Improving Neural Fine-Grained Entity Typing with Knowledge Attention

Ji Xin[1], Yankai Lin[2], Zhiyuan Liu[2]*, Maosong Sun[2]
[1]Deaprtment of Physics, Tsinghua University, Beijing, China
[2]State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing, China
{xinj14,linyk14}@mails.tsinghua.edu.cn {liuzy,sms}@tsinghua.edu.cn

Abstract

Fine-grained entity typing aims to identify the semantic type of an entity in a particular plain text. It is an important task which can be helpful for a lot of natural language processing (NLP) applications. Most existing methods typically extract features separately from the entity mention and context words for type classification. These methods inevitably fail to model complex correlations between entity mentions and context words. They also neglect rich background information about these entities in knowledge bases (KBs). To address these issues, we take information from KBs into consideration to bridge entity mentions and their context together, and thereby propose Knowledge-Attention Neural Fine-Grained Entity Typing. Experimental results and case studies on real-world datasets demonstrate that our model significantly outperforms other state-of-the-art methods, revealing the effectiveness of incorporating KB information for entity typing. Code and data for this paper can be found at https://github.com/thunlp/KNET.

1 Introduction

Entity typing is the task of detecting semantic types of a named entity in a plain text. It is an important task which can narrow down the range of candidates for an entity mention, and is therefore beneficial for a large number of natural language processing (NLP) tasks such as entity linking (Chabchoub, Gagnon, and Zouaq 2016), relation extraction (Liu et al. 2014), question answering (Yahya et al. 2013) and knowledge base population (Carlson et al. 2010).

Fine-grained entity typing (FET), which classifies entities into a large set of fine-grained types, is the more challenging new trend of entity typing. Conventional FET methods usually derive features using NLP tools such as POS tagging and parsing, and inevitably suffer from error propagation. Dong et al. (2015) make the first attempt to explore deep learning in entity typing by using only word embeddings as features and discarding complicated feature engineering. Shimaoka et al. (2016; 2017) further introduce attention into neural models for FET.

Neural models have achieved the state-of-the-art performance for FET. However, these models face the following non-trivial challenges:

Entity-Context Separation. Existing methods typically encode entity mentions and context words as separate features without considering correlations between them. However, it is intuitive that, the importance of each context word is significantly influenced by the entity mention concerned. For example, in the sentence "*Gates and Allen co-founded Microsoft, which became the largest software company*", the context word *company* is important when we are determining types of the entity mention *Microsoft*, but less important when determining types of *Gates*.

Text-Knowledge Separation. Knowledge bases (KBs, also known as Knowledge Graphs), such as YAGO, Freebase, provide rich information of relations between entities in form of triples (h, r, t), where h, t are head and tail entities, and r is the relationship between them. Such information describes relations and interactions between entities, and is therefore helpful for entity typing. For example, given a triple (*USA*, shares_border_with, *Canada*), it can be deduced that *Canada* in a certain sentence is likely to be a country. However, relational information has never been used in previous work yet to the best of our knowledge.

In order to address the issues of entity-context separation and text-knowledge separation, we propose **K**nowledge-Attention **N**eural Fine-Grained **E**ntity **T**yping (KNET). As illustrated in Figure 1, our model mainly consists of two parts. Firstly, we build a neural network to generate context and entity mention representations. Secondly, depending on the entity mention, we use knowledge attention to focus on important context words and improve the quality of context representation. Knowledge attention is computed using entity embeddings, which are learned from KB relational information and reconstructed from the text. Considering we will encounter both in-KB and out-of-KB entities in testing, we propose a disambiguation procedure, which can provide not only in-KB entities with precise KB information, but also out-of-KB entities with useful knowledge.

We establish two datasets for experiments: one automatically built from Wikipedia and Freebase, one manually labeled. Experiment results show that our model significantly improves the performance of entity typing compared with other state-of-the-art methods. Case studies demonstrate the effectiveness of knowledge attention, both for incorporating KB information into entity typing, and for capturing complex correlations between entities and context.

*Corresponding author: Zhiyuan Liu.

