Learning Entity Linking Features for Emerging Entities

Chenwei Ran, Wei Shen, Jianbo Gao, Yuhan Li, Jianyong Wang, *Fellow, IEEE*, and Yantao Jia

Abstract—Entity linking (EL) is the process of linking entity mentions appearing in text with their corresponding entities in a knowledge base. EL features of entities (e.g., prior probability, relatedness score, and entity embedding) are usually estimated based on Wikipedia. However, for newly emerging entities (EEs) which have just been discovered in news, they may still *not* be included in Wikipedia yet. As a consequence, it is unable to obtain required EL features for those EEs from Wikipedia and EL models will always fail to link ambiguous mentions with those EEs correctly as the absence of their EL features. To deal with this problem, in this paper we focus on a new task of learning EL features for emerging entities in a general way. We propose a novel approach called **STAMO** to learn high-quality EL features for EEs automatically, which needs just a small number of labeled documents for each EE collected from the Web, as it could further leverage the knowledge hidden in the unlabeled data. *STAMO* is mainly based on self-training, which makes it flexibly integrated with any EL feature or EL model, but also makes it easily suffer from the error reinforcement problem caused by the mislabeled data. Instead of some common self-training strategies that try to throw the mislabeled data away explicitly, we regard self-training as a multiple optimization process with respect to the EL features of EEs, and propose both intra-slot and inter-slot optimizations to alleviate the error reinforcement problem implicitly. We construct two EL datasets involving selected EEs to evaluate the quality of obtained EL features for EEs, and the experimental results show that our approach significantly outperforms other baseline methods of learning EL features.

Index Terms—Entity linking, entity linking feature, emerging entity, self-training

---

1 INTRODUCTION

ENTITY linking (EL), the task of mapping entity mentions in text to their corresponding entities in a target knowledge base (KB), is a fundamental step for various knowledge based applications [1], [2], [3], [4], [5], [6]. In general, it is challenging mainly due to the universal many-to-many ambiguity between mentions and entities. On one hand, a named entity may have multiple names (e.g., full name, partial name, nickname, and abbreviation). For example, the named entity *New York City* has a nickname *Big Apple* and an abbreviation *NYC*. On the other hand, a name could possibly denote different named entities. For instance, the name *Michael Jordan* may refer to an NBA basketball player, a Berkeley machine learning professor, or many other named entities which could be referred to as *Michael Jordan*. In recent years, EL models based on deep neural networks have achieved great success [2], [3]. For example, DeepED [7] is a remarkable work which embeds entities and words in a common vector space and leverages a neural attention mechanism over local context windows, while other architectures such as graph neural networks [8], [9], [10] and recurrent random walk networks [11] were investigated as well.

Besides the choice of EL model, another key factor in EL systems is the way to obtain EL features of entities. As reviewed in [1], [2], EL features of entities commonly utilized by EL models to rank candidate entities are prior probability [7], [10], [12], [13], [14], [15], relatedness score [11], [12], [13], [16], [17], [18], and entity embedding [2], [7], [19], [20], [21], [22], [23], [24]. These EL features of entities need to be estimated with the requirement of a labeled corpus where gold mention-entity pairs (i.e., pairs of mentions and their gold corresponding entities) are available [1], [2], [7]. This labeled corpus should cover the entities in the KB as many as possible. If some entities never show up in the labeled corpus, it is unable to estimate required EL features for those entities. In practice, Wikipedia, a large-scale encyclopedia which contains billions of hyperlinks in its articles that could be regarded as gold mention-entity pairs for millions of entities, is leveraged by most EL works as the best choice of the labeled corpus for estimating EL features of entities [1], [2], [3].

- Chenwei Ran is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China. E-mail: ranchenwei@gmail.com.
- Wei Shen, Jianbo Gao, and Yuhan Li are with TMCC, TKLNDST, the College of Computer Science, Nankai University, Tianjin 300350, China. E-mail: shenwei@nankai.edu.cn, {jianbo.gao, yuhanli}@mail.nankai.edu.cn.
- Jianyong Wang is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China, and also with the Jiangsu Collaborative Innovation Center for Language Ability, Jiangsu Normal University, Xuzhou, Jiangsu 221008, China. E-mail: jianyong@tsinghua.edu.cn.
- Yantao Jia is with Huawei Technologies Co., Ltd, Beijing 100077, China. E-mail: jamaths.h@163.com.
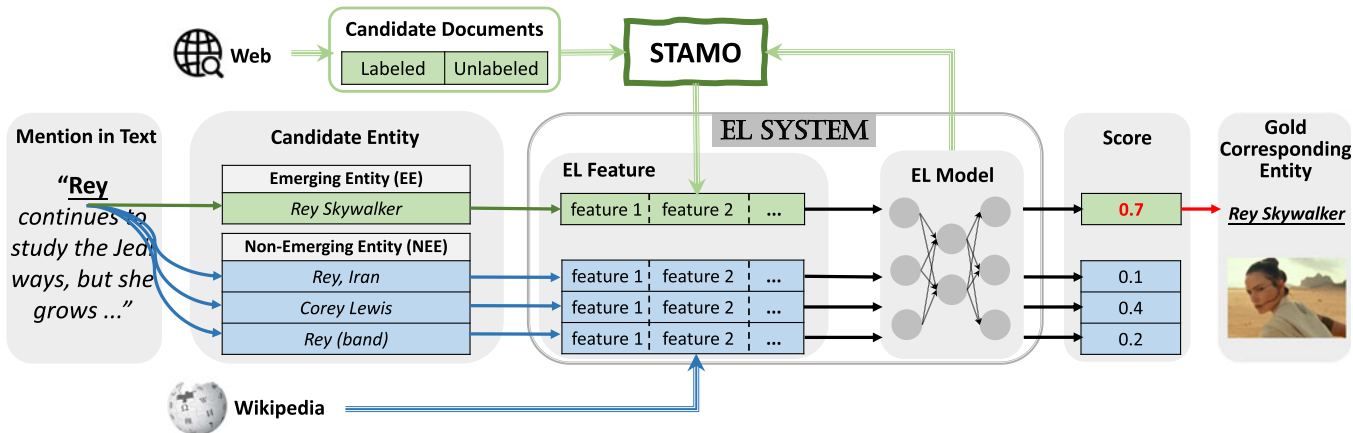
Fig. 1. An illustration for the task of learning entity linking features for an emerging entity. The ambiguous mention *Rey* refers to four candidate entities, among which the EE *Rey Skywalker* is the gold corresponding entity. EL features of the three non-emerging entities (NEEs) could be estimated from Wikipedia (highlighted in blue), whereas for the EE, we cannot obtain its EL features from Wikipedia. In this case, the EL model cannot calculate the score for this EE. To address this problem, the goal is to learn high-quality EL features for the EE based on the labeled and unlabeled candidate documents collected from the Web (highlighted in green). Then via leveraging the learned EL features for this EE, the EL model can calculate its score (i.e., 0.7), which is the highest among the candidates, and link the mention with this EE correctly.

However, it is observed that almost 50% of entities are first mentioned in news before they are included in Wikipedia, and the lags could be several months or even a year [25]. That is to say, although these emerging entities (EEs) could be discovered from news and added to a KB with a unique identifier by hand or automatic methods like [26], they cannot be linked with accurately by EL models because Wikipedia *cannot* provide required EL features for these EEs. To deal with this problem, in this paper we study a new task of learning EL features for emerging entities in a general way, which is illustrated in Fig. 1.

An intuitive idea to learn EL features for an EE is to collect some candidate documents where the EE may be mentioned from the Web, manually annotate them, and then estimate required EL features for the EE on these labeled data. However, this process is non-trivial in practice because manually annotating a large number of candidate documents for each EE is very time-consuming and labor-intensive. Additionally, a prior work [27] proposed a retrieval-based approach to extract keyphrase descriptions for EEs. However, keyphrase description is not a strong EL feature, and their approach cannot be extended to other commonly used EL features (e.g., prior probability, relatedness score, and entity embedding). It is also noted that zero-shot learning [28] has already been considered in the field of entity linking. In the setting of zero-shot EL [23], [29], [30], the entity being linked with has not been seen during training and is defined by a textual entity description from its corresponding Wikipedia page. Zero-shot EL models rely on language understanding of the given textual entity description to perform entity linking. However, this setting makes these zero-shot EL models unable to deal with our problem in this paper, as the EE has not been included in Wikipedia yet and does not have the textual entity description accordingly.

