End-to-End Entity Linking with Hierarchical Reinforcement Learning

Lihan Chen[1], **Tinghui Zhu**[1], **Jingping Liu**[2], **Jiaqing Liang**[3], **Yanghua Xiao**[1, 4*]
[1] Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
[2] East China University of Science and Technology, Shanghai, China
[3] School of Data Science, Fudan University, China
[4] Fudan-Aishu Cognitive Intelligence Joint Research Center
lhc825@gmail.com, thzhu22@fudan.edu.cn, jingpingliu@ecust.edu.cn, liangjiaqing@fudan.edu.cn, shawyh@fudan.edu.cn

Abstract

Entity linking (EL) is the task of linking the text segments to the referring entities in the knowledge graph, typically decomposed into mention detection, and entity disambiguation. Compared to traditional methods treating the two tasks separately, recent end-to-end entity linking methods exploit the mutual dependency between mentions and entities to achieve better performance. However, existing end-to-end EL methods have problems utilizing the dependency of mentions and entities in the task. To this end, we propose to model the EL task as a hierarchical decision-making process and design a hierarchical reinforcement learning algorithm to solve the problem. We conduct extensive experiments to show that the proposed method achieves state-of-the-art performance in several EL benchmark datasets. Our code is publicly available at https://github.com/lhlclhl/he2eel.

Introduction

Entity linking (EL) is the task of linking the text segments to the referring entities in the knowledge graphs (KG). EL is a fundamental task for machines to understand natural language via KGs; thus, it has been drawing much interest in research in the past decade (Hoffart et al. 2011; Kolitsas et al. 2018; Zhang et al. 2022). EL consists of two major steps: *mention detection* (MD, which identifies text segments referring to entities) and *entity disambiguation* (ED, which disambiguates mentions to respective entities). Traditional methods (Steinmetz et al. 2013; Moro et al. 2014) solve EL in a pipeline manner, treating MD and ED independently, which ignores the mutual dependency between these two tasks. To overcome this weakness, end-to-end solutions dominate recent efforts by solving EL with one model via joint training (Févry et al. 2020; De Cao et al. 2021b) or joint modeling (De Cao et al. 2021a; Mrini et al. 2022).

However, existing end-to-end EL methods have difficulty in fully exploiting the dependency residing in mentions and entities. First, the mutual *inter-dependency* between mentions and entities is not easy to be exploited. Most methods with MD-to-ED inference order ignore the dependency of mentions on entities. For example, typical multi-task learning models (Martins et al. 2019; Ravi et al. 2021) only exploit the dependency between MD and ED implied by the shared representation. However, these models are only trained with independent objectives for individual tasks, leaving the detection of mentions independent from ED results. Other works (Kolitsas et al. 2018; Zhang et al. 2022) perform ED for all candidate mentions and use ED scores to identify mentions. These methods take advantage of the explicit effects of entities on mentions but ignore that ED actually depends on the accurate identification of mention boundaries. Without considering MD results, they will produce ED errors due to noisy candidate mentions. Second, existing EL methods also have problems in utilizing the *intra-dependency* within mentions (or entities). Most methods model MD/ED in a document as multiple independent tasks (De Cao et al. 2021b; Zhang et al. 2022) instead of a single joint task. They obviously ignore that the detection/disambiguation of individual mentions/entities depends on other mentions/entities in a context. Other methods utilizing global context information suffer from either noisy candidate mentions and entities (Kolitsas et al. 2018) or the exposure bias problem of long-term decision-making (De Cao et al. 2021a; Mrini et al. 2022).

To address the above dependency problems, we propose to model the EL task as a *hierarchical sequential decision-making process*. Considering human readers annotating entities in the text (e.g., “Jobs founded Apple”), as shown in Figure 1, instead of detecting all the mentions first and then linking them to a knowledge graph, they will spot one mention first (e.g., “Jobs”) and try to decide what it refers to (e.g., “Steve Jobs”), after that they will then repeatedly spot another one (e.g., “Apple”) and do the same thing. Thus, for a human, the EL task is a sequence of *subtasks* (Pateria et al. 2021) of linking individual mentions to entities. Based on this intuition, MD is a high-level sequential decision-making process with each step choosing a *subtask* (i.e., identifying a mention to link), while ED for the mention can be viewed as making low-level decisions by completing the *subtask*. For low-level ED, we adopt the state-of-the-art generative entity retrieval (GENRE) (De Cao et al. 2021a), which is also a sequential decision-making process.

We then formalize the EL task as a Markov decision process (MDP) and adopt hierarchical reinforcement learning (HRL) algorithms (Pateria et al. 2021) to solve it. This solution deals with the above dependency problems from two

*Corresponding authors.

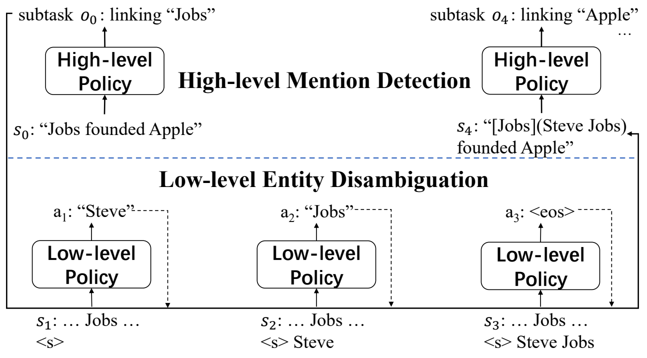
subtask o_0: linking "Jobs"
subtask o_4: linking "Apple"
...
High-level Policy
High-level Mention Detection
High-level Policy
s_0: "Jobs founded Apple"
s_4: "[Jobs](Steve Jobs) founded Apple"
Low-level Entity Disambiguation
a_1: "Steve"
a_2: "Jobs"
a_3: <eos>
Low-level Policy
Low-level Policy
Low-level Policy
s_1: ... Jobs ... <s>
s_2: ... Jobs ... <s> Steve
s_3: ... Jobs ... <s> Steve Jobs