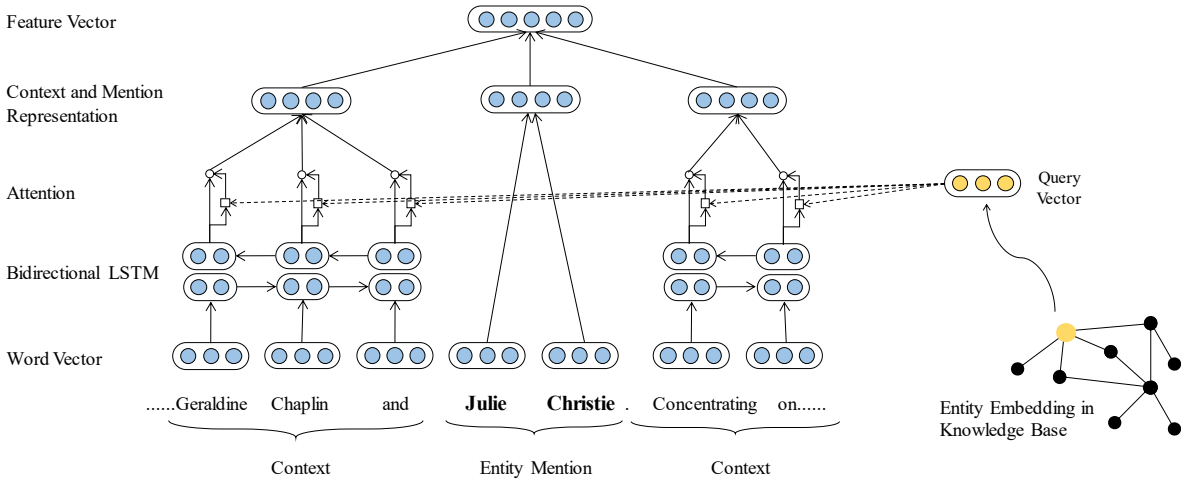

Figure 1: The framework of Neural Entity Typing with Knowledge Attention.

2 Related Work

2.1 Fine-Grained Entity Typing

Entity typing is conventionally a sub-task of named entity recognition (Collins and Singer 1999; Tjong Kim Sang and De Meulder 2003; Kozareva 2006; Jiang and Zhai 2006; Ratinov and Roth 2009), which typically classifies entity mentions into `person`, `location`, `organization` and `others`. However, these types are too coarse-grained to be helpful for other NLP applications. Recently, fine-grained entity typing (Ling and Weld 2012; Yosef et al. 2012; Gillick et al. 2014; Yogatama, Gillick, and Lazic 2015; Del Corro et al. 2015; Ren et al. 2016a) has been proposed to classify entities into a richer set of types, which typically constitute a hierarchy. Ling and Weld (2012); Yosef et al. (2012); Del Corro et al. (2015) all introduce their own fine-grained taxonomies with the size ranging from 112 to over 16,000.

All methods mentioned above rely on hand-crafted features. Dong et al. (2015) make the first attempt to explore deep learning in entity typing using only word embeddings as features. Further, Shimaoka et al. (2016) introduce an attention-based Long Short-Term Memory (LSTM) for FET, and Shimaoka et al. (2017) incorporate hand-crafted features into the attention-based neural models. These neural models, however, encounter the challenges of entity-context separation and text-knowledge separation. This paper seeks to address these issues by incorporating rich information of KBs.

KBs have been considered in a number of previous works (Del Corro et al. 2015; Ren et al. 2016a; Yaghoobzadeh and Schütze 2017). However, they only consider type information of each entity inside the KB and neglect rich relational information (the relationship between different entities), which happens to be an important part of the KB. In this paper, we incorporate relational information into entity typing by using knowledge representation learning (see next subsection for details).

Different from the works mentioned above, which can be called sentence-level entity typing, Yaghoobzadeh and Schütze (2015; 2017) consider corpus-level neural entity typing. Corpus-level entity typing aims to infer global types of an entity from a large corpus, usually by aggregating information from all sentences mentioning it. In contrast, sentence-level entity typing seeks to detect local types of an entity mention inside an individual sentence, and the same entity could have different types in different sentences. Our work focuses on sentence-level entity typing.

2.2 Knowledge Representation Learning

Knowledge Representation Learning (KRL) aims to encode semantic information of entities and relations inside triples into a low-dimensional semantic space. It can be further used as supplements in many tasks such as link prediction (Bordes et al. 2011) and question answering (Bordes, Weston, and Usunier 2014).

TransE (Bordes et al. 2013) is one of the most widely used KRL methods. It embeds entities and relations in the same space, and aims to ensure $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ when the triple (h, r, t) holds, where $\mathbf{h}$, $\mathbf{r}$ and $\mathbf{t}$ are corresponding embeddings.

TransE performs well for 1-to-1 relations, but is weak for other complicated relations. To address this issue, many other KRL models have been proposed to deal with various characteristics of KBs, such as TransH (Wang et al. 2014), TransR (Lin et al. 2015), TransD (Ji et al. 2015), TranSparse (Ji et al. 2016), KG2E (He et al. 2015), PTransE (Lin, Liu, and Sun 2015) and HolE (Nickel, Rosasco, and Poggio 2016). In this work we utilize TransE to examine the effectiveness of incorporating relational information of KB into entity typing.

3 Overall Framework of KNET

Given a sentence s which contains an entity mention and its context, and a set of entity types (the taxonomy) $\mathcal{T}$, our model aims to predict the probability of each type for this entity mention.

We denote the entity mention and the left/right context words in s as m_i, l_i, r_i respectively, and the sentence is

a sequence of words $s = \{..., l_2, l_1, m_1, m_2, ..., r_1, r_2, ...\}$. For each word, we use bold face to denote its corresponding word vector. For each entity mention, our model builds a feature vector $\mathbf{x}$ to infer the probability of each type by computing an entity type vector $\mathbf{y}$.

The framework of KNET consists of two parts: (1) *Sentence Encoder* which encodes the sentence s into the feature vector $\mathbf{x}$; (2) *Type Predictor* which infers entity types by computing $\mathbf{y}$ from $\mathbf{x}$.

