Entity Embedding Completion for Wide-Coverage Entity Disambiguation

Daisuke Oba[†‡*] **Ikuya Yamada**[‡◇] **Naoki Yoshinaga**[§] **Masashi Toyoda**[§]
† The University of Tokyo ‡ Studio Ousia ◇ RIKEN AIP
§ Institute of Industrial Science, The University of Tokyo
oba@tkl.iis.u-tokyo.ac.jp ikuya@ousia.jp
{ynaga,toyoda}@iis.u-tokyo.ac.jp

Abstract

Entity disambiguation (ED) is typically solved by learning to classify a given mention into one of the entities in the model's entity vocabulary by referring to their embeddings. However, this approach cannot address mentions of entities that are not covered by the entity vocabulary. Aiming to enhance the applicability of ED models, we propose a method of extending a state-of-the-art ED model by dynamically computing embeddings of out-of-vocabulary entities. Specifically, our method computes embeddings from entity descriptions and mention contexts. Experiments with standard benchmark datasets show that the extended model performs comparable to or better than existing models whose entity embeddings are trained for all candidate entities as well as embedding-free models. We release our source code and model checkpoints at https://github.com/studio-ousia/steel.

1 Introduction

Entity disambiguation (ED) (§ 2) has been studied as a method to organize information about real-world entities in text. Consequently, ED must be capable of handling diverse entities in various domains. Existing state-of-the-art ED models are based on huge pretrained language models (PLMs) (Devlin et al., 2019), and are trained to match a given mention with its candidate entities via embeddings (Broscheit, 2019; Ling et al., 2020; Févry et al., 2020; Yamada et al., 2022). Despite the advances in performance, these embedding-based ED models do not address out-of-vocabulary entities that are absent from the training data. Even worse, since maintaining millions of entity embeddings demands a huge memory space in training, the models are often trained only for candidate entities (§ 2) that appear in testing (Broscheit, 2019; Yamada et al., 2022).[1]

To address the above problems, researchers have explored ED models without entity embeddings. Cao et al. (2021) solves ED as autoregressive generation, which decodes the entity name token-by-token. Barba et al. (2022) formulates ED as span extraction, which selects a correct entity name from input concatenating mention contexts and candidate entity names. Although these approaches allow a model to handle millions of candidate entities, they still have difficulties in handling newly-emerged entities that are absent from the training data and evolved entities whose attributes have changed over time. Meanwhile, Logeswaran et al. (2019) and Wu et al. (2020) proposed zero-shot methods to handle any candidate entities by reading their descriptions at testing. The performance of these zero-shot methods, however, is inferior to the aforementioned models, probably because these methods represent each entity using short entity-related text instead of massive training examples used in the ED models.

In this paper, we propose an adaptation method that enables entity embedding-based ED models, which are trained for limited entities, to handle the other rare and emerging entities by dynamically predicting their embeddings. In particular, we actually extend the architecture of the state-of-the-art Transformer-based ED model developed in Yamada et al. (2022). We first train this model with an entity vocabulary consisting of only common entities (§ 4.1). Next, we extend the model (§ 4.3) by using our BERT-based encoders that dynamically predict entity embeddings from descriptions and relevant sentences (§ 4.2), thus enabling the model to handle the out-of-vocabulary entities.

[*] Work done as an intern at Studio Ousia.

[1] Reducing the entity vocabulary size is critical to train huge Transformer-based ED models (§ 3), as the entity embeddings can occupy GPU memory during training. For instance, seven million 256-dimensional embeddings for all entities in the English Wikipedia consume approximately 22GB of GPU memory in the training standard with the Adam optimizer.

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 6333–6344
December 7-11, 2022 ©2022 Association for Computational Linguistics

Through experiments (§ 5) with standard benchmark datasets for ED (i.e., AIDA-CoNLL (Hoffart et al., 2011), MSNBC, AQUAINT, ACE2004, WNED-CWEB, and WNED-WIKI (Guo and Barbosa, 2018)), we verify that our extended model performs competitively to the original state-of-the-art model, whose trained entity embeddings encompass all the candidate entities of the evaluation datasets (§ 5). We then perform an empirical analysis on the nature of our proposed encoders, and on reducing memory costs in the inference time (§ 6).

Our key contributions are:

- Adapting trained entity embedding-based models to any sets of candidate entities, by predicting embeddings of missing entities with neural encoders (§ 4).
- Showing that our method enables a model trained with a limited entity vocabulary to perform comparably with models trained with all candidate entities of the evaluation data (§ 5).
- Presenting an empirical analysis of our predicted embeddings, and the possibility for reducing memory costs during inference (§ 6).

2 Entity Disambiguation Task

Given a knowledge base (KB) such as Wikipedia, and a text with spans of mentions, the ED model assigns each mention span to an entity in the KB. The task is similar to word sense disambiguation.

Since the number of entities can be more than millions, during the evaluation, the ED models additionally assume a small list of candidate entities for each mention, using a mention-entity dictionary derived from hyperlinks in Wikipedia. Following existing embedding-based ED models (*e.g.*, Yamada et al. (2022)), we use the standard entity candidate set, KB-YAGO (Ganea and Hofmann, 2017).

3 Related Work

Several studies have proposed knowledge-intensive Transformer-based ED models that have entity embeddings as parameters, such as weights of the classifier (Broscheit, 2019; Ling et al., 2020; Févry et al., 2020; Yamada et al., 2022). Despite the state-of-the-art in the benchmark, they cannot handle out-of-vocabulary entities and entities of changed properties, since these embeddings are typically trained via ED task with labeled corpora. In addition, as argued by Barba et al. (2022), it is hard to integrate embeddings of all ever-increasing entities into the huge Transformer-based models, especially to maintain their gradient values during training. Ling et al. (2020) therefore demands expensive TPU clusters, and some employ transductive settings to train embeddings for a small subset of entities containing all candidate entities used in the evaluation (Broscheit, 2019; Yamada et al., 2022).

