The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

DeepType: Multilingual Entity Linking by Neural Type System Evolution

Jonathan Raiman
OpenAI
San Francisco, California
raiman@openai.com

Olivier Raiman
Agilience
Paris, France
or@agilience.com

Abstract

The wealth of structured (e.g. Wikidata) and unstructured data about the world available today presents an incredible opportunity for tomorrow's Artificial Intelligence. So far, integration of these two different modalities is a difficult process, involving many decisions concerning how best to represent the information so that it will be captured or useful, and hand-labeling large amounts of data. DeepType overcomes this challenge by explicitly integrating symbolic information into the reasoning process of a neural network with a type system. First we construct a type system, and second, we use it to constrain the outputs of a neural network to respect the symbolic structure. We achieve this by reformulating the design problem into a mixed integer problem: create a type system and subsequently train a neural network with it. In this reformulation discrete variables select which parent-child relations from an ontology are types within the type system, while continuous variables control a classifier fit to the type system. The original problem cannot be solved exactly, so we propose a 2-step algorithm: 1) heuristic search or stochastic optimization over discrete variables that define a type system informed by an Oracle and a Learnability heuristic, 2) gradient descent to fit classifier parameters. We apply DeepType to the problem of Entity Linking on three standard datasets (i.e. WikiDisamb30, CoNLL (YAGO), TAC KBP 2010) and find that it outperforms all existing solutions by a wide margin, including approaches that rely on a human-designed type system or recent deep learning-based entity embeddings, while explicitly using symbolic information lets it integrate new entities without retraining.

1 Introduction

Online encyclopaedias, knowledge bases, ontologies (e.g. Wikipedia, Wikidata, Wordnet), alongside image and video datasets with their associated label and category hierarchies (e.g. Imagenet (Deng et al. 2009), Youtube-8M (Abu-El-Haija et al. 2016), Kinetics (Kay et al. 2017)) offer an unprecedented opportunity for incorporating symbolic representations within distributed and neural representations in Artificial Intelligence systems. Several approaches exist for integrating rich symbolic structures within the behavior of neural networks: a label hierarchy aware loss function that relies on the ultrametric tree distance between labels (e.g. it is worse to confuse sheepdogs and skyscrapers than it is to confuse sheepdogs and poodles) (Wu, Tygert, and LeCun 2017), a loss function that trades off specificity for accuracy by incorporating hypo/hypernymy relations (Deng et al. 2012), using NER types to constrain the behavior of an Entity Linking system (Ling, Singh, and Weld 2015), or more recently integrating explicit type constraints within a decoder's grammar for neural semantic parsing (Krishnamurthy, Dasigi, and Gardner 2017). However, current approaches face several difficulties:

- Selection of the right symbolic information based on the utility or information gain for a target task.
- Design of the representation for symbolic information (hierarchy, grammar, constraints).
- Hand-labelling large amounts of data.

DeepType overcomes these difficulties by explicitly integrating symbolic information into the reasoning process of a neural network with a type system that is automatically designed without human effort for a target task. We achieve this by reformulating the design problem into a mixed integer problem: create a type system by selecting roots and edges from an ontology that serve as types in a type system, and subsequently train a neural network with it. The original problem cannot be solved exactly, so we propose a 2-step algorithm:

1. heuristic search or stochastic optimization over the discrete variable assignments controlling type system design, using an Oracle and a Learnability heuristic to ensure that design decisions will be easy to learn by a neural network, and will provide improvements on the target task,
2. gradient descent to fit classifier parameters to predict the behavior of the type system.

In order to validate the benefits of our approach, we focus on applying DeepType to Entity Linking (EL), the task of resolving ambiguous mentions of entities to their referent entities in a knowledge base (KB) (e.g. Wikipedia). Specifically we compare our results to state of the art systems on three standard datasets (WikiDisamb30, CoNLL (YAGO), TAC KBP 2010). We verify whether our approach can work in multiple languages, and whether optimization of the type system for a particular language generalizes to other lan-

guages[1] by training our full system in a monolingual (English) and bilingual setup (English and French), and also evaluate our Oracle (performance upper bound) on German and Spanish test datasets. We compare stochastic optimization and heuristic search to solve our mixed integer problem by comparing the final performance of systems whose type systems came from different search methodologies. We also investigate whether symbolic information is captured by using DeepType as pretraining for Named Entity Recognition (NER) on two standard datasets (i.e. CoNLL 2003 (Sang and Meulder 2003), OntoNotes 5.0 (CoNLL 2012) (Pradhan et al. 2012)).

Our key contributions in this work are as follows:

- A system for integrating symbolic knowledge into the reasoning process of a neural network through a type system, to constrain the behavior to respect the desired symbolic structure, and automatically design the type system without human effort.
- An approach to EL that uses type constraints, reduces disambiguation resolution complexity from $O(N^2)$ to $O(N)$, incorporates new entities into the system without retraining, and outperforms all existing solutions by a wide margin.

Moreover, we observe that disambiguation accuracy reaches 99.0% on CoNLL (YAGO) and 98.6% on TAC KBP 2010 when entity types are predicted by an Oracle, suggesting that EL would be almost solved if we can improve type prediction accuracy.

