Zero-shot Entity Linking with Less Data

G P Shrivatsa Bhargav[*], **Dinesh Khandelwal**[*], **Saswati Dana**[*], **Dinesh Garg**,
Pavan Kapanipathi, Salim Roukos, Alexander Gray, L Venkata Subramaniam
IBM Research AI
{gpshri27,dikhand1,sdana027,garg.dinesh}@in.ibm.com
{kapanipa@us,roukos@us,alexander.gray,lvsubram@in}.ibm.com

Abstract

Entity Linking (EL) maps an entity mention in a natural language sentence to an entity in a knowledge base (KB). The Zero-shot Entity Linking (ZEL) extends the scope of EL to unseen entities at the test time without requiring new labeled data. BLINK (Wu et al., 2020) (BERT-based) is one of the SOTA models for ZEL. Interestingly, we discovered that BLINK exhibits diminishing returns, i.e., it reaches 98% of its performance with just 1% of the training data and the remaining 99% of the data yields only a marginal increase of 2% in the performance. While this extra 2% gain makes a huge difference for downstream tasks, training BLINK on large amounts of data is very resource-intensive and impractical. In this paper, we propose a *neuro-symbolic, multi-task learning* approach to bridge this gap. Our approach boosts the BLINK's performance with much less data by exploiting an auxiliary information about *entity types*. Specifically, we train our model on two tasks simultaneously - *entity linking (primary task)* and *hierarchical entity type prediction (auxiliary task)*. The auxiliary task exploits the hierarchical structure of entity types. Our approach achieves superior performance on ZEL task with significantly less training data. On four different benchmark datasets, we show that our approach achieves significantly higher performance than SOTA models when they are trained with just 0.01%, 0.1%, or 1% of the original training data. Our code is available at https://github.com/IBM/NeSLET.

1 Introduction

Entity linking is a fundamental task in the field of Natural Language Processing (NLP) and plays an important role in numerous applications including Knowledge Base (KB) question answering, document understanding, dialogue systems, etc. Consider the sentence – "*She noticed a* **Jaguar** *speeding on the highway.*" In this sentence, the phrase **Jaguar** is called as *entity mention* and the task of mapping this mention to a real world entity from a given KB, e.g. Wikipedia, Wikidata, DBpedia, etc., is called as *entity linking (EL)*[1]. For Wikipedia KB, above mention of **Jaguar** should be mapped to **Jaguar car** and not to **Jaguar cat**.

Majority of the prior works on EL focus only on *in-domain* linking (Jiang et al., 2021; Orr et al., 2021; Yamada et al., 2016; Chisholm and Hachey, 2015) where, gold entities (ground truth entities) of the test examples are seen during training. In other words, they assume both KB and entity set are static. However, KBs evolve over time with addition of new entities and relations (Morsey et al., 2012) because they are inherently incomplete and facts change over time. This brings in a more useful and challenging scenario of *zero-shot* linking where the gold entity of test examples is unseen during training. This task is known as *zero-shot entity linking (ZEL)*[2] and has gained attention in recent times (Wu et al., 2020; Logeswaran et al., 2019; Vyas and Ballesteros, 2021; Onoe and Durrett, 2020; Gupta et al., 2017). A BERT-based model, called BLINK (Wu et al., 2020), is a state-of-the-art (SOTA) solution for the ZEL task. Recently, a BART-based model, called GENRE (Cao et al., 2021), was proposed and is claimed to outperform BLINK. While experimenting with BLINK, we observed an interesting phenomenon: *BLINK exhibits diminishing returns*. As shown in Figure 1, BLINK attains 98% of its performance with just 1% of training data and further increase in the training data from 1% to 100% yields an additional gain of just 2% in the performance. But note that

[*]Equal contribution

[1]In literature, this task is also called as Entity Disambiguation (ED) and the combined task of mention detection plus ED is called as EL. In this paper, we assume entity mentions are given to us.

[2]In practice, a small percentage of test entities do get seen during training time while working with large benchmark datasets. We call this phenomenon as *leakage*. In the computer vision field, the very same phenomenon of *leakage* is called as *Generalized Zero-Shot Learning (GZSL) (Jiang et al., 2019)*.

Findings of the Association for Computational Linguistics: NAACL 2022, pages 1681 - 1697

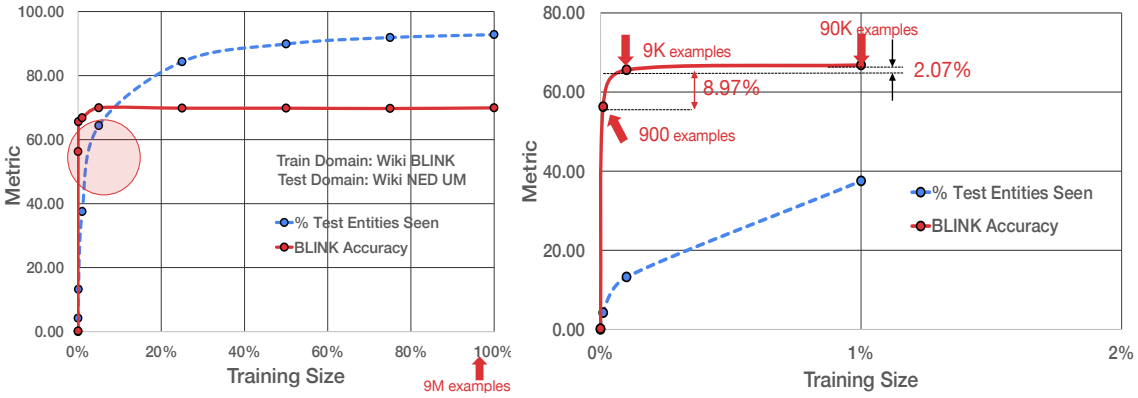

Figure 1: Figure on the right is a zoomed version of the figure on the left. The solid curve denotes diminishing returns for linking accuracy of BLINK's bi-encoder on the target domain as we increase the source domain's training data size. The dotted curve denotes the % of test entities leaking into training. A 10-fold increase in the training data (from 900 to $9K$ examples) yields a performance gain of less than 10%. A further 10-fold increase (from $9K$ to $90K$) merely yields a boost of 2.07%. As source domain training data increases, the leakage increases and problem drifts from ZEL to EL setting.

even a slight increase in entity linking performance has been shown to improve downstream tasks such as question answering significantly (Kapanipathi et al., 2021). However, training BLINK on a such a large volume of data just to offer a marginal gain of 2% is quite impractical due to heavy investment required in terms of data, time, and hardware. As per a tweet[3] from an author of the BLINK, its training took about a week's time on 8-GPUs for $9M$ examples. Based on the above observation, this paper aims to beat BLINK performance[4] while utilizing only 1% of training data.

To achieve above goal, we design a novel neuro-symbolic approach that combines an easily available *symbolic information*, called *entity types hierarchy*, with the BLINK model to reduce training data requirement without compromising ZEL performance. For example, most of the KBs would tag the entity *Mercedes-Benz* as `Company` or `Organization`. Type information can help disambiguate *Mercedes-Benz* even if it was unseen during training because mentions of similar-typed entities, say *Jaguar*, would have been seen. Our experiments empirically verify the above hypothesis. Previous works (Gupta et al., 2017; Raiman and Raiman, 2018; Onoe and Durrett, 2020) have also demonstrated performance gains by using entity types. However, their primary focus was on improving the performance, not on reducing the training data requirement, as they were not based on large language models. Even after utilizing type information, their performance is substantially lower than systems based on large language models such as BLINK and GENRE.

Furthermore, to the best of our knowledge, all of the existing systems that exploit entity types for entity linking task ignore the hierarchical structure of types. We hypothesize that exploiting the hierarchical structure of types can aid in the task of entity linking, especially when training with less data. A common strategy while training models with less training data is to apply a strong prior (arising from domain knowledge) on the parameters. The type hierarchy can be used as a prior and encoded into the model directly so that the model will not have to learn it from the very limited training data. We encode such a prior via ensuring the logical consistency (imposed by the hierarchy) of predicted entity types – *if a type is predicted, then its parent must also be predicted*. For this, we have developed a novel *neuro-symbolic* technique inspired from Gödel and Łukasiewicz norms (Klement et al., 2000).

