Class-Dynamic and Hierarchy-Constrained Network for Entity Linking

Kehang Wang[1], Qi Liu[1,2(✉)], Kai Zhang[1], Ye Liu[1], Hanqing Tao[1], Zhenya Huang[1], and Enhong Chen[1]

[1] Auhui Province Key Laboratory of Big Data Analysis and Application, University of Science and Technology of China, Hefei 230026, China
{wangkehang,kkzhang0808,liuyer,hqtao}@mail.ustc.edu.cn, {qiliuql,huangzhy,cheneh}@ustc.edu.cn

[2] Institute of Artificial Intelligence, Hefei Comprehensive National Science Center, Hefei, China

Abstract. Entity Linking (EL) aims to map mentions in a text to corresponding entities in a knowledge base. Existing EL methods usually rely on sufficient labeled data to achieve the best performance. However, the massive investment in data makes EL systems viable only to a limited audience. There is ample evidence that introducing entity types can provide the model prior knowledge to maintain the model performance in low-data regimes. Unfortunately, current low-data EL methods usually employ entity types by rule constraints, which are in a shallow manner. Furthermore, they usually ignore fine-grained interaction between mention and its context, resulting in insufficient semantic information of mention representation in low-data regimes. To this end, we propose a Class-Dynamic and Hierarchy-Constrained Network (CDHCN) for entity linking. Specifically, we propose a dynamic class scheme to learn a more effective representation for each entity type. Besides, we formulate a hierarchical constraint scheme to reduce the matching difficulty of the given mention and corresponding candidate entities by utilizing entity types. In addition, we propose an auxiliary task called mention position prediction (MPP) to obtain an informative mention representation in low-data regimes. Finally, extensive in-domain and out-of-domain experiments demonstrate the effectiveness of our method.

Keywords: Entity linking · Entity type · Hierarchical constraint

1 Introduction

Entity Linking (EL), also known as entity disambiguation, aims to link ambiguous textual mention to the correct entity in a particular knowledge base (KB). As a fundamental building block for many Natural Language Processing (NLP) applications, such as Question Answering [1] and Text Mining [2–4], this task has received increasing attention from researchers in recent years.

X. Wang et al. (Eds.): DASFAA 2023, LNCS 13944, pp. 622–638, 2023.
https://doi.org/10.1007/978-3-031-30672-3_42

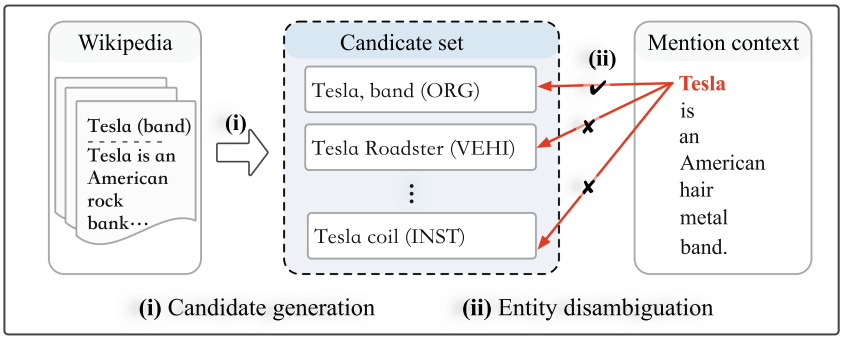

Fig. 1. A flowchart overview of the entity linking system.

Generally, as illustrated in Fig. 1, there are two main parts to the EL systems: the first part is the entity candidate generation module, which takes the given KB and selects a subset of entities that might be associated with mentions in the input text; the second part is the entity disambiguation module, which takes the given mentions and links them to the most corresponding entities in the candidates set. As Fig. 1 shows, in the sentence *"Tesla is an American hair metal band"*, the phrase *Tesla* is called a textual mention. Entity Linking seeks to map the mention to a real word entity from a specific KB. For instance, the mention *Tesla* in the above sentence should be mapped to *Tesla (band)* but not *Tesla (coil)* in Wikipedia. The entities to be linked may have similar word forms like *Tesla (band)* and *Tesla (coil)* which makes the EL task difficult.

Recently, learning better representations has been proposed as a promising direction for the EL tasks. For example, static word embeddings [5,6] and pre-trained models [8,9] have proved mention embedding's effectiveness for improving the performance of EL methods. Unfortunately, such methods often require a large amount of training data, and it is impractical for most researchers to repeatedly train EL models on massive data from KBs due to the constant updating of KBs (e.g., it took a week's time to train the BLINK [10] on 8-GPUs for $9M$ examples). For this reason, how to use external information to alleviate the problem of data dependence has received more attention in EL tasks [11–13] and other NLP tasks [14–16]. Specifically, Bhargav et al. [11] exploited external information about entity types by proposing an auxiliary task called entity type prediction in a low-data regime setting. Tedeschi et al. [13] proposed to improve the EL model trained on low amounts of labeled data by exploiting entity types.