The rest of this paper is structured as follows. In Section 2 we introduce EL and EL with Types, in Section 3 we describe DeepType for EL, In Section 4 we provide experimental results for DeepType applied to EL and evidence of cross-lingual and cross-domain transfer of the representation learned by a DeepType system. In Section 5 we relate our work to existing approaches. Conclusions and directions for future work are given in Section 6.

2 Task

Before we define how DeepType can be used to constrain the outputs of a neural network using a type system, we will first define the goal task of Entity Linking.

Entity Linking The goal is to recover the ground truth entities in a KB referred to in a document by locating mentions (text spans), and for each mention properly disambiguating the referent entity. Commonly, a lookup table that maps each mention to a proposal set of entities for each mention m: $\mathcal{E}_m = \{e_1, \ldots, e_n\}$ (e.g. "Washington" could mean **Washington, D.C.** or **George Washington**). Disambiguation is finding for each mention m the a ground truth entity e^{GT} in $\mathcal{E}_m$. Typically, disambiguation operates according to two criteria: in a large corpus, how often does a mention point to an entity, $\mathrm{LinkCount}(m, e)$, and how often does entity e_1 co-occur with entity e_2, an $O(N^2)$ process, often named *coherence* (Milne and Witten 2008; Ferragina and Scaiella 2010; Yamada et al. 2016).

Entity Linking with Types In this work we extend the EL task to associate with each entity a series of types (e.g. `Person`, `Place`, etc.) that if known, would rule out invalid answers, and therefore ease linking (e.g. the context now enables types to disambiguate "Washington"). Knowledge of the types T associated with a mention can also help prune entities from the the proposal set, to produce a constrained set: $\mathcal{E}_{m,T} \subseteq \mathcal{E}_m$. In a probabilistic setting it is also possible to rank an entity e in document x according to its likelihood under the type system prediction and under the entity model:

$$\mathbb{P}(e|x) \propto \mathbb{P}_{\text{type}}(\text{types}(e)|x) \cdot \mathbb{P}_{\text{entity}}(e|x, \text{types}(e)). \quad (1)$$

In prior work, the 112 FIGER Types (Ling and Weld 2012) were associated with entities to combine an NER tagger with an EL system (Ling, Singh, and Weld 2015). In their work, they found that regular NER types were unhelpful, while finer grain FIGER types improved system performance.

3 DeepType for Entity Linking

DeepType is a strategy for integrating symbolic knowledge into the reasoning process of a neural network through a type system. When we apply this technique to EL, we constrain the behavior of an entity prediction model to respect the symbolic structure defined by types. As an example, when we attempt to disambiguate "Jaguar" the benefits of this approach are apparent: our decision can be based on whether the predicted type is Animal or Road Vehicle as shown visually in Figure 1.

In this section, we will first define key terminology, then explain the model and its sub-components separately.

Terminology

Given some knowledge graph or feature set, a *relation* is a set of inheritance rules that define membership or exclusion from a particular group. For instance the relation `instance of`(**city**) selects all children of the *root* **city** connected by `instance of` as members of the group, depicted by outlined boxes in Figure 2.

In this work a *type* is a label defined by a *relation* (e.g. `IsHuman` is the type applied to all children of **Human** connected by `instance of`).

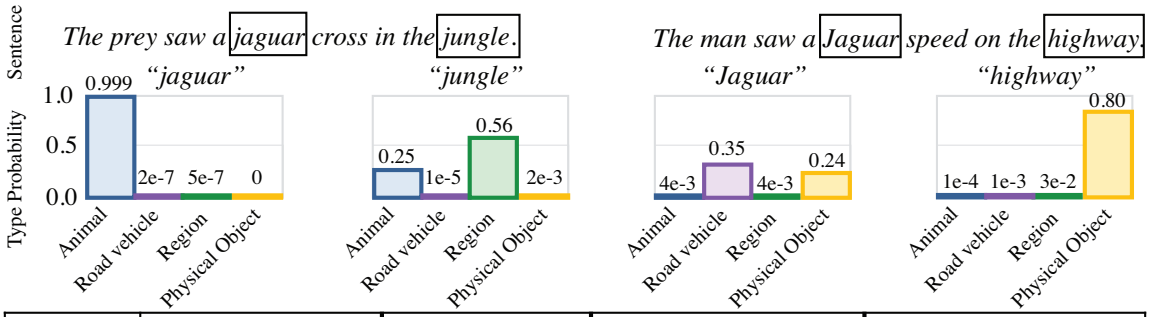
A *Type Axis* is a set of mutually exclusive types (e.g. $\texttt{IsHuman} \wedge \texttt{IsPlant} = \{\}$).

A *Type System* is a grouping of type axes, $\mathcal{A}$, along with a type labelling function: $\{t_1, \ldots, t_k\} = \mathrm{TypeLabeler}(e, \mathcal{A})$. For instance a type system with two axes $\{\texttt{IsA}, \texttt{Topic}\}$ assigns to **George Washington**: $\{\texttt{Person}, \texttt{Politics}\}$, and to **Washington, D.C.**: $\{\texttt{Place}, \texttt{Geography}\}$).