The following are the contributions of this paper:

- We show BLINK exhibits diminishing returns (Fig. 1).
- By utilizing *diminishing returns behavior of BLINK*, we propose a solution, called *Neuro-symbolic entity Linking using Entities Type (NeSLET)*, for the zero shot entity linking problem in low training data regimes. NeSLET is a novel combination of techniques from a diverse set of fields namely multi-task learning, fuzzy

[3] https://twitter.com/riedelcastro/status/1256283045855969286

[4] In our experiments BLINK consistently outperforms GENRE by a significant margin in low data regimes.

logic, and hierarchical multi-label classification. NeSLET is trained on two tasks simultaneously - entity linking (primary task) and hierarchical entity type prediction (auxiliary task).
- We are the first ones to show that accounting for hierarchical structure of entity types improves entity linking as compared to treating it flat.
- For the scenarios where a ZEL model is given to us in the form of just black-box, we have proposed a neuro-symbolic inference algorithm that explicitly uses types just at inference time to improve the performance of given black-box.
- Our experiments on four benchmark datasets show that NeSLET beats both SOTA baselines (BLINK and GENRE) in multiple low-data regimes.

2 Related Works

Entity Linking (EL): Recently, the models (Wu et al., 2020; Cao et al., 2021) based on large language models such as BERT (Devlin et al., 2019) and BART (Lewis et al., 2020) have achieved SOTA results for the EL task. Table 1 slices the EL literature based on the flavor of underlying neural model (BERT or non-BERT) and variants of the EL task. Here, end-to-end EL means solving both mention detection and entity disambiguation tasks at the same time. BERT-based models require large training data and hence leads to poor generalizability in the case of cross-domain EL (aka ZEL) when training data is less. This motivated us to use auxiliary information about entity types. As shown in Table 1, previous works have used entity types to help the EL task. However, to the best of our knowledge, none of these approaches are designed to work in a low-data regime. Moreover, these approaches do not leverage intrinsic hierarchical structure of types and instead work with a flat hierarchy. Logeswaran et al. (2019) proposed an interesting dataset for ZEL based on the community portal (https://www.fandom.com). We, however, do not work with this dataset and related approaches (Tang et al., 2021) as it lacks entity type information.

Fine-grained Entity Typing (FET): FET (Ling and Weld, 2012; Gillick et al., 2014) is the task of assigning types from a semantic hierarchy to the entity mentions in text. FET is a *hierarchical multi-label classification (HMLC)* task and our auxiliary task is also an FET task. The popular datasets for FET, such as AIDA, BBN (Weischedel and Brunstein, 2005), OntoNotes (Gillick et al., 2014), FIGER (Ling and Weld, 2012), etc., come with shallow hierarchies (2 to 3 levels deep) compared to our DBpedia type hierarchy (7 levels deep). Ultra-Fine entity typing dataset (Choi et al., 2018) comes with around $10k$ types, but the types are not arranged in a hierarchy. There are also independent studies on the abstract HMLC problem (Giunchiglia and Lukasiewicz, 2020; Srivastava et al., 2020) without tying it to any application. The latest work on HMLC (Giunchiglia and Lukasiewicz, 2020) uses a loss function that is similar to our Gödel and Łukasiewicz t-norms (Klement et al., 2000).

Multi-task Learning (MTL): An MTL framework (Caruana, 1995) is often used to improve the performance on the primary task by learning a shared representation between the primary and one or more closely related auxiliary tasks. Learning a joint representation between related tasks helps in preventing over-fitting (Maurer, 2006), even when the amount of training data is less for each task. In our case, the primary task is *entity linking* and the auxiliary task is *entity type prediction*.

3 Problem Definition

The ZEL task is akin to a cross-domain classification task where entities play the role of classes. It has two distinct characteristics - (i) number of classes (in both source and target domain) could be in the order of millions, e.g., Wikipedia has more than 5 million entities, (ii) classes are not merely labels but have rich features in the form of short textual descriptions. Formally, in a ZEL task, we are given an entity set $\mathcal{E}_S$, called *seen* (aka *train*) entities. Each entity $e \in \mathcal{E}_S$ is seen during training in the form of a linked gold entity for some training mention m. We are also given another entity set $\mathcal{E}_U$, called *unseen* (aka *test*) entities, where $\mathcal{E}_S \cap \mathcal{E}_U = \emptyset$[5]. The element-level structure of these entity sets is as follows: $\mathcal{E}_S = \{(e_i, d_i)\}_{i=1}^{K}$ and $\mathcal{E}_U = \{(e_i, d_i)\}_{i=(K+1)}^{L}$, where e_i is the unique title of the entity and d_i is a short textual description of the entity. The train, validation, and test sets look as follows. $\mathcal{D}^{zel}_{train} = \{(m_i, e_i) \mid e_i \in \mathcal{E}_S\}_{i=1}^{N}$; $\mathcal{D}^{zel}_{val} = \{(m_j, e_j) \mid e_j \in \mathcal{E}_S\}_{j=1}^{V}$; $\mathcal{D}^{zel}_{test} = \{(m_k, e_k) \mid e_k \in \mathcal{E}_U\}_{k=1}^{M}$. In these datasets, the

[5]In an ideal ZEL task, for each of the test example, the corresponding linked gold entity comes from the set $\mathcal{E}_U$. However, in practice, for a small fraction of test examples, we have their linked gold entities coming from the set $\mathcal{E}_S$.

EL Technique		Variants of EL Task		
		In-domain	Cross-domain	End-to-end
Neural (non-BERT)	w/o types	Chisholm and Hachey Yamada et al.	Logeswaran et al.,Le and Titov, Ganea and Hofmann	Banerjee et al.
	with types	Raiman and Raiman	Gupta et al., Onoe and Durrett	
Neural (BERT)	w/o types	Orr et al.	Wu et al., Vyas and Ballesteros, Chen et al.	Li et al.
	with types	Jiang et al.	**This Paper**	

Table 1: Slicing the space of prior art on EL.

first part m_i (and m_j, m_k) corresponds to the input text string along with the entity mention substring marked. The second part e_i (and e_j, e_k) corresponds to the gold entity that must be linked to this mention. A standard practice is to represent m_i as the following tuple - *(left context, mention, right context)*. For the sentence *"She noticed a* **Jaguar** *speeding on the highway"*, we have $m_i =$ (*She noticed a*, *Jaguar*, *speeding on the highway*) and $e_i =$ `Jaguar_Cars` .

A typical model for ZEL task is a scoring function $f_{zel} : \mathcal{M} \times \mathcal{E} \mapsto \mathbb{R}$, here $\mathcal{E} = \mathcal{E}_S \cup \mathcal{E}_U$. For any given mention $m \in \mathcal{M}$, it induces a score for each entity $e \in \mathcal{E}$. The ZEL model uses these induced scores $f_{zel}(m, e)$ to rank all the entities and highest-scoring entity is predicted as the final answer. The performance of a ZEL model is measured via Hits@k for $k \geq 1$. Hits@k measures if the gold entity appears within the top-k elements. For $k = 1$, it is called *accuracy*.