3.1 Sentence Encoder

Our model transforms word vectors into representations for entity mention and context with neural networks. The feature vector $\mathbf{x}$ is the concatenation of *entity mention representation* $\mathbf{m}$ and *context representations* $\mathbf{c}$:

$$\mathbf{x} = \begin{bmatrix} \mathbf{m} \\ \mathbf{c} \end{bmatrix}. \quad (1)$$

Entity mention representation. Following (Shimaoka et al. 2016), the representation $\mathbf{m}$ of entity mention is simply computed by averaging the word vectors of entity mention words $\{m_1, m_2, m_3, ...\}$:

$$\mathbf{m} = \frac{1}{n_m} \sum_{i=1}^{n_m} \mathbf{m}_i, \quad (2)$$

where n_m is the length of entity mention. Simple averaging is sufficient for entity mention representation, because an entity mention typically consists of a small number of words (in most cases 1 or 2), therefore complicated models (like CNN or RNN) tend to overfit.

Context representation. Bidirectional LSTM and attention mechanism are used to encode context representation. Word vectors of context words $\{..., l_3, l_2, l_1\}$ and $\{r_1, r_2, r_3, ...\}$ are fed into the LSTM, and context representation $\mathbf{c}$ is the weighted sum of the LSTM outputs:

$$\mathbf{c} = \frac{\sum_{i=1}^{L} \left(a_i^l \begin{bmatrix} \overrightarrow{\mathbf{h}_i^l} \\ \overleftarrow{\mathbf{h}_i^l} \end{bmatrix} + a_i^r \begin{bmatrix} \overleftarrow{\mathbf{h}_i^r} \\ \overrightarrow{\mathbf{h}_i^r} \end{bmatrix} \right)}{\sum_{i=1}^{L} a_i^l + a_i^r}, \quad (3)$$

where $\overrightarrow{\mathbf{h}_i}$ and $\overleftarrow{\mathbf{h}_i}$ are the outputs of the bidirectional LSTM, and arrows above them indicate the direction of the LSTM. a_i are the corresponding attention scores, which will be introduced in details in the next section. The superscripts l and r indicate left or right context parts. L is the window size of context words.

3.2 Type Predictor

The type vector $\mathbf{y}$ is computed from sentence vector $\mathbf{x}$ through a two-layer Multi-Layer Perceptron (MLP). Each entry of $\mathbf{y}$ indicates the predicted probability of the type for given entity mention:

$$\mathbf{y} = \sigma\big(\mathbf{W}_{y1} \tanh\left(\mathbf{W}_{y2}\mathbf{x}\right)\big), \quad (4)$$

$$\mathbf{y}^{(i)} = p(t^{(i)}|s, \theta), \quad (5)$$

where θ indicates all parameters of our model, σ is the sigmoid function, $\mathbf{W}_{y1}$, $\mathbf{W}_{y2}$ are MLP parameter matrices and $t^{(i)}$ means the ith type is predicted for this entity. A type is predicted as positive if its probability is greater than 0.5, and if none of them is greater, the type with the largest probability is regarded as positive.

The objective function is defined as element-wise cross entropy over all entity mentions:

$$J(\theta) = -\sum_{i,j} \mathbf{y}^{*(j)}_i \log \mathbf{y}^{(j)}_i + (1 - \mathbf{y}^{*(j)}_i) \log(1 - \mathbf{y}^{(j)}_i), \quad (6)$$

where $\mathbf{y}^*$ indicates the ground truth types of the mention, $\mathbf{y}^{(j)}_i$ is the jth entry of the vector $\mathbf{y}_i$. We learn to maximize J by optimizing parameters θ.

4 Attention Mechanism for KNET

Attention mechanism plays an important role in our model. In this section, we introduce the design of attention a_i^l and a_i^r in context representation as shown in Eq. (3). Attention is expected to take both entity-context and entity-KB correlations into consideration. In this paper, we explore three kinds of attention:

(1) *Semantic Attention* simply applies the context representation itself as attention query, which is proposed by (Shimaoka et al. 2017) and will be considered as our baseline method.

(2) *Mention Attention* applies the entity mention representation $\mathbf{m}$ as attention query, which is expected to capture the semantic correlations between entities and context information.

(3) *Knowledge Attention* applies entity representations learned from external KBs as attention query, which is expected to capture the semantic correlations of entity-context and entity-KB.

Details of attention are described as follows.

4.1 Semantic Attention (SA)

Shimaoka et al. (2017) apply an MLP to compute semantic attention as follows:

$$a_i^{\text{SA}} = \sigma\big(\mathbf{W}_{S1} \tanh\left(\mathbf{W}_{S2} \begin{bmatrix} \overrightarrow{\mathbf{h}_i} \\ \overleftarrow{\mathbf{h}_i} \end{bmatrix}\right)\big), \quad (7)$$

where $\mathbf{W}_{S1}$ and $\mathbf{W}_{S2}$ are MLP parameter matrices. Starting from here we omit the superscripts l and r since they are computed identically.

We notice that all entities share the same MLP used for computing SA. Therefore, the attention computed for context words are independent of the entity concerned. Hence, it is difficult for SA to focus on those context words that are highly related to the corresponding entity.