Model

To construct an EL system that uses type constraints we require: a type system, the associated type classifier, and a model for predicting and ranking entities given a mention.

[1] e.g. Do we overfit to a particular set of symbolic structures useful only in English, or can we discover a knowledge representation that works across languages?



Entity	**jaguar**	**Jaguar**	**jungle**	**jungle**	**jaguar**	**Jaguar**	**highway**	**Highway**
Type	Animal	Road vehicle	Region	Music	Animal	Road vehicle	Physical Object	Film
only link Prob.	0.29	**0.60**	**0.35**	0.17	0.29	**0.60**	**0.85**	0.04
Prob. w/. types	**1.0**	0.0	**1.0**	0.0	0.0	**1.0**	**1.0**	0.0

Figure 1: Example model output: "jaguar" refers to different entities depending on context. Predicting the type associated with each word (e.g. animal, region, etc.) helps eliminate options that do not match, and recover the true entity. Bar charts give the system's belief over the type-axis "IsA", and the table shows how types affects entity probabilities given by Wikipedia links.

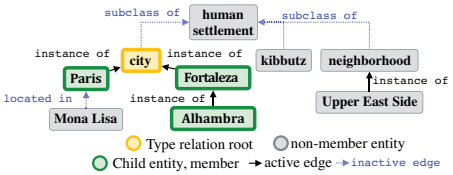


Figure 2: Defining group membership with a knowledge graph relation: children of root (city) via edge (instance of).

Instead of assuming we receive a type system, classifier, entity prediction model, we will instead create the type system and its classifier starting from a given entity prediction model and ontology with text snippets containing entity mentions (e.g. Wikidata and Wikipedia). For simplicity we use $\text{LinkCount}(e, m)$ as our entity prediction model.

We restrict the types in our type systems to use a set of parent-child relations over the ontology in Wikipedia and Wikidata, where each type axis has a root node r and an edge type g, that sets membership or exclusion from the axis (e.g. $r =$ **human**, $e =$ `instance of`, splits entities into: human vs. non-human[2]).

We then reformulate the problem into a mixed integer problem, where discrete variables control which roots $r_1, \ldots, r_k$ and edge types $g_1, \ldots, g_k$ among all roots $\mathcal{R}$ and edge types $\mathcal{G}$ will define type axes, while the continuous variables θ parametrize a classifier fit to the type system. Our goal in type system design is to select parent-child relations that a classifier easily predicts, and where the types improve disambiguation accuracy.

Objective

To formally define our mixed integer problem, let us first denote $\mathcal{A}$ as the assignment for the discrete variables that define our type system (i.e. boolean variables defining if a parent-child relation gets included in our type system), θ as the parameters for our entity prediction model and type classifier, and $S_{\text{model}}(\mathcal{A}, \theta)$ as the disambiguation accuracy given a test corpus containing mentions $M = \{(m_0, e_0^{\text{GT}}, \mathcal{E}_{m_0}), \ldots, (m_n, e_n^{\text{GT}}, \mathcal{E}_{m_n})\}$. We now assume our model produces some score for each proposed entity e given a mention m in a document D, defined $\text{EntityScore}(e, m, D, \mathcal{A}, \theta)$. The predicted entity for a given mention is thus: $e^* = \text{argmax}_{e \in \mathcal{E}_m} \text{EntityScore}(e, m, D, \mathcal{A}, \theta)$. If $e^* = e^{\text{GT}}$, the mention is disambiguated. Our problem is thus defined as:

$$\max_{\mathcal{A}} \max_{\theta} S_{\text{model}}(\mathcal{A}, \theta) = \frac{\sum_{(m, e_{\text{GT}}, \mathcal{E}_m) \in M} \mathbb{1}_{e_{\text{GT}}}(e^*)}{|M|}. \quad (2)$$

This original formulation cannot be solved exactly[3]. To make this problem tractable we propose a 2-step algorithm:

1. **Discrete Optimization of Type System**: Heuristic search or stochastic optimization over the discrete variables of the type system, $\mathcal{A}$, informed by a Learnability heuristic and an Oracle.
2. **Type classifier**: Gradient descent over continuous variables θ to fit type classifier and entity prediction model.

We will now explain in more detail discrete optimization of a type system, our heuristics (Oracle and Learnability heuristic), the type classifier, and inference in this model.

Discrete Optimization of a Type System

The original objective $S_{\text{model}}(\mathcal{A}, \theta)$ cannot be solved exactly, thus we rely on heuristic search or stochastic optimization to find suitable assignments for $\mathcal{A}$. To avoid training an entire type classifier and entity prediction model for each evaluation of the objective function, we instead use a proxy objective J for the discrete optimization[4]. To ensure

[2]Type "`instance of`:**human**" mimics the NER `PER` label.

[3]There are $\sim 2^{2.4 \cdot 10^7}$ choices if each Wikipedia article can be a type within our type system.

