bar{y}} = [\alpha\xi_{\mathrm{lev}(y)} - F(x,y) + F(x,\bar{y})]_+$$

$$J_{\bar{y} \succ y'} = \sum_{y' \in \mathrm{Sb}(y) \setminus Y} [(1-\alpha)\xi_{\mathrm{lev}(y)} - F(x,\bar{y}) + F(x,y')]_+$$

$$J_{y \succ y'} = \sum_{y' \in \mathrm{Sb}(y) \setminus Y} [\xi_{\mathrm{lev}(y)} - F(x,y) + F(x,y')]_+$$

$$J_{\mathrm{hier}}(x,Y) = \sum_{y \in Y} \left(J_{y \succ \bar{y}} + J_{\bar{y} \succ y'} + J_{y \succ y'}\right) \qquad (9)$$

4.4 Decoding

Predicting the types for each entity mention can be performed via iterative searching on the type tree, from the root `ENTITY` node to coarser types, then to finer-grained types. This ensures that our output does not violate the hierarchical property, i.e., if a subtype is output, its parent must be output.

Algorithm 1 Decoding for Hierarchical Typing

```
 1: function HIERTYPEDEC($F(x,\cdot)$)
 2:     $Q \leftarrow$ {ENTITY}                          ▷ queue for searching
 3:     $\hat{Y} \leftarrow \varnothing$                             ▷ set of output types
 4:     repeat
 5:         $y \leftarrow$ DEQUEUE($Q$)
 6:         $\theta \leftarrow F(x,y) + \delta_{\mathrm{lev}(y)}$    ▷ threshold value
 7:         $Z \leftarrow \{z \in \mathrm{Ch}(y) \mid F(x,z) > \theta\}$
                                        ▷ all decoded children types
 8:         $Z' \leftarrow$ TOPK($Z, k_{\mathrm{lev}(y)+1}, F(x,\cdot)$)
                         ▷ pruned by the max branching factors
 9:         $\hat{Y} \leftarrow \hat{Y} \cup Z'$
10:         for $z \in Z'$ do
11:             ENQUEUE($Q, z$)
12:         end for
13:     until $Q = \varnothing$                           ▷ queue is empty
14:     return $\hat{Y}$                         ▷ return all decoded types
15: end function
```

Given instance x we compute the score $F(x,y)$ for each type $y \in \mathcal{Y}$, the searching process starts with the root node `ENTITY` of the type tree in the queue. For each type y in the node, a child node $z <: y$ (subtypes) is added to the predicted type set if $F(x,z) > F(x,y)$, corresponding to the ranking relation in Equation 8 that the model has learned.[6]

Here we only take the top-k element to add to the queue to prevent from over-generating types. This can also be used to enforce the single-path property (setting $k = 1$) if the dataset is single-path. For each level i in the type hierarchy, we limit the branching factor (allowed children) to be k_i. The algorithm is listed in Algorithm 1, where the function TOPK(S, k, f) selects the top-k elements from S with respect to the function f.

4.5 Subtyping Relation Constraint

Each type $y \in \mathcal{Y}$ in the ontology is assigned a type embedding $\mathbf{y} \in \mathbb{R}^{d_t}$. We notice the binary subtyping relation "$<:$" $\subseteq \mathcal{Y} \times \mathcal{Y}$ on the types. Trouillon et al. (2016) proposed the relation embedding method ComplEx that works well with anti-symmetric and transitive relations such as subtyping. It has been employed in FET before

[6] For the OntoNotes dataset, we introduce another set of per-level hyperparameters $\delta_{\mathrm{lev}(y)}$, and the threshold value $F(x,y)$ is modified to $F(x,y) + \delta_{\mathrm{lev}(y)}$, akin to the adaptive threshold in Zhang et al. (2018). This is due to a large type distribution mismatch between the training and dev/test sets in OntoNotes (in dev/test there are a lot of instances with the single type `/other` but not in the training set). For other datasets they are unused, i.e. just 0.