4.2 Mention Attention (MA)

In order to take entity-context correlation into consideration, it is straightforward to take entity mentions as attention queries when computing attention.

Formally, given the entity mention representation $\mathbf{m}$ mentioned in Eq. (2), we compute attention as follows:

$$a_i^{\text{MA}} = f\left(\mathbf{m}\mathbf{W}_{\text{MA}}\left[\begin{array}{c}\overrightarrow{\mathbf{h}_i}\\ \overleftarrow{\mathbf{h}_i}\end{array}\right]\right), \quad (8)$$

where $\mathbf{W}_{\text{MA}}$ is a bi-linear parameter matrix, and $f()$ is a non-linear function. We simply choose the quadratic function $f(x) = x^2$, which is positive definite and easily differentiable (the same setting applies to Eq. (9) as well).

4.3 Knowledge Attention (KA)

Knowledge bases provide rich relational information about entities, which is important for entity typing. We use the most widely-used KRL method TransE (Bordes et al. 2013) to encode relational information into entity embeddings.

During the training process, the corresponding entity e for a certain entity mention is known, and hence, similar to Eq. (8), we can directly compute knowledge attention as follows:

$$a_i^{\text{KA}} = f\left(\mathbf{e}\mathbf{W}_{\text{KA}}\left[\begin{array}{c}\overrightarrow{\mathbf{h}_i}\\ \overleftarrow{\mathbf{h}_i}\end{array}\right]\right), \quad (9)$$

where $\mathbf{e}$ is the embedding for entity m, and $\mathbf{W}_{\text{KA}}$ is a bi-linear parameter matrix.

Knowledge Attention in Testing. Different from training, in the testing process there is a new challenge: we do not know in prior which entity in the KB is corresponding to a certain entity mention (it may even be out-of-KB). A straightforward solution is to perform entity linking, but entity linking itself is non-trivial and will inevitably introduce errors. Furthermore, entity linking does not work for out-of-KB entities.

To address this challenge, we try to reconstruct entity embeddings using text information. These embeddings are also learned during training. Concretely, for an entity e and its context sentence s, we encode its left / right context into $\mathbf{c}_l$ and $\mathbf{c}_r$ using a single-directional LSTM, and further learn the text-based representation $\hat{\mathbf{e}}$ as follows:

$$\hat{\mathbf{e}} = \tanh\left(\mathbf{W}\left[\begin{array}{c}\mathbf{m}\\ \mathbf{c}_l\\ \mathbf{c}_r\end{array}\right]\right), \quad (10)$$

where $\mathbf{W}$ is the parameter matrix, and $\mathbf{m}$ is the mention representation in Eq. (2). Note that, LSTM used here is different to those in Eq. (3) in order to prevent interference. In order to bridge text-reconstructed and KB-based representations, in the training process we simultaneously learn $\hat{\mathbf{e}}$ by putting an additional component in the objective function J in Eq. (6):

$$J_{\text{KB}}(\theta) = -\sum ||\mathbf{e} - \hat{\mathbf{e}}||^2, \quad (11)$$

where the summation is over all entities in the training set. In this way, in the testing process, we can directly use Eq. (10) to obtain the approximate entity embedding and compute knowledge attention using Eq. (9).

Knowledge Attention with Disambiguation (KA+D). It is straightforward that, if we can narrow down the range of candidate entities by using the surface name of the entity mention, we can obtain more accurate information. Such information can be used as a complement to the text-reconstructed embedding built from Eq. (10). We conduct entity disambiguation as follows: (1) We build a list of candidate entities by matching the surface names of entities in KBs and entity mentions. (2) We compute the L2 distance between the text-reconstructed embedding $\hat{\mathbf{e}}$ and candidate entity representations in KBs, and select the candidate with the smallest distance. An example is shown in Figure 2.

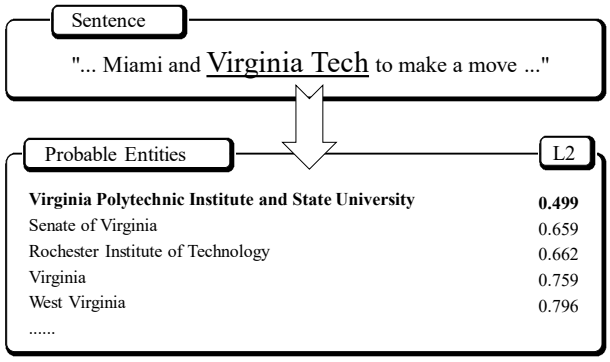

Figure 2: An example of how an entity mention is disambiguated to the correct entity.

To alleviate the harm caused by inevitable disambiguation errors, we set a threshold α for the L2 distance. For the chosen entity e and its L2 distance d, if d is smaller than α, which means we can be confident with the disambiguation result, we select $\mathbf{e}$ to compute knowledge attention. If d is greater than α, which means the disambiguation process is probably erroneous, or this is an out-of-KB entity without anything similar in the KB, we directly use $\hat{\mathbf{e}}$.