Overall, our goal is to learn high-quality EL features for the EE automatically. The proposed approach is expected to be applicable for any numerical EL feature (e.g., prior probability, relatedness score, and entity embedding) rather than a specific one, as long as the EL feature could be automatically estimated given a labeled corpus. It is also noted that this new task is defined as an orthogonal effort to the task of developing an EL model that most previous EL works focus on. This makes the proposed approach flexibly integrated with any state-of-the-art EL models such as DeepED [7] and Yamada et al. [19], while the prior work [27] is restricted to keyphrase-based EL models [13] only.

In this paper, we propose a novel approach called *STAMO* (Self-Training **A**s **M**ultiple **O**ptimizations) satisfying all the above requirements. *STAMO* is mainly based on self-training, a classic branch of semi-supervised learning. In general case, self-training iteratively assigns pseudo labels to the unlabeled data by a classification model and updates this classification model on both real labeled and pseudo labeled data. While in our case, the pseudo label means the entity that a mention refers to, the classification model is an EL model, and *the target to update* is the EL features of EEs (rather than the EL model) based on both labeled and unlabeled candidate documents collected from the Web, as shown in Fig. 1. *STAMO* needs just a small number of labeled documents for each EE, as it could further leverage the knowledge hidden in the unlabeled data. In addition, *STAMO* learns EL features for EEs *incrementally*. That is to say, given some non-emerging entities (NEEs) whose EL features have been estimated from Wikipedia and a trained EL model, we do not need to update the EL features of NEEs and the EL model every time when an EE is discovered.

The main advantage of self-training is that it is a wrapper method, which makes *STAMO* flexibly integrated with any numerical EL feature or EL model. On the other hand, the major drawback of self-training is that the errors introduced by the mislabeled data may be reinforced gradually [31]. In our task, we found that the vanilla self-training method collapses quickly after a few slots[1], i.e., the quality of EEs' EL features becomes worse after update. To alleviate this problem, a common assumption is that high confident predictions

---

1. In this paper, we will also call an iteration in self-training as a *slot*.

tend to be correct [32], and only a few unlabeled instances with the most confident predictions will be combined with the labeled data in the iterative process of self-training. However, the prediction score output by some EL models (e.g., DeepED [7] and Yamada et al. [19]) does not act as a competent measure of confidence, which is possibly caused by the use of max-margin loss. At the same time, the selection of instances may introduce additional bias, which would significantly affect the quality of some statistics-based features (e.g., prior probability). Therefore, common strategies based on this assumption are unsuitable for our task, and we keep all unlabeled data in all slots as suggested in [31] to update the EL features.

In *STAMO*, we creatively regard self-training as a multiple optimization process with respect to the EL features of EEs, which could alleviate the error reinforcement problem caused by the mislabeled data implicitly, rather than throw the mislabeled data away explicitly. Specifically, we propose both intra-slot optimization and inter-slot optimization. In intra-slot optimization, we regard the EL features of EEs estimated over the mix of the real labeled and pseudo labeled data as an initial guess instead of the final result, and consider that the EL features of EEs should also minimize the objective function which is used to train the given EL model only based on the real labeled data. In inter-slot optimization, we propose a hypothetical optimization process which makes a direct connection between the EL features of EEs in the current slot and the historical slots, and this novel perspective enables us to leverage the information provided by the historical slots to improve the future learning process.

Contributions. The main contributions of this paper can be summarized as follows:

- To the best of our knowledge, we are the first to study the task of learning EL features for EEs in a general way, an orthogonal effort to the task of developing an EL model that most previous EL works focus on.
- We propose a novel approach *STAMO* based on self-training, which makes it flexibly integrated with any numerical EL feature or EL model. We regard self-training as a multiple optimization process, and propose both intra-slot and inter-slot optimizations to alleviate the error reinforcement problem implicitly.
- In inter-slot optimization, we propose a novel hypothetical optimization process, which enables us to leverage the information provided by the historical slots to improve the future learning process.
- We construct two EL datasets to evaluate the quality of the obtained EL features for EEs, and the experimental results show that our approach significantly outperforms other baseline methods of learning EL features.

2 PRELIMINARIES

2.1 Notations and Problem Definition

We begin with some basic concepts in EL. Given a document d, we assume a set of K mentions $\mathbf{m} = \{m_1, m_2, ..., m_K\}$ has been detected (e.g., *Rey* in Fig. 1) via named entity recognition methods [33], [34], [35]. The goal of EL is mapping these mentions to their corresponding entities (e.g., *Rey Skywalker* in Fig. 1) in a target KB which comprises a set of entities E. Each entity $e \in E$ could have different aliases, and the set of its aliases is denoted by $A(e)$. According to the alias dictionary, we can identify a set of candidate entities $E(m) = \{e : m \in A(e)\}$ for a mention m (e.g., *Rey Skywalker* and *Rey (band)* for the mention *Rey* in the aforementioned example). In real-world data, it is usually found that $|A(e)| > 1$ and $|E(m)| > 1$ simultaneously, which inevitably brings the many-to-many ambiguity problem.

To identify the corresponding entity for mention m_i, a real-number probability or score is assigned to each candidate entity $e_i \in E(m_i)$ by an EL model g parameterized by ϕ and the candidate entity with the highest probability or score is output as the linking result. In general, EL model can be divided into two categories: local model and global model. A local model $g : E(m_i) \to R$ links one mention at a time. It predicts the corresponding entity for mention m_i by the following formula:

$$\hat{e}_i = \underset{e_i \in E(m_i)}{\arg\max} \, g_\phi(e_i; m_i)$$

where $\hat{e}_i$ denotes the predicted corresponding entity for mention m_i. Therefore, we need to repeat it for K mentions in document d such that every mention is linked. A global model $g : E(m_1) \times ... \times E(m_K) \to R$ collectively links all mentions in document d at the same time by the following formula:

$$\hat{e}_1, ..., \hat{e}_K = \underset{e_1 \in E(m_1), ..., e_K \in E(m_K)}{\arg\max} \, g_\phi(e_1, ..., e_K; \mathbf{m})$$

Given a labeled corpus S with gold mention-entity pairs $\{(m, e^\dagger)\}$, model parameters ϕ for the EL model g can be optimized with respect to a suitable objective function $\mathcal{O}$. The optimal model parameters are denoted by $\hat{\phi}$.

Next, we introduce more details about EL features for entities. In the context of EL, an EL feature for entities means a table-lookup feature function $f : E \times ... \times E \to \mathbb{R}$ parameterized by θ. In this paper, we call θ feature parameters in order to distinguish them from general model parameters ϕ introduced above. Here we give two examples as follows:

Example 1. The feature function of entity embedding $f : E \to \mathbb{R}^Q$ maps an input entity to a Q-dimensional real vector space. It can be defined as $f(e) = \theta \mathbf{x}_e$, where $\mathbf{x}_e \in \mathbb{R}^{|E|}$ is one-hot encoding of entity e and $\theta \in \mathbb{R}^{Q \times |E|}$ is the corresponding feature parameters.

Example 2. The feature function of relatedness score $f : E \times E \to R$ maps two entities jointly to a real number. It can be defined as $f(e, e') = \mathbf{x}_e^\intercal \theta \mathbf{x}_{e'}$, where $\theta \in \mathbb{R}^{|E| \times |E|}$ is the corresponding feature parameters.

In practice, usually more than one feature would be leveraged. The corresponding feature parameters are grouped together as $\theta = (\theta_1, \theta_2, ...)$. The part of θ that associates with an entity e is denoted by $\theta(e)$.

Then, an EE which has just been discovered and added to a KB is denoted by e^*. e^* has not been included in Wikipedia, and the only information known about EE e^* is its

aliases $A(e^*)$. According to aliases $A(e^*)$, the mentions that may refer to EE e^* could be identified. These mentions are called *candidate mentions*, while other mentions that cannot refer to EE e^* are called *non-candidate mentions*. A set of candidate documents W each of which contains at least one candidate mention is collected from the Web, and a small subset $L \subseteq W$ is manually annotated (i.e., the gold corresponding entities for candidate mentions in L have been labeled manually). Here, it is noted that the gold corresponding entity of a candidate mention is not always the EE, but may be another entity that has the same mention as this EE, as the candidate mention may be ambiguous. The remaining unlabeled data is denoted by U, whose size is usually much larger than the labeled data (i.e., $|U| >> |L|$). Besides candidate mentions, we assume all the non-candidate mentions in both labeled data L and unlabeled data U have been linked to their corresponding entities by an existing EL system (e.g., DeepED [7] and Yamada el al. [19]). In the following, we define our task of learning EL features for EE formally.