Before Transformer becomes the dominant architecture for the ED task, knowledge-intensive ED models were developed (Yamada et al., 2016; Ganea and Hofmann, 2017; Le and Titov, 2018). Since these models require relatively few parameters to be learned (*e.g.*, several linear transformation matrices (Ganea and Hofmann, 2017)), loading and training embeddings of entire entities may not be a problem compared to the Transformer-based models. However, they still do not address entities absent in or updated from the training phase.

Zero-shot systems, which represent entities as descriptions or contexts rather than embeddings, can convey textual information pertaining to novel or changing entities. However, a single entity description, used by Logeswaran et al. (2019) and Wu et al. (2020), has less information compared to embeddings learned from many examples. FitzGerald et al. (2021) embeds each mention context as keys to retrieve associated entities from the target mention contexts (query). However, they also suffers from less informative contexts per key embedding.[2]

Generation- or extraction-based approaches can reduce computational costs by not integrating entity-level knowledge such as entity embeddings; Cao et al. (2021) generates tokens of entity titles and Barba et al. (2022) selects the correct title from the concatenated titles of candidate entities. However, while they can generate or extract any entities, they cannot properly model novel and evolving entities since they are trained with labeled data based on a specific KB. Moreover, the performance still falls short of entity embedding-based models.

Along with the fact that Transformer-based models performed well on the other entity-related tasks, such as question answering, by using entity embeddings (Zhang et al., 2019; Yamada et al., 2020; Févry et al., 2020; de Jong et al., 2021), we can infer the importance of explicitly modeling knowledge as the parameters to optimize the task performance. This study focuses on this powerful entity embedding-based models.

[2] lower Recall@1 than that of Wu et al. (2020)

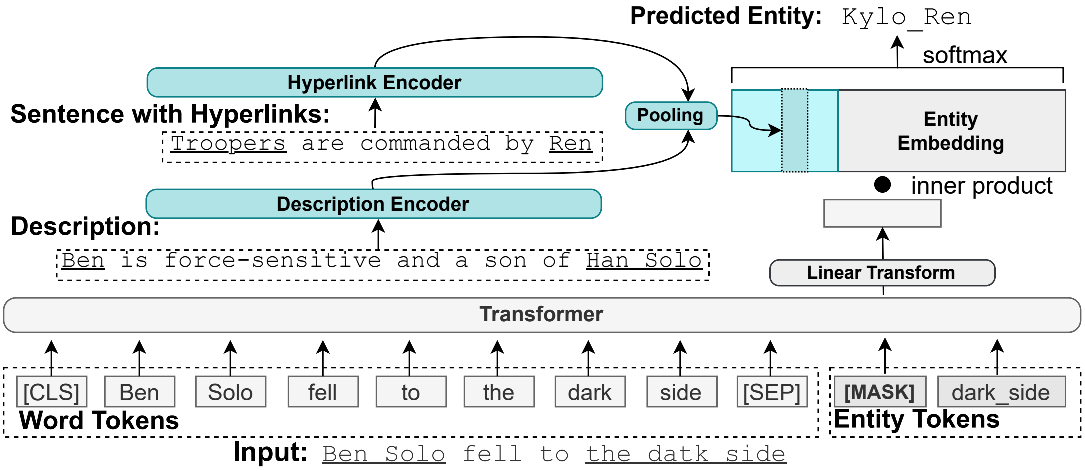

Figure 1: Our method complements the entity embedding matrix by encoding entities' descriptions and relevant sentences. The underlines refer to mentions. `Ben Solo`, `Ben`, and `Ren` refer to the same entity `Kylo_Ren`. Here, `Kylo_Ren` is newly added to the model's entity vocabulary by our method while it is originally not included in it.

4 STEEL

In this section, we present our adaptation method, **STEEL**, which complements **ST**atic and **E**ncoder-based entity embeddings for **E**ntity **L**inking. While our method is applicable to a variety of entity embedding-based methods, in this paper, we apply it to the architecture of the state-of-the-art ED model developed in Yamada et al. (2022).

We begin by training the base ED model, whose entity embeddings comprise only a reasonable number of common entities V_e (§ 4.1). We then estimate the embeddings for other entities $\notin V_e$ using our proposed encoders, and extend the model (§ 4.2). Hereafter, V_w and V_e denote the word and entity vocabularies of the base model. H and D denote the size of the hidden states and of the embeddings.

4.1 Base ED Model

Input Representations Our base ED model is adopted from Yamada et al. (2022), an extension of BERT (Devlin et al., 2019). This model takes a concatenation of word token sequence and entity token sequence as an input, and entity tokens of target mentions are masked (Figure 1). Each entity token corresponds to an entity in the word token sequence (*e.g.*, `dark_side` stands for word tokens of `the`, `dark`, and `side`). Each token is represented by summing the following three embeddings:

- **Token Embeddings** correspond to each word $w_i \in V_w$ and entity $e_i \in V_e$. We denote the word token embeddings as $\mathbf{A} \in \mathbb{R}^{|V_w| \times H}$. We denote entity token embeddings as $\mathbf{BU}$ with two smaller matrices, $\mathbf{B} \in \mathbb{R}^{|V_e| \times D}$ and $\mathbf{U} \in \mathbb{R}^{D \times H}$, to reduce computational cost where $D \ll H$, following Yamada et al. (2020).
- **Type Embeddings** represent the type of token, i.e., word or entity. Each representation is $\mathbf{C}_w \in \mathbb{R}^H$ and $\mathbf{C}_e \in \mathbb{R}^H$.
- **Position Embeddings** correspond to the positions of tokens. We represent words and entities at position i in the word sequence by $\mathbf{D}_i \in \mathbb{R}^H$ and $\mathbf{E}_i \in \mathbb{R}^H$. For entities spanning multiple word tokens (*e.g.*, `dark_side`), we average their position embeddings.