5 Experiments

5.1 Dataset Construction

FIGER is a widely used dataset for entity typing proposed in (Ling and Weld 2012). However, the training set of FIGER does not include the linked entities of mentions required by KNET. Moreover, the test set is not fine-grained enough (e.g., over 38% entities are only annotated with `person` and no more fine-grained labels).

Therefore, we build our own dataset, which consists of an automatically labeled part and a manually labeled part:

- **Automatically labeled dataset** WIKI-AUTO. Similar to (Ling and Weld 2012), we employ Wikipedia and Freebase to generate *train, validation and test* datasets using distant supervision (Mintz et al. 2009). Concretely, we search Wikipedia text for sentences with an anchor link which links to another Wikipedia page. The page can be further connected to a Freebase entity, whose type labels are shown in Freebase. Without loss of generality, we choose entities in FB15K when searching through Wikipedia. FB15K is a subset of Freebase containing common entities introduced in (Bordes et al. 2013). To better simulate practical application, entities in the test set

are unseen in the train set. Also, they do not necessarily need to be inside the KB.

Freebase contains thousands of types, which are typically noisy and confusing. For example, the entity *New York City* has as many as 85 types, including `citytown`, `city_with_dogs` and `award_winner`. To avoid such confusion, we only keep types that have at least 50 instances in FB15K, and then manually map them into a two-layer hierarchical taxonomy with 74 types.

- **Manually labeled dataset** WIKI-MAN. Distant supervision inevitably introduces noise into WIKI-AUTO (Ren et al. 2016b; Yaghoobzadeh, Adel, and Schütze 2017). As a result, we randomly pick 100 entity mentions and their sentences from Wikipedia, and manually label them using the same taxonomy as WIKI-AUTO. This dataset, WIKI-MAN, is *only* used for testing.

Automatic and manual datasets have their own weakness. The weakness of manual datasets is the inevitably limited size, which is not a problem for automatic labeling. The weakness of automatic datasets is the distant supervision assumption that all labels from KBs are correct for any context. From the observation of manually labeled dataset, however, this weakness of automatic datasets is not severe: only a small proportion of entities would have different labels in different contexts (e.g., only 3.9% in the test set of FIGER). These two kinds of datasets are complementary for each other's weakness. Consequently, we conduct experiments and report results on both datasets, and the results are generally consistent. We compare the test sets of FIGER, WIKI-AUTO and WIKI-MAN in Table 1.

Dataset	WIKI-AUTO	WIKI-MAN	FIGER
Total Entities	100,000	100	562
Labels per Entity	3.07	2.32	1.38
Person	22.47%	16.00%	43.42%
Organization	14.76%	11.00%	28.11%
Location	39.90%	52.00%	18.15%
Others	22.87%	21.00%	12.81%

Table 1: Comparison of various test sets.

Note that, in our experiments Freebase plays two roles: (1) providing triples for KRL with TransE; (2) providing type information for annotating datasets. These two roles do not necessarily need to be done by one KB; rather, they can be accomplished by two independent KBs. And in our experiments, we also make sure triples in (1) and entity type information in (2) do not interfere with each other, which ensures the model can be generalized to using different KBs[1].

5.2 Experiment Settings

Following (Ling and Weld 2012), we use macro-F1, micro-F1, and accuracy to evaluate the performance of models. In general, we consider micro-F1 to be the metric that can best represent the performance of fine-grained entity typing because entities with more fine-grained labels will have greater proportion. Further details can be found in their paper.

Following (Shimaoka et al. 2017), we use pre-trained word embeddings from (Pennington, Socher, and Manning 2014). We use Adam Optimizer (Kingma and Ba 2014) and mini-batch of size B for parameter optimization. We also use implementation of TransE from (Lin et al. 2015) to obtain entity embeddings.

To avoid overfitting, we employ Dropout (Srivastava et al. 2014) on entity mention representation. The reason for only applying Dropout to entity mention is that entities of test set are probably unseen in training, while context words are not so different.

We explore different sets of hyperparameter settings and determine Adam optimizer learning rate λ among $\{0.01, 0.005, 0.001\}$, hidden-size of LSTM among $\{100, 150, 200\}$, word vector size among $\{50, 100, 300\}$, window size L among $\{5, 10, 15\}$ and batch size B among $\{100, 500, 1{,}000\}$, based on performance on the validation set. The hyperparameter settings are shown in Table 3.

Hyperparameter	Value
Learning rate	0.005
LSTM hidden-size	100
Word vector size	300
Window size	15
Batch size	1,000

Table 3: Hyperparameter settings.

5.3 Evaluation Results

Recent neural models have been shown to outperform most feature-based models like (Ling and Weld 2012; Yosef et al. 2012; Yogatama, Gillick, and Lazic 2015). Hence we consider the following two neural models as our baseline:

(1) **Neural Model with Semantic Attention (SA).** To the best of our knowledge, (Shimaoka et al. 2017) achieves the state-of-the-art performance. Since their codes are not yet publicly available, we implement their model by ourselves and achieves comparable results as reported by the authors. This model is described in the Semantic Attention (SA) part.

(2) **Hybrid Neural Model (HNM).** We also implement the model HNM (Dong et al. 2015), which is also a neural model with fully-connected layers and recurrent layers, but without the attention mechanism.