Definition 1 (EL feature learning for EE). *Given an emerging entity e^*, a set of candidate documents W of which a subset L is manually labeled and an EL model g, the goal of our task is to learn the optimal EL feature parameters $\hat{\theta}(e^*)$ for EE e^*.*

Note that since we want incremental learning, only feature parameters for EE $\theta(e^*)$ is modifiable and trainable in our proposed approach, while the optimal model parameters $\hat{\phi}$ has been learned with non-emerging entities (NEEs) only, and the optimal feature parameters for NEEs $\hat{\theta}(\overline{e^*})$ have been estimated on Wikipedia, both of which will be frozen all the time. Some notations used in this paper are summarized in Table 1.

2.2 Entity Linking Feature

In this section, we introduce three widely used EL features: prior probability, relatedness score, and entity embedding. The estimation methods and incremental update rules for the corresponding feature parameters are stated as follows.

(1) Prior Probability.

Prior popularity is the probability of the appearance of a candidate entity given an entity mention without considering the context where the mention appears. It has been adopted by most EL methods [2], [7], [10], [12], [13], [14] since it is a simple but strong signal in EL. In many real-world datasets, the accuracy of using prior probability alone can reach more than 80% [7]. Specifically, its parameters $\theta_1 \in \mathbb{R}^{|E| \times |M|}$ are estimated on Wikipedia article corpus (denoted by S_{wiki}) as follows:

$$\theta_1^{i,j} = P(e^i | m^j) = \frac{\text{count}(e^i, m^j; S_{wiki})}{\sum_{e \in E(m^j)} \text{count}(e, m^j; S_{wiki})} \quad (1)$$

where $\text{count}(e, m; S_{wiki})$ denotes the total times of entity e referred by mention m in the Wikipedia corpus S_{wiki}, e^i and m^j denote the ith entity in E and jth mention in M (the set of all mentions) respectively, and $\theta_1^{i,j}$ denotes the element of θ_1 with index (i, j).

For EE e^*, without loss of generality, we assume it is the last entity in E (i.e., $e^* = e^{|E|}$). Given some labeled candidate documents (they could be either the real labeled data or the

TABLE 1
Notations

Symbol	Description
d	A document
m	A mention in d
E	A set of entities in a KB
e	An entity in E
$\hat{e}$	The predicted corresponding entity for mention m
$e^\dagger$	The gold corresponding entity for mention m
e^*	An emerging entity (EE)
$\overline{e^*}$	A non-emerging entity (NEE)
$E(m)$	A set of candidates entities for mention m
$A(e)$	A set of aliases for entity e
g	An EL model
ϕ	EL model parameters w.r.t. EL model g
$\hat{\phi}$	Optimal EL model parameters w.r.t. EL model g
f	EL feature function
θ	EL feature parameters
$\hat{\theta}$	Optimal EL feature parameters
$\theta(e)$	EL feature parameters associated with entity e
θ^t	EL feature parameters at iteration t
W	A set of candidate documents
L	A labeled subset of W
U	An unlabeled subset of W
$\mathcal{O}$	Objective function w.r.t. EL model g
c	A word context surrounding mention m
$\hat{e}$	An entity context surrounding mention m

mix of the real labeled and pseudo labeled data) for $e^{|E|}$ collected from the Web, we denote them by S_{web} and the parameters associated with $e^{|E|}$ are incrementally updated as follows:

$$\theta_1^{|E|,j} = \frac{\text{count}(e^{|E|}, m^j; S_{web})}{\sum_{e \in E(m^j)} \text{count}(e, m^j; S_{web})} \quad (2)$$

Finally, the corresponding feature function is defined as $f_1(e, m) = \mathbf{x}_e^\intercal \theta_1 \mathbf{x}_m$, where $\mathbf{x}_e \in \mathbb{R}^{|E|}$ is one-hot encoding of entity e, and $\mathbf{x}_m \in \mathbb{R}^{|M|}$ is one-hot encoding of mention m.

(2) Relatedness Score.

Relatedness score is usually used to calculate the topical coherence between candidate entities in EL. Wikipedia Link-based Measure (WLM) [36] is the most widely utilized relatedness score in existing EL systems [11], [12], [13], [16], [17]. It is defined under the assumption that two entities tend to be related if there are many documents that link to both. Here, we choose WLM as the relatedness score between entities and define it as follows:

$$WLM(e^i, e^j; S) = \\ 1 - \frac{\log(max(|D(e^i)|, |D(e^j)|) - \log(|D(e^i) \cap D(e^j)|))}{\log(|D|) - \log(min(|D(e^i)|, |D(e^j)|))} \quad (3)$$

where D denotes the set of all documents in the corpus S and $D(e)$ denotes the set of documents where entity e is referred. Therefore, its parameters $\theta_2 \in \mathbb{R}^{|E| \times |E|}$ are estimated on Wikipedia as $\theta_2^{i,j} = WLM(e^i, e^j; S_{wiki})$. For the EE $e^* = e^{|E|}$, we incrementally update the parameters in the $|E|$th row as $\theta_2^{|E|,j} = WLM(e^{|E|}, e^j; S_{web})$ and the parameters in the $|E|$th column as $\theta_2^{j,|E|} = WLM(e^j, e^{|E|}; S_{web})$. The feature function is defined as $f_2(e, e') = \mathbf{x}_e^\intercal \theta_2 \mathbf{x}_{e'}$.

(3) Entity Embedding.

Entity embedding that represents entities in a continuous vector space has been developed rapidly in recent years and leveraged by many deep learning based EL models [2], [7], [19], [20], [21], [22], [23], [24] to calculate the context similarity and the topical coherence between candidate entities. For this feature, the ith column of its parameters $\theta_3 \in \mathbb{R}^{Q \times |E|}$ is set to the entity embedding of e^i, and the feature function is defined as $f_3(e) = \theta_3 \mathbf{x}_e$. We estimate entity embeddings according to the method proposed in DeepED [7]. Specifically, for an entity, we first collect its co-occurrence words from the Wikipedia corpus S_{wiki} and regard them as positive words of that entity. In addition, we randomly sample negative words unrelated to that entity in S_{wiki}. Finally, we use a max-margin loss to infer the optimal embedding of the entity with a goal that embeddings of positive words are closer to the embedding of that entity compared with embeddings of negative words. For the EE $e^* = e^{|E|}$, we incrementally estimate its embedding based on S_{web}.

2.3 Entity Linking Model

In our experiments, we choose two mainstream and advanced EL models to work with our *STAMO*. One is the EL model proposed in Yamada et al. [19], and the other is an extended variation of the local EL model proposed in DeepED [7]. In this section, we give a brief introduction of these two selected EL models and the learning strategy.

2.3.1 Yamada

The model proposed by Yamada et al. [19] (called Yamada in the remainder) is an influential EL model, which mainly relies on several similarity scores to perform linking.

We formally define the word context c of a candidate mention m that needs to be linked as $c = \{w_1, ..., w_H\}$. Since other non-candidate mentions in the same document can be linked by existing EL systems, we consider their corresponding entities as an entity context $\hat{\mathbf{e}} = \{\hat{e}_1, ..., \hat{e}_K\}$ for the mention m. Based on the above definitions, we introduce six kinds of scores leveraged by Yamada in the following. The first one is the prior probability $\Psi_1^{(Y)}(e, m) = f_1(e, m)$ introduced in Section 2.2, which captures the probability of the appearance of an entity given a mention without considering the context. The second score is the context similarity, which captures the similarity between the entity and the words in the context. Following the same setting in Yamada, word embeddings are trained to align with entity embeddings, and a word in the vocabulary (denoted by $w \in V$) is mapped to its embedding by a feature function $f_4^{(Y)}(w) : V \rightarrow \mathbb{R}^Q$. Hence, the context similarity score $\Psi_2^{(Y)}(e, c)$ is derived as follows:

$$\Psi_2^{(Y)}(e, c) = cos(f_3(e), \frac{1}{H} \sum_{w \in c} f_4^{(Y)}(w)) \tag{4}$$

where $cos(\cdot)$ denotes the cosine similarity measure and H is the number of words in the word context c. The third score is the topical coherence, which captures the similarity between the candidate entity and the entity context based on entity embeddings. The topical coherence score $\Psi_3^{(Y)}(e, \hat{\mathbf{e}})$ is defined as follows:

$$\Psi_3^{(Y)}(e, \hat{\mathbf{e}}) = cos(f_3(e), \frac{1}{K} \sum_{e' \in \hat{\mathbf{e}}} f_3(e')) \tag{5}$$

where K is the number of entities in the entity context $\hat{\mathbf{e}}$. In addition, three surface form similarity scores are leveraged by Yamada to capture the similarity between the surface forms of the mention m and the candidate entity e. They are the edit distance, whether the surface form of the candidate entity exactly equals or contains the mention, and whether the surface form of the candidate entity starts or ends with the mention, which are denoted by $\Psi_4^{(Y)}(e, m)$, $\Psi_5^{(Y)}(e, m)$ and $\Psi_6^{(Y)}(e, m)$, respectively. Based on these six scores, Yamada calculates the prediction score $g(e; m)$ for each candidate entity $e \in E(m)$ as follows:

$$g(e; m) = \Gamma^{(Y)}(\Psi_1^{(Y)}(e, m), \Psi_2^{(Y)}(e, c), \Psi_3^{(Y)}(e, \hat{\mathbf{e}}),$$
$$\Psi_4^{(Y)}(e, m), \Psi_5^{(Y)}(e, m), \Psi_6^{(Y)}(e, m)) \tag{6}$$

where $\Gamma^{(Y)}$ is a fully connected layer that generates the final score.