Training We first build and train an ED model that includes entity embeddings $\mathbf{B} \in \mathbb{R}^{|V_e| \times D}$ for only the top-K frequent entities V_e in Wikipedia articles as trainable parameters. The model performs entity disambiguation in the manner of masked language modeling (Devlin et al., 2019).

Using hyperlinks in Wikipedia articles as entity annotations, we mask each mention token with a certain probability, and optimize the model to estimate the entity $e \in V_e$ from the BERT's output $\mathbf{h}_e \in \mathbb{R}^H$ corresponding to `[MASK]`:

$$\mathbf{m}_e = \text{layernorm}\big(\text{gelu}(\mathbf{W}_f \mathbf{h}_e + \mathbf{b}_f)\big) \quad (1)$$

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{B}\mathbf{m}_e) \quad (2)$$

where $\mathbf{W}_f \in \mathbb{R}^{D \times H}$ and $\mathbf{b}_f \in \mathbb{R}^D$ are trainable parameters, gelu (Hendrycks and Gimpel, 2016) is the activation function, and layernorm (Lei Ba et al., 2016) is the layer normalizer. Compared to the model of Yamada et al. (2022), we eliminate a bias term in Eq. 2, as STEEL extends only the entity embedding $\mathbf{B}$. We maximize the log-likelihood of the estimation $\hat{\mathbf{y}}$.

4.2 Our Encoders for Embedding Prediction

We propose two BERT-based encoders (Figure 1) to predict the embeddings of out-of-vocabulary entities $e_i \notin V_e$. Each encoder follows the same architecture as our base ED model which is an extension of BERT, while removing the linear transformation head to estimate entities in Eq. 1 and 2. Encoders use **entity descriptions** and **mention contexts** (sentences referring to the entity) as inputs respectively, which are language resources easily obtained from a KB. Each of them corresponds to connotative and denotative definitions of entities.

Description Encoder: The encoder predicts entity embeddings from descriptions. We first convert each description into an input representation in the same way as § 4.1 (i.e., word tokens and entity tokens). We then obtain the BERT output $\mathbf{h}_{\text{CLS}} \in \mathbb{R}^H$ corresponding to [CLS], and compute the embedding of the entity e_i as follows:

$$\hat{\mathbf{e}}_{\text{DESC}_i} = \mathbf{W}_{\text{DESC}}\mathbf{h}_{\text{CLS}} + \mathbf{b}_{\text{DESC}} \tag{3}$$

where $\mathbf{W}_{\text{DESC}} \in \mathbb{R}^{D \times H}$ and $\mathbf{b}_{\text{DESC}} \in \mathbb{R}^D$ are the trainable parameters.

Hyperlink Encoder: The encoder predicts embedding for entity e_i from the sentences with hyperlinks of e_i, which we hereafter refer to as *mention contexts*. We first obtain mention contexts referring to e_i from KB, and convert them to input representations in the same way as § 4.1 (i.e., word token sequences and entity token sequences). We then replace the entity token corresponding to e_i with [MASK]. By using N mention contexts for e_i, we obtain N outputs from the BERT corresponding to each [MASK] (i.e., $\mathbf{h}_{\text{MASK}_1} \dots \mathbf{h}_{\text{MASK}_N}$), and we average them as the predicted embeddings for e_i in order to embed multiple contexts:

$$\overline{\mathbf{h}}_{\text{MASK}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{h}_{\text{MASK}_n} \tag{4}$$

$$\hat{\mathbf{e}}_{\text{HL}_i} = \mathbf{W}_{\text{HL}}\overline{\mathbf{h}}_{\text{MASK}} + \mathbf{b}_{\text{HL}} \tag{5}$$

where $\mathbf{W}_{\text{HL}} \in \mathbb{R}^{D \times H}$ and $\mathbf{b}_{\text{HL}} \in \mathbb{R}^D$ are trainable parameters.

Encoder Training: We have trained the above two encoders as follows. We first initialize parameters of the two encoders using the trained parameters of the base ED model (§ 4.1), since these encoders adopt a similar architecture and input representations as ED model. As the supervision to train the encoders, we utilize embeddings $\mathbf{B}$ for the common entities V_e, which are integrated in and jointly trained with our base model (§ 4.1). During training the encoders, we freeze $\mathbf{B}$. With this approach, the predicted embeddings are placed in the same space as that of the base ED model. We minimize the mean squared error between the trained embedding $\mathbf{B}_i$ for the entity $e_i \in V_e$, and the corresponding predicted embedding. To reduce training costs, we optimize each encoder separately.

4.3 Entity Embedding Completion

We predict embeddings for the out-of-vocabulary entities $V_e^{\text{oov}} \not\subset V_e$ using our encoders, and complement the entity embedding matrix $\mathbf{B}$.