3.1 Entity Type Prediction - Auxiliary Task

We use *entity type prediction (ETP)* as an auxiliary task. The goal behind ETP is to link an entity mention to one or more type classes from a given *entity type set* $\mathcal{T} = \{t_j\}_{j=1}^{|\mathcal{T}|}$. for e.g., in sentence *"She noticed a* **Jaguar** *speeding on the highway"*, the mention *Jaguar* is classified into *Organization* and *Company* classes as per DBpedia's (Lehmann et al., 2015) type classes hierarchy. Training data for such a task is given in the form of $\mathcal{D}_{train}^{type} = \{(m_i, \mathbf{t}_i)\}_{i=1}^{N}$ where $\mathbf{t}_i = [t_{ij}]_{j=1}^{|\mathcal{T}|}$ is a binary vector of size $|\mathcal{T}|$ where t_{ij} equals 1 if $t_j \in \mathcal{T}$ is a valid type for the corresponding gold entity e_i, and 0 otherwise. A typical model for the entity type prediction is given by $f_{type} : \mathcal{M} \times \mathcal{T} \mapsto [0, 1]$. For any given mention $m \in \mathcal{M}$, it induces a probability score for each type class $t_j \in \mathcal{T}$. That is, $p(t_j|m) = f_{type}(m, t_j), \ \forall t_j \in \mathcal{T}$. The performance of the ETP task is measured via F_1 score computed over predicted type set and gold type set. We make a simplifying assumption that size of gold type set is known to us; this helps us avoid hassle of setting a threshold.

4 NeSLET

As mentioned earlier, NeSLET is an MTL-based neuro-symbolic approach where we use ETP as an auxiliary task and jointly learn it with the primary task of EL. Both the primary and auxiliary tasks are classification tasks for a given mention m except that: (i) the corresponding classes are different – entities for the primary task and types for the auxiliary task, and (ii) auxiliary task is a HMLC task. In our modeling, we make a simplified assumption that the probability of entity e being the gold entity of a mention m is conditionally independent of the probability of any type t_j being the valid type of the gold entity. In other words, $p(e, \mathbf{t} \mid m) = p(e \mid m) \cdot \prod_{j=1}^{|\mathcal{T}|} p(t_j \mid m), \ \forall e, \mathbf{t}, m.$

Neural Model for $p(e \mid m)$: We use following models for $p(e \mid m)$ where, V_m and V_e are the vector representations of mention m and entity e, respectively.

$$p(e \mid m) = f_{zel}(m, e) = \frac{\exp(V_m^\top \cdot V_e)}{\sum_{e' \in \mathcal{E}} \exp(V_m^\top \cdot V_{e'})} \quad (1)$$

As shown in Figure 3, we use BERT to obtain vector representation for both mention m and the entity e (as described in Wu et al. (2020)). Parameters of these BERT models are denoted by θ_m and θ_e, respectively. Thus, we can say $V_m =$ mention-bert(m, θ_m) and $V_e =$ entity-bert(e, θ_e).

Neuro-symbolic Model for $p(t_j \mid m)$: In practice, KB's organize entity types in the form of a logical hierarchy as shown in Figure 2. In such a hierarchy, if we traverse along a path from leaf node to the root, e.g. *cat* $\rightarrow$ *mammal* $\rightarrow$ *animal*, the corresponding types become coarse grained. To ensure this logical consistency property of the type

hierarchy, we require that our proposed model satisfies the following *path monotonicity* property.
[Path Monotonicity Property]: Let t_k be the parent node of t_j in the given type hierarchy. For any mention m, we must ensure that type probabilities predicted by our model satisfy the following monotonicity condition: $0 \leq p(t_j \mid m) \leq p(t_k \mid m) \leq 1$. We ensure this by exploiting Gödel or Łukasiewicz t-norm (Klement et al., 2000) used in the field of *fuzzy logic* (Klir and Yuan, 1995). As per these t-norms, if t_j is an internal node in the hierarchy then we compute its probability purely in a symbolic fashion by using the probabilities assigned to its children nodes. We use logical OR formula (given below) of the *Gödel* (equation 2a) or *Łukasiewicz* (equation 2b) t-norm for this purpose. $\mathcal{C}_{t_j}$ in these formulas denotes the set of children nodes for t_j.

$$p(t_j|m) = \begin{cases} \max_{t_k \in \mathcal{C}_{t_j}} \; p(t_k|m) & \text{(2a)} \\ \min\left\{1, \sum_{t_k \in \mathcal{C}_{t_j}} p(t_k|m)\right\} & \text{(2b)} \end{cases}$$

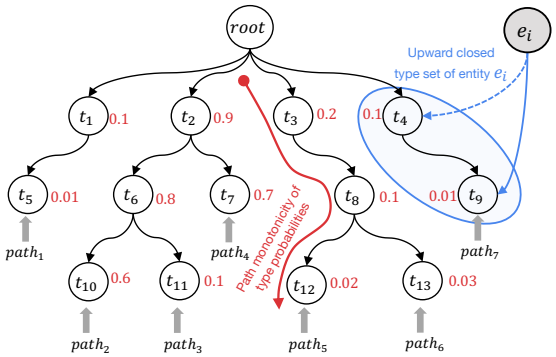

Figure 2: An illustration of type hierarchy and issue of path consistency during training.

For above model to work, we still need to address the issue of computing type probability for each leaf node t_j in the hierarchy. We achieve this via a neural model where we obtain vector representations V_{t_j} for each leaf node by using a *type network*. The *type network* comprises an initial embedding vector for each type class and an optional linear layer followed by non-linearity. The parameters of this type network are denoted by θ_t and hence, we have $V_{t_j} = \text{type-network}(t_j, \theta_t)$. Type probability of each leaf node is computed using:

$$p(t_j \mid m) = f_{type}(m, t_j) = \frac{1}{1 + \exp(-V_m^\top \cdot V_{t_j})} \quad (3)$$

Because of equations (4) and (3), we call this model as *neuro-symbolic*. While training the above model, we need to prepare our data so as to ensure a property called *upward closure of entity types*.
[Upward Closure of Entity Types]: If t_j is given to be a type of an entity e_i in a training example then all the nodes on the path from t_j till root must also be considered as its valid types. For e.g., suppose $\{t_9\}$ is given as a valid type for entity e_i as shown in Figure 2. Then, we must augment its type set by adding all the ancestor nodes of $\{t_9\}$. This results in $\{t_4, t_9\}$ as the upward closed type set for e_i.
Model Training: For training our model parameters θ_m, θ_e, and θ_t; we define loss ℓ_{zel} for the primary task and ℓ_{type} for the auxiliary task.

$$\ell_{zel}(\theta_m, \theta_e) = -\sum\nolimits_{(m_i, e_i) \in \mathcal{D}^{zel}_{train}} \log p(e_i|m_i)$$

$$\ell_{type}(\theta_m, \theta_t) = -\sum_{(m_i, e_i) \in \mathcal{D}^{zel}_{train}} \sum_{t_j \in \mathcal{T}} \mathbb{1}_{e_i}(t_j) \log p(t_j|m_i) + (1 - \mathbb{1}_{e_i}(t_j)) \log(1 - p(t_j|m_i))$$

where, $\mathbb{1}_{e_i}(t_j)$ is an indicator variable capturing whether t_j is a valid type of entity e_i or not. The loss function ℓ_{zel} is similar to the bi-encoder loss function used in BLINK (Wu et al., 2020). The combined loss for the two tasks is given by

$$\ell_{mtl}(\theta_m, \theta_e, \theta_t) = \ell_{zel}(\theta_m, \theta_e) + \alpha \, \ell_{type}(\theta_m, \theta_t)$$

where α is a hyperparameter.

Thus, learning of NeSLET involves solving the following optimization problem.

$$\theta_m^*, \theta_e^*, \theta_t^* = \underset{(\theta_m, \theta_e, \theta_t)}{\arg\min} \; \ell_{mtl}(\theta_m, \theta_e, \theta_t) \quad (4)$$

Observe, the parameter θ_m is common across both the tasks' loss terms. Due to this, both these tasks get tied together during training and the ETP task *induces a bias* in the hypothesis selection for the EL task. If there are multiple equally good EL hypotheses (i.e. model parameters θ_m and θ_e), the inductive bias forces model to pick an hypothesis that does well on the ETP task. Such inductive bias helps in better generalization for the EL task across new domain even when trained with the less data. We train EL and ETP tasks jointly using *hard parameter sharing* (Sun et al., 2020; Ruder, 2017) strategy. In this strategy, we train $\theta_m, \theta_e, \theta_t$ simultaneously. The backpropagation scheme for this scenario is depicted in Figure 3.
Inference: Given a mention m, the trained EL model is used to predict entities in a ranked order.