However, the above approaches employ external knowledge (i.e., entity types) by rule constraints or simply splicing entity types and entity texts, which are in a shallow manner. For example, Tedeschi et al. [13] simply concatenated entity types (e.g., ORG) with entity description to enrich each candidate entity representation and use entity types to constrain the EL model's output. In this way, the models cannot make the best use of semantic information associated with entity types. Some deep semantic solutions (e.g., type embeddings) can pro-

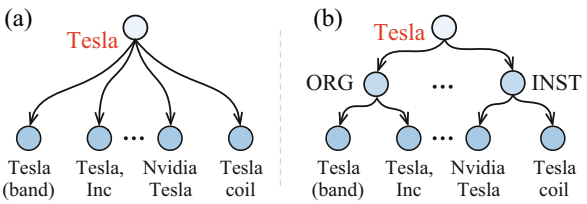

Fig. 2. The illustration of mention-entities selecting. (a) Traditional entity linking. These methods select the corresponding entity from a candidate set directly. (b) Hierarchical constraint entity linking. Our methods score the entity types for mention first and then use this score to select the corresponding entity.

vide more information for entity linking. Meantime, the previous models select the most suitable entity from the candidate set via similarity matching directly. However, these candidates are often highly similar. As shown in Fig. 2 (a), the candidate entity titles in Tesla's candidate set are similar, causing candidate entity representations generated to be similar, which is hard for models to select the mention's suitable entity. Therefore, it is necessary to exploit the semantic information of entity types to lower EL tasks difficulty.

With the above analysis, in this paper, we propose a **C**lass-**D**ynamic and **H**ierarchy-**C**onstrained **N**etwork (CDHCN) for effectively entity linking. Unlike traditional label embedding methods [17] embedded entity types statistically, we argue that the entity type representation should be dynamic as the meanings of the same entity type for different candidate entities maybe be different. For example, Tesla (band) and the European Union, the two mentions have the same entity type (ORG). However, their meanings are inconsistent, and we believe their entity type representations should be far away in the feature space. In response to this, we propose a dynamic class embedding generation method to learn a more effective representation for each entity type, which can ensure that the mention's unique characteristics can be retained as much as possible. Besides, different from traditional methods that match the mention and its candidate entities directly, we propose to use entity types help pre-classify of candidate entities and formulate a hierarchical scheme as shown in Fig. 2 (b). Specifically, the hierarchical constraint module in our method first classifies candidate entities according to entity types (e.g., ORG, INST) and then matches mention and entity types, as shown in Fig. 2 (b). In this way, we can substantially reduce the matching difficulty of the given mention and corresponding candidate entities. Furthermore, in order to solve the problem of the insufficient mention representation trained in low-data regimes, we propose an auxiliary task called *Mention Position Prediction (MPP)* to conduct multi-task learning. This task enables fine-grained interactions between mention and its context, which could capture more semantic information about the mention. In summary, the main contributions of our work could be summarized as follows.

– For the first time, we formulate a dynamic class scheme and hierarchical constraint scheme to fully utilize entity types in low-data regimes.

– We present an auxiliary task called *Mention Position Prediction* to obtain a more informative mention representation in low-data regimes.
– Extensive experiments on in-domain and out-of-domain datasets demonstrate the effectiveness of the proposed method. Our code is available at https://github.com/bigdata-ustc/CDHCN.

2 Related Work

2.1 Entity Linking

In recent years, the EL systems have made significant progress with the development of contextualized word embedding and representation methods [6,18,19]. There are a variety of different approaches proposed by researchers to tackle with EL task and its variants. For example, in order to maximize the similarity score between a mention embedding and its most corresponding entity embedding, Botha et al. [20] designed a dual-encoder architecture, consisting of two encoders for mentions and entities separately. GENRE, proposed by De Cao et al. [8], treats EL task as a generation problem by employing an auto-regressive formulation to teach a transformer-based model to produce a unique name for the mention. Gu et al. [21] proposed a machine reading comprehension framework for short text EL.

These methods described above require large amounts of training data, which is resource-intensive. This motivated researchers to make greater use of external information such as entity relations, entity types to provide the model with more prior information and reduce the scale of training data without performance degradation. Therefore, Raiman and Raiman [22] proposed DeepType, which makes use of type information to enhance the EL model's performance. Another noteworthy direction is how to fully leverage the entity types' commonalities between NER tasks and EL tasks. To take advantage of their relatedness, Martins et al. [23] performed joint learning of NER and EL, and obtain a robust system. Fine-grained NER labels are used to place limitations on the EL model's behavior in the most recent study [13]. Even though there is ample proof that introducing entity types (NER labels) into EL approaches is beneficial, current methods don't take fully utilize entity types information. For more than just labelling purposes: it can also be used to create informative embeddings that can be investigated with fine-grained features.

2.2 Name Entity Recognition

Most NER systems regard the task as sequence labelling and usually model it using conditional random fields (CRFs) or bi-directional Long Short Term Memory Networks (LSTMs). The performance of NER was further improved by recent large-scale pre-trained language models like BERT [9], which produced state-of-the-art results comparable to almost any other area in NLP. Thanks to NER classes, NER can cluster entities and solve the problem of intrinsic

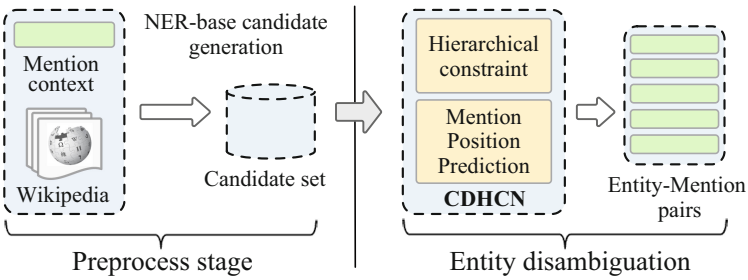

Fig. 3. The flowchart overview of our work.

sparsity in EL tasks. Besides, the relatedness of NER and EL has been proven to improve the performance of the entity linking model [23]. Nevertheless, there is a paucity of research on the effectiveness of enhancing EL models' capability by adding NER classes information or the relatedness of EL and NER. Previous approaches in this area have directly either used NER for mention detection or train a multi-task model that learns NER and EL jointly [23,24].