[4]Training of the type classifier takes ~3 days on a Titan X Pascal, while our Oracle can run over the test set in 100ms.

that maximizing $J(\mathcal{A})$ also maximizes $S_{\text{model}}(\mathcal{A}, \theta)$, we introduce a Learnability heuristic and an Oracle that quantify the disambiguation power of a proposed type system, an estimate of how learnable the type axes in the selected solution will be. We measure an upper bound for the disambiguation power by measuring disambiguation accuracy S_{oracle} for a type classifier Oracle over a test corpus.

To ensure that the additional disambiguation power of a solution $\mathcal{A}$ translates in practice we weigh by an estimate of solution's learnability $\text{Learnability}(\mathcal{A})$ improvements between S_{oracle} and the accuracy of a system that predicts only according to the entity prediction model[5], S_{greedy}.

Selecting a large number of type axes will provide strong disambiguation power, but may lead to degenerate solutions that are harder to train, slow down inference, and lack higher-level concepts that provide similar accuracy with less axes. We prevent this by adding a per type axis penalty of λ.

Combining these three terms gives us the equation for J:

$$J(\mathcal{A}) = (S_{\text{oracle}} - S_{\text{greedy}}) \cdot \text{Learnability}(\mathcal{A}) + S_{\text{greedy}} - |\mathcal{A}| \cdot \lambda. \quad (3)$$

Oracle Our Oracle is a methodology for abstracting away machine learning performance from the underlying representational power of a type system $\mathcal{A}$. It operates on a test corpus with a set of mentions, entities, and proposal sets: $m_i, e_i^{\text{GT}}, \mathcal{E}_{m_i}$. The Oracle prunes each proposal set to only contain entities whose types match those of e_i^{GT}, yielding $\mathcal{E}_{m,\text{oracle}}$. Types fully disambiguate when $|\mathcal{E}_{m,\text{oracle}}| = 1$, otherwise we use the entity prediction model to select the right entity in the remainder set $\mathcal{E}_{m_i,\text{oracle}}$:

$$\text{Oracle}(m) = \underset{e \in \mathcal{E}_{m,\text{oracle}}}{\text{argmax}} \ \mathbb{P}_{\text{entity}}(e | m, \text{types}(x)). \quad (4)$$

If $\text{Oracle}(m) = e^{\text{GT}}$, the mention is disambiguated. Oracle accuracy is denoted S_{oracle} given a type system over a test corpus containing mentions $M = \{(m_0, e_0^{\text{GT}}, \mathcal{E}_{m_0}), \ldots, (m_n, e_n^{\text{GT}}, \mathcal{E}_{m_n})\}$:

$$S_{\text{oracle}} = \frac{\sum_{(m, e_{\text{GT}}, \mathcal{E}_m) \in M} \mathbb{1}_{e_{\text{GT}}}(\text{Oracle}(m))}{|M|}. \quad (5)$$

Learnability To ensure that disambiguation gains obtained during the discrete optimization are available when we train our type classifier, we want to ensure that the types selected are easy to predict. The Learnability heuristic empirically measures the average performance of classifiers at predicting the presence of a type within some Learnability-specific training set.

To efficiently estimate Learnability for a full type system we make an independence assumption and model it as the mean of the Learnability for each individual axis, ignoring positive or negative transfer effects between different type axes. This assumption lets us parallelize training of simpler classifiers for each type axis. We measure the area under its receiver operating characteristics curve (AUC) for each classifier and compute the type system's learnability: $\text{Learnability}(\mathcal{A}) = \frac{\sum_{t \in \mathcal{A}} \text{AUC}(t)}{|\mathcal{A}|}$. We use a text window classifier trained over windows of 10 words before and after a mention. Words are represented with randomly initialized word embeddings; the classifier is illustrated in Figure 3a. AUC is averaged over 4 training runs for each type axis.

[5]For an entity prediction model based only on link counts, this means always picking the most linked entity.

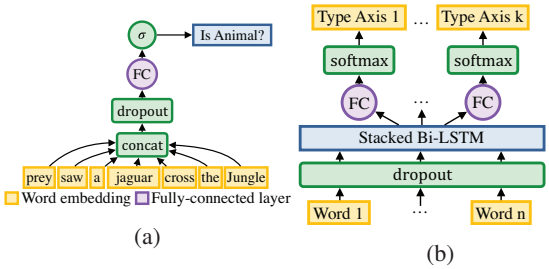


Figure 3: Text window classifier in (a) serves as type Learnability estimator, while the network in (b) takes longer to train, but discovers long-term dependencies to predict types and jointly produces a distribution for multiple type axes.

Type Classifier

After the discrete optimization has completed we now have a type system $\mathcal{A}$. We can now use this type system to label data in multiple languages from text snippets associated with the ontology[6], and supervize a Type classifier.

The goal for this classifier is to discover long-term dependencies in the input data that let it reliably predict types across many contexts and languages. For this reason we select a bidirectional-LSTM (Lample et al. 2016) with word, prefix, and suffix embeddings as done in (Andor et al. 2016). Our network is shown pictorially in Figure 3b. Our classifier is trained to minimize the negative log likelihood of the per-token types for each type axis in the document D with L tokens: $-\sum_{i=1}^{k} \log \mathbb{P}_i(t_{i,1}, \ldots, t_{i,L} | D)$. When using Wikipedia as our source of text snippets our label supervision is partial[7], so we make a conditional independence assumption about our predictions and use Softmax as our output activation: $-\sum_{i=1}^{k} \sum_{j=1}^{L} \log \mathbb{P}_i(t_{i,j} | w_j, D)$.