Figure 1: End-to-end entity linking task as a hierarchical decision-making process.

aspects. First, the hierarchical framework fully expresses the two-way *inter-dependency* between mentions and entities. The low-level ED depends on the mention detected by the high-level decision, while the high-level RL for MD will be optimized to maximize the EL task reward in terms of the outcome of ED, which enforces MD to be dependent on the low-level ED. Second, the *intra-dependency* within mentions (or entities) is exploited by the mention-by-mention sequential decision-making process.

Based on the HRL framework, we construct a deep RL agent with a hierarchical policy optimized for EL. In general, the intrinsic properties of EL tasks pose great challenges for hierarchical policy learning. First, high-level policy learning is unstable due to the variant extrinsic cumulative reward caused by the variant number of mentions in a document. A standard solution is to utilize the value function as a baseline in policy learning (Sutton and Barto 2018), but learning a robust value function via a naive approach is still difficult. We propose a novel value function approximation approach specifically for EL to achieve more accurate value estimation and further reduce the influence of variant reward during training. Second, in order to adapt the low-level policy for ED as generative entity retrieval, we adopt offline reinforcement learning (Levine et al. 2020) to take advantage of candidate entities for efficient optimization.

In summary, our contribution is threefold.

- As far as we know, we are the first to model EL task as a hierarchical decision-making process. This framework fully exploits the dependency in mentions and entities.
- Based on the proposed framework, we design a novel HRL algorithm to solve the EL problem. We propose a specific value function approximation method for EL to tackle the challenges of variant extrinsic reward in hierarchical policy optimization.
- Extensive experiments show that our method achieves state-of-the-art performance in EL benchmark datasets.

Overview

Problem Definition

The input of EL is a text document (including short phrases, like a query or a tweet) given as a sequence $X = x_1 x_2 ... x_n$ of words from a dictionary, i.e., $x_k \in \mathcal{V}$. The goal of EL is to detect the mentions, $M = [m_1, m_2, ..., m_T]$, where each mention is a word subsequence of the input document, i.e. $m_i = x_p ... x_q \ (1 \leq p \leq q \leq n), \forall 1 \leq i \leq T$, referring to entities, and to link the mentions to corresponding entities, $E = [e_1, e_2, ..., e_T]$, where each entity is an entry in the given KG, i.e., $e_i \in \mathcal{E}, \forall 1 \leq i \leq T$. For example, given a text "Jobs founded Apple" as input, the output of the task is [(Jobs, Steve Jobs), (Apple, Apple Inc.)], where "Jobs" and "Apple" are mentions in the text and "Steve Jobs" and "Apple Inc." are entities in KG.

Solution Framework

As shown in Figure 1, we model the EL task as a hierarchical decision-making process. Specifically, we define the *subtask* as linking a given mention to an entity in KG (e.g., linking "Jobs" to the correct entity). Then, the MD procedure is a high-level decision-making process of sequentially identifying mentions as subtasks, and the ED procedure is a low-level decision-making process of finishing subtasks by autoregressively generating entity names. To this end, we formalize EL as an MDP and design a hierarchical reinforcement learning algorithm to solve it. We build an HRL agent consisting of two primary modules, as shown in Figure 2. The agent is trained to find an optimized *high-level policy* for mention detection and a *low-level policy* for entity disambiguation. For each episode (EL process for a document), the agent first identifies a mention as a subtask according to the high-level policy. After the mention is chosen, the agent then disambiguates the mention as a specific entity by sequentially generating the entity name according to the low-level policy. Once the low-level policy arrives at a terminal, the high-level policy retakes control and resumes the above procedure until the high-level terminal action is chosen. Unlike those architectures only considers the separate task information for MD, this high-level policy aims to maximize the reward of the global task objective (Pateria et al. 2021), which takes into account the results of low-level ED.

Methodology

We elaborate on the details of our method in this section. We first describe the MDP components of the hierarchical decision-making process. After that, we then introduce our algorithm for hierarchical policy learning.

High-level Mention Detection

The process of high-level decision-making is to detect mentions from the text sequentially. This section first introduces the definition and corresponding representation for its MDP components, including state, action and reward, then formalizes the parameterized high-level policy π_ω.

State The high-level decision points exist only in part of MDP states (s_0 and s_4 in the example of Figure 1). Thus, from the view of high-level policy, the state contains the input sequence as well as the detected mentions (e.g., "Jobs") and corresponding linked entities (e.g., "Steve Jobs") from past actions. To represent the state $s_t \in S$ of high-level

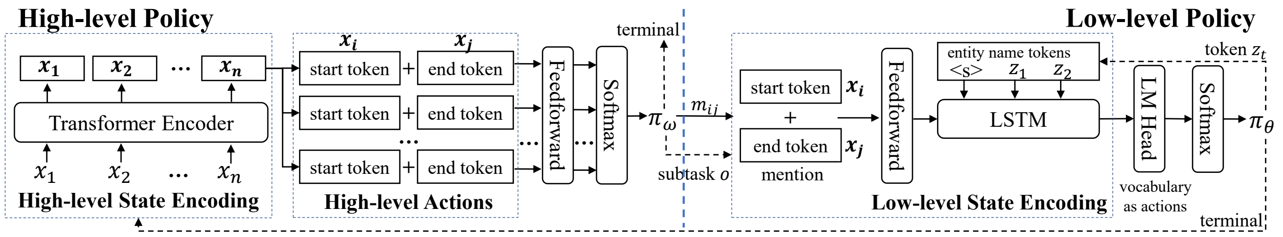

Figure 2: The architecture of the hierarchical policy of the HRL agent. The high-level policy identifies mentions as subtasks, and the low-level policy executes the given subtasks by linking mentions via generative entity retrieval.