In this paper, we take the best of NER information. Following Tedeschi et al. [13], we add constraints to the candidate entities generation and inference phase. Additionally, we show using entity types embeddings is a feasible way to exploit NER for EL. Furthermore, we propose an auxiliary task called *mention position prediction*. This task is similar to NER but does not identify the type of detected mention, which can retain the mention's specific features for mention embedding generation as much as possible.

3 The Framework of CDHCN

In this section, we first present the problem statement of entity linking (EL), and then give an overview of our proposed CDHCN method. After that, we explain the technical details of CDHCN.

3.1 Problem Statement

The EL task can be formulated as a multi-label classification task where classes are represented by entities. Specifically, a text S contains a set of identified mentions $M = \{m_1, m_2, \ldots, m_n\}$. The goal of the EL task is to find a mapping function that links each mention m_i to a unique entity e_i which is an unambiguous page in a given KB (e.g., Wikipedia).

Before entity disambiguation, we have a preprocessing step called entity candidate generation that chooses potential candidate entities $\Theta_i = \{e_1, e_2, \ldots, e_K\}$ from a specific KB for each mention m_i to improve recall, where K is a pre-defined number to prune the candidate set. In this work, we adopt NER-enhanced candidate generation in line with Tedeschi et al. [13]. In a nutshell, by the use of NER classes, we can add constraints to the entity candidate set, which help decrease the size of the candidate set without reducing its recall.

3.2 An Overview of CDHCN

Figure 3 shows the framework of our method. In the preprocessing stage, our method makes use of mention's entity types (NER classes) to help generate a candidate set [13]. Note that each candidate has one pertinent description in the given KB. After obtaining the candidate set, we simultaneously input entities and mentions into the model. Through the hierarchical constraint module, we can obtain two similarity scores: one for the mention and entity type, and one for the mention and entity itself. Then, the former similarity score that we call it hierarchical constraint scores can be used to constrain the latter. Finally, we use the similarity score that is obtained after constraining for selecting the best mention-entity pair.

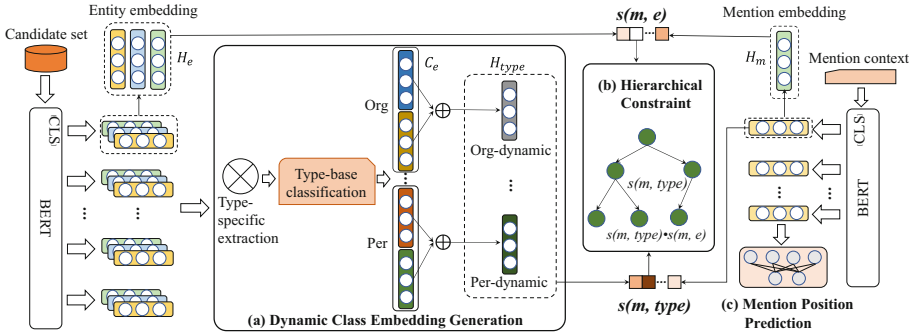

Fig. 4. Illustration of our proposed CDHCN framework for entity linking. (a) The dynamic class embeddings generation module can use candidate entities information to generate class embeddings for mention. (b) The hierarchical constraint module can use class embeddings to constrain the similarity scores between mention and its candidate entities. (c) The mention position prediction module can use auxiliary task to learn a more robust mention embedding.

3.3 Components of CDHCN

In this subsection, we will introduce the technical details of CDHCN. As shown in Fig. 4, CDHCN mainly contains three parts: (1) dynamic class embedding generation; (2) hierarchical constraint; (3) metion position prediction. We first obtain mention-specific entity type embeddings by dynamic class embedding generation module. Then the embedding can be used in hierarchical constraint module. Additionally, the mention position prediction module aims to obtain a more robust mention embeddings.

Input Representation. For a more accurate representation of mention and entity, we need to map each word in contextual mention and entity description into a low-dimensional vector. In this paper, we splice mention and mention's context as the input of transformer-base encoder to obtain initial mention

embeddings. Specifically, we construct the input of each mention example as:

$$I_m = [CLS]\ lctx\ [M_s]\ mention\ [M_e]\ rctx\ [SEP], \tag{1}$$

where *lctx* and *rctx* denote context to the left of the mention and right of the mention, respectively. To address the problem of quadratic dependency in the transformer encoder when dealing with long texts, we limit the mention context length L to 128. Similarly, The entity input is as follows:

$$I_e = [CLS]\ title\ [SEP]\ description\ [SEP], \tag{2}$$

where *[CLS]* and *[SEP]* are special placeholders and *title* denotes entity. We consider the first token embedding, which is the output of the last hidden layer corresponding to the position of the *[CLS]*, as mention or entity representation. Both mention context and candidate entity are encoded into vectors:

$$H_m = BERT_{1[CLS]}(I_m) \in \mathbb{R}^h, \tag{3}$$

$$H_e = BERT_{2[CLS]}(I_e) \in \mathbb{R}^h, \tag{4}$$

where H_m and H_e are all h-dimensional vectors which represent mention and candidate entity respectively. Note that $BERT_1$ and $BERT_2$ are two different encoders. Besides, we also obtain every token representation as follows:

$$[H_{e_i^1}, H_{e_i^2}, ..., H_{e_i^L}]^T = BERT_1(I_{e_i}) \in \mathbb{R}^{L \times h}, \tag{5}$$

$$[H_{m^1}, H_{m^2}, ..., H_{m^L}]^T = BERT_2(I_m) \in \mathbb{R}^{L \times h}, \tag{6}$$

where H is an h-dimensional vector and L is text length.