Inference

At inference-time we incorporate classifier belief into our decision process by first running it over the full context and obtaining a belief over each type axis for each input word $w_0, \ldots, w_L$. For each mention m covering words $w_x, \ldots, w_y$, we obtain the type conditional probability for all type axes i: $\{\mathbb{P}_i(\cdot | w_x, D), \ldots, \mathbb{P}_i(\cdot | w_y, D)\}$. In multi-word mentions we must combine beliefs over multiple tokens $x \ldots y$: the product of the beliefs over the mention's tokens is correct but numerically unstable and slightly less

[6]Wikidata's ontology has cross-links with Wikipedia, IMDB, Discogs, MusicBrainz, and other encyclopaedias with snippets.

[7]We obtain type labels only on the intra-wiki link anchor text.

performant than max-over-time[8], which we denote for the i-th type axis: $\mathbb{P}_{i,*}(\cdot|m, D)$.

The score $s_{e,m,D,\mathcal{A},\theta} = \text{EntityScore}(e, m, D, \mathcal{A}, \theta)$ of an entity e given these conditional probability distributions $\mathbb{P}_{1,*}(\cdot|m, D), \ldots, \mathbb{P}_{k,*}(\cdot|m, D)$, and the entities' types in each axis $t_1, \ldots, t_k$ can then be combined to rank entities according to how predicted they were by both the entity prediction model and the type system. The chosen entity e^* for a mention m is chosen by taking the option that maximizes the score among the $\mathcal{E}_m$ possible entities; the equation for scoring and e^* is given below, with $\mathbb{P}_{\text{Link}}(e|m) = \frac{\text{LinkCount}(m,e)}{\sum_{j \in \mathcal{E}_m} \text{LinkCount}(m,j)}$, α_i a per type axis smoothing parameter, β is a smoothing parameter over all types:

$$s_{e,m,D,\mathcal{A},\theta} = \mathbb{P}_{\text{Link}}(e|m) \cdot \left(1 - \beta + \beta \cdot \left\{ \prod_{i=1}^{k} (1 - \alpha_i + \alpha_i \cdot \mathbb{P}_{i,*}(t_i|m, D)) \right\} \right). \quad (6)$$

4 Results

Type System Discovery

In the following experiments we evaluate the behavior of different search methodologies for type system discovery: which method best scales to large numbers of types, achieves high accuracy on the target EL task, and whether the choice of search impacts learnability by a classifier or generalisability to held-out EL datasets.

For the following experiments we optimize DeepType's type system over a held-out set of 1000 randomly sampled articles taken from the Feb. 2017 English Wikipedia dump, with the Learnability heuristic text window classifiers trained only on those articles. The type classifier is trained jointly on English and French articles, totalling 800 million tokens for training, 1 million tokens for validation, sampled equally from either language.

We restrict roots $\mathcal{R}$ and edges $\mathcal{G}$ to the most common $1.5 \cdot 10^5$ entities that are entity parents through `wikipedia category` or `instance of` edges, and eliminate type axes where $\text{Learnability}(\cdot)$ is 0, leaving 53,626 type axes.

Human Type System Baseline To isolate discrete optimization from system performance and gain perspective on the difficulty and nature of the type system design we incorporate a human-designed type system. Human designers have access to the full set of entities and relations in Wikipedia and Wikidata, and compose different inheritance rules through Boolean algebra to obtain higher level concepts (e.g. `woman` $=$ `IsHuman` $\wedge$ `IsFemale`, or `animal` $=$ `IsTaxon` $\wedge \neg\{$`IsHuman` $\vee$ `IsPlant`$\}$[9]). The final human system uses 5 type axes[10], and 1218 inheritance rules.

[8]The choice of max-over-time is empirically motivated: we compared product mean, min, max, and found that max was comparable to mean, and slightly better than the alternatives.

[9]Taxon is the general parent of living items in Wikidata.

[10]`IsA`, `Topic`, `Location`, `Continent`, and `Time`.

Search methodologies

Beam Search and Greedy selection We iteratively construct a type system by choosing among all remaining type axes and evaluating whether the inclusion of a new type axis improves our objective: $J(\mathcal{A} \cup \{t_j\}) > J(\mathcal{A})$. We use a beam size of b and stop the search when all solutions stop growing.

Cross-Entropy Method (CEM) (Rubinstein 1999) is a stochastic optimization procedure applicable to the selection of types. We begin with a probability vector $\vec{P}_0$ set to p_{start}, and at each iteration we sample M_{CEM} vectors $\vec{s}$ from the Bernoulli distribution given by $\vec{P}_i$, and measure each sample's fitness with Eq. 3. The N_{CEM} highest fitness elements are our winning population $\mathcal{S}_t$ at iteration t. Our probabilities are fit to $\mathcal{S}_t$ giving $P_{t+1} = \frac{\sum_{\vec{s} \in \mathcal{S}_t} \vec{s}}{N_{\text{CEM}}}$. The optimization is complete when the probability vector is binary.