decision points at time step t, we adopt the markup language (De Cao et al. 2021a) that marks the detected mention and adds the corresponding linked entities in the input text sequence. Figure 1 presents an example. From the high-level view, the procedure usually starts with an initial state of plain text without any markups ("Jobs founded Apple" as s_0). Then, after the first subtask (o_0 to link "Jobs") is chosen and finished (via low-level policy), the next state of high-level view is represented as a markup text ("[Jobs] (Steve Jobs) founded Apple" as s_4). To provide a rich state representation for long input texts like documents, we encode it using a Longformer (Beltagy, Peters, and Cohan 2020), which is a Transformer pre-trained with a masked language model objective designed to support long sequences. Thus, as shown in Figure 2, the encoding of a state $s \in S$ is a sequence of contextualized token embeddings computed from the Longformer encoder, i.e., $\boldsymbol{s} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n]$.

Action For a state $s_t \in S$ in a high-level decision point, an action $o_t \in O$ is to choose a mention as a subtask, e.g., linking "Jobs" as o_0 in Figure 1. As a subtask, the high-level action o_t is technically the action defined in the Semi-Markov decision process (SMDP) (Baykal-Gürsoy 2010), where the concept of *time* is involved for the execution of an action after it has been chosen, e.g., the execution of o_0 takes four time steps. We then denote $c(o_t)$ as the number of time steps for which o_t is executed. Specifically, o_t is chosen from the set O_t of all the candidate mentions plus a terminal action, i.e., $O_t = \{o_t | o_t = m_{ij} \vee o_t = \eta\} \subset O$, where $m_{ij} = x_i ... x_j$ $(1 \leq i \leq j \leq n)$ denotes a possible mention from the input text and η is a special token indicating terminal. The possible mentions come from all the text spans within a specific length limit, except for the mentions and entities already marked up in the state representation. As shown in Figure 2, an action encoding $\boldsymbol{o}_t$ corresponding to a mention m_{ij} is calculated as the concatenation of contextualized encodings of the start token $\boldsymbol{x}_i$ and the end token $\boldsymbol{x}_j$, i.e., $\boldsymbol{o}_t = [\boldsymbol{x}_i; \boldsymbol{x}_j]$. For the encoding of the terminal action η, we use the "CLS" token encoding in state representation.

Reward The reward is defined as the feedback after an action is executed. For an agent that only performs mention detection, the feedback can be immediately received when an action o is chosen. However, in the hierarchical framework, the function of the high-level policy is not only detecting a mention, but acting as a global strategy optimizing for the whole EL task. Thus, the high-level reward should respond to the outcome of performing the subtask, i.e., the execution of the low-level actions (Pateria et al. 2021). We define the reward received from choosing and executing o based on s and arriving at s' as:

$$r^h(s, o, s') \begin{cases} 1, \text{if the linked entity for } o \text{ is correct} \\ 0, \text{if } o = \eta \\ \delta^h, \text{if the linked entity for } o \text{ is wrong} \end{cases}, \quad (1)$$

where $\delta^h \leq 0$ is the hyper-parameter as the negative reward for incorrectly linked entities. For a high-level state-action trajectory $s_0, o_0, s_{c(o_0)}, ...$, we use r^h_t to denote $r^h(s_t, o_t, s_{t+c(o_t)})$ for simplicity.

Policy The policy is a mapping from state to action. The stochastic high-level policy $\pi_\omega : S \times O \mapsto [0, 1]$ specifies a probability distribution over actions given the state:

$$\pi_\omega(o|s_t) = \frac{\exp f(\boldsymbol{o})}{\sum_{o' \in O_t} \exp f(\boldsymbol{o'})}, \quad (2)$$

where f is a feedforward neural network (FNN) with output dimension one and ω represents parameters of the policy network, including the parameters of f and the encoder.

Low-level Entity Disambiguation

Once the high-level policy has chosen a mention, the low-level policy will be triggered to link the detected mention to an entity. To this end, we adopt state-of-the-art generative entity retrieval (De Cao et al. 2021a) and present the details of this process from the view of the low-level policy π_θ. The notation t in this section is re-used as the low-level time step.

State After a subtask o is chosen, the agent takes a special transition to a low-level state s^o_1 (shown as s_1 in Figure 1). The low-level state $s^o_t \in S$ under subtask o consists of the context of the mention and the generated token sequence of entity name before the low-level time step t from the start of this subtask, i.e., $z_0 z_1 ... z_{t-1}$, where $z_0 = \langle \text{s} \rangle$ is a pre-defined start token. Following (De Cao et al. 2021b), we use a long-short term memory (LSTM) network (Hochreiter and Schmidhuber 1997) to represent the entity token sequence, as shown in Figure 2. Specifically, we first project the subtask encoding $\boldsymbol{o}$ with an FNN g to the initial state of LSTM, i.e., $\boldsymbol{h}_0 = g(\mathbf{o})$. Then we represent s^o_t as the LSTM output state at time step t: $\langle \boldsymbol{s}^o_t, \boldsymbol{h}_t \rangle = \text{LSTM}(\boldsymbol{h}_{t-1}, z_{t-1})$.