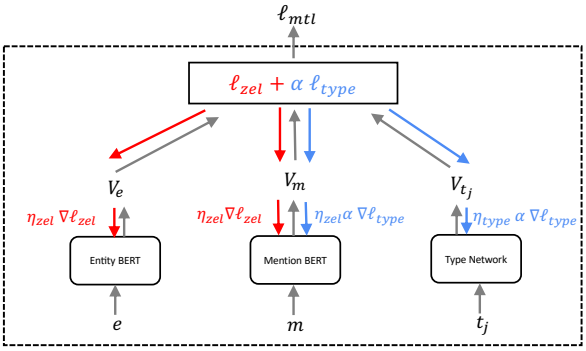

Figure 3: Training strategy with hard parameter sharing.

5 Experiments

Datasets and Type Hierarchy: Table 2 summarizes various benchmark datasets used in our experiments. We use *Wiki BLINK* (Wu et al., 2020) and *Wiki FGET* (Onoe and Durrett, 2020) as two different datasets for the training domain. Each of these datasets are based on Wikipedia. For the test domain, we use two benchmark datasets – *WikilinksNED UM* (NED for short) (Onoe, 2020; Onoe and Durrett, 2020) and *CoNLL-YAGO* (CoNLL for short) (Hoffart et al., 2011; Max Planck Institute for Informatics, 2013). For the Wiki FGET dataset, we have two variants available – one for each of the test domains. These two variants were prepared by the authors so as to ensure a good amount of entity types being covered between train and test domains where entity types were taken from Wikipedia categories. For each entity across all datasets, we use DBpedia (Lehmann et al., 2015) to get the entity types. DBpedia ontology (dbp, 2020) contains 769 types including root type *owl:Thing*. These types are arranged in a tree structure having 7 levels and 611 leaves.

Domain	Dataset	# Examples			# Unique Entities		
		Train	Val.	Test	Train	Val.	Test
Train	Wiki BLINK	9M	10K	–	1.49M	8.7K	–
	Wiki FGET (NED)	5.6M	3K	–	1.3M	2.8K	–
	Wiki FGET (CoNLL)	6M	5K	–	1.14M	4.5K	–
Test	NED	–	10K	10K	–	2.3K	2.5K
	CoNLL	–	4791	4485	–	1.6K	1.5K

Table 2: Datasets summary. For train (test) domain datasets, we have specified a dash (–) in its test (train) column as that set is never used. For the last row, the splits are as per Onoe and Durrett (2020).

Implementation Details: We used the BLINK source code from (Wu, 2020) as the base model and implemented NeSLET on top of it. The hyperparameters that we used closely follow BLINK. For both the mention and entity BERT models, we use the *bert-base-uncased* model. We use a batch size of 128, maximum context length of 64 (32 tokens on each side of the mention), and the maximum entity description length is set to 128. The learning rates are $\eta_{zel} = 10^{-5}$ and $\eta_{type} = 10^{-3}$. We use ADAM (Kingma and Ba, 2015) to optimize the objective. For BLINK and NeSLET models, we train each of them in two stages – first with *in-batch negative* entities, followed by *hard-negative* (Gillick et al., 2019) entities. We obtained the hard-negatives similar to Wu et al. (2020) by finding the top-10 predicted entities for each training example. These hard negatives are combined with the random in-batch negatives during training. We do not perform *hard negative mining* for the types because we use all the negatives types while computing ℓ_{type}. In both these stages, the number of training epochs is 30 for 0.01% and 0.1% data splits, and 4 for the 1% data splits. The best model is selected based on the source domain validation set accuracy which is computed after each epoch. We initialize the NeSLET model with the weights obtained by training BLINK on the corresponding training data splits (0.01%, 0.1% and 1%). We use BERT to compute the initial embedding for types based on their names. The type loss weight, α, for each iteration is obtained using: $\alpha = \frac{2}{1+e^{-\gamma p}} - 1$ where, γ is set to 10 and $p \in [0, 1]$ is the training progress. In the NeSLET model, we use a weighted sum (learnt) of the CLS vectors from layer #5 to #11 of the mention BERT as the *mention representation* for the purpose of type prediction.

Computing Infrastructure: We trained our models on a single machine having 2 × 20 core POWER9 processors, 6×Nvidia Volta V100 GPUs with 32GB memory, and 512GB system RAM. A single training epoch for the NeSLET model on a train set of 900 samples takes 2 to 3 minutes.

Results: Table 3 summarizes our experimental results, where we have compared performance of NeSLET with two baselines – BLINK and GENRE (Cao et al., 2021). For training and inference of GENRE, we use the hyperparameters reported in (Cao et al., 2021) (more details given in Section B.4 of the appendix). The performance

Domain		Training Data %	Method						
Train	Test		GENRE	BLINK	NeSLET-G	NeSLET-L	NeSLET-F	Gain	Hierarchy Gain
Wiki BLINK	NED	0.01	33.1	56.6	57.0	57.0	**57.7**	1.9	-1.2
		0.1	44.3	65.4	64.6	**65.6**	65.4	0.3	0.3
		1	55.0	67.7	**70.0**	69.7	69.9	3.4	0.1
	CoNLL	0.01	49.4	61.2	62.6	**64.3**	63.5	5.1	1.3
		0.1	60.2	71.6	70.5	**73.0**	70.6	2.0	3.4
		1	68.5	72.3	74.7	**75.2**	74.4	4.0	1.1
Wiki FGET	NED	0.01	22.3	47.6	52.8	52.1	**55.3**	16.2	-4.5
		0.1	36.8	62.0	**63.7**	63.6	63.0	2.7	1.1
		1	50.7	67.3	68.1	67.6	**69.2**	2.8	-1.6
	CoNLL	0.01	37.6	52.8	**60.1**	**60.1**	58.4	13.8	2.9
		0.1	53.2	68.4	**70.5**	69.8	69.6	3.1	1.3
		1	62.5	73.1	75.0	**75.1**	73.8	2.7	1.8

Table 3: Performance of NeSLET compared to GENRE and BLINK on target domain's test set. NeSLET-G, NeSLET-L, and NeSLET-F correspond to the variants of NeSLET that use Gödel (G), Łukasiewicz (L) norms to exploit the type hierarchy, or, assume that the types are Flat (F) i.e, no hierarchy. The Gain column denotes the performance gain (in %) obtained by the best of NeSLET-G and NeSLET-L and NeSLET-F relative to the best of GENRE and BLINK. The Hierarchy Gain column denotes the performance gain (in %) obtained using the hierarchical structure of entity types and is calculated as the best of NeSLET-G and NeSLET-L relative to NeSLET-F.

numbers for NeSLET in Table 3 were reported using the models (hyperparameter configurations) that resulted in the best performance on the source domain's validation set. In all of our experiments, BLINK performed better than GENRE in low training data regimes. The detailed performance numbers for validation and test sets across tuning ranges of various hyperparameters are captured in Tables 16, 17, and 18 of the appendix. These tables also capture the performance on the auxiliary task. Table 10 of the appendix shows the performance saturation trend for BLINK beyond 1% training data. We have captured the statistics related to leakage of entities and mentions in Tables 6, 7, and 8 of the appendix.

Domain		Method	
Train	Test	BLINK (100%)	NeSLET (1%)
Wiki BLINK	NED	**71.7**	70.0
	CoNLL	71.5	**75.2**
Wiki FGET	NED	67.7	**69.2**
	CoNLL	74.4	**75.1**

Table 4: Comparing the accuracy of BLINK trained on 100% data and NeSLET trained on 1% data.