Genetic Algorithm The best subset of type axes can be found by representing type axes as genes carried by $N_{\text{population}}$ individuals in a population undergoing mutations and crossovers (Harvey 2009) over G generations. We select individuals using Eq. 3 as our fitness function.

Search Methodology Performance Impact To validate that λ controls type system size, and find the best tradeoff between size and accuracy, we experiment with a range of values and find that accuracy grows more slowly below 0.00007, while system size still increases.

From this point on we keep $\lambda = 0.00007$, and we compare the number of iterations needed by different search methods to converge, against two baselines: the empty set and the mean performance of 100 randomly sampled sets of 128 types (Table 1a). We observe that the performance of stochastic optimizers GA and CEM is similar to heuristic search, but requires orders of magnitude less function evaluations.

Next, we compare the behavior of the different search methods to a human designed system and state of the art approaches on three standard datasets (i.e. WIKI-DISAMB30 (WKD30) (Ferragina and Scaiella 2010)[11], CoNLL(YAGO) (Hoffart et al. 2011), and TAC KBP 2010 (Ji et al. 2010)), along with test sets built by randomly sampling 1000 articles from Wikipedia's February 2017 dump in English, French, German, and Spanish which were excluded from training the classifiers. Table 1c has Oracle performance for the different search methods on the test sets, where we report disambiguation accuracy per annotation. A LinkCount baseline is included that selects the mention's most frequently linked entity[12]. All search techniques' Oracle ac-

[11]We apply the preprocessing and link pruning as (Ferragina and Scaiella 2010) to ensure the comparison is fair.

[12]Note that LinkCount accuracy is stronger than the one found in (Ferragina and Scaiella 2010) or (Milne and Witten 2008) because newer Wikipedia dumps improve link coverage and reduce link distribution noisiness.

Table 1: Method comparisons. Highest value in **bold**, excluding oracles.

(a) Type system discovery method comparison

Approach	Evals	Accuracy	Items
BeamSearch	$5.12 \cdot 10^7$	97.84	130
Greedy	$6.40 \cdot 10^6$	97.83	130
GA	$116,000$	96.959	128
CEM	$43,000$	96.26	89
Random	N/A	92.9 ± 0.28	128
No types	0	92.10	0

(b) NER F1 score comparison for DeepType pretraining vs. baselines.

Model	CoNLL 2003		OntoNotes	
	Dev	Test	Dev	Test
Bi-LSTM (Chiu and Nichols 2015)	-	76.29	-	77.77
Bi-LSTM-CNN + emb + lex (Chiu and Nichols 2015)	**94.31**	**91.62**	84.57	**86.28**
Bi-LSTM (Ours)	89.49	83.40	82.75	81.03
Bi-LSTM-CNN (Ours)	90.54	84.74	83.17	82.35
Bi-LSTM-CNN (Ours) + types	93.54	88.67	**85.11**	83.12

(c) Entity Linking model Comparison. Significant improvements over prior work denoted by * for $p < 0.05$, and ** for $p < 0.01$.

Model		enwiki	frwiki	dewiki	eswiki	WKD30	CoNLL	TAC 2010
M&W(Milne and Witten 2008)						84.6	-	-
TagMe (Ferragina and Scaiella 2010)		83.224		80.711		90.9	-	-
(Globerson et al. 2016)						-	91.7	87.2
(Yamada et al. 2016)						-	91.5	85.2
NTEE (Yamada et al. 2017)						-	-	87.7
LinkCount only		89.064**	92.013	92.013**	89.980	82.710	68.614	81.485
Ours	manual	**94.331****	92.967			91.888**	93.108**	90.743*
	manual (oracle)	97.734	98.026	98.632	98.178	95.872	98.217	98.601
	greedy	93.725**	**92.984**			**92.375****	94.151**	**90.850***
	greedy (oracle)	98.002	97.222	97.915	98.246	97.293	98.982	98.278
	CEM	93.707**	92.415			92.247**	93.962**	90.302*
	CEM (oracle)	97.500	96.648	97.480	97.599	96.481	99.005	96.767
	GA	93.684**	92.027			92.062**	**94.879****	90.312*
	GA (oracle)	97.297	96.783	97.408	97.609	96.268	98.461	96.663
	GA (English only)	93.029**				91.743**	93.701**	-

curacy significantly improve over LinkCount, and achieve near perfect accuracy on all datasets (97-99%); furthermore we notice that performance between the held-out Wikipedia sets and standard datasets sets is similar, supporting the claim that the discovered type systems generalize well. We note that machine discovered type systems outperform human designed systems: CEM beats the human type system on English Wikipedia, and all search method's type systems outperform human systems on WIKI-DISAMB30, CoNLL(YAGO), and TAC KBP 2010.