Action The action of the low-level decision-making process is to generate the next word of the entity. Thus, the action space is defined as $A = \mathcal{V}$, where $\mathcal{V}$ denotes the token vocabulary, including an end-of-sentence token (EOS) representing the termination of the low-level decision process.

Reward For low-level actions, the reward is zero until the entity name generation procedure is complete; thus, the low-level reward $r^o(s^o, a)$ on subtask o is defined as:

$$r^o(s^o, a) \begin{cases} I(s^o) + \delta^l(1 - I(s^o)), \text{if } a = \texttt{EOS} \\ 0, \text{otherwise} \end{cases}, \quad (3)$$

where $I(s^o) \in \{0, 1\}$ is the identifier determining whether the generated entity in s^o is equal to the ground truth entity of subtask o, and $\delta^l \leq 1$ is a hyper-parameter representing the reward for an incorrectly linked entity.

Policy Similar to high-level policy, we define the low-level policy as a stochastic policy, i.e., $\pi_\theta : S \times A \mapsto [0, 1]$. Specifically, we use the language model head of Longformer, with parameter matrix $\boldsymbol{W}$ and $\boldsymbol{b}$, to project the state encoding to the action distribution:

$$a \sim \pi_\theta(\cdot | s_t^o) = \text{softmax}(\boldsymbol{W} \boldsymbol{s}_t^o + \boldsymbol{b}), \quad (4)$$

where θ represents parameters of this policy network, including $\boldsymbol{W}$, $\boldsymbol{b}$, and parameters of LSTM, g, and the Longformer encoder.

Hierarchical Policy Optimization

In this section, we first elaborate on the details of high-level and low-level policy learning, respectively, then introduce the training algorithm for hierarchical policy optimization.

High-level Policy Learning Objective The objective of the high-level policy is to acquire the maximum expected cumulative reward r^h from the environment. Thus, the optimization objective of the high-level policy is directly formalized as finding the parameters ω that maximizes the expected cumulative reward of trajectories[1] under π_ω:

$$J(\pi_\omega) = \mathbb{E}_{s_0, o_0, s_1, \ldots, s_T \sim \pi_\omega}[\sum_{t=0}^{T-1} \gamma^t r_t^h], \quad (5)$$

where π_ω is parameterized by ω, γ is a discount factor, and T is the terminal time step of an episode. An intuitive solution to optimize this objective is REINFORCE algorithm (Williams 1992), which updates the policy with its gradients via Monte Carlo sampling:

$$\nabla_\omega J(\pi_\omega) = \mathbb{E}_{s_0, o_0, s_1, \ldots, s_T \sim \pi_\omega}[\sum_{t=0}^{T-1} G_t \nabla_\omega \log \pi_\omega(o_t | s_t)], \quad (6)$$

where $G_t = \sum_{k=t}^{T-1} \gamma^{k-t} r_k^h$ is the return, i.e., discounted cumulative reward, from t to terminal time step T.

However, in our EL problem, the return G_t is of high variance because the number of mentions in input documents is various. We adopt actor-critic methods to address this problem and to further enable bootstrapping with temporal-difference learning (Sutton and Barto 2018). Specifically, in order to update the policy π_ω, i.e., the actor, we optimize the objective by introducing an estimated value function V_ϕ parameterized by ϕ, as a *critic*, and further use the current state value as a baseline:

$$\begin{aligned} \nabla_\omega J(\pi_\omega) = \mathbb{E}_{s_t, o_t, s_{t+c(o_t)}, r_t^h \sim \pi_\omega}[(r_t^h + \gamma V_\phi(s_{t+c(o_t)}) - \\ V_\phi(s_t)) \nabla_\omega \log \pi_\omega(o_t | s_t)] \end{aligned}. \quad (7)$$

The estimated value function $V_\phi(s)$ approximates the expected return from state s under policy π_ω, i.e. $V(s) = \mathbb{E}_{\pi_\omega}(G_t | s_t = s)$. Actor-critic methods usually simultaneously learn the policy π_ω and the value function V_ϕ.

In fact, in the entity linking task, the state value is hard to estimate because the representation of state s is based on the average encoding of the sequence, while the state value is of high variance and depends on many factors such as the length of the text and the number of mentions. To this end, we design a novel approach to compute the value function $V_\phi(s)$ via a pipeline inference for the state. Specifically, we obtain a mention set $M_s = [m_1, m_2, ...]$ by sequentially choosing non-overlapping mentions with maximum probability and the probability larger than the terminal action, i.e., $\pi_\omega(o|s) > \pi_\omega(\eta|s)$, only based on state s. Then, for each $o \in M_s$, we execute π_θ to perform ED and obtain a corresponding set of low-level terminal states of linked entities, $E_s = [s_{t_1}^{m_1}, s_{t_2}^{m_2}, ...]$. Note that the low-level ED for those mentions is based on the same state, which can be efficiently computed in parallel. Thus, the value function can be approximated as:

$$V_\phi(s) = \sum_{i=1}^{|E_s|} [D(\boldsymbol{s}_{t_i}^{m_i}) + \delta^h(1 - D(\boldsymbol{s}_{t_i}^{m_i}))], \quad (8)$$

where $D(\boldsymbol{s}_{t_i}^{m_i}) \in [0, 1]$ is a discriminator realized by an FNN ϕ. We directly train $D(\cdot)$ for each subtask with the ground truth entity via binary cross-entropy objective. This formalization of the value function is easy to estimate since only a discriminator for the individual mention-entity pair is additionally required. Moreover, it makes the policy learning algorithm more intuitive. Based on this design, the intuition of the term $r_t^h + \gamma V_\phi(s_{t+c(o_t)}) - V_\phi(s_t)$ can be regarded as the performance gain of completing a subtask (o_t) first ($r_t^h + \gamma V_\phi(s_{t+c(o_t)})$) over directly performing a pipeline inference ($V_\phi(s_t)$). As a result, Eq (7) encourages the subtask o_t that is helpful for future linking decisions and discourages those that are not.