— in Murty et al. (2018), ComplEx is added to the loss to regulate the type embeddings. ComplEx operates in the complex space — we use the natural isomorphism between real and complex spaces to map the type embedding into complex space (first half of the embedding vector as the real part, and the second half as the imaginary part):

$$\phi : \mathbb{R}^{d_t} \to \mathbb{C}^{d_t/2} \tag{10}$$

$$\mathbf{t} = [\ \mathrm{Re}\,\phi(\mathbf{t})\ ;\ \ \mathrm{Im}\,\phi(\mathbf{t})\ ] \tag{11}$$

We learn a single relation embedding $\mathbf{r} \in \mathbb{C}^{d_t/2}$ for the subtyping relation. Given type y and z, the subtyping statement $y <: z$ is modeled using the following scoring function:

$$r(y,z) = \mathrm{Re}\left(\mathbf{r} \cdot \left(\phi(\mathbf{y}) \odot \overline{\phi(\mathbf{z})}\right)\right) \tag{12}$$

where $\odot$ is element-wise product and $\overline{x}$ is the complex conjugate of x. If $y <: z$ then $r(y,z) > 0$; and vice versa, $r(y,z) < 0$ if $y \not<: z$.

Loss Given instance (x, Y), for each positive type $y \in Y$, we learn the following relations:

$$\begin{aligned} & y <: \bar{y} \\ & y \not<: y', \quad \forall y' \in \mathrm{Sb}(y) \\ & y \not<: y', \quad \forall y' \in \mathrm{Sb}(\bar{y}) \end{aligned} \tag{13}$$

Translating these relation constraints as a binary classification problem (”is or is not a subtype”) under a primal SVM, we get a hinge loss:

$$J_{\mathrm{rel}}(x, Y) = \sum_{y \in Y} \left([1 - r(y, \bar{y})]_+ + \sum_{y' \in \mathrm{Sb}(y) \cup \mathrm{Sb}(\bar{y})} [1 + r(y, y')]_+ \right). \tag{14}$$

This is different from Murty et al. (2018), where a binary cross-entropy loss on randomly sampled (y, y') pairs is used. Our experiments showed that the loss in Equation 14 performs better than the cross-entropy version, due to the structure of the training pairs: we use siblings and siblings of parents as negative samples (these are types closer to the positive parent type), hence are training with more competitive negative samples.

4.6 Training and Validation

Our final loss is a combination of the hierarchical ranking loss and the subtyping relation constraint loss, with L_2 regularization:

$$J_{\mathrm{hier}}(x, Y) + \beta J_{\mathrm{rel}}(x, Y) + \frac{\lambda}{2} \|\mathbf{\Theta}\|_2^2 . \tag{15}$$

The AdamW optimizer (Loshchilov and Hutter, 2019) is used to train the model, as it is shown to be superior than the original Adam under L_2 regularization. Hyperparameters α (ratio of margin above/below threshold), β (weight of subtyping relation constraint), and λ (L_2 regularization coefficient) are tuned.

At validation time, we tune the maximum branching factors for each level $k_1, \cdots, k_L$.[7] These parameters tune the trade-off between the precision and recall for each layer and prevents over-generation (as we observed in some cases). All hyperparameters are tuned so that models achieve maximum micro F_1 scores (see subsection 5.4).

5 Experiments

5.1 Datasets

AIDA The AIDA Phase 1 practice dataset for hierarchical entity typing comprises of 297 documents from `LDC2019E04` / `LDC2019E07`, and the evaluation dataset is from `LDC2019E42` / `LDC2019E77`. We take only the English part of the data, and use the practice dataset as train/dev, and the evaluation dataset as test. The practice dataset comprises of 3 domains, labeled as `R103`, `R105`, and `R107`. Since the evaluation dataset is out-of-domain, we use the smallest domain `R105` as dev, and the remaining `R103` and `R107` as train.

The AIDA entity dataset has a 3-level ontology, termed *type*, *subtype*, and *subsubtype*. A mention can only have one label for each level, hence the dataset is *single-path*, thus the branching factors (k_1, k_2, k_3) for the three layers are set to $(1, 1, 1)$.

BBN Weischedel and Brunstein (2005) labeled a portion of the one million word Penn Treebank corpus of Wall Street Journal texts (`LDC95T7`) using a two-level hierarchy, resulting in the BBN Pronoun Coreference and Entity Type Corpus. We follow the train/test split by Ren et al. (2016b), and follow the train/dev split by Zhang et al. (2018).