Insights: In Table 3, we see that NeSLET outperforms the baselines in all twelve training and test domain combinations (2 training datasets x 2 test sets x 3 training data percentages). These results validate our claim that learning entity linking and entity type prediction in a multi-task learning fashion leads to improved performance on entity linking in low training data regimes. This characteristic is beneficial for real-world applications, where acquiring training data is quite expensive. From Table 3, we can see that the variants of NeSLET that exploit the type hierarchy using Gödel and Łukasiewicz norms (NeSLET-G and NeSLET-L) outperform NeSLET-F (which ignores the type hierarchy) in nine out of the twelve experiments. It shows that exploiting type hierarchy boosts the entity linking accuracy most times. The experiments in Table 3 also suggest that one fuzzy logic operator need not perform the best on all domains. The choice of these operators can be considered as a hyperparameter. In Table 4, we compare the accuracy of BLINK trained on 100% data with NeSLET trained on 1% data. For the purpose of this study, we consider the version of NeSLET (Gödel, Łukasiewicz or Flat) that achieves the highest accuracy on the given combination of train and test datasets. NeSLET trained on 1% data outperforms BLINK trained on 100% data in three out of the four experiments. NeSLET not only manages to recover the accuracy that BLINK loses due to

the reduction in training data size, but also goes beyond. The positive transfer of knowledge from the auxiliary task of entity type prediction to the primary task of entity linking enables NeSLET to outperform the baselines in low data regimes.

6 What if ZEL Model is a Black-Box?

Note that NeSLET model exploits entity type information only at the time of training a ZEL model (BLINK in this case). However, what if the ZEL model is available only as a black-box? For such a scenario, which is quite plausible in practice, it is not possible to train NeSLET like models that exploit entity types. To address this, we propose a neuro-symbolic inference Algorithm 1 that leverages entity type information only at inference time and thereby improves the performance of the given black-box ZEL model. Algorithm 1 assumes access to two black-box models - one for ZEL task and other for entity types prediction. Both of these models may be trained on separate datasets and made available as pre-trained black-box models. Algorithm 1 uses the types model to re-rank entities outputted by the ZEL model.

Algorithm 1: Type-based Inference

1. For a given m_{test}, pick a set $\mathcal{S}$ of top-k entities from $\mathcal{E}$ using the *zel* score $S_{zel}(e \mid m_{test}) = p(e \mid m_{test};\ \theta_m^{zel^*}, \theta_e^*)$;
2. For each entity $e \in \mathcal{S}$, compute *type based* ranking score by $S_{type}(e \mid m_{test}) = \sum_{t_j \in \mathcal{T}} \mathbb{1}_e(t_j) \cdot p(t_j \mid m_{test};\ \theta_m^{type^*}, \theta_t^*)$;
3. For each entity $e \in \mathcal{S}$, compute a *total* ranking score given by $S_{total} = (\beta_{zel} \cdot S_{zel}) + (\beta_{type} \cdot S_{type}) + (\beta_{both} \cdot S_{zel} \cdot S_{type})$ where, $0 \leq \beta_{zel}, \beta_{type}, \beta_{both}$ are weights of different terms and act as hyperparameters;
4. Output the entity with highest joint score.

When $\beta_{zel} = 1, \beta_{type} = \beta_{both} = 0$, Algorithm 1 gets simplified to the inference strategy of the ZEL model. The inference strategy of Algorithm 1 runs into a problem if types are arranged in a hierarchy. Consider two entities e_1 and e_2. After type augmentation, suppose valid type sets of e_1 and e_2 are given by the paths $root \rightarrow t_2 \rightarrow t_6$ and $root \rightarrow t_2 \rightarrow t_6 \rightarrow t_{10}$, respectively, in type hierarchy of Figure 2. In this case, the *type-based* score $S_{type}(\cdot)$ will always score e_2 higher than e_1. Thus, it has a bias in terms of preferring the entities having deeper penetration in the hierarchy. If we modify the formula $S_{type}(\cdot)$ and divide it by the *path length* then also it will be problematic as it will now favor shallow paths. To mitigate this bias, we suggest following revision in this formula: $\tilde{S}_{type}(e \mid m_{test}) = \sum_{path_j} p(path_j \mid m_{test}) \cdot p(path_j \mid e)$ where, sum is taken over all the *paths from root till leaves* in the type hierarchy (see Figure 2). For each such path, $p(path_j \mid m_{test})$ denotes predicted probability for mention's type being the leaf node on that path. The entity's path probability $p(path_j \mid e)$ depends on whether $path_j$ contains any valid type of e. Details of computing $p(path_j \mid e)$ are given in Section B.6 of the appendix.

Hierarchy Pruning: Computation of $p(path_j \mid e)$ involves estimating prior probability $p(t_b|t_a)$ of an entity having finer type t_b given it has a coarse type t_a (parent of t_b). If t_b is too fine-grained, we do not have enough data to reliably estimate $p(t_b|t_a)$. To mitigate such data sparsity issues, we consider only top-k levels of the hierarchy from the root and restrict all the calculations on these types (see Figure 2). Our experiments show strong evidence that pruning improves the performance.

Results: Table 5 shows the performance gain obtained by using type-based inference on top of the NeSLET-L model (treated as black-box). There are two hyperparameters in Algorithm 1 – top-k entities return by ZEL model, and pruning level of the type hierarchy. The values and tuning ranges of these hyperparameters are given in Table 9 of the appendix. The detailed performance numbers are given in Tables 16, 17, and 18 of the appendix. Anecdotal examples given in Tables 11 and 12 show how our inference scheme pushes the gold entity to Rank-1 which otherwise lies at Rank-2 in ZEL model's prediction. Also, in few cases, our type-based inference pushes down the correct prediction from Rank-1 position. Two such examples are given in Tables 13 and 14 of appendix.

Domain		Method	
Train	Test	NeSLET-L	NeSLET-LT
Wiki BLINK	NED	**69.65**	69.64
	CoNLL	75.24	**75.37**
Wiki FGET	NED	67.58	**67.67**
	CoNLL	75.06	**75.39**

Table 5: Accuracy boost for ZEL black-box model (NeSLET-L trained on 1% data) when applied type inference (NeSLET-LT).

7 Conclusions

We have developed a multi-task approach called NeSLET wherein one can leverage the auxiliary domain knowledge about entity types so as to improve the performance on zero-shot entity linking task, beyond what SOTA methods such as BLINK and GENRE offer, in an extremely low training data regime. We believe, this research opens up an avenue for such deep learning based methods to be tried in real applications where training examples are very less. The future directions include exploring other architectures and learning schemes to train our hard parameter sharing model. Design of newer auxiliary tasks based on self-supervision is another potential direction.

8 Ethical Considerations

Any ZEL technique, like ours, that advances the SOTA with less training data is a boon for applications such as KB question answering, document understanding, dialogue systems, etc. Our solution, however, comes with its own limitations and risks as follows. 1) Our assumption of entity types coming from the same hierarchy for both train and test domains is not always true in practice. Moreover, if there is no entity type information available in the first place, then our approach can not even be used. 2) The sensitivity of our predictions with respect to mild perturbation in input text could be a risk factor while deploying it in real-life applications. It warrants a rigorous study that we leave as future work.

We use publicly available datasets for training and testing our models. The Wiki BLINK and CoNLL-YAGO datasets are available under the MIT and CC-BY-3.0 licenses respectively. For the other datasets mentioned in Table 2, we obtained permissions from the authors (Onoe and Durrett, 2020) as the license was not explicitly mentioned on their Github. All of these datasets were constructed using publicly available sources such as Wikipedia for the purpose of developing entity linking systems and they are being used as intended. We believe these datasets do not contain any information that is offensive or uniquely identifies any individual.

Acknowledgments

We would like to thank the entire team from the IBM Research AI Hardware Center and the Center for Computational Innovation at Rensselaer Polytechnic Institute for computational resources on the AiMOS Supercomputer. We conducted most of our experiments on the AiMOS computing platform. We would also like to acknowledge IBM Cognitive Computing Cluster (CCC) for providing resources to carry out various experiments.

References

2020. DBpedia Ontology. http://akswnc7.informatik.uni-leipzig.de/dstreitmatter/archivo/dbpedia.org/ontology--DEV/2020.10.15-031000/ontology--DEV_type=parsed.owl.

Debayan Banerjee, Debanjan Chaudhuri, Mohnish Dubey, and Jens Lehmann. 2020. PNEL: Pointer network based end-to-end entity linking over knowledge graphs. In *ISWC*, pages 21–38.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *ICLR*.