We also consider a recent feature-based baseline:

(3) **AFET** (Ren et al. 2016a), which also uses auxiliary information from the KB, but does not consider relational knowledge.

Considering the external relational knowledge injected by KB embedding, we further introduce a baseline:

(4) **KB-ONLY**, which only uses KB embedding for entity typing (replacing $\mathbf{x}$ in Eq. 4 with $\mathbf{e}$ or $\hat{\mathbf{e}}$, controled by the threshold α).

We compare these four baselines with our neural entity typing models with Mention Attention (MA), Knowledge

[1] In some cases such as specific academic area or non-English language, KBs are limited both in size and content. These two kinds of information may be separated in different KBs.

Dateset	WIKI-AUTO							WIKI-MAN						
Metrics	Strict	Macro			Micro			Strict	Macro			Micro		
	Acc	Pre	Rec	F1	Pre	Rec	F1	Acc	Pre	Rec	F1	Pre	Rec	F1
AFET	20.32	67.00	45.82	54.75	69.29	42.40	52.61	18.00	64.50	50.00	56.33	64.29	50.43	56.52
KB-ONLY	35.12	69.65	71.35	70.49	54.85	74.99	63.36	17.00	55.50	72.83	63.00	27.81	74.57	40.52
HNM	34.88	68.09	61.03	64.37	72.80	64,48	68.39	15.00	61.80	68.00	64.75	62.35	68.53	65.30
SA	42.77	75.33	69.69	72.40	77.35	72.63	74.91	18.00	66.67	73.67	69.44	65.54	75.43	70.14
MA	41.58	73.64	71.71	72.66	75.94	75.52	75.72	26.00	65.13	78.50	71.19	64.09	82.33	72.08
KA	45.49	74.82	72.46	73.62	76.96	75.49	76.22	23.00	64.69	78.92	71.10	63.25	82.68	71.67
KA+D	**47.20**	**75.72**	**74.03**	**74.87**	**77.96**	**77.87**	**77.92**	**34.00**	**68.41**	**82.83**	**74.94**	**66.12**	**87.50**	**75.32**

Table 2: Performance of entity typing, evaluated by strict accuracy, micro and macro precison, recall and F1 score. (%)

Attention (KA), Knowledge Attention with Disambiguation (KA+D). The results are shown in Table 2. From the table we observe that:

(1) All neural models perform better than AFET, demonstrating their power in making use of the large-scale training dataset.

(2) MA performs slightly better compared to SA by making a primitive attempt at entity-discriminated attention. This indicates the benefits of entity-related attention.

(3) KA and KA+D achieve the best results among all methods. The reason is that both KA and KA+D introduce rich entity information from KBs, and produce more accurate attention over context words as compared to other methods. It indicates the significance of adding the entity information from KBs into entity typing.

(4) KA+D has better performance than KA under all evaluation metrics. It demonstrates that, by conducting entity disambiguation based on surface name matching and embedding similarity between entity mention and candidate entity, the model can utilize more precise information from KBs.

(5) The performance of KB-ONLY is considerably worse than KA and KA+D. It indicates that, although KB information can be beneficial for entity typing, it cannot work alone. Instead, it has to be considered jointly with text information in a more sophisticated way.

5.4 Effectiveness on Different Entities

To study the details of our models, we further compare them with baselines on different subsets of the test set. The division is based on entity coarse types or disambiguation difficulty.

Entity Coarse Types We explore the performance of various methods for three coarse-grained entity types, `person`, `organization` and `location`. For better illustration, we also compare them with a naïve baseline, M-ONLY, which only uses entity mention representations $\mathbf{m}$ for entity typing (replacing $\mathbf{x}$ with $\mathbf{m}$ in Eq. 4). The results are shown in Table 4.

From the table, we can observe that: KA and KA+D achieve larger improvements on coarse types which are less "easy", such as `person` and `organization`. The reason is that, it is easier to determine fine-grained types of a `location` entity simply according to entity mentions, which often contain informative words like *River* or *Road*. But for `person` and `organization`, we have to rely more on context information. In this case, KA and KA+D show their superiority for modeling context information. The performance of M-ONLY shows the degree of "easy" for each coarse type.

Type	`Person`	`Organization`	`Location`
Dataset		WIKI-AUTO	
M-ONLY	58.64	63.95	87.65
HNM	63.79	66.85	86.26
SA	68.47	71.85	90.74
KA	70.77	74.18	91.23
KA+D	**74.87**	**75.16**	**91.75**
Type	`Person`	`Organization`	`Location`
Dataset		WIKI-MAN	
M-ONLY	52.63	71.19	75.54
HNM	54.00	50.00	76.69
SA	55.77	81.36	79.26
KA	67.72	75.41	79.29
KA+D	**67.14**	**90.32**	**81.62**

Table 4: Comparison on entities of different coarse-grained types, evaluated by micro-F1 (%).

Disambiguation Difficulty In KA+D, disambiguation of in-KB entities mention depends on different context environments. The context either provides rich and helpful information about the attributes of the entity, or hardly contains any useful hint. Disambiguation of out-of-KB entities, on the other hand, will no doubt be erroneous. We divide the test set into two subsets named *Correct* and *Erroneous*, according to whether the disambiguation process is correct or not, and explore the performance of various methods. The results are shown in Table 5.