Dynamic Class Representation. To capture more fine-grained semantic features of entity types (e.g., Organization), we need to encode entity types into low-dimensional vectors. This can be done quickly and easily by using BERT's built-in support for entity types as input text. However, due to the limited text (i.e., entity types are only one word in general), the embeddings that the model extract are insufficient to represent types features. Moreover, as mentioned above, even the meanings of the same entity type for different mentions are diverse. For example, *Tesla*, *the European Union* and *Paribas* have same entity type - Organization. However, there are clear distinctions between them, i.e., Tesla is a rock band, the European Union is a political and economic union and Paribas is a banking group. Fortunately, the candidate entity descriptions provide us with rich type information. We propose two simple yet effective methods to extract type-specific information for each candidate entity: one is context-aware type extraction and the other is mention-aware type extraction.

1) Context-aware Type Extraction. The local context in the candidate entity description provides most of the type-specific information. Therefore, we focus on how to extract these crucial local features. In this method, multiple

CNNs are used for generating entity types features because of the excellent ability of CNN in extracting local features [25]. Specifically, given a candidate entity description representation $[H_{e_i^1}, H_{e_i^2}, ..., H_{e_i^L}]^T$ generated in the above step, we need to splice them together:

$$H_{e_i^{1:L}} = H_{e_i^1} \oplus H_{e_i^2} \oplus ... \oplus H_{e_i^L}, \tag{7}$$

where $\oplus$ denotes concatenation operation.

Then, the local feature $C_{e_i^x}$ of candidate entity e_i can be generated by convolution operation with different convolution kernel sizes ks:

$$C_{e_i^x} = f(W_{CNN}^T \cdot H_{e_i^{x+ks-1}} + b), \tag{8}$$

where $W_{CNN} \in \mathbb{R}^{ks \times h}$ is a learnable matrix and b is a bias term. f is a nonlinear function. Particularly, the kernel is applied to each possible windows of word embddings $[H_{e_i^{1:ks}}, H_{e_i^{2:ks+1}}, \ldots, H_{e_i^{L-ks+1:L}}]$ in the sentence to produce a feature map:

$$\mathbf{C}_{e_i} = [C_{e_i^1}, C_{e_i^2}, \ldots, C_{e_i^{L-ks+1}}]. \tag{9}$$

After that, we use a max-pooling layer over the feature map to extract key information, and we can obtain the continuous fine-grained representations $[\mathbf{C}_{e_1}, \mathbf{C}_{e_2}, ..., \mathbf{C}_{e_K}]$ for each candidate entity.

2) Mention-aware Type Extraction. Lacking complex mention-entity interactions in the type-specific extraction stage might lead to the extracted features not being the ones that are required by the mention-entity disambiguation task. Therefore, we focus on how to utilize the mention information to assist in extracting type-specific features for each candidate entity. Some researches have proven that the associative attention can improve feature learning [26]. In this method, the multi-head attention mechanism [27] is used to help the model focus on features more relevant to the mention-entity disambiguation task. Specifically, we regard mention embedding H_{m_i} as *query*, candidate entity token embedding $[H_{e_j^1}, H_{e_j^2}, ..., H_{e_j^L}]^T$ as *key* and *value*. Then the mention-aware entity embeddings are computed by:

$$\begin{aligned} \mathbf{C} &= \{\mathbf{C}_{e_i} | i = 1, 2, \ldots, K\} \\ &= MultiHead(query * W_q, key * W_k, value * W_v) \end{aligned} \tag{10}$$

where W_q, W_k and W_v are learnable parameters. The output feature $\mathbf{C} = \{\mathbf{C}_{e_i} | i = 1, 2, \ldots, K\}$ is mention-aware representations for each candidate entity. For detailed implementation of MultiHead Attention Mechanism, please refer to Transformer [27].

After extracting type-specific information, we need to generate entity type representations by aggregating fine-grained candidate entity representations with the same entity type. In detail, we use entity type (e.g., Organization) to cluster candidate type representations, and the final mention-specific entity type

representation $\mathbf{H}_{type}$ is computed by:

$$\mathbf{H}_{type_j} = \sum_{i=1}^{n} C_{e_{ij}}/n, \tag{11}$$

where e_{ij} $(i = 1, 2, \ldots, \mathrm{n})$ denotes that candidate entity e_i type is $type_j$.