OntoNotes Gillick et al. (2014) sampled sentences from the OntoNotes corpus and annotated the entities using 89 types. We follow the train/dev/test data split by Shimaoka et al. (2017).

[7] For the OntoNotes dataset, this also includes the per-level threshold $\delta_{\mathrm{lev}(k)}$.

Dataset	Train	Dev	Test	# Levels	# Types	Multi-path?	α	β	λ	p_D	$k_{1,\cdots,L}$
AIDA	2,492	558	1,383	3	187	single-path	0.1	0.3	0.1	0.5	(1,1,1)
BBN	84,078	2,000	13,766	2	56	multi-path	0.2	0.1	0.003	0.5	(2,1)
OntoNotes	251,039	2,202	8,963	3	89	multi-path	0.15	0.1	0.001	0.5	(2,1,1)
FIGER	2,000,000	10,000	563	2	113	multi-path	0.2	0.1	0.0001	0.5	(2,1)

Table 1: Statistics of various datasets and their corresponding hyperparameter settings.

FIGER Ling and Weld (2012) sampled a dataset from Wikipdia articles and news reports. Entity mentions in these texts are mapped to a 113-type ontology derived from Freebase (Bollacker et al., 2008). Again, we follow the data split by Shimaoka et al. (2017).

The statistics of these datasets and their accompanying ontologies are listed in Table 1, together with their respective hyperparameters.[8]

5.2 Setup

To best compare to recent prior work, we follow Lin and Ji (2019) where the ELMo encodings of words are fixed and not updated. We use all 3 layers of ELMo output, so the initial embedding has dimension $d_w = 3072$. We set the type embedding dimensionality to be $d_t = 1024$. The initial learning rate is 10^{-5} and the batch size is 256.

Hyperparameter choices are tuned on dev sets, and are listed in Table 1. We employ early stopping: choosing the model that yields the best micro F_1 score on dev sets.

Our models are implemented using AllenNLP (Gardner et al., 2018), with implementation for subtyping relation constraints from OpenKE (Han et al., 2018).

5.3 Baselines

We compare our approach to major prior work in FET that are capable of *multi-path* entity typing.[9] For AIDA, since there are no prior work on this dataset to our knowledge, we also implemented multi-label classification as set of binary classifier models (similar to Lin and Ji (2019)) as a baseline, with our mention feature extractor. The results are shown in Table 2 as "Multi-label".

[8] The OntoNotes dataset has an additional set of hyperparameters, i.e. the per-level threshold $\delta_{1,2,3} = (2.5, 3.0, 0.0)$.

[9] Zhang et al. (2018) included document-level information in their best results—for fair comparison, we used their results without document context, as are reported in their ablation tests.

5.4 Metrics

We follow prior work and use strict accuracy (Acc), macro F_1 (MaF), and micro F_1 (MiF) scores. Given instance x_i, we denote the gold type set as Y_i and the predicted type set $\hat{Y}_i$. The strict accuracy is the ratio of instances where $Y_i = \hat{Y}_i$. Macro F_1 is the average of all F_1 scores between Y_i and $\hat{Y}_i$ for all instances, whereas micro F_1 counts total true positives, false negatives and false positives globally.

We also investigate per-level accuracies on AIDA. The accuracy on level l is the ratio of instances whose predicted type set and gold type set are identical at level l. If there is no type output at level l, we append with `OTHER` to create a dummy type at level l: e.g. `/person/OTHER/OTHER`. Hence accuracy of the last level (in AIDA, level 3) is equal to the strict accuracy.

5.5 Results and Discussions

All our results are run under the two conditions regarding partial type paths: exclusive or undefined. The result of the AIDA dataset is shown in Table 2. Our model under the exclusive case outperforms a multi-label classification baseline over all metrics.

Of the 187 types specified in the AIDA ontology, the train/dev set only covers 93 types. The test set covers 85 types, of which 63 are seen types. We could perform zero-shot entity typing by initializing a type's embedding using the type name (e.g. `/fac/structure/plaza`) together with its description (e.g. "*An open urban public space, such as a city square*") as is designated in the data annotation manual. We leave this as future work.