Embedding Prediction and Ensemble: We first predict the embeddings of out-of-vocabulary entities V_e^{oov} using each of our neural encoders. We then simply average the embeddings predicted by our encoders (Eq. 3 and 5), to leverage different language resources:

$$\hat{\mathbf{e}}_i = \frac{1}{2}(\hat{\mathbf{e}}_{\text{DESC}_i} + \hat{\mathbf{e}}_{\text{HL}_i}) \tag{6}$$

Completion: We extend the trained entity embedding matrix $\mathbf{B}$ using the predicted embeddings of entities V_e^{oov}, as illustrated in Figure 1. Accordingly, we rearrange Eq. 2 as follows:

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{B}^{\diamond}\mathbf{m}_e) \tag{7}$$

where $\mathbf{B}^{\diamond} \in \mathbb{R}^{|V_e \cup V_e^{\text{oov}}| \times D}$ denotes extended entity embedding matrix. Thus, the trained model can address out-of-vocabulary entities. In this paper, we focus on *adding* new embeddings for out-of-vocabulary entities. However, it is also capable of *replacing* the trained embeddings $\in \mathbf{B}$ of entities whose meaning has changed due to updates of the associated KB such as Wikipedia.

Inference: The model performs inferences sequentially in several steps. We first mask all entity tokens. The model estimates the entities in each step (Eq. 7), and then replaces the [MASK] of the highest probability with the estimated entity token. We repeat this process until all [MASK] tokens have been disambiguated.

4.4 Implementation Details

Base ED Model: To ensure computational efficiency and improve the reproducibility on a common computational environment, we implement an ED model based on the $\text{BERT}_{\text{BASE}}$ (Devlin et al., 2019), whereas Yamada et al. (2022) exploited $\text{BERT}_{\text{LARGE}}$. As the training corpus, we use the December 20th, 2018 version of English Wikipedia articles consisting of over three billion words and ten million entity annotations. We set the entity vocabulary size of our base ED model to $|V_e| = 500K$ consisting of the most frequent entities in the articles. We split the articles into the sets of up to 512 words, and extract up to 128 hyperlinks for each set. We then tokenize them using BERT's tokenizer with the vocabulary V_w consisting of 30K tokens. We initialize all parameters common with BERT using the pre-trained parameters, and set other parameters randomly. We train the model in two steps; we first train the model with the BERT's parameters fixed for 10 epochs, and we then update entire parameters for 10 epochs. Additional details are described in Appendix A. We designate this model as $\text{YAMADA}_{\text{BASE}}^{500K}$, which will be extended by STEEL.

Encoders for Embedding Prediction: We initialize the parameters by using those of our trained base model, $\text{YAMADA}_{\text{BASE}}^{500K}$. As the training corpus, we exploit the descriptions and mention contexts corresponding to V_e from the same Wikipedia articles. As the entity description, we extract the first 512 tokens and up to 128 hyperlinks contained in them, from the corresponding entity's page. As the mention context, we extract the sentences with hyperlinks pointing to the corresponding entity. We here use OpenNLP sentence tokenizer[3] to detect sentences. As an extracted mention context, we include both or either of the before and after sentences altogether for exploiting longer contexts per mention. We exploit up to eight mention contexts per entity (i.e., $N = 8$ in Eq. 4). Further details are described in Appendix B.

5 Experiments

We conduct experiments on entity disambiguation with the goal of verifying that the performance of $\text{YAMADA}_{\text{BASE}}^{500K}$ with STEEL is comparable to that of state-of-the-art models, whose entity vocabulary contains all candidate entities in the evaluation as well as to that of embedding-free models.

5.1 Settings

Evaluation Datasets and Metric: We follow the settings specified in previous studies (Ganea and Hofmann, 2017; Le and Titov, 2018; Yamada et al., 2022). Specifically, we use benchmark datasets; AIDA-CoNLL (Hoffart et al., 2011), MSNBC, AQUAINT, ACE2004, WNED-CWEB, and WNED-WIKI (Guo and Barbosa, 2018). For entity candidate sets, we use KB+YAGO (Ganea and Hofmann, 2017), and select top-30 candidates per mention based on the associated prior probability. All entity candidates used in these evaluation data consist of 140K entities (hereafter, V_e^{all}). **52.7%** of candidate entities V_e^{all} are not included in the entity vocabulary V_e of our base ED model. Moreover, **13.3%** of gold entities are not included in V_e, which will never be solved without STEEL's extension. We report In-KB accuracy for AIDA-CoNLL and Micro-F1 for the others.

Fine-tuning for AIDA-CoNLL: In the evaluation with AIDA-CoNLL, we conduct fine-tuning using the attached training set (AIDA-train) as the baselines did, while freezing the extended entity embedding matrix $\mathbf{B}^{\diamond}$. We prepare inputs in the same way as in the training step (§ 4.1), and mask the entity tokens with a probability of 90% following Yamada et al. (2022). We optimize the model by maximizing the log-likelihood of the prediction in Eq. 7. The best batch size, learning rate, and epochs are searched from the same space as Devlin et al. (2019) based on the development set (AIDA-testa). Further details are described in Appendix C.

Baselines: We employ embedding-based models (Ganea and Hofmann, 2017; Le and Titov, 2018; Broscheit, 2019; Févry et al., 2020; Yamada et al., 2022), whose entity vocabulary covers all the candidate entities V_e^{all} in their training phase. In addition, we train a full-coverage version of our base ED model whose entity vocabulary V_e is equal to V_e^{all}, which we refer to as $\text{YAMADA}_{\text{BASE}}$. This full coverage model can be regarded as the smaller version of Yamada et al. (2022) without bias terms in Eq. 3. As the baselines trained without entity embeddings, we employ the generation-based (Cao et al., 2021) and extraction-based (Barba et al., 2022) models.