2.3.2 DeepED

DeepED [7] is a remarkable work which embeds entities and words in a common vector space and leverages a neural attention mechanism over local context windows.

We adopt an extended variation of the local EL model proposed in DeepED as the EL model, which utilizes four kinds of scores to perform linking. The first one is the prior probability $\Psi_1^{(D)}(e, m) = f_1(e, m)$. The second score, i.e., the context similarity score, is obtained via applying an attention mechanism to give different attention weights to the words in the context. Similar to Yamada, a word in the vocabulary V is mapped to its embedding via a feature function $f_4^{(D)}(w) : V \rightarrow \mathbb{R}^Q$. We define the context similarity score $\Psi_2^{(D)}(e, c)$ as follows:

$$\Psi_2^{(D)}(e, c) = \sum_{w \in c} \alpha(w) \cdot f_3^{\intercal}(e) \phi_1 f_4^{(D)}(w) \tag{7}$$

where ϕ_1 is a diagonal matrix and $\alpha(w)$ is the attention weight of word w. Before defining $\alpha(w)$, we first introduce a support score $u(w)$ as follows:

$$u(w) = \max_{e' \in E(m)} f_3^{\intercal}(e') \phi_2 f_4^{(D)}(w) \tag{8}$$

where ϕ_2 is another diagonal matrix. We identify the top $H' \leq H$ words with respect to the support score $u(w)$, and the set of these selected words is denoted by c', and then the attention weight is defined as:

$$\alpha(w) = \begin{cases} \frac{\exp(u(w))}{\sum_{w' \in c'} \exp(u(w'))} & \text{for } w \in c' \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The third score is the topical coherence $\Psi_3^{(D)}(e, \hat{\mathbf{e}})$ based on entity embeddings, which is defined as follows:

$$\Psi_3^{(D)}(e, \hat{\mathbf{e}}) = \frac{1}{K} \sum_{e' \in \hat{\mathbf{e}}} (f_3^{\intercal}(e) \phi_3 f_3(e')) \tag{10}$$
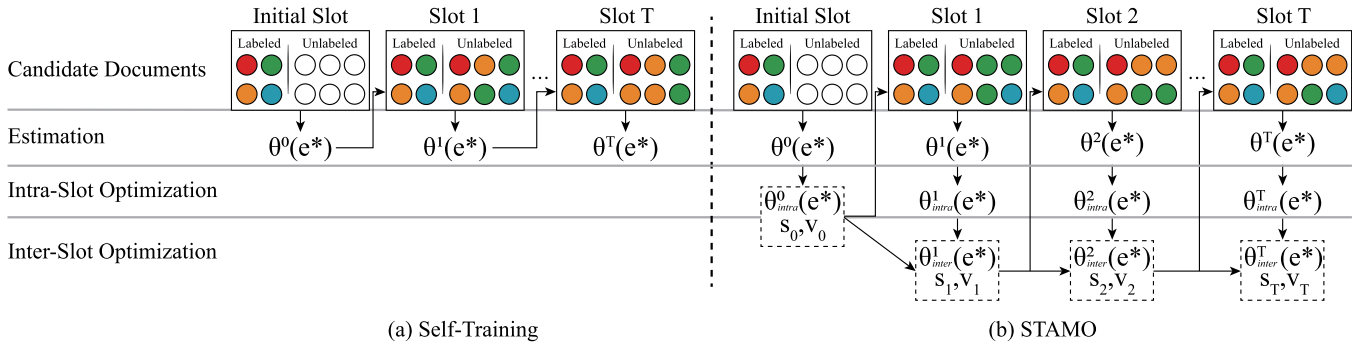
Fig. 2. The iterative process of *Self-Training* and our proposed *STAMO*. Circles of different colors represent different labels of candidate mentions in these candidate documents. Note that s_t and v_t ($t = 0, 1, hellip;, T$) are the gradients (defined in Equation 15) used in inter-slot optimization.

where ϕ_3 is a diagonal matrix as well. The last score is defined as $\Psi_4^{(D)}(e, \hat{\mathbf{e}}) = \frac{1}{K}\sum_{e' \in \hat{\mathbf{e}}} f_2(e, e')$, which captures the topical coherence between entities in a document based on relatedness scores introduced in Section 2.2. DeepED also introduces a fully connected layer $\Gamma^{(D)}$ to generate the prediction score $g(e; m)$ for each candidate entity $e \in E(m)$, which is defined as follows:

$$g(e; m) = \Gamma^{(D)}(\Psi_1^{(D)}(e, m), \Psi_2^{(D)}(e, c), \Psi_3^{(D)}(e, \hat{\mathbf{e}}), \Psi_4^{(D)}(e, \hat{\mathbf{e}}))$$
(11)

2.3.3 Learning

The model parameters of Yamada only include the weights of the function $\Gamma^{(Y)}$, while the model parameters of DeepED include the three diagonal matrices mentioned above (i.e., $\phi_1, \phi_2,$ and ϕ_3) and the weights of the function $\Gamma^{(D)}$. The model parameters of these two introduced EL models are optimized with the same objective function $\mathcal{O}$ with a max-margin loss:

$$\mathcal{O}(\phi; S_{model}) = \sum_{(m, e^\dagger) \in S_{model}} \sum_{e \in E(m)} [\mu - g(e^\dagger; m) + g(e; m)]_+$$
(12)

where ϕ denotes the model parameters of the EL model. S_{model} is a labeled corpus for the EL model training and it only involves NEEs. In our experiments, S_{model} is randomly collected from the annotated Web document corpus introduced in Section 4.1.2 and it contains 10K mention-entity pairs in total.

3 THE PROPOSED APPROACH: STAMO

3.1 Vanilla Self-Training

The simplest way to get feature parameters for EE $\theta(e^*)$ is just estimating them on the labeled data L, but we found when $|L|$ is small, simple estimation results in poor performance. Therefore, we try to leverage the knowledge hidden in the unlabeled data by self-training.

As shown in Fig. 2(a), we initially estimate the EL features of EE e^* on the labeled data L and get $\theta^0(e^*)$ (w.r.t. line 1 in Algorithm 1). Next, we could link the candidate mentions to their mapping entities in the unlabeled data U by using an EL model $g_{\theta^0(e^*)}$ (w.r.t. line 5 in Algorithm 1). Then, we estimate the EL features of EE e^* on both real labeled and pseudo labeled data again and get $\theta^1(e^*)$ (w.r.t. line 6 in Algorithm 1). We repeat these steps until convergence and

finally get $\theta^T(e^*)$ as the best result $\hat{\theta}(e^*)$ (w.r.t. line 15 in Algorithm 1). However, we found this vanilla self-training method did not work well either due to the error reinforcement problem. Specifically, in each slot of self-training, we link the candidate mentions in the unlabeled data U to their mapping entities using an EL model. In most cases, the candidate mention is ambiguous and has more than one candidate entity. Some mapping entity of the candidate mention output by the EL model might be incorrect in reality, which makes some of the pseudo labels over the unlabeled data U incorrect. In the next slot, the quality of EEs' EL features estimated over these mislabeled data becomes worse and subsequently the linking result based on these learned EL features also becomes worse. Accordingly, this leads to the error reinforcement problem.

As shown in Fig. 2(b), in *STAMO* we regard self-training as a multiple optimization process with respect to $\theta(e^*)$, and propose both intra-slot optimization and inter-slot optimization to alleviate the error reinforcement problem implicitly, which are introduced in the following.