Low-level Policy Learning Objective The policy learning for the low-level entity disambiguation is a much simpler task because the number of the decision time step is much smaller. Since the gold entities and candidate entities are generated in the training set, we treat them as trajectories that have already been acquired in the RL problem. Thus, the optimization of the low-level policy is an offline reinforcement learning (Levine et al. 2020) problem where we can optimize the policy with off-policy policy gradient

[1]Note that state subscript indices for a high-level trajectory are not consecutive, we write like that here for simplicity.

Algorithm 1: Hierarchical Policy Optimization for EL

```
Input: Training set with samples $\{(X, Y)\}$
Output: Model parameters $\omega$, $\phi$, $\theta$
 1: Initialize parameters $\omega$, $\phi$, and $\theta$
 2: for each sample $(X, Y)$ do
 3:     Initial state $s \leftarrow X$
 4:     while $s$ is non-terminal do
 5:         Select a high-level subtask $o$ based on $\pi_\omega$
 6:         Execute subtask $o$ based on $\pi_\theta$ obtaining entity $e$
 7:         $s' \leftarrow s$ with annotation $o, e$
 8:         $r \leftarrow r^h(s, o, s')$
 9:         $M \leftarrow M \cup \{(s, o, s', r, Y)\}$
10:         Sample a batch $B = (s_B, o_B, s'_B, r_B, Y_B)$ from $M$
11:         Update $\omega$ using $B$ with Eq (7)
12:         Update $\phi$ and $\theta$ using subtasks in $s_B$ with $Y_B$
13:         $s \leftarrow s'$
14:     end while
15: end for
```

via importance sampling based on these acquired trajectories without more exploration:

$$\nabla_\theta J(\pi_\theta) = \sum_{s_1^o, a_1^o, \ldots, s_T^o \sim \mathcal{D}} \sum_{t=1}^{T-1} G_t^o \frac{\pi_\theta(a_t|s_t^o)}{\pi_\beta(a_t|s_t^o)} \nabla_\theta \log \pi_\theta(a_t|s_t^o), \tag{9}$$

where o is the chosen mention, and $\mathcal{D}$ is the offline acquired trajectory set. Since the reward is 0 except for the final state, the low-level return G_t^o is the final low-level reward of that trajectory. π_β is the behavior policy that acquires the trajectories, which we estimate with another network trained by candidate entities in the training set via supervised learning.

Algorithm for Hierarchical Policy Learning We illustrate our training process in Algorithm 1. We regard each document-annotation sample in the dataset as an episode (line 2). For each high-level step in an episode (line 4), we first choose a mention o as a subtask via the high-level policy π_ω (line 5) and then complete the subtask via the low-level policy π_θ by generating the entity name (line 6). After that, the agent get to the next state s' (line 7) and receives a reward r (line 8). We push the high-level tuple with the label Y (s, o, s', r, Y) into the replay memory (line 9) and sample a batch of tuples from it (line 10). We first update the high-level policy π_ω by Eq (7) with the batch of tuples (line 11). Then, with the states of the sampled batch, we update the low-level policy π_θ and the discriminator D of the value function on all the annotated mentions (subtasks) (line 12).

Experiments

In this section, we present empirical results of our method and verifications of our designs on several benchmark datasets compared with state-of-the-art EL methods.

Experimental Setup

We use the standard English AIDA-CoNLL splits (Hoffart et al. 2011) for training, validation, and in-domain test. AIDA provides full supervision for both MD and ED. In addition, we further evaluate our method on four out-of-domain test sets: N3-RSS-500 (R500) (Röder et al. 2014), OKE challenge 2015 and 2016 (OKE15 and OKE16) (Nuzzolese et al. 2015), and Derczynski (Der) (Derczynski et al. 2015). Following most previous works (Zhang et al. 2022; Mrini et al. 2022), we report the InKB Micro-F$_1$ score as the evaluation metric. As in previous approaches, we assume the availability of a pre-computed set of candidates for entity disambiguation, for which we use the same candidate generation as (Kolitsas et al. 2018). We also use these candidates for offline RL training of low-level policy and to provide negative samples for training D in the value function. For pre-training, we use the Wikipedia corpus (De Cao et al. 2021a) with supervised multi-task learning objectives like (De Cao et al. 2021b). The detailed settings including dataset statistics, training details and hyper-parameters settings are presented in supplementary materials.

Method	Micro-F$_1$
Pipeline Entity Linking	
(Hoffart et al. 2011)	72.8
(Steinmetz et al. 2013)	42.3
(Moro et al. 2014)	48.5
(Kolitsas et al. 2018) w/ NER	74.6
(van Hulst et al. 2020)	80.5
End-to-end Entity Linking	
(Kolitsas et al. 2018)	82.4
(Broscheit 2019)	79.3
(Martins et al. 2019)	81.9
(Févry et al. 2020)	76.7
(Ravi et al. 2021)	83.1
(De Cao et al. 2021a)	83.7
(De Cao et al. 2021b)	85.5
(Mrini et al. 2022)	85.7
(Zhang et al. 2022)	85.8
Ours (PI)	87.1
Ours (HI)	**87.6**

Table 1: Evaluation results on in-domain test set.