5.2 Results

Table 1 shows the results for AIDA-CoNLL. First, by applying STEEL to $\text{YAMADA}_{\text{BASE}}^{500\text{K}}$, we achieve comparable performance against the full-coverage

[3] https://opennlp.apache.org

Systems	CoNLL
w/ entity embeddings:	
Ganea and Hofmann (2017)*	92.2
Le and Titov (2018)*	93.1
Broscheit (2019)*	87.9
Févry et al. (2020)*	92.5
Yamada et al. (2022) $^{*}_{\text{LARGE}}$	95.0
YAMADA$_{\text{BASE}}$	94.7
w/o entity embeddings:	
Cao et al. (2021)* (generation-based)	93.3
Barba et al. (2022)* (extraction-based)	92.6
YAMADA$^{\text{500K}}_{\text{BASE}}$ + STEEL	94.3
YAMADA$^{\text{500K}}_{\text{BASE}}$ + STEEL w/o fine-tuning	91.8
YAMADA$^{\text{500K}}_{\text{BASE}}$ ♠	90.8

Table 1: In-KB accuracy on AIDA-CONLL. Models are fine-tuned unless otherwise stated. The figures of systems marked with $*$ are quoted from existing studies. Entity candidate sets used in the system marked with ♠ are limited to those included in the 500k entities V_e.

version: YAMADA$_{\text{BASE}}$ (−0.4pt.). In addition, this adapted model outperforms all other entity embedding-based baselines with the exception of Yamada et al. (2022)$_{\text{LARGE}}$, which is the larger version of YAMADA$_{\text{BASE}}$. Note that their entity embeddings encompass all candidate entities in the training phase. Furthermore, our model outperforms both of generation- and extraction-based models.

Table 2 shows the experimental results on the other benchmark datasets. On average (**avg.**), our model successfully achieved identical performance to that of YAMADA$_{\text{BASE}}$, and comparable performance (−0.1pt.) even against its larger version: Yamada et al. (2022) $_{\text{LARGE}}$. Similar to the tendency on AIDA-CoNLL, our model outperforms all other entity embedding-based systems, whose entity vocabulary encompasses all the candidate entities (Ganea and Hofmann, 2017; Le and Titov, 2018), as well as the systems trained without entity embeddings (Cao et al., 2021; Barba et al., 2022). More specifically, our model advanced the new state-of-the-art on ACE2004, and performed better on CWEB than the full-coverage version of our base ED model: YAMADA$_{\text{BASE}}$.

Although we cannot perform a fair comparison between the entity embedding-based models if the models' entity vocabularies do not cover the same set of candidate entities, we quickly evaluated YAMADA$^{\text{500K}}_{\text{BASE}}$ by using zero embeddings for out-of-vocabulary candidate entities (i.e., systems marked with ♠ in Table 1 and Table 2). From the tables, we can see that adaptation with STEEL improved the performance on the six benchmark datasets consistently. If the candidate entities are limited to those included in 500K frequent entities, it may reduce the possibility of making mistakes regarding the frequent entities' mentions. On the other hand, mentions for out-of-vocabulary entities, 13.3% of mentions in our setting, can never be solved in any way without applying STEEL.

Table 3 shows the accuracy on AIDA-CoNLL compared to the reported score[4] of the state-of-the-art zero-shot system (Wu et al., 2020), which addresses ED by comparing the mention contexts and candidate entities' descriptions without using entity embeddings. Note that it is not a fair comparison as their candidate sets are not identical to KB+YAGO, since their model uses their own candidate generator (i.e., bi-encoder). However, the gold recall of their candidate set 97% is comparable to KB+YAGO's 98%. Even taking into account the difference in entity candidates, our model yielded higher performance with more than 10% margin. This is probably because we solve ED by using entity embeddings predicted from descriptions or mention contexts only for V_e^{oov} and by using high-quality entity embeddings optimized via ED task with massive examples for the other frequent entities. Entity embedding-based models advanced the state-of-the-arts in the past studies, but such models are generally not good at handling rare and emerging entities as discussed in § 1 and § 3. Our approach solves these critical weaknesses while inheriting the strengths of the conventional entity embedding-based models.

Overall, the above observations indicate that STEEL obtains entity embeddings of comparable quality to that of state-of-the-art system (Yamada et al., 2022), and successfully integrates them to the model to perform ED on out-of-vocabulary entities.

6 Analysis

In this section, we provide a detailed analysis of our proposed adaptation method, STEEL.

6.1 Ablation Study

We investigate the impact of ensembling two types of predicted entity embeddings, as derived from the description and hyperlink encoders (§ 4.2). By

[4] https://github.com/facebookresearch/BLINK

Systems	ACE2004	AQUAINT	CWEB	WIKI	MSNBC	avg.
w/ entity embeddings:						
Ganea and Hofmann (2017)*	88.5	88.5	77.9	77.5	93.7	85.2
Le and Titov (2018)*	89.9	88.3	77.5	78.0	93.9	85.5
Yamada et al. (2022) $^{*}_{\text{LARGE}}$	91.9	93.5	78.9	89.1	96.3	89.9
YAMADA$_{\text{BASE}}$	91.1	94.2	78.6	90.0	95.0	89.8
w/o entity embeddings:						
Cao et al. (2021)* (generation-based)	90.1	89.9	77.3	87.4	94.3	87.8
Barba et al. (2022)* (extraction-based)	91.8	91.6	77.7	88.8	94.7	88.9
YAMADA$^{\text{500K}}_{\text{BASE}}$ + STEEL	92.3	94.0	78.8	89.2	94.6	89.8
YAMADA$^{\text{500K}}_{\text{BASE}}$ ♠	91.5	93.7	76.4	83.9	94.1	87.9

Table 2: Micro-F1 on the standard evaluation datasets, with the exception of AIDA-CoNLL. The figures of systems marked with $*$ are quoted from existing studies. Entity vocabulary sets used by the system marked with ♠ are limited to those included in 500K frequent entities V_e. **avg.** indicates the averaged score over all datasets.