3.2 Intra-Slot Optimization

In each slot of self-training, we try to find the best feature parameters of EE e^* that could minimize the objective function $\mathcal{O}$ (which was used to optimize the model parameters ϕ defined in Equation 12) on the labeled data L as follows:

$$\theta_{intra}^t(e^*) = \arg\min_{\theta(e^*)} \mathcal{O}(\theta(e^*); L)$$
(13)

and we consider $\theta^t(e^*)$, the estimated EL feature parameters of EE e^* on both real labeled and pseudo labeled data, as an initial guess of this optimization process rather than the final result (w.r.t. lines 2 and 7 in Algorithm 1). Intra-slot optimization could help the EL feature parameters of EE e^* to be better combined with the optimal EL model parameters $\hat{\phi}$ and the optimal EL feature parameters of non-emerging entities $\hat{\theta}(\overline{e^*})$ estimated on Wikipedia, both of which are well trained before the EE is discovered and are frozen all the time when learning the EL feature parameters for the EE. This capacity is essential since we want to learn the EL features of EE e^* incrementally.

3.3 Inter-Slot Optimization

Moreover, we try to optimize the EL features of EE e^* by leveraging the information provided by the historical slots in self-training. To have this capacity, let us consider how

$\theta_{intra}^t(e^*)$ is generated given the optimal EL feature parameters of the previous slot $\hat{\theta}^{t-1}(e^*)$. The *real procedure* is: we use the EL model to link candidate mentions in U with feature parameters $\hat{\theta}^{t-1}(e^*)$, and then we learn $\theta_{intra}^t(e^*)$ on both real labeled and pseudo labeled data as proposed in Section 3.2. Instead of it, we construct a *hypothetical procedure* which makes a direct connection between $\theta_{intra}^t(e^*)$ and $\hat{\theta}^{t-1}(e^*)$. We first rewrite $\theta_{intra}^t(e^*)$ by an identical equation:

$$\theta_{intra}^t(e^*) \equiv \hat{\theta}^{t-1}(e^*) - 1 \cdot (\hat{\theta}^{t-1}(e^*) - \theta_{intra}^t(e^*)) \tag{14}$$

This identical equation can be explained as follows: consider a hypothetical objective function with respect to $\theta(e^*)$ whose analytical solution of derivative is unknown, but the gradient at the point $\hat{\theta}^{t-1}(e^*)$ is known, whose value is $\delta^t = \hat{\theta}^{t-1}(e^*) - \theta_{intra}^t(e^*)$ (w.r.t. line 8 in Algorithm 1). With this hypothetical objective function, we apply the gradient descent method at the point $\hat{\theta}^{t-1}(e^*)$ and set the learning rate to $\eta = 1$. According to the update rule of gradient descent, the updated point should be $\hat{\theta}^{t-1}(e^*) - \eta \cdot \delta^t$, which is equal to the right part of Equation 14. Therefore, we finally get $\theta_{intra}^t(e^*)$ after the update step. That is to say, in the hypothetical procedure, $\theta_{intra}^t(e^*)$ is generated by an inter-slot optimization process which has an update step for the feature parameters of e^* in the previous slot, rather than the real intra-slot optimization process.

This novel perspective of self-training enables us to improve the EL features of EEs by leveraging the information of historical slots. Specifically, we adjust the gradient $\delta^t = \hat{\theta}^{t-1}(e^*) - \theta_{intra}^t(e^*)$ in consideration of the historical gradients, i.e., its moment estimates. First, the exponential moving averages of the gradient s_t and the squared gradient v_t are iteratively updated (w.r.t. line 9 in Algorithm 1) as follows:

$$\begin{aligned} s_t &= \beta_1 \cdot s_{t-1} + (1 - \beta_1) \cdot \delta^t \\ v_t &= \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot (\delta^t)^2 \end{aligned} \tag{15}$$

where β_1 and β_2 are hyper-parameters that control the exponential decay rates and the initial values of s_t and v_t are both **0** (w.r.t. line 3 in Algorithm 1). Then the bias-corrected estimates are derived (w.r.t. line 10 in Algorithm 1) as follows:

$$\hat{s}_t = s_t / (1 - \beta_1^t) \quad \hat{v}_t = v_t / (1 - \beta_2^t) \tag{16}$$

where β^t denotes β to the power t. Finally, the adaptive gradient is defined as $\hat{\delta}^t = \hat{s}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (w.r.t. line 11 in Algorithm 1), where ϵ is a hyper-parameter to avoid division by zero error. This adjustment method for gradients is first proposed by Adam [37] for the gradient descent method. Besides the adjustment of gradients, we further stabilize inter-slot optimization by decreasing the learning rate η from 1 to a small value, and we use a warm-up scheduling which increases the learning rate linearly in the first γ slots. Therefore, the learning rate in the tth slot is defined as $\eta^t = \min(\eta, \eta \cdot t / \gamma)$ (w.r.t. line 12 in Algorithm 1). The final EL feature parameters of e^* with the improved inter-slot optimization process are defined as follows:

$$\theta_{inter}^t(e^*) = \hat{\theta}^{t-1}(e^*) - \eta^t \cdot \hat{\delta}^t \tag{17}$$

and it is also the optimal EL feature parameters of EE e^* in the tth slot (i.e., $\hat{\theta}^t(e^*) = \theta_{inter}^t(e^*)$) (w.r.t. line 13 in Algorithm 1). The pseudo code of our approach is depicted in Algorithm 1.

Algorithm 1. *STAMO* (Self-Training as Multiple Optimizations)

Input: Labeled set L, unlabeled set U, entity linking model g, learning rate η, warm-up period γ
Output: Optimal feature parameters $\hat{\theta}(e^*)$
1: Estimate $\theta^0(e^*)$ on L
2: $\hat{\theta}^0(e^*) \leftarrow \theta_{intra}^0(e^*) \leftarrow$ intra-slot optimization with the initial guess $\theta^0(e^*)$ according to Equation 13
3: $s_0, v_0 \leftarrow 0$
4: **for** $t = 1, 2, ..., T$ **do**
5:      Apply g with $\hat{\theta}^{t-1}(e^*)$ to all unlabeled candidate mentions in U
6:      Estimate $\theta^t(e^*)$ on $L \cup \{(m, \hat{e}); m \in U\}$
7:      $\theta_{intra}^t(e^*) \leftarrow$ intra-slot optimization with the initial guess $\theta^t(e^*)$ according to Equation 13
8:      $\delta^t \leftarrow \hat{\theta}^{t-1}(e^*) - \theta_{intra}^t(e^*)$
9:      Compute s_t, v_t according to Equation 15
10:     Compute $\hat{s}_t, \hat{v}_t$ according to Equation 16
11:     $\hat{\delta}^t \leftarrow \hat{s}_t / (\sqrt{\hat{v}_t} + \epsilon)$
12:     $\eta^t \leftarrow \min(\eta, \eta \cdot t / \gamma)$
13:     $\hat{\theta}^t(e^*) \leftarrow \theta_{inter}^t(e^*) \leftarrow \hat{\theta}^{t-1}(e^*) - \eta^t \cdot \hat{\delta}^t$
14: **end for**
15: **return** $\hat{\theta}^T(e^*)$

We analyze the time and space complexity for Algorithm 1. The time complexity depends primarily on the iteration part (i.e., lines 4–14). At each iteration, the time complexity of line 5 is $O(K_l \cdot |U|)$, where K_l is the cost to apply the EL model g to one document and $|U|$ is the number of documents in U; the time complexity of line 6 is $O(K_e \cdot (|L| + |U|))$, where K_e is the cost to estimate feature parameters with one document and $|L|$ is the number of documents in L; the time complexity of intra-slot optimization (i.e., line 7) is $O(K_f \cdot |L|)$, where K_f is the cost to optimize feature parameters with one document; and the time complexity of inter-slot optimization (i.e., lines 8–13) is $O(1)$. Therefore, the time complexity of the whole algorithm is $O(T \cdot (K_l \cdot |U| + K_e \cdot (|L| + |U|) + K_f \cdot |L|))$. As the size of the unlabeled data is far larger than that of the labeled data (i.e., $|U| >> |L|$), it could be simplified as $O(T \cdot (K_l + K_e) \cdot |U|)$. The space complexity of the algorithm is $O(d)$, where d is the sum of the dimensions of the learned EL features.

3.4 STAMO Instantiations

It is noted that *STAMO* is an approach of learning EL features for EEs based on self-training, which makes it flexibly integrated with any numerical EL feature or EL model. Depending on which specific EL features and EL models we choose to integrate with our approach *STAMO*, the instantiated whole EL systems may have different architectures and characteristics. However, these variations of instantiated whole EL systems are not the focus of this paper, as our interest is to provide an approach of EL feature learning for EEs rather than enumerating all the possibilities of combinations for the whole EL system. In the following experiment, we choose the two EL models (i.e., Yamada

introduced in Section 2.3.1 and DeepED introduced in Section 2.3.2) and the EL features they utilize to work with our *STAMO*.