Search Methodology Learnability Impact To understand whether the type systems produced by different search methods can be trained similarly well we compare the type system built by GA, CEM, greedy, and the one constructed manually. EL Disambiguation accuracy is shown in Table 1c, where we compare with recent deep-learning based approaches (Globerson et al. 2016), or recent work by Yamada et al. for embedding word and entities (Yamada et al. 2016), or documents and entities (Yamada et al. 2017), along with count and coherence based techniques Tagme (Ferragina and Scaiella 2010) and Milne & Witten (Milne and Witten 2008). To obtain Tagme's Feb. 2017 Wikipedia accuracy we query the public web API[13] available in German and English, while other methods can be compared on CoNLL(YAGO) and TAC KBP 2010. Models trained on a human type system outperform all previous approaches to entity linking, while type systems discovered by machines lead to even higher performance on all datasets except English Wikipedia.

Cross-Lingual Transfer

Type systems are defined over Wikidata/Wikipedia, a multilingual knowledge base/encyclopaedia, thus type axes are language independent and can produce cross-lingual supervision. To verify whether this cross-lingual ability is useful we train a type system on an English dataset and verify whether it can successfully supervize French data. We also measure using the Oracle (performance upper bound) whether the type system is useful in Spanish or German. Oracle performance across multiple languages does not appear to degrade when transferring to other languages (Table 1c). We also notice that training in French with an English type system still yields improvements over LinkCount for CEM, greedy, and human systems.

[13] https://tagme.d4science.org/tagme/

Because multi-lingual training might oversubscribe the model, we verified if monolingual would outperform bilingual training: we compare GA in English + French with only English (last row of Table 1c). Bilingual training does not appear to hurt, and might in fact be helpful.

We follow-up by inspecting whether the bilingual word vector space led to shared representations: common nouns have their English-French translation close-by, while proper nouns do not (French and US politicians cluster separately).

Named Entity Recognition Transfer

The goal of our NER experiment is to verify whether DeepType produces a type sensitive language representation useful for transfer to other downstream tasks. To measure this we pre-train a type classifier with a character-CNN and word embeddings as inputs, following (Kim et al. 2015), and replace the output layer with a linear-chain CRF (Lample et al. 2016) to fine-tune to NER data. Our model's F1 scores when transferring to the CoNLL 2003 NER task and OntoNotes 5.0 (CoNLL 2012) split are given in Table 1b. We compare with two baselines that share the architecture but are not pre-trained, along with the current state of the art (Chiu and Nichols 2015).

We see positive transfer on Ontonotes and CoNLL: our baseline Bi-LSTM strongly outperforms (Chiu and Nichols 2015)'s baseline, while pre-training gives an additional 3-4 F1 points, with our best model outperforming the state of the art on the OntoNotes development split. While our baseline LSTM-CRF performs better than in the literature, our strongest baseline (CNN+LSTM+CRF) does not match the state of the art with a lexicon. We find that DeepType always improves over baselines and partially recovers lexicon performance gains, but does not fully replace lexicons.

5 Related Work

Neural Network Reasoning with Symbolic structures Several approaches exist for incorporating symbolic structures into the reasoning process of a neural network by designing a loss function that is defined with a label hierarchy. In particular the work of (Deng et al. 2012) trades off specificity for accuracy, by leveraging the hyper/hyponymy relation to make a model aware of different granularity levels. Our work differs from this approach in that we design our type system within an ontology to meet specific accuracy goals, while they make the accuracy/specificity tradeoff at training time, with a fixed structure. More recently (Wu, Tygert, and LeCun 2017) use a hierarchical loss to increase the penalty for distant branches of a label hierarchy using the ultrametric tree distance. We also aim to capture the most important aspects of the symbolic structure and shape our loss function accordingly, however our loss shaping is a result of discrete optimization and incorporates a learnability heuristic to choose aspects that can easily be acquired.

A different direction for integrating structure stems from constraining model outputs, or enforcing a grammar. In the work of (Ling, Singh, and Weld 2015), the authors use NER and FIGER types to ensure that an EL model follows the constraints given by types. We also use a type system and constrain our model's output, however our type system is task-specific and designed by a machine with a disambiguation accuracy objective, and unlike the authors we find that types improve accuracy. The work of (Krishnamurthy, Dasigi, and Gardner 2017) uses a type-aware grammar to constrain the decoding of a neural semantic parser. Our work makes use of type constraints during decoding, however the grammar and types in their system require human engineering to fit each individual semantic parsing task, while our type systems are based on online encyclopaedias and ontologies, with applications beyond EL.

Neural Entity Linking Current approaches to entity linking make extensive use of deep neural networks, distributed representations. In (Globerson et al. 2016) a neural network uses attention to focus on contextual entities to disambiguate. While our work does not make use of attention, RNNs allow context information to affect disambiguation decisions. In the work of (Yamada et al. 2016) and (Yamada et al. 2017), the authors adopt a distributed representation of context which either models words and entities, or documents and entities such that distances between vectors informs disambiguation. We also rely on word and document vectors produced by RNNs, however entities are not explicitly represented in our neural network, and we use context to predict entity types, thereby allowing us to incorporate new entities without retraining.