Approach	L1	L2	L3	MaF	MiF
Ours (exclusive)	**81.6**	43.1	**32.0**	**60.6**	**60.0**
Ours (undefined)	80.0	**43.3**	30.2	59.3	58.0
– Subtyping constraints	80.3	40.9	29.9	59.1	58.3
– Multi-level margins	76.9	40.2	29.8	57.4	56.9
Multi-label	80.5	42.1	30.7	59.7	57.9

Table 2: Results on the AIDA dataset.

Approach	**BBN**			**OntoNotes**			**FIGER**		
	Acc	MaF	MiF	Acc	MaF	MiF	Acc	MaF	MiF
Ling and Weld (2012)	46.7	67.2	61.2		– [†]		52.3	69.9	69.3
Ren et al. (2016b)	49.4	68.8	64.5	51.6	67.4	62.4	49.4	68.8	64.5
Ren et al. (2016a)	67.0	72.7	73.5	55.1	71.1	64.7	53.3	69.3	66.4
Abhishek et al. (2017)	60.4	74.1	75.7	52.2	68.5	63.3	59.0	78.0	74.9
Shimaoka et al. (2017)		– [†]		51.7	71.0	64.9	59.7	79.0	75.4
Murty et al. (2018)		– [†]			– [†]		59.7	78.3	75.4
Zhang et al. (2018)	58.1	75.7	75.1	53.2	72.1	66.5	*60.2*[‡]	*78.7*[‡]	*75.5*[‡]
Lin and Ji (2019)	55.9	79.3	78.1	*63.8*[*]	*82.9*[*]	*77.3*[*]	62.9	**83.0**	79.8
Ours (exclusive)	48.2	63.2	61.0	58.3	72.4	67.2	**69.1**	82.6	**80.8**
Ours (undefined)	**75.2**	**79.7**	**80.5**	**58.7**	**73.0**	**68.1**	65.5	80.5	78.1
– Subtyping constraint	73.2	77.8	78.4	58.3	72.2	67.1	65.4	81.4	79.2
– Multi-level margins	68.9	73.2	74.2	58.5	71.7	66.0	68.1	80.4	78.0

[†]: Not run on the specific dataset; [*]: Not strictly comparable due to non-standard, much larger training set;
[‡]: Result has document-level context information, hence not comparable.

Table 3: Results of common FET datasets: BBN, OntoNotes, and FIGER. Numbers in italic are results obtained with various augmentation techniques, either larger data or larger context, hence not directly comparable.

Results for the BBN, OntoNotes, and FIGER can be found in Table 3. Across 3 datasets, our method produces the state-of-the-art performance on strict accuracy and micro F_1 scores, and state-of-the-art or comparable (±0.5%) performance on macro F_1 score, as compared to prior models, e.g. (Lin and Ji, 2019). Especially, our method improves upon the strict accuracy substantially (4%–8%) across these datasets, showing our decoder are better at outputting exact correct type sets.

Partial type paths: exclusive or undefined? Interestingly, we found that for AIDA and FIGER, partial type paths should be better considered as *exclusive*, whereas for BBN and OntoNotes, considering them as *undefined* leads to better performance. We hypothesize that this comes from how the data is annotatated—the annotation manual may contain directives as whether to interpret partial type paths as exclusive or undefined, or the data may be non-exhaustively annotated, leading to undefined partial types. We advocate for careful investigation into partial type paths for future experiments and data curation.

Ablation Studies We compare our best model with various components of our model removed, to study the gain from each component. From the best of these two settings (*exclusive* and *undefined*), we report the performance of (i) removing the subtyping constraint as is described in subsection 4.5; (ii) substituting the multi-level margins in Equation 7 with a "flat" margin, i.e., margins on all levels are set to be 1. These results are shown in Table 2 and Table 3 under our best results, and they show that both multi-level margins and subtyping relation constraints offer orthogonal improvements to our models.