4 EXPERIMENTS

4.1 Experimental Setting

To the best of our knowledge, there is no publicly available benchmark dataset for the task of learning entity linking features for emerging entities. According to the task definition introduced in Section 2.1, for each EE, a partially labeled Web document corpus W should be given as the input for learning its EL features. W is a reasonable number of candidate documents each of which contains at least one candidate mention w.r.t. the EE, and a small subset of W should be manually labeled, i.e., the gold corresponding entities for candidate mentions in documents are annotated. However, all the publicly available entity linking datasets (e.g., AIDA-CoNLL [13], TAC-KBP2010 [38], ACE2004 [12], MSNBC [39], AQUAINT [40] and so on) are constructed for the case of linking with non-emerging entities, and do not have a partially labeled Web document corpus for each entity as each entity is mentioned just a few times (mostly once or twice) in these datasets. This limited occurrence of the entity in these datasets cannot provide adequate context for EL feature learning. Accordingly, these publicly available EL datasets do not meet the requirements of our new task, and are unsuitable for the evaluation of our new task of learning EL features for EEs. Therefore, we create a partially labeled Web document corpus W for each EE and additionally construct two EL test datasets involving the selected EEs for our new task. We make the datasets and the source code used in this paper publicly available for future research[2].

4.1.1 Emerging Entities

Since collecting and annotating data for *real* EEs are difficult, we select some entities following the criteria proposed in the prior work [27] and regard them as EEs. The first criterion is that the selected entity should not occur very frequently. On the other hand, a reasonable number of candidate documents for this entity is also required, as we need to create a partially labeled Web document corpus W for this entity. Therefore, we bound the document frequency of the selected entity in the range of 1000 to 2000. The second criterion is *ambiguous*. An alias is considered as ambiguous if it could refer to more than one entity. We provide the top-3 most popular aliases for each selected entity, and we require them all to be ambiguous in order to make the EL problem challenging. Otherwise, if an alias of an entity is unambiguous, it is effortless to link this alias mentioned in text with the only mapping entity correctly. We further define the ambiguity rate of an entity as the proportion of candidate mentions that really refer to it in all its candidate mentions. We bound the ambiguity rate of the selected entity in the range of 0.2 to 0.8, as we hope a reasonable proportion of candidate mentions refer to the selected entity, not too much or too little. Finally, we randomly select 50 entities meeting the above criteria and regard them as

EEs. Their average ambiguity rate is 0.61. We initialize the feature parameters associated with them to zero, and the only information known about them is at most 3 aliases.

4.1.2 Wikipedia and Web Documents

The august 2017 version of English Wikipedia dump is used in our experiments. It contains about 5.5 million article pages, each of which contains 14.2 hyperlinks (i.e., gold mention-entity pairs) on average. The hyperlinks in Wikipedia are expressed by handcraft internal links using wikitext markup[3] and could be identified easily. Wikipedia is used to estimate EL features for non-emerging entities (NEEs), and it is also the target KB in our experiment.

Following the prior work [27], the ClueWeb12-B13 dataset[4] is chosen as the source of Web documents in our experiments, and it contains more than 52 million Web documents in total. The FACC1 dataset [41] provides annotations of mention-entity pairs for the Web documents in ClueWeb by an automatic EL method, and the annotation precision is believed to be around 80-85%. Despite that the annotation provided by the FACC1 dataset is not perfect, it is the largest and the most suitable dataset providing the EL annotation for the Web documents in ClueWeb to the best of our knowledge. Each Web document in ClueWeb has 13 mentions annotated on average.

For each EE, we randomly sampled 1000 candidate documents each of which contains at least one candidate mention of the EE from ClueWeb, and regarded them as the Web document corpus W. Then we leverage W as the source to learn EL features for the EE (i.e., $|W| = 1000$), among which only 20 candidate documents were labeled (i.e., $|L| = 20$). The labels of the candidate mentions in this labeled subset L are provided by FACC1, while the labels of the candidate mentions in the remaining unlabeled set U remain unknown.

4.1.3 EL Datasets

After learning EL features for the EE based on the Web document corpus W, the next step is to evaluate the quality of the learned EL features. Directly evaluating the feature quality is difficult, as we cannot obtain the gold standard EL feature parameters for emerging entities. Thus, we evaluate the EL feature quality via the target downstream task (i.e., entity linking). Specifically, we evaluate the quality of the learned EL features, via assessing the entity linking result of some EL model that leverages the learned EL features for linking candidate mentions of the selected EEs. Therefore, we need to construct EL datasets containing candidate mentions of the selected EEs.

We first construct the EL dataset *Test-Web* from ClueWeb. It is comprised of 200 reserved candidate documents in ClueWeb (i.e., the documents not appearing in W) for each of the 50 selected EEs, and thus this EL dataset *Test-Web* contains 10K documents in total.

Since the labels of the candidate mentions in *Test-Web* are generated by an automatic EL method and thus imperfect, we construct the second EL dataset *Test-Wiki*. It is comprised of 200 candidate documents in Wikipedia for each of

2. https://github.com/stamo4el/STAMO

3. https://en.wikipedia.org/wiki/Help:Wikitext
4. http://lemurproject.org/clueweb12

TABLE 2
Statistics of the Constructed Datasets

Dataset	Web document corpus W	EL dataset *Test-Web*	EL dataset Test-Wiki
# candidate documents	50,000	10,000	10,000
# candidate documents per EE	1,000	200	200

the 50 selected EEs and thus it also contains 10K documents in total. In this *Test-Wiki* dataset, the labels of the candidate mentions are generated using hyperlinks in Wikipedia, which are usually correct annotations. Therefore, we consider that the evaluation results on *Test-Wiki* are more convincing compared with *Test-Web*. Additionally, the *Test-Wiki* dataset is helpful for us to verify if the learned EEs' features can generalize well to different data sources, as EL features are learned based on the Web documents from ClueWeb. The statistics of the constructed datasets are shown in Table 2.

4.1.4   Baselines and Metrics

We study the task of learning EL features for EEs, which is orthogonal to the task of developing an EL model. In order to evaluate different approaches of learning EL features for EEs, we assess the quality of their learned EL features, whereas the performance of different EL models is not the focus of this paper. Therefore, to demonstrate the effectiveness of our proposed approach *STAMO*, we compare it with different approaches of learning EL features for EEs, rather than different EL models.

To the best of our knowledge, there is only one prior work [27] (called *Keyphrase*) addressing the task of learning EL features for EEs. In addition to it, we also create several baselines which are introduced in the following:

- *Keyphrase*: the keyphrase-based method proposed in [27]. Following the setting in their original paper, the keyphrase descriptions mined from Wikipedia for EEs are regarded as gold standard. We assume this method could successfully mine all the gold standard keyphrase descriptions for EEs and thus the result we report for this method could be regarded as the upper bound of this method. According to their setting, after mining keyphrase descriptions for EEs, candidate mentions in text could be linked based on context overlap with keyphrases via the EL model proposed in [13], while our EL models (introduced in Section 2.3) cannot leverage the mined keyphrase descriptions of EEs well for entity linking.
- *BERT*: the method that leverages BERT [42] to learn representations for entities to perform entity linking. Following the same setting in existing BERT-based EL works [23], [30], [43], we employ a BERT-based cross-encoder to jointly encode the mention context and the entity description to perform cross-attention between each other. For each NEE, we could obtain its entity description from its corresponding Wikipedia page. Since EEs have not been included in

Wikipedia yet, we form the entity description for each EE via concatenating contexts of mentions that are labeled as that EE from the labeled data L. For linking, the concatenation of the mention context and the entity description is taken as the input sequence of BERT and a linear layer is applied to the output representation of the [CLS] token to generate the ranking score for each candidate entity. The parameters of BERT are initialized by the BERT base-uncased model in the experiment. We fine-tune this method using the labeled data S_{model} for the EL model training introduced in Section 2.3.3, based on a cross-entropy loss, the same as existing BERT-based EL works [23], [30], [43].

- *Estimation*: the method that simply estimates EEs' EL features based on the given labeled data L. To demonstrate the performance of simply estimating EL features over a large number of labeled candidate documents for each EE, we also tried to increase the size of L from 20 to 1000 (i.e., 1K) and this method is denoted by *Estimation(1K)*.
- *Self-Training*: the vanilla self-training method as introduced in Section 3.1. It is noted that only 20 labeled candidate documents are provided for each EE in the *Self-Training* method. We also tried to improve it via adding intra-slot optimization or inter-slot optimization to it separately. These self-training based baselines are different variants of our proposed *STAMO*. Performance comparison of these baselines and *STAMO* could be regarded as an ablation study to investigate the contribution of each component in *STAMO*.