Hierarchical Constraint. As we discussed before, external knowledge can be used to improve model performance without requiring a larger data set. In the paper, we use entity types information as an EL task external knowledge. Unlike previous literature, we model entity type implicitly so that the proposed model can capture specific type information for each different mention. As shown in Fig. 2 (b), before calculating the similarity scores between mention and candidate entities, we need to obtain the similarity scores between mention and mention-specific entity types, which can also be called hierarchical constraint scores. The hierarchical constraint scores are given by the cosine similarity score:

$$s(m, type_i) = \frac{H_m^T \cdot \mathbf{H}_{type_i}}{||H_m||\ ||\mathbf{H}_{type_i}||}, i \in (1, 2, \ldots, 18), \tag{12}$$

where $type_i$ represents entity type, and we have 18 kinds of entity types. $\mathbf{H}_{type}$ are entity type embedding that is computed in equation (11).

After obtaining the hierarchical constraint scores between mention and entity types $s(m, type_i)$, we could use these scores to constrain the similarity scores between mention and candidate entities. Prior to matching mention and candidate entities, it is intended that the model have pertinent knowledge about candidate entity types. The similarity scores between mention and candidate entities are also calculated by cosine similarity score:

$$s(m, e_i) = \frac{H_m^T \cdot H_{e_i}}{||H_m||\ ||H_{e_i}||}, i \in (1, 2, \ldots, K). \tag{13}$$

And the final constrained score function is as follows:

$$s_{type_i}(m, e_j) = s(m, type_i) \cdot s(m, e_j), \tag{14}$$

where $s_{type_i}(m, e_j)$ denotes the similarity score between mention m and candidate entity e_j whose entity type is $type_i$. Then the network is trained to maximize the score of the gold mention-entity pairs. The training loss function of gold entity prediction is defined as cross-entropy:

$$\mathcal{L}_{EL} = -\frac{1}{N} \sum_{k=1}^{N} \sum_{j=1}^{K} y_k^j log(s_{type_i}(m_k, e_j)), \tag{15}$$

where y_k^j denotes gold entity of m_k. N is the number of examples. K is the number of candidate entities.

Mention Position Prediction. In order to enhance low-data mention representation through fine-grained interaction between mention and its context, we

introduce *mention position prediction (MPP)* as an auxiliary task. We design an *MPP* task to teach the model to distinguish whether a token is a part of the mention. *MPP* is a token-level prediction task because we require tokens belonging to the correct mention span with the same position. In this way, the model can learn more token-level feature which is more compatible with the mention. This task is, in essence, a binary classification task that model predicts if a word is part of a mention or not. We feed mention context token embedding $[H_{m^1}, H_{m^2}, ..., H_{m^L}]$ obtained above into a feed-forward layer with Softmax function:

$$\tilde{y}_i^j = Softmax(W_{MPP}^T H_{m_j^i} + b_{MPP}), \tag{16}$$

where $W_{MPP} \in \mathbb{R}^h$ and $b \in \mathbb{R}^L$ are the learnable parameter and bias respectively. The training loss function of *MPP* is defined as cross-entropy:

$$\mathcal{L}_M = -\frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{L}[y_i^j log(\tilde{y}_i^j) + (1 - y_i^j) log(1 - \tilde{y}_i^j)], \tag{17}$$

where $\tilde{y}_i^j$ denotes the prediction and y_i^j is the target indicating whether $H_{m_j^i}$ is mention embedding. N is the number of training examples.

3.4 Training Strategy

Our approach, in contrast to conventional methods, consists of two tasks: entity linking task and mention position prediction task. we need to optimize simultaneously for these two tasks by a single loss function. Given the losses defined in equation (15) and equation (17), our loss function is as follows:

$$\mathcal{L} = \lambda_{EL}\mathcal{L}_{EL} + \lambda_M \mathcal{L}_M, \tag{18}$$

where $\mathcal{L}$ is the final optimization target, λ_{EL} and λ_M are hyper-parameters that denote task weights for entity linking and mention position prediction respectively. In our experiment setup, λ_{EL} and λ_M are all set to 1.

In the same manner as Tedeschi et al. [13], we use NER classes (e.g., ORG, LOC) to constrain the model's output. In particular, for each mention, we require the model to output entities whose entity types is consistent with the prediction of an NER classifier.

4 Experiments

4.1 Datasets Preparation

For the reliability and authority of experimental results, we conduct experiments on both the in-domain and out-of-domain datasets. Specifically, we use AIDA-YAGO-CoNLL [28] as our in-domain dataset. This dataset contains AIDA-train for training, AIDA-A for development, and AIDA-B for testing. It is important to note that we train each of our models on only the AIDA-train (18k

label instances). For out-domain datasets, we evaluated models on five popular datasets: MSNBC, AQUAINT, ACE2004 [29] and WNED-WIKI, WNED-CWEB [29,30]. The statistics of these datasets are shown in Table 1. For the KB, we use the November 2020 dump of the English Wikipedia labeled by Tedeschi et al. [13]. Each entity in Wikipedia was labeled by a new set of 18 fine-grained NER classes.

Table 1. Statistics of the datasets. It is important to note that MSNBC, AQUAINT, ACE2004, WNED-WIKI, and WNED-CWEB are out-of-domain datasets, meaning they do not have training data and development data. We train each of our models on only the AIDA-training.