Error Analysis We identify common patterns of errors, coupled with typical examples:

- Confusing types: In BBN, our model outputs `/gpe/city` when the gold type is `/location/region` for "... *in shipments from the Valley of either hardware or software goods.*" These types are semantically similar, and our model failed to discriminate between these types.
- Incomplete types: In FIGER, given instance "... *multi-agency investigation headed by the U.S. Immigration and Customs Enforcement 's homeland security investigations unit*", the gold types are `/government_agency` and `/organization`, but our model failed to output `/organization`.
- Focusing on only parts of the mention: In AIDA, given instance "... *suggested they were the work of Russian special forces assassins*

out to blacken the image of Kievs pro-Western authorities", our model outputs `/org/government` whereas the gold type is `/per/militarypersonnel`. Our model focused on the "Russian special forces" part, but ignored the "assassins" part. Better mention representation is required to correct this, possibly by introducing type-aware mention representation—we leave this as future work.

6 Conclusions

We proposed **(i)** a novel multi-level learning to rank loss function that operates on a type tree, and **(ii)** an accompanying coarse-to-fine decoder to fully embrace the ontological structure of the types for hierarchical entity typing. Our approach achieved state-of-the-art performance across various datasets, and made substantial improvement (4–8%) upon strict accuracy.

Additionally, we advocate for careful investigation into *partial type paths*: their interpretation relies on how the data is annotated, and in turn, influences typing performance.

Acknowledgements

We thank our colleague Guanghui Qin and the anonymous reviewers for their insightful suggestions and comments. This research benefited from support by the JHU Human Language Technology Center of Excellence (HLTCOE), and DARPA AIDA. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA or the U.S. Government.

References

Abhishek, Ashish Anand, and Amit Awekar. 2017. 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, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 797–807.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 87–96.

Hongliang Dai, Donghong Du, Xin Li, and Yangqiu Song. 2019. 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)*, pages 6209–6214, Hong Kong, China. Association for Computational Linguistics.

Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. FINET: context-aware fine-grained named entity typing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 868–878.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *CoRR*, abs/1412.1820.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2681–2690.

Xu Han, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. 2018. OpenKE: An open toolkit for knowledge embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, pages 139–144.

Hailong Jin, Lei Hou, Juanzi Li, and Tiansi Dong. 2019. Fine-grained entity typing via hierarchical multi graph convolutional networks. 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)*, pages 4968–4977, Hong Kong, China. Association for Computational Linguistics.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 133–142.

Ying Lin and Heng Ji. 2019. 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)*, pages 6198–6203, Hong Kong, China. Association for Computational Linguistics.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, pages 94–100.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421.

Shikhar Murty, Patrick Verga, Luke Vilnis, Irena Radovanovic, and Andrew McCallum. 2018. 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, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 97–109.

Yasumasa Onoe and Greg Durrett. 2019. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2407–2417.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237.

Jonathan Raiman and Olivier Raiman. 2018. Deeptype: Multilingual entity linking by neural type system evolution. 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*, pages 5406–5413.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016a. AFET: automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1369–1378.

Xiang Ren, Wenqi He, Meng Qu, Clare R. Voss, Heng Ji, and Jiawei Han. 2016b. 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, San Francisco, CA, USA, August 13-17, 2016*, pages 1825–1834.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. An attentive neural architecture for fine-grained entity type classification. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction, AKBC@NAACL-HLT 2016, San Diego, CA, USA, June 17, 2016*, pages 69–74.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural architectures for fine-grained entity type classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 1271–1280.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 2071–2080.

Ralph Weischedel and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. *Philadelphia: Linguistic Data Consortium*.

Jason Weston and Chris Watkins. 1999. Support vector machines for multi-class pattern recognition. In *ESANN 1999, 7th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 21-23, 1999, Proceedings*, pages 219–224.

Wenhan Xiong, Jiawei Wu, Deren Lei, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2019. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 773–784.

Peng Xu and Denilson Barbosa. 2018. 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, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 16–25.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 715–725.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. 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 of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 291–296.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. HYENA: hierarchical type classification for entity names. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Posters, 8-15 December 2012, Mumbai, India*, pages 1361–1370.

Sheng Zhang, Kevin Duh, and Benjamin Van Durme. 2018. Fine-grained entity typing through increased discourse context and adaptive classification thresholds. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics, *SEM@NAACL-HLT 2018, New Orleans, Louisiana, USA, June 5-6, 2018*, pages 173–179.