We assess the quality of the learned EL features for EEs via the entity linking result performed on candidate mentions of these EEs. It is noted that all the baselines introduced above except *Keyphrase* and *BERT* are integrated with the same EL model as our proposed approach *STAMO* to perform entity linking. Therefore, the different EL results of those baselines and our proposed *STAMO* are only determined by the EL features they learn. That is to say, the better the final EL result is, the better the quality of the learned EL features is.

The first metric we choose to assess the EL result is accuracy. Given some test candidate mentions whose gold entities are $\{e_1^\dagger, ..., e_N^\dagger\}$ and predicted entities are $\{\hat{e}_1, ...\hat{e}_N\}$, accuracy is defined as follows:

$$Acc = \frac{\sum_{i=1}^N \mathbb{1}(\hat{e}_i = e_i^\dagger)}{N}$$

where $\mathbb{1}(x)$ is an indicator function whose value is 1 if condition x is true, otherwise the value is 0. Since in this paper we focus on EEs, we further consider an EE as a query, and define precision, recall, and F_1-score with respect to the EE as follows:

$$P = \frac{\sum_{i=1}^N \mathbb{1}(\hat{e}_i = e_i^\dagger \;\; and \;\; \hat{e}_i = e^*)}{\sum_{i=1}^N \mathbb{1}(\hat{e}_i = e^*)}$$

$$R = \frac{\sum_{i=1}^N \mathbb{1}(\hat{e}_i = e_i^\dagger \;\; and \;\; e_i^\dagger = e^*)}{\sum_{i=1}^N \mathbb{1}(e_i^\dagger = e^*)} \quad F_1 = 2 \cdot \frac{P \cdot R}{P + R}$$

TABLE 3
Experimental Results Over the two EL Datasets

Methods		Test-Web				Test-Wiki				Avg.		Time (min)
		Acc	P	R	F_1	Acc	P	R	F_1	Acc	F_1	
Keyphrase		33.32	82.56	48.83	61.39	44.96	87.15	65.23	74.61	39.14	68.00	-
BERT		53.88	98.24	29.51	45.39	52.44	95.53	38.32	54.70	53.16	50.04	-
Estimation		46.03	88.75	45.03	59.75	43.03	92.43	26.24	40.87	44.53	50.31	-
Estimation(1K)		**65.39**	87.77	78.76	**83.02**	59.16	88.59	57.74	69.92	62.27	76.47	-
Self-Training	+ Yamada	47.07	71.97	57.17	63.72	48.93	80.70	41.62	54.92	48.00	59.32	13.52
+Intra-Slot Optimization		62.00	79.92	79.44	79.68	68.13	91.20	72.43	80.74	65.06	80.21	13.82
+Inter-Slot Optimization		51.42	87.44	55.47	67.88	49.48	91.41	37.77	53.46	50.45	60.67	13.63
STAMO		63.86	89.95	76.28	82.55	**68.94**	94.53	71.18	**81.21**	66.40	81.88	13.95
Estimation		49.50	78.87	55.75	65.32	49.06	94.23	18.24	30.56	49.28	47.94	-
Estimation(1K)		68.11	81.41	87.72	**84.45**	64.43	90.81	47.79	62.63	66.27	73.54	-
Self-Training	+ DeepED	55.25	63.50	77.57	69.83	59.47	86.40	40.76	55.39	57.36	62.61	12.13
+Intra-Slot Optimization		63.32	69.74	87.27	77.52	81.85	93.67	77.40	84.76	72.54	81.14	12.60
+Inter-Slot Optimization		62.36	66.42	89.63	76.30	66.13	88.33	52.38	65.76	64.25	71.03	12.72
STAMO		**68.48**	70.06	97.87	81.66	**86.19**	92.71	86.03	**89.24**	77.34	85.45	13.25

4.1.5 Preprocessing

For the ClueWeb12-B13 dataset, we extract main textual contents of Web documents by Boilerpipe[5]. For Wikipedia, we choose WikiExtractor[6] to preprocess the dump so that we can identify these mentions and their corresponding entities in Wikipedia articles. As the label of the mention in the Web document provided by the FACC1 dataset is its corresponding Freebase entity, we align the entity in Freebase to Wikipedia via the attribute "key:wikipedia.en_id".

4.1.6 Training Details

In our experiments, most hyper-parameters are set to the recommended value, and we only describe the different parts here. When updating entity embeddings of EEs, we train them for 35 epochs with the learning rate of 1e-2. For the EL model described in Section 2.3, we keep the top 30 candidate entities for each mention based on prior probability when training, while all candidate entities are kept when testing. The fully connected layer $\Gamma^{(Y)}$ used in Yamada sequentially applies the neural layers [Linear(6, 100), Dropout(0.1), ReLU(), Linear(100, 1)], while the fully connected layer $\Gamma^{(D)}$ used in DeepED sequentially applies the neural layers [Linear(4, 100), Dropout(0.1), ReLU(), Linear(100, 1)]. In intra-slot optimization, we train $\theta(e^*)$ for 50 epochs with the learning rate of 1e-4, and the batch size is set to 32. In inter-slot optimization, we apply different learning rates η to different groups of feature parameters as follows: $\eta(\theta_1) = 5e\text{-}2, \eta(\theta_2) = 5e\text{-}2, \eta(\theta_3) = 1e\text{-}3$. The warm-up period γ is set to 5.

4.2 Effectiveness Study

The experimental results of all approaches over the two EL datasets (i.e., *Test-Web* and *Test-Wiki*) are shown in Table 3. From the results, it can be seen that even though the gold standard keyphrase descriptions are provided for EEs, *Keyphrase* does not perform well since keyphrase description is not a strong and effective EL feature. The baseline *BERT*

also performs poorly on both datasets, which may be due to the reason that it is unable to learn high-quality representations for EEs as the absence of their corresponding Wikipedia pages. In addition to the two baselines above (i.e., *Keyphrase* and *BERT*), other approaches shown in Table 3 (e.g., *Estimation*, *Estimation(1K)*, *Self-Training*, and *STAMO*) are integrated with each of the two EL models (i.e., Yamada and DeepED) to perform entity linking. From the results, we can see that the performance of these approaches has a similar trend when they are integrated with Yamada and DeepED. It is also noted that we should compare their performance when they are integrated with the same EL model, either Yamada or DeepED. Specifically, when 20 labeled candidate documents are provided for each EE, *Estimation* (+Yamada or +DeepED) which simply estimates EEs' EL features on these labeled data does not perform well either. When the number of labeled candidate documents increases greatly to 1K, we could see that *Estimation(1K)* (+Yamada or +DeepED) achieves the highest F1-score on *Test-Web*, which is consistent with our intuition. However, this method is impractical in reality because manually annotating such a large number (i.e., 1K) of candidate documents for each EE is very time-consuming and labor-intensive. To our surprise, *Estimation(1K)* (+Yamada or +DeepED) does not achieve good performance on the *Test-Wiki* dataset. The reason may be that the underlying distributions of documents from Wikipedia and Web are different. EL features simply estimated from Web documents in ClueWeb do not generalize well to Wikipedia documents in *Test-Wiki*.

Compared with *Estimation*, *Self-Training* with the same size of the labeled data L achieves much better results. Specifically, when integrated with Yamada, *Self-Training* achieves improvements of 3.47% average accuracy and 9.01% average F1-score compared with *Estimation*. When integrated with DeepED, *Self-Training* achieves improvements of 8.08% average accuracy and 14.67% average F1-score compared with *Estimation*. It indicates that *Self-Training* leverages the knowledge hidden in the unlabeled data effectively to learn better EL features. Then, adding intra-slot optimization or inter-slot optimization to *Self-Training* (+Yamada or +DeepED) separately can both
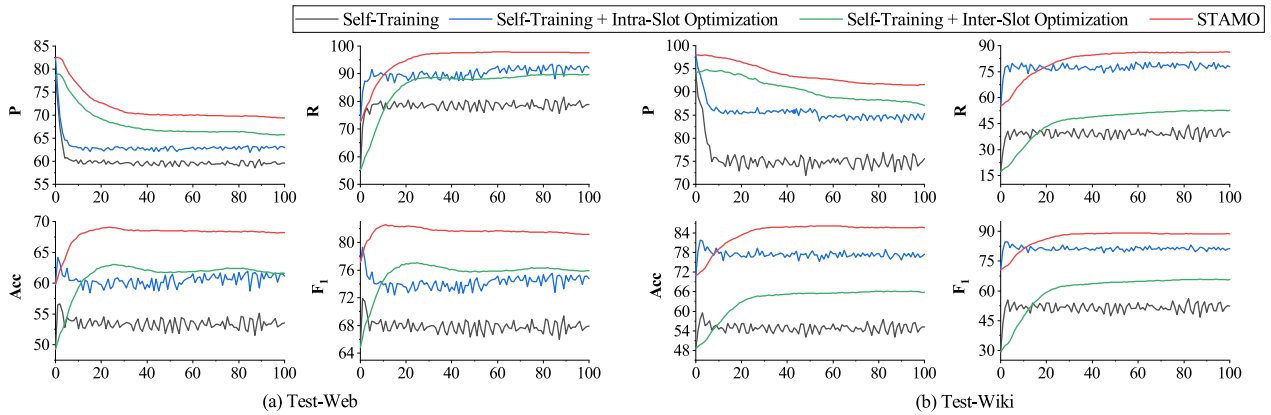
5. https://boilerpipe-web.appspot.com/
6. https://github.com/attardi/wikiextractor