Overall Evaluation

Table 1 summarizes the results of our method on the in-domain AIDA test set. We provide two inference settings for our methods, *pipeline inference* (PI) and *hierarchical inference* (HI). The first setting is the same as previous multi-task learning methods (Févry et al. 2020; Ravi et al. 2021), where we first predict all the mentions using the high-level policy and then predict entities for the detected mentions. The second setting follows the procedure during our HRL training, where we switch the high-level mention detection and low-level ED iteratively. The advantage of the first setting is *efficiency* since it only has to encode the context once. The advantage of the second setting is *effectiveness* since it makes decisions at each time step by relying on the results of previous decisions. Results show that our method with HI reduces the Micro-F$_1$ error from the previous state-of-the-art method by 12%. We find that our RL training can also

Method	R500	OKE15	OKE16	Der	Avg.
(Hoffart et al. 2011)	<u>42.4</u>	**63.1**	0.0	32.6	34.5
(Steinmetz et al. 2013)	20.5	46.2	46.4	26.5	34.9
(Moro et al. 2014)	29.1	41.9	37.7	29.8	34.6
(Kolitsas et al. 2018)	38.2	<u>61.9</u>	52.7	34.1	46.7
(van Hulst et al. 2020)	35.0	**63.1**	**58.3**	41.1	49.4
(De Cao et al. 2021a)	40.3	56.1	50.0	<u>54.1</u>	50.1
(Zhang et al. 2022)	41.9	61.1	51.3	52.9	<u>51.8</u>
Ours	**42.8**	<u>61.9</u>	<u>52.9</u>	**55.9**	**53.4**

Table 2: Micro-F_1 score on out-of-domain test sets. Bold indicates the best, and underline indicates the second best.

Method	F_1^{MD}	F_1^{ED}
(Kolitsas et al. 2018)	-	87.3
(Broscheit 2019)	-	87.9
(van Hulst et al. 2020)	-	89.4
(Ravi et al. 2021)	-	85.7
(De Cao et al. 2021a)	-	**93.3**
(De Cao et al. 2021b)	93.5	91.5
(De Cao et al. 2021b) w/ pre-training	93.5	92.1
HRL (PI)	94.3	92.4
HRL (HI)	**94.4**	92.8

Table 3: Decomposed task results on the AIDA test set.

improve the performance in PI setting, which is useful in practice because the sequential decision-making of mention detection is a time-consuming inference process. Due to the similar architecture, the pipeline inference is as efficient as (De Cao et al. 2021b); the hierarchical inference is around 20 times slower than the pipeline inference on the validation set because of its mention-by-mention prediction.

To validate the generalization ability of our method, we further conduct experiments on out-of-domain datasets. The results are presented in Table 2. Our method achieves the best macro-averaged Micro-F_1 score across the four out-of-domain evaluation datasets (+1.6). Specifically, our method achieves the best test Micro-F_1 scores for Derczynski (+1.8) and N3-RSS-500 (+0.4) and is also competitive on the other two (close second-best on OKE15 and OKE16). We observe that the performance of our method lags behind pipeline methods (e.g., (van Hulst et al. 2020)) in OKE15 and OKE16. It is because of the domain shift of the entity annotation rules between these two datasets and the training set. These datasets are annotated with many coreferences (e.g., pronouns) and common nouns (e.g., "professors" and "director"), while our model was not explicitly trained for them. Nevertheless, pipeline methods with more generalized MD modules are more robust to those changes.

Decomposed Task Evaluation

To provide a clear view of how our method works, we present its performance on decomposed tasks, i.e., MD and ED, for a detailed analysis of the AIDA test set. The EL task scores a prediction correct when both MD and ED are done correctly, i.e., $F_1^{EL} \approx F_1^{MD} \times F_1^{ED}$. Unfortunately, this decomposition is not usually reported in previous end-to-end EL literature, so it is not easy to systematically investigate how our method performs from the decomposed views of MD and ED. Thus, we compare several systems that report the ED performance (Kolitsas et al. 2018; Broscheit 2019; van Hulst et al. 2020; Ravi et al. 2021; De Cao et al. 2021a), and the re-run version of paralleled autoregressive EL (De Cao et al. 2021b) with pre-training on Wikipedia anchor texts. The results are presented in Table 3. The upper part of the table reports results from previous literature; the lower part presents the results from our experiments. The results of baselines (De Cao et al. 2021b) in the lower part are produced by re-running their source code in our experiments. Our method achieves the best performance on MD and competitive performance on ED. The inferior performance on ED results from the limited capacity of our generative entity retriever. Our low-level policy based on LSTM (around 10M parameters) is much smaller than the BART (Lewis et al. 2020) decoder (over 50M parameters) used in (De Cao et al. 2021a)). We believe a larger model for ED can bridge the performance gap, which we will leave for future work.

Ablation Study We further conduct ablation analysis based on the comparison with (De Cao et al. 2021b) which has the similar model architecture to ours while different in the training and inference mechanism. First, the pre-training improves the performance of ED ($91.5 \rightarrow 92.1$) while having no effects on MD. The performance gain in ED is predictable because autoregressive generation of entity names heavily relies on seen entities from a large corpus. Since the annotation rule for mentions varies from different tasks (AIDA only concerns named entities), the invariance of MD performance is also reasonable. Second, in the context of the same pipeline inference mechanism, the superiority of our method over paralleled autoregressive EL is mainly on MD ($93.5 \rightarrow 94.3$). It is because our RL training strategy gives the mention detector a high-level view, optimized based on the global task with the consideration of the dependency on low-level ED. It thus validates the effectiveness of our method in exploiting the *inter-dependency* between mentions and entities. Third, in the context of the same model, the hierarchical inference strategy has mainly positive influence on the performance of ED ($92.4 \rightarrow 92.8$). The reason is that the hierarchical inference takes the global ED assumption (Ratinov et al. 2011) and considers the dependency among the linking decisions of different mentions in a context instead of treating each linking task independently like PI. It thus validates the effectiveness of our method in exploiting the *intra-dependency* within entities.