From the table we can observe that:

(1) All methods perform better in the *Correct* subset than in the *Erroneous* subset. The results are reasonable, since the contexts of in-KB entity mentions in the *Correct* subset provide more accurate information than those in the *Erroneous* subset.

Subset	*Correct*		*Erroneous*	
Metrics	strict	micro-F1	strict	micro-F1
Wiki-auto	80.53%		19.47%	
HNM	37.60	68.39	23.60	52.15
SA	46.66	78.63	26.64	57.61
MA	44.32	79.29	28.26	59.05
KA	49.24	79.83	**29.99**	**59.42**
KA+D	**51.77**	**82.33**	28.27	57.56
Subset	*Correct*		*Erroneous*	
Metrics	strict	micro-F1	strict	micro-F1
Wiki-man	83.00%		17.00%	
HNM	15.66	67.80	11.76	51.95
SA	20.48	75.05	5.88	47.37
MA	28.92	75.22	11.76	53.85
KA	24.10	75.23	**17.65**	53.93
KA+D	**34.94**	**78.32**	12.50	**54.77**

Table 5: Performance on correct/erroneous subsets, with percentages indicating the ratios among all entity mentions.

(2) KA consistently outperforms all baselines in both subsets. It indicates that employing KB information can robustly achieve improvements on entity typing.

(3) In the *Correct* subset, KA+D can obtain precise entity information from KBs via disambiguation, and thus significantly outperforms all other methods. The superiority of KA+D is smaller in the *Erroneous* subset due to unsuccessful disambiguation, but it still outperforms the baselines. The reason is that, controlled by the threshold α, an entity mention in this subset will either be disambiguated to a similar entity (whose embedding will also be useful), or keep the original text constructed embedding, and therefore alleviate the error.

We further demonstrate the impact of the threshold α on KA+D in Figure 3. It is shown that, as α increases (i.e., more confident with disambiguation results), the performance of KA+D is improved in the *Correct* subset, but decays in the *Erroneous* subset. Hence, in real-world applications, we have to tune α to achieve the trade-off according to the ratio of *Correct*/*Erroneous* subsets (i.e., disambiguation difficulty of the dataset). Considering the sharp decreasing in *Erroneous* and the relatively slow increasing in *Correct* when α goes from 0.55 to 0.7, we set α to be 0.55 in Table 3.

5.5 Case Analysis

In Figure 4, we visualize and compare attention computed by SA and KA+D in an example.

From the example we can see that: SA fails to concentrate on useful words for entity typing. KA+D, by disambiguating to the correct entity in KB, is able to focus on those informative words, such as *starred*, *the film*, *Omar Sharif* and *Geraldine Chaplin*[2]. Types predicted by KA+D are `person`, `artist` and `actor`, which are identical with

[2]Julie Christie, Omar Sharif and Geraldine Chaplin are actors and actresses. They acted together in a film *Doctor Zhivago*.

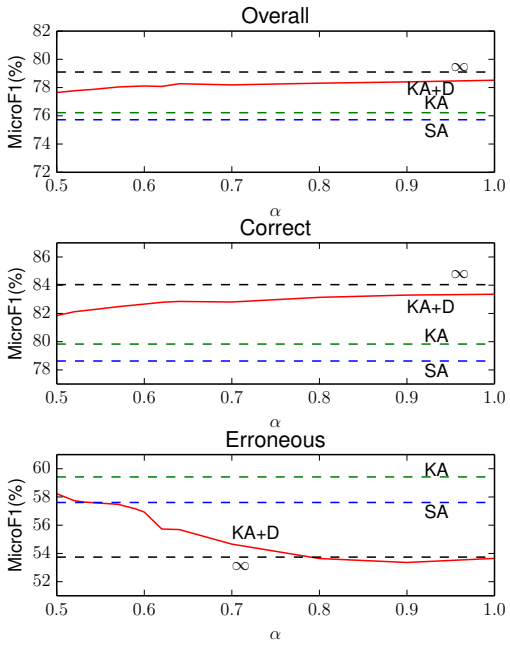

Figure 3: Evaluating the effect of disambiguation threshold α on KA+D in the validation set. ∞ indicates KA+D without using α.

Model	Sentence and Attention
SA	...tradition starred Omar Sharif, Geraldine Chaplin and **Julie Christie**. Concentrating on the love triangle aspects of the novel, the film ...
KA+D	...tradition starred Omar Sharif, Geraldine Chaplin and **Julie Christie**. Concentrating on the love triangle aspects of the novel, the film ...

Figure 4: Case analysis for the entity *Julie Christie*. Deeper background color indicates higher attention.

annotated labels. Besides these three types, SA also predicts three more superfluous types.

6 Conclusion and Future Work

In this paper, we propose a novel attention mechanism which leverages information from KBs and joint bring text and KB into consideration. We introduce a new dataset, experiments based on which clearly demonstrate the effectiveness and significance of the proposed knowledge attention. Among several methods compared, our new model KA+D achieves the state-of-the-art performance of Micro-F1 score 77.92 in the WIKI-AUTO dataset.