Dataset	Train	Dev	Test
AIDA [28]	18,395	4,784	4,463
MSNBC [29]	–	–	656
AQUAINT [29]	–	–	727
ACE2004 [29]	–	–	257
WNED-WIKI [29,30]	–	–	6,821
WNED-CWEB [29,30]	–	–	11,154

4.2 Experiment Setup

In this paper, our goal is to demonstrate the superiority of our method in low-data regimes entity linking. For the low-data regimes scenario, we follow the setting of Tedeschi et al. [13]. A complete EL systems consists of mention detection, candidate generation and entity disambiguation. Here, we mainly focus on entity disambiguation methods. For mention detection and candidate generation, we follow the setting of Tedeschi et al. [13]. Note that, we do not have to figure out the right span of mention because it is provided in context. For candidate generation, we adopt NER-Enhance strategy to find a trade-off between recall and set size. Specifically, we train and employ an NER classifier to predict the entity type of the given mention in context and then try to choose the entities whose NER classes are consistent with the mention as candidate entities.

In our experiment, we use the pre-trained uncased BERT-based model with a 768 dimensions hidden representation as our backbone. We trained each model for 30 epochs, adopting an early stopping strategy with a patience value of 5. We adopt Adam [31] as optimizer with learning rate 1e−5 and maximum sequence length L 128. For each mention, the candidates' number K is set to 40. All experiments are performed on a NVIDIA GTX2080Ti with 11G GPU memory.

4.3 Baseline Methods

In our experiments, we compare our model with existing state-of-the-art baselines in entity linking:

- **Global-RNN** [32] is a method based on convolutional neural networks and recurrent neural networks.
- **Deep-ed** [6] is a joint document-level entity disambiguation method which leverage neural attention to reinforce context representations.
- **WNED** [29] is a greedy and global NED algorithm based on a sound information-theoretic notion of semantic relatedness derived from random walks on carefully constructed disambiguation graphs.
- **E-ELMo** [7] is a method to learn an entity-aware extension of pretrained ELMo [33]. The model obtains significant improvements.
- **DCA** [34] is a global method which sequentially accumulates context information to make efficient, collective inference.
- **WNEL** [19] is a two-stages method which exploits unlabelled documents.
- **GENRE** [8] exploits a sequence-to-sequence architecture to generate entity names in an autoregressive fashion conditioned on the context.
- **NER-EL** [13] is a method that use NER classes to improve performance in low-data regimes.
- **EXTEND** [12] is Transformer-based architectures framing EL task as a text extraction problem.

Most EL systems actually make use of massive additional data and information originating from Wikipedia at training time, i.e., GENRE benefits greatly from drastically increasing the size of the training set from 18K to 9000K labelled instances, gaining almost 5 points. Our focus, however, is to improve EL model performance in low-data regimes. Among the above methods, only **GENRE**, **NER-EL** and **EXTEND** have low-data regimes setting.

Table 2. Results (InKB accuracy) on the in-domain settings when training on the low-data regime, i.e., AIDA-training only (right) and when using additional resources coming from Wikipedia (left). We mark in **bold** the best scores.

Model(high-data)	AIDA-B	Model(low-data)	AIDA-B
Global-RNN	90.7	GENRE	88.6
Deep-ed	92.2	NER-EL	89.0
WNED	89.0	EXTEND(base)	87.9
E-ELMo	93.5	*CDHCN(w/o MPP, w CAT)*	89.6
DCA	**93.7**	*CDHCN(w/o MPP, w MAT)*	89.7
WNEL	89.6	*CDHCN(w MPP, w/o CAT)*	89.2
GENRE	93.3	*CDHCN(w MPP, w CAT)*	90.0
EXTEND(large)	92.6	***CDHCN(w MPP, w MAT)***	**90.3**

Table 3. Results (InKB accuracy) on the out-of-domain settings when training on the low-data regime, i.e., AIDA-training only. We mark in **bold** the best scores.

	Model	MSNBC	AQUAINT	ACE2004	CWEB	WIKI
low-data	NER-EL	84.9	67.2	86.3	63.7	60.0
	CDHCN(w/o MPP, w CAT)	85.4	67.3	86.7	64.0	60.4
	CDHCN(w/o MPP, w MAT)	**86.5**	67.1	85.9	64.1	60.0
	CDHCN(w MPP, w/o CAT)	86.0	67.2	86.7	64.0	60.0
	CDHCN(w MPP, w CAT)	84.2	**67.7**	**87.5**	64.2	60.2
	CDHCN(w MPP, w MAT)	84.6	67.4	86.7	**64.6**	**60.7**

4.4 Experimental Results

We present the results of our approaches for EL on an in-domain dataset known as AIDA-B and discuss the merits of each contribution in turn. Then, we conduct experiments on out-of-domain datasets named MSNBC, AQUAINT, ACE2004, WNED-WIKI and WNED-CWEB to prove that our contributions are robust and beneficial for EL task in low-data regimes. In order to demonstrate the effects of each contribution, we compare five CDHCN variants. Specifically, MPP denotes the mention position prediction module, CAT denotes the context-aware type extraction module and MAT denotes the mention-aware type extraction module.

In-domain Results. We present the in-domain entity linking evalution results in low-data regimes in Table 2 (right). From Table 2 (right), we can observe that compared to other strong baseline methods, CDHCN (w MPP, w MAT) has state-of-the-art performance in low-data regimes and achieves 1.3% absolute improvement in terms of accuracy over the strong baseline [13]. On the AIDA-B dataset, CDHCN (w MPP, w CAT) performs better than other methods but slightly worse than CDHCN (w MPP, w MAT) in low-data regimes. Two CDHCNs (w/o MPP) all perform better compared to the strong baseline [13]. The performance of our model further increases to 90.0 and 90.3 respectively by combining auxiliary task MPP and two hierarchical constraint methods, indicating that our menthod enhances mention representation and makes use of entity type information to tackle EL task successfully. To further validate the effectiveness of our method, we also compare with baselines which are trained in high-data regimes in Table 2 (left). Although our method is trained in low-data regimes, it is still competitive and surpasses WNED [29] and WNEL [19] that are trained in high-data regimes.