Rich Caruana. 1995. Learning many related tasks at the same time with backpropagation. In *NeurIPS*, pages 657–664.

Shuang Chen, Jinpeng Wang, Feng Jiang, and Chin-Yew Lin. 2020. Improving entity linking by modeling latent entity type information. In *AAAI*, pages 7529–7537.

Andrew Chisholm and Ben Hachey. 2015. Entity disambiguation with web links. *Transactions of the Association for Computational Linguistics*, 3:145–156.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *ACL*, pages 87–96.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2009. *Introduction to Algorithms*. MIT press.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *EMNLP*, pages 2619–2629.

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

Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. 2019. Learning dense representations for entity retrieval. In *CoNLL*, pages 528–537.

E Giunchiglia and T Lukasiewicz. 2020. Coherent hierarchical multi-label classification networks. In *NeurIPS*, pages 9662–9673.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *EMNLP*, pages 2681–2690.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *EMNLP*, pages 782–792.

Hang Jiang, Sairam Gurajada, Qiuhao Lu, Sumit Neelam, Lucian Popa, Prithviraj Sen, Yunyao Li, and Alexander Gray. 2021. LNN-EL: A neuro-symbolic approach to short-text entity linking. In *ACL*, pages 775–787.

Huajie Jiang, Ruiping Wang, Shiguang Shan, and Xilin Chen. 2019. Transferable contrastive network for generalized zero-shot learning. In *ICCV*, pages 9764–9773.

Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander Gray, Ramón Fernandez Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue-Nkoutche, et al. 2021. Leveraging abstract meaning representation for knowledge base question answering. In *Findings of ACL-IJCNLP 2021*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Erich-Peter Klement, Radko Mesiar, and Endre Pap. 2000. *Triangular Norms*, volume 8 of *Trends in Logic*. Springer.

George Klir and Bo Yuan. 1995. *Fuzzy sets and fuzzy logic*, volume 4. Prentice hall New Jersey.

Phong Le and Ivan Titov. 2018. Improving entity linking by modeling latent relations between mentions. In *ACL*, pages 1595–1604.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, pages 7871–7880.

Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. Efficient one-pass end-to-end entity linking for questions. In *EMNLP*, pages 6433–6441.

Xiao Ling and Daniel S Weld. 2012. Fine-Grained entity recognition. In *AAAI*, pages 94–100.

Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. 2019. Zero-shot entity linking by reading entity descriptions. In *ACL*, pages 3449–3460.

Andreas Maurer. 2006. Bounds for linear multi-task learning. *Journal of Machine Learning Research*, 7(1):117–139.

Max Planck Institute for Informatics. 2013. AIDA CoNLL-YAGO Dataset. `https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/ambiverse-nlu/aida/downloads`.

Mohamed Morsey, Jens Lehmann, Sören Auer, Claus Stadler, and Sebastian Hellmann. 2012. Dbpedia and the live extraction of structured data from wikipedia. *Program: Electronic Library and Information Systems*, 46(2):157–181.

Yasumasa Onoe. 2020. Et4el. `https://github.com/yasumasaonoe/ET4EL`.

Yasumasa Onoe and Greg Durrett. 2020. Fine-grained entity typing for domain independent entity linking. In *AAAI*, pages 8576–8583.

Laurel Orr, Megan Leszczynski, Simran Arora, Sen Wu, Neel Guha, Xiao Ling, and Christopher Re. 2021. Bootleg: Chasing the tail with self-supervised named entity disambiguation. In *CIDR*.

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

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Santosh K. Srivastava, Dinesh Khandelwal, Dhiraj Madan, Dinesh Garg, Hima Karanam, and L. Venkata Subramaniam. 2020. Inductive quantum embedding. In *NeurIPS*.

Tianxiang Sun, Yunfan Shao, Xiaonan Li, Pengfei Liu, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2020. Learning sparse sharing architectures for multiple tasks. In *AAAI*, pages 8936–8943.

Hongyin Tang, Xingwu Sun, Beihong Jin, and Fuzheng Zhang. 2021. A bidirectional multi-paragraph reading model for zero-shot entity linking. In *AAAI*, pages 13889–13897.

Yogarshi Vyas and Miguel Ballesteros. 2021. Linking entities to unseen knowledge bases with arbitrary schemas. In *NAACL-HLT*, pages 834–844.

Ralph Weischedel and Ada Brunstein. 2005. Bbn pronoun coreference and entity type corpus. *Linguistic Data Consortium, Philadelphia*, 112.

Ledell Wu. 2020. BLINK. https://github.com/facebookresearch/BLINK.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *EMNLP*, pages 6397–6407.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *SIGNLL*, pages 250–259.

A Leakage Analysis for Entity and Mention

Tables 6, 7, and 8 show the percentage of test examples in our benchmark datasets whose gold entities (e), mentions (m), entity and mention pairs (e, m) were seen during training, respectively.

B Experiments

B.1 Type-based Inference Hyperparameters

Table 9 shows hyperparameters and their tuning ranges related to our proposed type-based inference Algorithm 1. Initially, we tuned these hyperparameters on the source domain's validation set, but that did not result in any performance gain. The reason could be a significant difference in the source and target domain's data distributions. To explore further, we tuned these hyperparameters on the target domain's validation set and obtained a boost in entity linking performance. Improving the type-based inference algorithm to obtain gains by tuning on the source domain alone is a direction of future work.

B.2 BLINK Accuracy

Table 10 shows BLINK's bi-encoder accuracy with varying level of training set size. This serves as a key baseline for our results.

B.3 Anecdotal Examples

Tables 11 and 12 show anecdotal examples where, NeSLET model predicts the gold entity at Rank-2 but, our proposed type-based inference corrects the prediction of NeSLET by promoting it to Rank-1 during inference time. Tables 13 and 14, on the other hand, show anecdotal examples where type-based inference makes the prediction of the ZEL model incorrect. However, at an aggregate level, our type-based inference overall improves ZEL performance. Here the type model comes from NeSLETwith the Łukasiewicz norm.

B.4 GENRE Experiment Details

GENRE (Cao et al., 2021) is a SOTA entity linking system based on BART (Lewis et al., 2020). Given an input mention m to GENRE, it autoregressively generates an entity name e. We initialized BART weights with pre-trained BART from (Lewis et al., 2020) and fine-tuned it on *Wiki BLINK* (Wu et al., 2020) and *Wiki FGET* (Onoe and Durrett, 2020) datasets for different training data splits ($0.01\%, 0.1\%$ and 1%). We used the hyperparameters reported in (Cao et al., 2021) and the training script available at https://github.com/facebookresearch/GENRE for training GENRE. For different training data splits, we used different numbers of training update steps as reported in Table 15. We selected the model using test domain's validation set. For inference, we used the same hyperparameters as used by (Cao et al., 2021) for the entity disambiguation task. We have constructed a trie (Cormen et al., 2009) using BLINK's $5.9M$ entities set to perform constrained decoding at the inference time.

B.5 Comparison of NeSLET vs. BLINK

Tables 16, 17, and 18 show detailed comparison of NeSLET with BLINK when both are trained with 0.01%, 0.1%, and 1% of training data splits. The ETP F_1 is the performance of the auxiliary ETP task. Different rows capture variations in train/test domain datasets as well as key hyperparameters for NeSLET. 'Flat' loss means treating the type hierarchy as flat. Column $\boldsymbol{\beta}$ corresponds to the values for $(\beta_{zel}, \beta_{type}, \beta_{both})$. The NeSLET accuracy when inferred w/o types corresponds to the case $\beta_{zel} = 1, \beta_{type} = \beta_{both} = 0$. For each row, the NeSLET performance w/ types is taken for the combination of 'Tree Level', 'top-k', and $\boldsymbol{\beta}$ that resulted in the best NeSLET performance on test domain's validation set.