We will explore the following directions as future work: (1) Our KNET framework can incorporate any kinds of KRL methods with no difficulty, and we will explore the effectiveness of other KRL methods besides TransE. (2) We

will examine the effectiveness of our KNET methods on a more complicated taxonomy of entity types, with either more classes or deeper hierarchy. (3) Directly using existing entity linking tools will inevitably introduce noise. Alleviating such noise and incorporating entity linking in our model will be interesting to explore in the future. (4) Existing works about FET have used a number of different datasets and taxonomies (Shimaoka et al. 2017), and we will also further explore our model on various datasets.

7 Acknowledgments

This work is supported by the 973 Program (No. 2014CB340501), the National Natural Science Foundation of China (NSFC No. 61572273, 61661146007), China Association for Science and Technology (2016QNRC001), and Tsinghua University Initiative Scientific Research Program (20151080406).

References

Bordes, A.; Weston, J.; Collobert, R.; and Bengio, Y. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of AAAI*.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of NIPS*.

Bordes, A.; Weston, J.; and Usunier, N. 2014. Open question answering with weakly supervised embedding models. In *Proceedings of ECML-PKDD*.

Carlson, A.; Betteridge, J.; Wang, R. C.; Hruschka Jr, E. R.; and Mitchell, T. M. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of WSDM*.

Chabchoub, M.; Gagnon, M.; and Zouaq, A. 2016. Collective disambiguation and semantic annotation for entity linking and typing. In *Proceedings of SWEC*.

Collins, M., and Singer, Y. 1999. Unsupervised models for named entity classification. In *Proceedings of EMNLP*.

Del Corro, L.; Abujabal, A.; Gemulla, R.; and Weikum, G. 2015. Finet: Context-aware fine-grained named entity typing. In *Proceedings of EMNLP*.

Dong, L.; Wei, F.; Sun, H.; Zhou, M.; and Xu, K. 2015. A hybrid neural model for type classification of entity mentions. In *Proceedings of IJCAI*.

Gillick, D.; Lazic, N.; Ganchev, K.; Kirchner, J.; and Huynh, D. 2014. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820*.

He, S.; Liu, K.; Ji, G.; and Zhao, J. 2015. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of CIKM*.

Ji, G.; He, S.; Xu, L.; Liu, K.; and Zhao, J. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of ACL*.

Ji, G.; Liu, K.; He, S.; and Zhao, J. 2016. Knowledge graph completion with adaptive sparse transfer matrix. *Proceedings of AAAI*.

Jiang, J., and Zhai, C. 2006. Exploiting domain structure for named entity recognition. In *Proceedings of NAACL*.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kozareva, Z. 2006. Bootstrapping named entity recognition with automatically generated gazetteer lists. In *Proceedings of EACL*.

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI*.

Lin, Y.; Liu, Z.; and Sun, M. 2015. Modeling relation paths for representation learning of knowledge bases. *Proceedings of EMNLP*.

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

Liu, Y.; Liu, K.; Xu, L.; and Zhao, J. 2014. Exploring fine-grained entity type constraints for distantly supervised relation extraction. In *Proceedings of COLING*.

Mintz, M.; Bills, S.; Snow, R.; and Jurafsky, D. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL*.

Nickel, M.; Rosasco, L.; and Poggio, T. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of AAAI*.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, volume 14.

Ratinov, L., and Roth, D. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of CoNLL*.

Ren, X.; He, W.; Qu, M.; Huang, L.; Ji, H.; and Han, J. 2016a. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of EMNLP*.

Ren, X.; He, W.; Qu, M.; Voss, C. R.; Ji, H.; and Han, J. 2016b. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of KDD*.

Shimaoka, S.; Stenetorp, P.; Inui, K.; and Riedel, S. 2016. An attentive neural architecture for fine-grained entity type classification. *arXiv preprint arXiv:1604.05525*.

Shimaoka, S.; Stenetorp, P.; Inui, K.; and Riedel, S. 2017. Neural architectures for fine-grained entity type classification. In *Proceedings of EACL*.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR* 15(1).

Tjong Kim Sang, E. F., and De Meulder, F. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of HLT-NAACL*.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of AAAI*.

Yaghoobzadeh, Y.; Adel, H.; and Schütze, H. 2017. Noise mitigation for neural entity typing and relation extraction. In *Proceedings of EACL*.

Yaghoobzadeh, Y., and Schütze, H. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of EMNLP*.

Yaghoobzadeh, Y., and Schütze, H. 2017. Multi-level representations for fine-grained typing of knowledge base entities. In *Proceedings of EACL*.

Yahya, M.; Berberich, K.; Elbassuoni, S.; and Weikum, G. 2013. Robust question answering over the web of linked data. In *Proceedings of CIKM*.

Yogatama, D.; Gillick, D.; and Lazic, N. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of ACL*.

Yosef, M. A.; Bauer, S.; Hoffart, J.; Spaniol, M.; and Weikum, G. 2012. HYENA: Hierarchical type classification for entity names. In *Proceedings of COLING*.