Systems	Accuracy
Wu et al. (2020)*	80.3
YAMADA$^{\text{500K}}_{\text{BASE}}$ + STEEL	91.8

Table 3: Accuracy on the AIDA-CoNLL. The figures of systems marked with $*$ are quoted from their official website.

Method	Micro-F1
YAMADA$^{\text{500K}}_{\text{BASE}}$ + STEEL	90.9
w/o Description Encoder	89.7
w/o Hyperlink Encoder	90.7

Table 4: Averaged Micro-F1 on all datasets when only one of the proposed encoders is ablated.

ablating one of the two types of embeddings from YAMADA$^{\text{500K}}_{\text{BASE}}$ + STEEL, we report an avg. Micro-F1 over the six datasets: AIDA-CoNLL, MSNBC, AQUAINT, ACE2004, CWEB, and WIKI.

Table 4 lists the ablation results. We can observe that performance consistently decreased when ablating each encoder. This suggests that even the straightforward approach of averaging different types of entity embeddings can effectively exploit different language resources (*mention contexts* and *entity descriptions*), each of which represents connotative and denotative meanings of entities. Furthermore, by comparing the drops in performance, we can see that *entity descriptions* are more valuable than *mention contexts* for overall performance.

6.2 Entity Disambiguation Performance for Mentions of Out-of-Vocabulary Entities

We investigate how well our extended model disambiguates the mentions of out-of-vocabulary entities V_e^{oov}, whose embeddings are predicted and integrated into the model by STEEL. We report precision regarding those mentions over the six datasets. We also report the performance of the simple baseline, which selects the entity with the highest prior-probability $\hat{p}(entity|mention)$ (Ganea and Hofmann, 2017).

Table 5 (V_e^{oov} column) shows the results. Our model correctly handles approximately 80% mentions of the out-of-vocabulary entities, whereas the prior probability baseline, which exploits surface information, solves less than 70% of the mentions. Moreover, the performance gap with oracle model YAMADA$_{\text{BASE}}$, which optimizes the embeddings of all candidate entities via ED task, is only 1.2 pt. As for the comparison to the state-of-the-art zero-shot system, Wu et al. (2020), we achieved huge performance improvements. Again, we here state that Wu et al. (2020) use candidate sets not identical to KB+YAGO as described in § 5.2.

The performance experienced a more significant drop (-2.2 pt.) when ablating the hyperlink encoder. It indicates the worth of *mention contexts* to embed out-of-vocabulary entities. This trend differs from that of the ablation results of overall mentions in Table 4. It indicates that the utilization of only *mention contexts* has a side effect of inhibiting the inference of mentions for in-vocabulary entities V_e.

Method	V_e^{oov}	# contexts	
		≤ 2	$7 \leq$
YAMADA$^{500\text{K}}_{\text{BASE}}$ + STEEL	79.0	66.5	79.8
w/o Description Encoder	79.7	66.1	80.8
w/o Hyperlink Encoder	76.8	68.1	77.0
YAMADA$_{\text{BASE}}$	80.8	63.3	82.8
Wu et al. (2020)	59.3	49.3	60.2
prior probability	68.1	64.9	66.1

Table 5: V_e^{oov} column indicates precision for the mentions of the out-of-vocabulary entities. # contexts column shows precision for the subset of those mentions, which are distinguished by the number of *mention contexts* (max. 8) used to encode the corresponding gold entity.

Method	$V_e^{oov,\leq 8}$
YAMADA$^{500\text{K}}_{\text{BASE}}$ + STEEL	76.3
YAMADA$_{\text{BASE}}$	75.3

Table 6: Precision on all datasets for the mentions of specific out-of-vocabulary entities whose mention contexts exist at most eight on Wikipedia articles.

6.3 Effects of the Number of Hyperlink Sentences Fed into Hyperlink Encoder

In this section, we analyzed the impact of the number of *mention contexts* to be fed into hyperlink encoder on the quality of predicted embeddings. In the settings of our study, hyperlink encoder uses up to eight mention contexts per entity. Therefore, the quality of predicted embeddings can be partially determined by the number of mention contexts used. Here, we extract the subsets of mentions of out-of-vocabulary entities, who have ≤ 2 and ≥ 7 mention contexts available for the hyperlink encoder, and compute the precision for each mention subset.

# contexts column in Table 5 shows the results on each mention subset. For mentions of relatively frequent entities (≥ 7), the use of predicted embeddings from hyperlink encoder leads to more correct inferences. Conversely, for mentions of rare entities (≤ 2), the description encoder performs more effectively. This is likely because *entity descriptions* corresponding to connotative meanings are a frequency-independent resource. Future research directions include the selection of different types of predicted embeddings based on the volume of available resources, and weighted averaging of two types of embeddings.

6.4 On Whether Embeddings of Rare Entities Should be Learned or Predicted

We measured the performance on the mention subset of out-of-vocabulary entities that have eight or fewer mentions on Wikipedia articles (say, $V_e^{oov,\leq 8}$). In our experimental settings (§ 5.1), up to eight mentions per entity are extracted and used for predicting embeddings for out-of-vocabulary entities, even if there are more than eight mentions available on Wikipedia. By focusing on $V_e^{oov,\leq 8}$, the number of mention contexts used to **optimize** embeddings by YAMADA$_{\text{BASE}}$ via ED task and to **predict** embeddings by STEEL are both the same.

Table 6 shows that if only the same number of mention contexts are available, prediction by STEEL yields better quality entity embeddings than learning them through the ED task. It suggests that it is not always practical to apply optimization of entity embeddings via ED task when their resources are scarce, whereas STEEL makes relatively effective use of limited resources. In practice, STEEL can utilize more mention contexts than the trained base model as KB updates after the model training. The gain therefore will become larger.