Domain		% Test entities (e) seen during training						
Train	Test	Varying levels of training set						
		0.01%	0.1%	1%	5%	25%	50%	100%
Wiki BLINK	NED	4.21	13.24	37.51	64.33	84.30	89.85	92.78
	CoNLL	13.42	35.70	63.14	76.41	85.15	88.21	91.55
Wiki FGET	NED	1.04	7.60	26.26	51.88	75.24	82.48	87.09
	CoNLL	8.74	31.62	53.94	69.45	82.23	86.20	89.16

Table 6: Percentage of test examples whose gold entities (e) were seen during training time.

Domain		% Test mentions (m) seen during training						
Train	Test	Varying levels of training set						
		0.01%	0.1%	1%	5%	25%	50%	100%
Wiki BLINK	NED	5.27	15.52	44.85	72.74	90.81	94.35	96.40
	CoNLL	8.23	29.92	51.48	62.34	70.97	74.49	77.35
Wiki FGET	NED	0.06	9.47	34.11	62.84	85.48	91.09	94.26
	CoNLL	6.62	26.71	46.93	58.55	68.52	72.00	75.03

Table 7: Percentage of test examples whose mentions (m) were seen during training time.

Domain		% Test (entity, mention) pairs seen during training						
Train	Test	Varying levels of training set						
		0.01%	0.1%	1%	5%	25%	50%	100%
Wiki BLINK	NED	3.24	10.40	26.75	48.92	68.93	77.06	82.47
	CoNLL	6.44	23.79	39.42	51.17	61.38	65.71	69.41
Wiki FGET	NED	0.00	4.33	17.07	36.69	60.32	68.49	75.18
	CoNLL	4.93	19.62	35.01	46.27	57.44	61.56	65.28

Table 8: Percentage of test examples whose gold entity and mention pairs (e, m) were seen during training time.

Parameter	Symbol	Tuning Range
Parameter for type-based inference strategy	β_{zel}	$\{0, 1\}$
Parameter for type-based inference strategy	β_{type}	$\{0, 1\}$
Parameter for type-based inference strategy	β_{both}	$\{0, 1\}$
Parameter for type-based inference strategy	top-k	$\{2, 3, 5, 10, 100\}$
Pruning level of type hierarchy during inference	Tree Level	$\{1, 2, 3, 4, 5, 6, 7\}$

Table 9: List of hyperparameters and tuning range. For optimal value of the hyperparameters, please refer Tables 16, 17, and 18.

Domain		ZEL Accuracy				
Train	Test	Varying levels of training set				
		0.01%	0.1%	1%	5%	100%
Wiki BLINK	NED	56.62	65.38	67.66	68.87	71.72
	CoNLL	61.18	71.61	72.27	73.50	71.50
Wiki FGET	NED	47.62	62.00	67.34	68.59	67.74
	CoNLL	52.77	68.40	73.08	72.99	74.36

Table 10: BLINK's bi-encoder accuracy with varying level of training set size.

Mention (m)	in Marin County, California, Lucasfilm Ltd. is one of the most ... The Seattle Mariners are an American professional baseball team based in Seattle, **Washington**. Enfranchised in 1977 , the Mariners are a member of the Western Division of Major League Baseball 's American League . Safeco Field has
Gold entity (e^*)	**Washington (state)**
NeSLET's top-3 predictions with scores $S_{zel}(e \mid m_{test})$	[*Seattle (*0.54*), Washington, D.C. (*0.23*), Washington (state) (*0.22*)*]
Type set for *Seattle*	[*City, Settlement, Place, PopulatedPlace*]
Type set for *Washington (state)*	[*Settlement, AdministrativeRegion, PopulatedPlace, Place, Region*]
Our predictions $p(t_j \mid m_{test})$	[*City* (0.99), *Settlement* (0.99), *Place* (0.99), *PopulatedPlace* (0.99), *AdministrativeRegion* (0.99), *Region* (0.99)]
Type Score $\tilde{S}_{type}(e \mid m_{test})$	[*Seattle (*0.21*), Washington, D.C. (*0.21*), Washington (state) (*0.57*)*]
Type-based inference top-3 predictions with scores $S_{total}(e \mid m_{test})$	[*Washington (state) (*0.79*), Seattle (*0.75*), Washington, D.C. (*0.44*)*]

Table 11: An anecdotal example where the NeSLET model predicts the gold entity at Rank-2 position but type-based inference promotes it to Rank-1.

Mention (m)	Pink Floyd song 'Keep Talking'. Hawking's early life and the onset of his illness was the subject of the 2004 BBC4 TV film **Hawking** in which he was portrayed by Benedict Cumberbatch. In 2008, Hawking was the subject of and featured in the documentary series Stephen Hawking, Master.
Gold entity (e^*)	**Hawking (2004 film)**
NeSLETś top-3 predictions with scores $S_{zel}(e \mid m_{test})$	[*Stephen Hawking (*0.51*), Hawking (2004 film) (*0.48*), Hawking (2013 film) (*0.0042*)*]
Type set for *Stephen Hawking*	[*Scientist, Person, Agent*]
Type set for *Hawking (2004 film)*	[*TelevisionShow, Film, Work*]
Our predictions $p(t_j \mid m_{test})$	[*Scientist* (6.6×10^{-8}), *Person* (0.061), *Agent* (0.061), *TelevisionShow* (0.87), *Film* (0.99), *Work* (0.99)]
Type Score $\tilde{S}_{type}(e \mid m_{test})$	[*Stephen Hawking (*0.097*), Hawking (2004 film) (*0.63*), Hawking (2013 film) (*0.26*)*]
Type-based inference top-3 predictions with scores $S_{total}(e \mid m_{test})$	[*Hawking (2004 film) (*1.11*), Stephen Hawking (*0.60*), Hawking (2013 film) (*0.26*)*]

Table 12: One more anecdotal example where the NeSLET model predicts the gold entity at Rank-2 position but type-based inference promotes it to Rank-1.

Mention (m)	way she does. All this makes me feel this character is made up of a strongly ironic stance on Atwood's part. More ironic than in **Cat's Eye**. Yet a pattern to identify with is emerging also. Joan is growing up. She is beginning to break away from Arthur to question him.
Gold entity (e^*)	**Cat's Eye (novel)**
NeSLETś top-3 predictions with scores $S_{zel}(e \mid m_{test})$	[*Cat's Eye (novel) (*0.52*), Eye of the Cat (*0.30*), Eye of Cat (*0.16*)*]
Type set for *Cat's Eye (novel)*	[*Book, Work, WrittenWork*]
Type set for *Eye of the Cat*	[*Film, Work*]
Our predictions $p(t_j \mid m_{test})$	[*Book* (0.07), *Work* (0.89), *WrittenWork* (0.07), *Film* (0.89)]
Type Score $\tilde{S}_{type}(e \mid m_{test})$	[*Cat's Eye (novel) (*0.23*), Eye of the Cat (*0.53*), Eye of Cat (*0.23*)*]
Type-based inference top-3 predictions with scores $S_{total}(e \mid m_{test})$	[*Eye of the Cat (*0.83*), Cat's Eye (novel) (*0.75*), Eye of Cat (*0.39*)*]

Table 13: An anecdotal example where the NeSLET model predicts the gold entity at Rank-1 position but type-based inference pushes it down to Rank-2.

Mention (m)	for National Harbor to take a stroll and do some window shopping. The kids had fun on 'the man in the sand' better known as **The Awakening**. We also walked along the trail that connects to the Woodrow Wilson Bridge and leads into Alexandria - lots of bikers and walkers
Gold entity (e^*)	**The Awakening (sculpture)**
NeSLETś top-3 predictions with scores $S_{zel}(e \mid m_{test})$	[*The Awakening (sculpture) (*0.35*), Dalai Lama Awakening (*0.33*), The Great Awakening (*0.31*)*]
Type set for *The Awakening (sculpture)*	[*Artwork, Work*]
Type set for *Dalai Lama Awakening*	[*Film, Work*]
Our predictions $p(t_j \mid m_{test})$	[*Artwork* (5.2×10^{-7}), *Work* (0.99), *Film* (0.99), *Work* (0.99)]
Type Score $\tilde{S}_{type}(e \mid m_{test})$	[*The Awakening (sculpture) (*0.19*), Dalai Lama Awakening (*0.53*), The Great Awakening (*0.26*)*]
Type-based inference top-3 predictions with scores $S_{total}(e \mid m_{test})$	[*Dalai Lama Awakening (*0.86*), The Great Awakening (*0.57*), The Awakening (sculpture) (*0.54*)*]

Table 14: One more anecdotal example where the NeSLET model predicts the gold entity at Rank-1 position but type-based inference pushes it down to Rank-2.