Value Function Evaluation

To overcome the problem of value function approximation, we propose a particular design to estimate the value via a pipeline inference. In this section, we conduct experiments to show the advantage of our design compared to naive value function approximation. We consider two baselines for comparison, the naive actor-critic method with baseline (AC+BL), where we replace our value function in Eq (8)

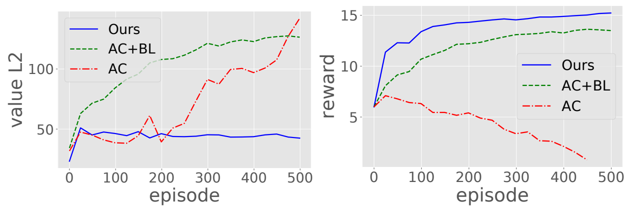

Figure 3: Comparison between our value function approximation and the standard baselines.

Error	Example (**G**: ground truth; **P**: prediction)
MD miss (150)	**G**: off-spinner [such]$_{\text{Peter Such}}$ had scuttled their ... **P**: off-spinner such had scuttled their hopes
MD error (288)	**G**: another against [british universities]$_{\text{BU cricket team}}$ **P**: another against [british]$_{\text{United Kingdom}}$ universities
ED error (208)	**G**: the team is as follows: [given]$_{\text{Shay Given}}$, ... **P**: the team is as follows: [given]$_{\text{Given name}}$, ...

Table 4: Error analysis on the validation set.

with a naive value network, and the actor-critic method without baseline (AC), where we train a naive value network for bootstrapping but not use it as a baseline in Eq (7). The naive value network implemented as an FNN takes the "`CLS`" token encoding of the state as input and outputs a scalar as the estimated value. We approximate the naive value function with a standard actor-critic algorithm (Sutton and Barto 2018). To prove the effectiveness of our value function design, we evaluate them with two metrics on the validation set during training, 1) L2 distance between the estimated value and the actual return, and 2) average reward. The former shows the fidelity of value networks, and the latter shows the effects of value networks on high-level policy learning. We present the results in Figure 3. The results show that the learning process quickly fails without the variance control of the baseline, and our design of the value network has better fidelity and more positive influence on policy learning.

Error Analysis

We conduct a qualitative experiment and analyze typical errors of our method in this section. Although achieving a high Micro-F$_1$ score of 95.5% on MD and 91.2% on EL, our method still got some typical errors. To better understand those errors, we examine the error cases and divide them into three categories: 1) *MD miss*, where the gold mentions are not detected by our method; 2) *MD error*, where our method detects wrong mentions; and 3) *ED error*, where the detected mentions are correct, but linked entities are wrong. We present the qualitative results in Table 4. Most mentions our method misses are those sharing the same form of polysemous words. Many MD errors come from missing annotations or out-of-KB annotations we do not count for the EL evaluation. Another typical MD error is the nested annotation, where our method detects a part of a gold mention. Finally, ED errors are induced by various reasons. We hypothesize that it is because of the limited capacity of low-level policy. We believe a larger model will bring a performance gain, which we will leave for future work.

Related Work

End-to-end Entity Linking EL is composed of mention detection (MD) and entity disambiguation (ED). Traditional methods (Hoffart et al. 2011; Steinmetz et al. 2013; Moro et al. 2014; van Hulst et al. 2020) treat these sub-tasks separately, training different modules. More modern approaches adopt multi-task learning of MD and ED to enforce them to benefit from each other (Martins et al. 2019; Broscheit 2019; De Cao et al. 2021b; Ravi et al. 2021). Because of the separate training objective and pipeline inference process, they only consider the dependency in the representation while ignoring the dependency in the goals. To make MD explicitly benefits from ED, some works (Kolitsas et al. 2018; Chen et al. 2018; Zhang et al. 2022) first perform ED for all possible mentions and then use the linking score to guide the mention detection. Without considering MD, the mention boundary is hard to identify accurately, and the explicit effect from MD to ED is usually ignored. Another line of work (De Cao et al. 2021a; Mrini et al. 2022) utilizes EL for all mention-entity pairs jointly by autoregressively generating a version of the input markup-annotated with the entities' unique identifiers expressed in natural language. But this complicated formalization will suffer from the exposure bias problem of generation models.

Reinforcement Learning Previous works have shown that reinforcement learning methods are good at exploring and making decisions in large search spaces (Mnih et al. 2013; Silver et al. 2017). It has been widely applied in NLP and KG-related tasks that can be modeled as sequential decision-making problems, such as text generation (Kostrikov, Nair, and Levine 2021), reasoning (Hou et al. 2021) and recommendation (Xian et al. 2019). It has also been used to solve the global entity disambiguation problem (Fang et al. 2019), where an RL agent is trained to disambiguate mentions in the context sequentially. As far as we know, we are the first to apply RL in end-to-end EL literature. Hierarchical RL (Pateria et al. 2021) has also been applied to tasks with complicated formalization, such as relation extraction (Takanobu et al. 2019), and KG reasoning with fine-grained relations (Wan et al. 2020).