6 Conclusion

In this work we introduce DeepType, a method for integrating symbolic knowledge into the reasoning process of a neural network. We've proposed a mixed integer reformulation for jointly designing type systems and training a classifier for a target task, and empirically validated that when this technique is applied to EL it is effective at integrating symbolic information in the neural network reasoning process. When pre-training with DeepType for NER, we observe improved performance over baselines and a new state of the art on the OntoNotes dev set, suggesting there is cross-domain transfer: symbolic information is incorporated in the neural network's distributed representation. Furthermore we find that type systems designed by machines outperform those designed by humans on three benchmark datasets, which is attributable to incorporating learnability and target task performance goals within the design process. Our approach naturally enables multilingual training, and our experiments show that bilingual training improves over monolingual, and type systems optimized for English operate at similar accuracies in French, German, and Spanish, supporting the claim that the type system optimization leads to the discovery of high level cross-lingual concepts useful for knowledge representation. We compare different search techniques, and observe that stochastic optimization has comparable performance to heuristic search, but with orders of magnitude less objective function evaluations.

The main contributions of this work are a joint formulation for designing and integrating symbolic information into neural networks, that enable us to constrain the out-

puts to obey symbolic structure, and an approach to EL that uses type constraints. Our approach reduces EL resolution complexity from $O(N^2)$ to $O(N)$, while allowing new entities to be incorporated without retraining, and we find on three standard datasets (WikiDisamb30, CoNLL (YAGO), TAC KBP 2010) that our approach outperforms all existing solutions by a wide margin, including approaches that rely on a human-designed type system (Ling, Singh, and Weld 2015) and the more recent work by Yamada et al. for embedding words and entities (Yamada et al. 2016), or document and entities (Yamada et al. 2017). As a result of our experiments, we observe that disambiguation accuracy using Oracles reaches 99.0% on CoNLL (YAGO) and 98.6% on TAC KBP 2010, suggesting that EL would be almost solved if we can close the gap between type classifiers and the Oracle.

The results presented in this work suggest many directions for future research: we may test how DeepType can be applied to other problems where incorporating symbolic structure is beneficial, whether making type system design more expressive by allowing hierarchies can help close the gap between model and Oracle accuracy, and seeing if additional gains can be obtained by relaxing the classifier's conditional independence assumption.

Acknowledgments We would like to thank the anonymous reviewers for their valuable feedback. In addition, we thank John Miller, Andrew Gibiansky, and Szymon Sidor for thoughtful comments and fruitful discussion.

References

Abu-El-Haija, S.; Kothari, N.; Lee, J.; Natsev, P.; Toderici, G.; Varadarajan, B.; and Vijayanarasimhan, S. 2016. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*.

Andor, D.; Alberti, C.; Weiss, D.; Severyn, A.; Presta, A.; Ganchev, K.; Petrov, S.; and Collins, M. 2016. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042*.

Chiu, J. P., and Nichols, E. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 248–255. IEEE.

Deng, J.; Krause, J.; Berg, A. C.; and Fei-Fei, L. 2012. Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 3450–3457. IEEE.

Ferragina, P., and Scaiella, U. 2010. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, 1625–1628. ACM.

Globerson, A.; Lazic, N.; Chakrabarti, S.; Subramanya, A.; Ringaard, M.; and Pereira, F. 2016. Collective entity resolution with multi-focal attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 621–631.

Harvey, I. 2009. The microbial genetic algorithm. In *European Conference on Artificial Life*, 126–133. Springer.

Hoffart, J.; Yosef, M. A.; Bordino, I.; Fürstenau, H.; Pinkal, M.; Spaniol, M.; Taneva, B.; Thater, S.; and Weikum, G. 2011. Robust Disambiguation of Named Entities in Text. In *Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, Edinburgh, Scotland*, 782–792.

Ji, H.; Grishman, R.; Dang, H. T.; Griffitt, K.; and Ellis, J. 2010. Overview of the tac 2010 knowledge base population track. In *Third Text Analysis Conference (TAC 2010)*, volume 3, 3–3.

Kay, W.; Carreira, J.; Simonyan, K.; Zhang, B.; Hillier, C.; Vijayanarasimhan, S.; Viola, F.; Green, T.; Back, T.; Natsev, P.; et al. 2017. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*.

Kim, Y.; Jernite, Y.; Sontag, D.; and Rush, A. M. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

Krishnamurthy, J.; Dasigi, P.; and Gardner, M. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *EMNLP*, volume 17, 1532–1543.

Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Ling, X., and Weld, D. S. 2012. Fine-grained entity recognition. In *AAAI*. Citeseer.

Ling, X.; Singh, S.; and Weld, D. S. 2015. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics* 3:315–328.

Milne, D., and Witten, I. H. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, 509–518. ACM.

Pradhan, S.; Moschitti, A.; Xue, N.; Uryupina, O.; and Zhang, Y. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *EMNLP-CoNLL Shared Task*.

Rubinstein, R. 1999. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability* 1(2):127–190.

Sang, E. F. T. K., and Meulder, F. D. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Wu, C.; Tygert, M.; and LeCun, Y. 2017. Hierarchical loss for classification. *arXiv preprint arXiv:1709.01062*.

Yamada, I.; Shindo, H.; Takeda, H.; and Takefuji, Y. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. *arXiv preprint arXiv:1601.01343*.

Yamada, I.; Shindo, H.; Takeda, H.; and Takefuji, Y. 2017. Learning distributed representations of texts and entities from knowledge base. *arXiv preprint arXiv:1705.02494*.