Out-of-domain Results. Among all the baselines described in this paper, only GENRE, NER-EL and EXTEND have low-data regime settings. GENRE [8] and EXTEND [12] are trained based on Bart [35], and NER-EL is trained based on Bert [9]. Since Bart is more generalized than Bert, the models based on Bart can perform much better on out-of-domain datasets. Although our methods can also be realized on the Bart, in this paper, we only implemented on the Bert

Table 4. Per-class accuracy (%) of the entities on the AIDA-B and ACE2004.

	Entity type	AIDA-B			ACE2004		
		NER-EL	**CDHCN**	Δ	NER-EL	**CDHCN**	Δ
AIDA	PER	95.9	95.7	−0.2	88.9	88.9	+0.0
	ORG	81.4	83.9	+2.5	84.5	85.9	+1.4
	LOC	93.1	93.5	+0.4	87.7	89.0	+1.3
	ANIM	100.0	100.0	+0.0	–	–	–
	EVE	46.3	46.4	+0.1	–	–	–
	VEHI	100.0	100.0	+0.0	100.0	100.0	+0.0

and left the implementation on the Bart as the future work. So we evaluate five CDHCNs on the out-of-domain setting and compare it to NER-EL which is the only model based on Bert in low-data regimes. Similarly to the in-domain evaluations, our method consistently improve the results across five out-of-domain test sets. Table 3 shows that CDHCN (w MPP, w MAT) presents consistent performance in term of accuracy by 0.7% points compared to baseline methond [13] on WIKI. CDHCN (w/o MPP, w MAT) presents consistent performance by 1.6% in term of accuracy compared to baseline methond [13] on MSNBC. On the ACE2004 and AQUAINT datasets, CDHCN (w MPP, w CAT) achieves a 1.2 and 0.5% point improvement over baseline approaches [13]. The performance on out-of-domain datasets reveals the robustness of our model.

4.5 Type-Base Results

To better evaluate the role of entity types in our method, we analyze the accuracy of each entity type. Table 4 shows the results of a comparison between NER-EL [13] and CDHCN (w MPP, w MAT) when dealing with partial entity types. We can observe that our approach improves the results of most entity types. Under the type ORG, CDHCN pushes performances up by 2.5 and 1.4% on the AIDA-B and ACE2004 respectively. Under the type LOC, CDHCN performs slightly better than NER-EL by 0.4% on the AIDA-B but far better than NER-EL by 1.3%. There is a main reason that our improvement in mainly focused on ORG and LOC: some entity types features are difficult to extract since the description information of these entities has low degree distinction (e.g., PER). These type-based results corroborate our hypothesis that using external knowledge (i.e., entity type) can boost entity linking performance.

4.6 Case Study

To better explain the CDHCN model's research results, we conduct a case study. As shown in Fig. 5, the model without hierarchical constraints incorrectly predicts "HSBC" as "The Hongkong and Shanghai Banking Corporation", predicts

"Yale University" as "Yale University Press". The model can learn partial mention's entity type according to the mention context, however, it does not dare to choose the gold candidate entity because of the lack of specific candidate entities' types information. After learning entity type by giving hierarchical constraints, the model will tend to select the correct candidate entity whose entity type is the same as mention.

Text	Gold Entity	Prediction
HSBC has moved its Shanghai branch …	HSBC	NER-EL: The Hongkong and Shanghai Banking Corporation **CDHCN: HSBC**
… to head one of the departments at Yale University.	Yale University	NER-EL: Yale University Press **CDHCN: Yale University**
… International Airport in the southern Gaza Strip was …	Gaza Strip	NER-EL: Gaza **CDHCN: Gaza Strip**
The New York Times on Wednesday reported that al Faroon told …	The New York Times	NER-EL: Time (magazine) **CDHCN: The New York Times**

Fig. 5. Examples of sentences where CDHCN can correctly link entity in the KB, but NER-EL is not."NER-EL" and "CDHCN" stand for the baseline system and CDHCN (w MPP, w MAT), respectively.

5 Conclusions

In this paper, we studied the problem of entity linking in a low-data regime and proposed the CDHCN model, which can take into account the information from entity types. Specifically, to better capture information from entity type, we proposed a method to generate a class embedding for each mention and then used the embedding to constrain the similarity score between the mention and its candidate entities. Additionally, we utilized an auxiliary task called *Mention Position Prediction* to generate a more robust embedding for mention. Experiments on in-domain and out-of-domain datasets verified the effectiveness of CDHCN. Since we proposed to study entity type representations in low-data regimes for the first time, we hope this work could lead to more research in the future.

Acknowledgement. This research was partially supported by grant from the National Natural Science Foundation of China (Grant No. 61922073), and the University Synergy Innovation Program of Anhui Province (GXXT-2021-002).