Training Data Split	Number of Update Steps
0.01%	$1k$
0.1%	$4k$
1%	$10k$

Table 15: Number of update steps used for fine-tuning GENRE for different data splits.

Domain		ETP Loss	Tree Level	top-k	β	BLINK Acc.		NeSLET Acc. when inferred				ETP F_1	
								w/o types		w/ types			
Train	Test					Val	Test	Val	Test	Val	Test	Val	Test
Wiki BLINK	NED	Gödel	1	10	$(1,1,0)$	56.55	56.62	57.49	56.99	57.63	57.07	22.50	22.03
		Łukasiewicz	3	2	$(1,1,0)$	56.55	56.62	58.41	56.98	58.46	56.95	25.15	24.98
		Flat	3	2	$(1,1,0)$	56.55	56.62	56.25	57.69	56.66	57.72	54.35	52.66
	CoNLL	Gödel	1	3	$(1,1,0)$	61.73	61.18	62.90	62.57	63.26	63.07	14.19	13.27
		Łukasiewicz	3	3	$(1,1,0)$	61.73	61.18	63.41	64.28	63.72	64.46	27.83	25.32
		Flat	2	2	$(1,1,0)$	61.73	61.18	64.67	63.47	65.11	63.95	62.74	62.16
Wiki FGET	NED	Gödel	1	5	$(1,1,0)$	45.44	47.62	52.18	52.82	52.52	52.79	22.42	21.74
		Łukasiewicz	1	10	$(1,1,0)$	45.44	47.62	52.33	52.11	52.54	52.17	31.29	31.31
		Flat	3	3	$(1,0,1)$	45.44	47.62	55.71	55.33	55.90	55.30	47.36	47.23
	CoNLL	Gödel	1	5	$(1,1,0)$	55.79	52.77	62.00	60.10	62.23	60.14	12.79	11.61
		Łukasiewicz	1	3	$(1,1,0)$	55.79	52.77	62.06	60.08	62.27	60.03	12.79	11.63
		Flat	3	3	$(1,1,0)$	55.79	52.77	61.46	58.39	62.06	58.73	64.87	61.64

Table 16: Performance of proposed NeSLET vis-à-vis BLINK at 0.01% training dataset.

Domain		ETP Loss	Tree Level	top-k	β	BLINK Acc.		NeSLET Acc. when inferred				ETP F_1	
								w/o types		w/ types			
Train	Test					Val	Test	Val	Test	Val	Test	Val	Test
Wiki BLINK	NED	Gödel	1	3	$(1,1,0)$	66.08	65.38	64.94	64.57	65.22	64.72	57.64	57.50
		Łukasiewicz	1	2	$(1,1,0)$	66.08	65.38	66.31	65.58	66.57	65.58	46.73	46.47
		Flat	1	3	$(1,1,0)$	66.08	65.38	65.70	65.35	66.00	65.25	65.31	64.19
	CoNLL	Gödel	3	2	$(1,1,0)$	75.74	71.61	73.92	70.47	74.26	70.74	67.45	62.06
		Łukasiewicz	3	2	$(1,1,0)$	75.74	71.61	76.00	73.01	76.44	73.26	54.47	52.97
		Flat	3	3	$(1,1,0)$	75.74	71.61	74.28	70.58	74.76	71.14	77.28	74.94
Wiki FGET	NED	Gödel	1	3	$(1,1,0)$	61.14	62.00	62.88	63.72	63.11	63.88	55.92	54.33
		Łukasiewicz	1	100	$(1,1,0)$	61.14	62.00	63.18	63.61	63.20	63.59	26.56	25.92
		Flat	3	3	$(1,1,0)$	61.14	62.00	62.78	62.98	63.30	62.81	61.55	60.56
	CoNLL	Gödel	1	2	$(1,1,0)$	71.93	68.40	73.23	70.49	73.60	70.60	70.05	65.98
		Łukasiewicz	2	2	$(1,1,0)$	71.93	68.40	73.21	69.84	73.81	70.06	62.08	56.61
		Flat	1	2	$(1,1,0)$	71.93	68.40	73.44	69.59	73.81	69.64	75.72	70.77

Table 17: Performance of proposed NeSLET approach vis-à-vis BLINK at 0.1% training dataset.

Domain			ETP Loss	Tree Level	top-k	β	BLINK Acc.		NeSLET Acc. when inferred				ETP F_1	
									w/o types		w/ types			
Train		Test					Val	Test	Val	Test	Val	Test	Val	Test
Wiki BLINK		NED	Gödel	3	2	$(1,1,0)$	69.37	67.66	70.91	70.02	71.32	69.69	64.53	62.76
			Łukasiewicz	5	3	$(1,1,0)$	69.37	67.66	70.68	69.65	70.93	69.64	46.81	47.28
			Flat	1	2	$(1,1,0)$	69.37	67.66	70.85	69.91	71.10	70.09	69.38	68.60
		CoNLL	Gödel	4	2	$(1,1,0)$	76.12	72.27	78.70	74.74	79.14	75.06	73.80	71.02
			Łukasiewicz	1	3	$(1,1,0)$	76.12	72.27	78.51	75.24	78.72	75.37	56.47	53.81
			Flat	3	2	$(1,1,0)$	76.12	72.27	77.89	74.38	78.43	74.92	81.31	78.73
Wiki FGET		NED	Gödel	1	5	$(1,1,0)$	67.89	67.34	69.40	68.08	69.54	68.16	62.34	59.68
			Łukasiewicz	1	2	$(1,1,0)$	67.89	67.34	68.73	67.58	68.86	67.67	41.94	41.04
			Flat	1	2	$(1,1,0)$	67.89	67.34	70.50	69.21	70.58	69.05	66.75	66.48
		CoNLL	Gödel	3	3	$(1,1,0)$	76.88	73.08	78.18	75.03	78.47	74.90	73.45	69.78
			Łukasiewicz	2	5	$(1,1,0)$	76.88	73.08	78.41	75.06	78.72	75.39	56.27	54.22
			Flat	3	2	$(1,0,1)$	76.88	73.08	78.37	73.77	78.66	74.13	79.01	75.52

Table 18: Performance of proposed NeSLET approach vis-à-vis BLINK at 1% training dataset.

B.6 Entity Path Probability $p(path_j \mid e)$

The entity's path probability $p(path_j \mid e)$ is computed as follows.

1. $p(path_j \mid e) \leftarrow 0$ if leaf node of $path_j$ is not reachable from any valid type of e. For example, in Figure 2), we would be having $p(path_j \mid e_i) = 0,\ \forall j = 1 \rightarrow 6$.

2. $p(path_j \mid e) \leftarrow 0$ if leaf node of $path_j$ is reachable from some valid type (say t_k) of e but a direct child of t_k not lying on the $path_j$ is also a valid type for e. For e.g., in Figure 2, imagine node t_4 having one more child called t_{14} and the corresponding path being $path_8$. Then, we would be having $p(path_8 \mid e_i) = 0$.

3. For each of the remaining $path_j$, let t_k be the deepest node which is also a valid type of e. Traverse the lower portion of $path_j$ starting from node t_k all the way up to leaf node and multiply *conditional probabilities* of the edges on the way to get $p(path_j \mid e)$ as follows: $p(path_j \mid e) = \prod_{(b,a) \in path_j, b \text{ is descendant of } t_k} p(t_b \mid t_a)$. We approximate the conditional probabilities $p(t_b \mid t_a)$ with prior probabilities from the given dataset about entities and the corresponding type set.