6.5 Qualitative Analysis of Predicted Entity Embeddings

Using STEEL, we predict each embedding of entire entities in English Wikipedia articles except the most frequent 500K entities in $\in V_e$, and perform k-nearest neighbors-search among the trained and predicted embeddings, separately.

Table 7 shows that each predicted embedding is close to embeddings of relevant entities. For instance, the embedding of a station in Denver, `Perry_station`, is located near embeddings of other stations in Denver. An high-end-audio company, `Pass_Labs`, is close to a sound-related term, `Passband`, and a semiconductor maker, `Intersil`. Among the infrequent entities, it is close to more directly related audio device-makers such as `Valve_Amplification_Company`.

6.6 Quantization of Entity Embeddings

In this section, we quantized entity embeddings in the extended model (YAMADA$^{500\text{K}}_{\text{BASE}}$ + STEEL) to reduce the memory cost at inference time. This experiment was motivated by the argument in Barba et al. (2022) that embedding-based ED models are flawed because entire entity embeddings need to be stored in memory at runtime. Our method, which is

Entity Title	Among the most frequent 500K entities	Among the other entities
Perry_station	RTD_bus_and_rail_services, Denver_Union_Station, U.S._Route_287_in_Colorado	Colorado_station, Knox_station, Evans_station_(RTD), 25th_%26_Welton_station
Word_embedding	Word-sense_disambiguation, WordNet, Automatic_summarization,	Sentence_embedding, Word-sense_induction, Font_embedding,
Pass_Labs	Passband, Intersil, Keysight, TEAC_Corporation, Patch_panel	Valve_Amplification_Company, Nelson_Pass, Class-T_amplifier

Table 7: Nearest neighbors of entity embeddings predicted by our encoder, among entities in V_e whose embeddings are contained in our base model, as well as other entities whose embeddings are dynamically predicted.

Format	Size	CoNLL	Others
Original: FP32	7.73GB	91.8	89.8
INT8	1.93GB	91.9	89.8
INT6	1.45GB	91.8	89.9
INT4	0.97GB	91.7	89.6

Table 8: Micro-F1 of our extended model (w/o fine-tuning) when quantizing its entity embeddings, as well as the required memory to store embeddings for entire English Wikipedia titles (∼7 million) during inference.

based on a limited entity vocabulary, has somewhat reduced the memory cost during **training** while maintaining high performance. However, we still need to store the extended embedding matrix in memory during **inference**. To increase portability when deploying, we use the scalar quantizer provided by the open source library Faiss[5] (Johnson et al., 2019) to quantize the entity embeddings to 4-, 6-, or 8-bit integers from 32-bit floats.

Table 8 lists the In-KB accuracy for AIDA-CoNLL and the avg. Micro-F1 for the other five datasets. From the results, we can see that quantization makes almost no difference in performance. By doing so, the model consumes less than 1GB memory even when all entities from English Wikipedia articles (∼7M) are integrated within the model. These observations indicate that entity embedding-based approaches can be operated without excessive concern for memory issues during inference, and present a realistic research direction to be continuously explored in the future.

7 Conclusions

In this study, we proposed an adaptation method, STEEL, which makes trained entity embedding-based ED models applicable to candidate entities that are not integrated into the model (§ 4). Our methods extend trained models by predicting entity embeddings of other rare or emerging entities by encoding the corresponding descriptions or relevant sentences. In the experiments with standard benchmark datasets (§ 5), our model performed competitive or better than existing entity embedding-based models trained for all candidate entities used in the evaluation, and also outperformed embedding-free models. We also conducted analyses concerning the predicted embeddings (§ 6.1 to 6.5), and evaluated the capability of reducing memory costs required during inference via quantization (§ 6.6).

Our methods, in addition to out-of-vocabulary entities, can also handle evolving entities whose properties have changed after training the model, even though they were included in the entity vocabulary set during training. This capability is vital in ED since a KB to be linked can be updated intermittently. For example, Wikipedia articles before and after "Joe Biden" have become U.S. president are very different. His embedding trained using the former periods of articles may not be optimal for processing mentions in the newer articles. The performance of replacing trained embeddings with new ones should be evaluated in the future.

As a base model to which our method is applied, we adopted Yamada et al. (2022) since it is the state-of-the-art and its source code is publicly available. We release our implementation at https://github.com/studio-ousia/steel for the other models to enjoy our method in the future.

[5] https://github.com/facebookresearch/faiss

Limitations

The proposed method works for any rare or emerging entities only when relevant descriptions or sentences are available, as is the case with most knowledge bases such as Wikipedia, Fandom[6], and UMLS (Bodenreider, 2004). If these resources are not available, for example, we can use entity-entity relations collected by knowledge graphs alternatively. We may also utilize the standard entity titles (e.g., `Manchester_Airport`) as descriptions or as search queries to retrieve possible mention contexts from the web.

The proposed method can handle entities in languages other than English; however, when handling several multilingual entities simultaneously, it is unknown whether it can successfully absorb differences in word order and volume of resources across languages.

Acknowledgement

We thank the anonymous reviewers for their attentive reading of our manuscript and author response, and for their valuable comments and suggestions. This work was partially supported by JSPS KAKENHI Grant Number 21H03494, and by JST CREST Grant Number JPMJCR19A4 including AIP challenge program, Japan.

References

Edoardo Barba, Luigi Procopio, and Roberto Navigli. 2022. ExtEnD: Extractive entity disambiguation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2478–2488, Dublin, Ireland. Association for Computational Linguistics.

Olivier Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic Acids Research*, 32(1):D267–D270.

Samuel Broscheit. 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)*, pages 677–685, Hong Kong, China. Association for Computational Linguistics.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive Entity Retrieval. In *9th International Conference on Learning Representations*.