References

1. Dubey, M., Banerjee, D., Chaudhuri, D., Lehmann, J.: EARL: joint entity and relation linking for question answering over knowledge graphs. In: Vrandečić, D., et al. (eds.) ISWC 2018. LNCS, vol. 11136, pp. 108–126. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00671-6_7

2. Ji, H., Grishman, R.: Knowledge base population: successful approaches and challenges. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 1148–1158 (2011)
3. Liu, Y., et al.: Technical phrase extraction for patent mining: a multi-level approach. In: ICDM, pp. 1142–1147. IEEE (2020)
4. Huang, Z., et al.: Disenqnet: disentangled representation learning for educational questions. In: SIGKDD, pp. 696–704 (2021)
5. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: ICML, pp. 1188–1196. PMLR (2014)
6. Ganea, O.-E., Hofmann, T.: Deep joint entity disambiguation with local neural attention. In: EMNLP, pp. 2619–2629 (2017)
7. Shahbazi, H., Fern, X.Z., Ghaeini, R., Obeidat, R., Tadepalli, P.: Entity-aware ELMO: learning contextual entity representation for entity disambiguation. arXiv preprint arXiv:1908.05762 (2019)
8. De Cao, N., Izacard, G., Riedel, S., Petroni, F.: Autoregressive entity retrieval. In: ICLR (2020)
9. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of NAACL-HLT, pp. 4171–4186 (2019)
10. Wu, L., Petroni, F., Josifoski, M., Riedel, S., Zettlemoyer, L.: Scalable zero-shot entity linking with dense entity retrieval. In: EMNLP, pp. 6397–6407 (2020)
11. Shrivatsa Bhargav, G.P., et al.: Zero-shot entity linking with less data. In: NAACL 2022, pp. 1681–1697 (2022)
12. Barba, E., Procopio, L., Navigli, R.: Extend: extractive entity disambiguation. In: ACL, pp. 2478–2488 (2022)
13. Tedeschi, S., Conia, S., Cecconi, F., Navigli, R.: Named entity recognition for entity linking: what works and what's next. In: EMNLP 2021, pp. 2584–2596 (2021)
14. Zhang, K., Zhang, H., Liu, Q., Zhao, H., Zhu, H., Chen, E.: Interactive attention transfer network for cross-domain sentiment classification. In: AAAI, vol. 33, pp. 5773–5780 (2019)
15. Zhang, K., et al.: Graph adaptive semantic transfer for cross-domain sentiment classification. In: ACM SIGIR, pp. 1566–1576 (2022)
16. Zhang, K.: Incorporating dynamic semantics into pre-trained language model for aspect-based sentiment analysis. In: ACL 2022, pp. 3599–3610 (2022)
17. Honglun Zhang, Liqiang Xiao, Wenqing Chen, Yongkun Wang, and Yaohui Jin. Multi-task label embedding for text classification. In EMNLP, pages 4545–4553, 2018
18. Le, P., Titov, I.: Improving entity linking by modeling latent relations between mentions. In: ACL, pp. 1595–1604 (2018)
19. Le, P., Titov, I.: Boosting entity linking performance by leveraging unlabeled documents. In: ACL, pp. 1935–1945 (2019)
20. Botha, J.A., Shan, Z., Gillick, D.: Entity linking in 100 languages. In: EMNLP, pp. 7833–7845 (2020)
21. Gu, Y., et al.: Read, retrospect, select: an MRC framework to short text entity linking. In: AAAI , vol. 35, pp. 12920–12928 (2021)
22. Jonathan Raiman and Olivier Raiman. Deeptype: multilingual entity linking by neural type system evolution. In AAAI, volume 32, 2018
23. Martins, Z., Marinho, P.H., Martins, A.F.T.: Joint learning of named entity recognition and entity linking. In: ACL, pp. 190–196 (2019)

24. Mrini, K., Nie, S., Gu, J., Wang, S., Sanjabi, M., Firooz, H.: Detection, disambiguation, re-ranking: autoregressive entity linking as a multi-task problem. In: ACL, pp. 1972–1983 (2022)
25. Chen, Y.: Convolutional neural network for sentence classification. Master's thesis, University of Waterloo (2015)
26. Tao, H., et al.: A schema-aware radical-guided associative model for Chinese text classification. IEEE TKDE, Learning from ideography and labels (2022)
27. Vaswani, A.. et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
28. Hoffart, J., et al.: Robust disambiguation of named entities in text. In: EMNLP, pp. 782–792 (2011)
29. Guo, Z., Barbosa, D.: Robust named entity disambiguation with random walks. Semantic Web **9**(4), 459–479 (2018)
30. Gabrilovich, E., Ringgaard, M., Subramanya, A.: Facc1: freebase annotation of clueweb corpora, version 1 (release date 2013–06-26, format version 1, correction level 0)
31. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (Poster) (2015)
32. Nguyen, T.H., Fauceglia, N.R., Muro, M.R., Hassanzadeh, O., Gliozzo, A., Sadoghi, M.: Joint learning of local and global features for entity linking via neural networks. In: COLING, pp. 2310–2320 (2016)
33. Peters, M., Neumann, M., Iyyer, M., Gardner, M., Zettlemoyer, L.: Deep contextualized word representations (2018)
34. Yang, X., et al.: Learning dynamic context augmentation for global entity linking. arXiv preprint arXiv:1909.02117 (2019)
35. Lewis, M., et al.: Bart: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: ACL, pp. 7871–7880 (2020)