Conclusion

This paper presents a novel hierarchical framework to formalize the entity linking problem. In the proposed framework, mention detection is a high-level decision-making process that identifies mentions as subtasks; entity disambiguation is a low-level decision-making process that finishes subtasks by linking mentions to entities. This hierarchical framework fully exploits the dependency lying in mentions and entities. Based on the novel framework, we design a hierarchical RL algorithm to solve EL. Extensive experiments prove the effectiveness of our approach. Future work could further improve the performance of low-level ED via high-capacity generative entity retrievers.

Acknowledgements

We thank anonymous reviewers for their comments and suggestions. This work was supported by Shanghai Sailing Program (No. 23YF1409400), National Key Research and Development Project (No. 2020AAA0109302), Shanghai Municipal Science and Technology Major Project (No. 2021SHZDZX0103), Science and Technology Commission of Shanghai Municipality Grant (No. 22511105902), National Natural Science Foundation of China (No. 62102095), and Shanghai Municipal Special Fund for Promoting High-quality Development of Industries (No. 2021-GZL-RGZN-01018). Yanghua Xiao is also a member of Research Group of Computational and AI Communication at Institute for Global Communications and Integrated Media, Fudan University.

References

Baykal-Gürsoy, M. 2010. Semi-Markov Decision Processes. *Wiley Encyclopedia of Operations Research and Management Science*.

Beltagy, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Broscheit, S. 2019. Investigating Entity Knowledge in BERT with Simple Neural End-To-End Entity Linking. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, 677–685.

Chen, L.; Liang, J.; Xie, C.; and Xiao, Y. 2018. Short text entity linking with fine-grained topics. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 457–466.

De Cao, N.; Izacard, G.; Riedel, S.; and Petroni, F. 2021a. Autoregressive Entity Retrieval. In *International Conference on Learning Representations*.

De Cao, N.; et al. 2021b. Highly Parallel Autoregressive Entity Linking with Discriminative Correction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 7662–7669.

Derczynski, L.; Maynard, D.; Rizzo, G.; Van Erp, M.; Gorrell, G.; Troncy, R.; Petrak, J.; and Bontcheva, K. 2015. Analysis of named entity recognition and linking for tweets. *Information Processing & Management*, 51(2): 32–49.

Fang, Z.; et al. 2019. Joint entity linking with deep reinforcement learning. In *The World Wide Web Conference*, 438–447.

Févry, T.; FitzGerald, N.; Soares, L. B.; and Kwiatkowski, T. 2020. Empirical evaluation of pretraining strategies for supervised entity linking. *arXiv preprint arXiv:2005.14253*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Hoffart, J.; et al. 2011. Robust disambiguation of named entities in text. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, 782–792.

Hou, Z.; Jin, X.; Li, Z.; and Bai, L. 2021. Rule-Aware Reinforcement Learning for Knowledge Graph Reasoning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 4687–4692.

Kolitsas, N.; et al. 2018. End-to-End Neural Entity Linking. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, 519–529.

Kostrikov, I.; Nair, A.; and Levine, S. 2021. Offline Reinforcement Learning with Implicit Q-Learning. In *Deep RL Workshop NeurIPS 2021*.

Levine, S.; Kumar, A.; Tucker, G.; and Fu, J. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7871–7880.

Martins, P. H.; et al. 2019. Joint Learning of Named Entity Recognition and Entity Linking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, 190–196.

Mnih, V.; et al. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Moro, A.; et al. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2: 231–244.

Mrini, K.; et al. 2022. Detection, Disambiguation, Reranking: Autoregressive Entity Linking as a Multi-Task Problem. In *Findings of the Association for Computational Linguistics: ACL 2022*, 1972–1983.

Nuzzolese, A. G.; Gentile, A. L.; Presutti, V.; Gangemi, A.; Garigliotti, D.; and Navigli, R. 2015. Open knowledge extraction challenge. In *Semantic Web Evaluation Challenges*, 3–15. Springer.

Pateria, S.; Subagdja, B.; Tan, A.-h.; and Quek, C. 2021. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5): 1–35.

Ratinov, L.; Roth, D.; Downey, D.; and Anderson, M. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 1375–1384.

Ravi, M. P. K.; et al. 2021. CHOLAN: A Modular Approach for Neural Entity Linking on Wikipedia and Wikidata. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 504–514.

Röder, M.; Usbeck, R.; Hellmann, S.; Gerber, D.; and Both, A. 2014. N^3-a collection of datasets for named entity recognition and disambiguation in the nlp interchange format. In *Proceedings of the ninth international conference on language resources and evaluation (LREC'14)*, 3529–3533.

Silver, D.; et al. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676): 354–359.

Steinmetz, N.; et al. 2013. Semantic multimedia information retrieval based on contextual descriptions. In *Extended Semantic Web Conference*, 382–396. Springer.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Takanobu, R.; Zhang, T.; Liu, J.; and Huang, M. 2019. A hierarchical framework for relation extraction with reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 7072–7079.

van Hulst, J. M.; et al. 2020. Rel: An entity linker standing on the shoulders of giants. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2197–2200.

Wan, G.; et al. 2020. Reasoning Like Human: Hierarchical Reinforcement Learning for Knowledge Graph Reasoning. In *International Joint Conference on Artificial Intelligence 2020*, 1926–1932. Association for the Advancement of Artificial Intelligence (AAAI).

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3): 229–256.

Xian, Y.; et al. 2019. Reinforcement Knowledge Graph Reasoning for Explainable Recommendation. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Zhang, W.; et al. 2022. EntQA: Entity Linking as Question Answering. In *International Conference on Learning Representations*.