Michiel de Jong, Yury Zemlyanskiy, Nicholas FitzGerald, Fei Sha, and William W Cohen. 2021. Mention memory: incorporating textual knowledge into transformers through entity mention attention. In *10th International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Thibault Févry, Livio Baldini Soares, Nicholas FitzGerald, Eunsol Choi, and Tom Kwiatkowski. 2020. Entities as experts: Sparse memory access with entity supervision. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4937–4951, Online. Association for Computational Linguistics.

Thibault Févry, Nicholas FitzGerald, Livio Baldini Soares, and Tom Kwiatkowski. 2020. Empirical Evaluation of Pretraining Strategies for Supervised Entity Linking. In *Automated Knowledge Base Construction*.

Nicholas FitzGerald, Dan Bikel, Jan Botha, Daniel Gillick, Tom Kwiatkowski, and Andrew McCallum. 2021. MOLEMAN: Mention-only linking of entities with a mention annotation network. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 278–285, Online. Association for Computational Linguistics.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2619–2629, Copenhagen, Denmark. Association for Computational Linguistics.

Zhaochen Guo and Denilson Barbosa. 2018. Robust Named Entity Disambiguation with Random Walks. *Semantic Web*, 9(4):459–479.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415v3*.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 782–792, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

[6] `https://www.fandom.com`

Phong Le and Ivan Titov. 2018. Improving entity linking by modeling latent relations between mentions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1595–1604, Melbourne, Australia. Association for Computational Linguistics.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer Normalization. *arXiv preprint arXiv:1607.06450v1*.

Jeffrey Ling, Nicholas FitzGerald, Zifei Shan, Livio Baldini Soares, Thibault Févry, David Weiss, and Tom Kwiatkowski. 2020. Learning Cross-Context Entity Representations from Text. *arXiv preprint arXiv:2001.03765v1*.

Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. 2019. Zero-Shot Entity Linking by Reading Entity Descriptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3449–3460.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online. Association for Computational Linguistics.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 250–259, Berlin, Germany. Association for Computational Linguistics.

Ikuya Yamada, Koki Washio, Hiroyuki Shindo, and Yuji Matsumoto. 2022. Global entity disambiguation with BERT. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3264–3271, Seattle, United States. Association for Computational Linguistics.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.

number of hidden layers	12
hidden size H	768
entity embedding size D	256
mask probability of entity tokens	0.3
mask probability of word tokens	0
attention heads	12
attention head size	64
activation function	gelu
maximum word length	512
batch size	2048
total epochs	20
learning rate (1st 10 epochs)	5e-4
learning rate decay (1st 10 epochs)	linear
warmup steps (1st 10 epochs)	2500
learning rate	5e-5
learning rate decay	linear
warmup steps	2500
dropout	0.1
weight decay	0.01
gradient clipping	1000
adam β_1	0.9
adam β_2	0.999
adam ϵ	1e-6

Table 9: Hyper-parameters used for training our base ED model.

Appendix

A Details of Training the Base ED model

We implement our base ED model based with PyTorch (v.1.10.1) (Paszke et al., 2019) and Hugging Face's Transformers (v4.15.0) (Wolf et al., 2019). We exploit the base-uncased version of BERT.[7] We optimize the model using AdamW optimizer. Training takes about 10 days using eight Tesla V100 GPUs. Table 9 describes our hyper-parameters.

[7] https://huggingface.co/docs/transformers/v4.15.0/en/model_doc/bert#overview

number of hidden layers	12
hidden size H	768
entity embedding size D	256
attention heads	12
attention head size	64
activation function	gelu
maximum word length	512
batch size	256
learning rate	1e-4
learning rate decay	linear
warmup steps (description-based)	500
warmup steps (hyperlink-based)	1000
dropout	0.1
weight decay	0.01
gradient clipping	1000
adam β_1	0.9
adam β_2	0.999
adam ϵ	1e-6

Table 10: Hyper-parameters used for training our encoders.

B Details of Training the Encoders

We implement the encoders with PyTorch (v.1.10.1) and Hugging Face's Transformers (v4.15.0). We exploit the same version of BERT as in Appendix A. Training of each of the description- and hyperlink-based encoders takes about 7 hours and 17 hours, using a single Quadro RTX 5000 GPU. We optimize the model using AdamW optimizer. We follow most of the hyper-parameters adopted in the base ED model, with the exception of the learning rate. By actually extending the base model with the trained encoder, and by checking its performance on the development set of AIDA-CoNLL (AIDA-testa), we search the best learning rate from {1e-4, 2e-5, 5e-5} for each encoder separately. We describe our hyper-parameters in Table 10.

C Details of Fine-tuning for the evaluation on AIDA-CoNLL dataset

Following the settings of training our base ED model (Appendix A), we split the datasets into sequences consisting of up to 512 word tokens and of up to 128 entity tokens corresponding hyperlinks in the word tokens. We conduct fine-tuning with PyTorch and Trainer provided by Hugging Face. We use a single Quadro RTX 5000 GPU. We search hyper-parameters from the same space as that of Devlin et al. (2019) based on the development set (AIDA-testa):

- batch size: {16, 32}
- learning rate: {2e-5, 3e-5, 5e-5}
- epochs: {2, 3, 4}

Table 11 describes our hyper-parameters used in conducting fine-tuning on the CoNLL dataset.

maximum word length	512
mask probability of entity tokens	0.9
mask probability of word tokens	0
batch size	32
epochs	2
learning rate	3e-5
learning rate decay	linear
warmup propotion	0.1
dropout	0.1
weight decay	0.01
adam β_1	0.9
adam β_2	0.999
adam ϵ	1e-6

Table 11: Hyper-parameters used for fine-tuning our extended model on AIDA-CoNLL.