Fig. 3. Performances of different self-training based methods as iterations progress. Horizontal axes represent the number of iterations t.

improve this vanilla self-training method in terms of average accuracy and average F_1-score. Finally, with the use of both intra-slot optimization and inter-slot optimization, *STAMO* (+Yamada or +DeepED) achieves the best performance in terms of average accuracy and average F1-score. Specifically, when integrated with Yamada, *STAMO* achieves improvements of 27.26% (13.24%) average accuracy and 13.88% (31.84%) average F_1-score compared with the baseline *Keyphrase* (*BERT*). When integrated with DeepED, *STAMO* achieves improvements of 38.20% (24.18%) average accuracy and 17.45% (35.41%) average F_1-score compared with the baseline *Keyphrase* (*BERT*). Moreover, *STAMO* even outperforms *Estimation(1K)* on the *Test-Wiki* dataset and on average by a large margin, under the condition that *Estimation(1K)* uses a labeled data whose size is 50 times larger than *STAMO*. This demonstrates that our proposed *STAMO* could learn high-quality EL features for EEs with the requirement of just a small number of labeled documents for each EE and each component of *STAMO* has a significant positive contribution to the *STAMO*'s performance.

From the results in Table 3, we can also see that DeepED performs better than Yamada in most cases when they work with the same EL feature learning approach, as DeepED leverages a neural attention mechanism over local context windows and the topical coherence based on relatedness scores. However, as stated before, the performance of different EL models is not the focus of this paper since this paper focuses on the task of learning EL features for EEs rather than developing an EL model.

The run-times of *STAMO* (and its several variants) and the baselines (i.e., *Keyphrase* and *BERT*) are not comparable, as they learn EL features in totally different ways. Therefore, we just demonstrate the run-times of our *STAMO* and its variants (i.e., the other three self-training based methods). As the goal of our task is to learn EL features for EEs based on the Web document corpus W, we show the average EL feature learning time per iteration for each self-training based method in Table 3. From the results, we can see that the vanilla self-training method *Self-Training* has the least time consumption among all the self-training based methods when integrated with Yamada or DeepED. Adding intra-slot or inter-slot optimization to *Self-Training* would increase a small amount of learning time. Finally, the complete method *STAMO* is the most time-consuming, which is

consistent with our intuition. From another perspective, we can see that among the whole process of *STAMO*, the vanilla self-training process (including applying the EL model to the unlabeled data and estimating EL features based on both real labeled and pseudo labeled data) consumes most of the learning time, while the two optimization processes consume much less learning time, which is consistent with our time complexity analysis depicted in Section 3.3.

4.3 Convergence Study

Self-training does not guarantee to converge since early errors may be reinforced. In this experiment, we evaluate the performances of different self-training based methods as iterations progress. Their experimental results based on DeepED are shown in Fig. 3. We could see intra-slot optimization and inter-slot optimization are both helpful, while inter-slot optimization plays a critical role in the stabilization of the iterative process. Then, with the help of intra-slot optimization and inter-slot optimization together, *STAMO* achieves the best performance on the two EL datasets in terms of all the metrics among these self-training based methods.

In addition, it is observed that for the *STAMO* method, precision is the only metric decreasing in the iterative process. We believe it is caused by the error reinforcement problem. In our case, "errors" mainly means false positive predictions with respect to EEs. These wrong predictions would mislead the feature learning approach to regard wrong entities or words as relevant context of EEs, which further brings more false positive predictions. *STAMO* achieves the highest precision when convergence, which demonstrates its effectiveness in alleviating the error reinforcement problem.

4.4 Ablation Study

We conduct an ablation study to verify the effectiveness of different modules applied in inter-slot optimization. There are three modules: (1) the adaptive adjustment of gradient δ, whose original value is $\theta^{t-1} - \theta^t$; (2) setting learning rate η to a small value, whose original value is 1; and (3) the warm-up scheduling. We remove each of them separately and the experimental results of *STAMO*+DeepED are shown in Table 4. We can see that the adjustments of δ and

TABLE 4
Ablation Study of Inter-Slot Optimization

Methods	Test-Web		Test-Wiki		Avg.	
	Acc	F_1	Acc	F_1	Acc	F_1
STAMO	**68.48**	**81.66**	**86.19**	**89.24**	**77.34**	**85.45**
w/o changing δ	66.83	81.03	84.09	87.05	75.50	84.03
w/o changing η	64.25	79.46	85.31	89.05	74.78	84.26
w/o warm-up	68.32	81.53	85.66	88.89	76.99	85.21

TABLE 5
Ablation Study of EL Features

EL features	Avg.			
	Acc	ΔAcc	F_1	ΔF_1
Three EL features	**77.34**	-	**85.45**	-
w/o prior probability	50.26	-27.08	56.50	-28.95
w/o relatedness score	67.92	-9.42	83.22	-2.23
w/o entity embedding	63.44	-13.90	72.37	-13.08

η are both important, while the warm-up step also makes a slight improvement. We further investigate the change of average F_1-score on the two datasets as iterations progress. The experimental results of STAMO+DeepED are shown in Fig. 4. It can be seen that setting η to a small value is a simple but effective way to keep the iterative process stable, which is similar to the effect of learning rate in the gradient descent method: a too large learning rate may never reach the optimal solution while a too small learning rate will take a long time to converge. In addition, the adaptive adjustment of δ makes self-training converge to a better result. As analyzed by [37], the adaptive adjustment of δ can be regarded as the signal-to-noise ratio. Since noise represents uncertainty, we believe such an adaptive adjustment could help our approach focus on the confident part when updating feature parameters for EEs, and thus it could alleviate the error reinforcement problem implicitly. Warm-up is usually used in batch gradient descent approaches to alleviate the problem of instable gradient at the beginning of training. While in our case, the problem of instable gradient may not be obvious and thus the contribution of warm-up seems slight from the experimental results.

We also conduct an ablation study to show the effectiveness of different EL features (i.e., prior probability, relatedness score, and entity embedding) used in this paper. We remove each of them separately and the experimental results of STAMO+DeepED are shown in Table 5. We can see that each of the three EL features has a positive contribution to the final performance and when all the three EL features are leveraged together, it yields the best performance, which is consistent with our intuition. It is also noted that our proposed STAMO could be flexibly integrated with any other numerical EL feature besides these three features.
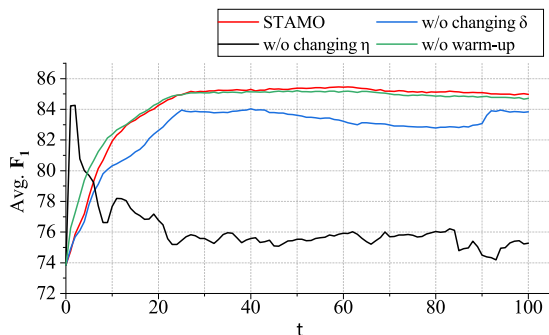
4.5 Effect Analysis of Labeled Data Size

Finally, we aim at answering the following question: how does the number of labeled documents influence the quality of EEs' features? We vary the labeled data size $|L|$, and the experimental results based on DeepED are shown in Fig. 5. It is observed that for the approach Estimation, the labeled data size has a significant impact on its results, while STAMO is much more robust. When reducing the labeled data size from 250 to 20, the F_1-score of Estimation changes -17.66% and -19.98% on Test-Web and Test-Wiki respectively, while the F_1-score of our proposed approach STAMO changes -3.26% and $+1.27\%$ respectively. It demonstrates that our approach is highly effective to leverage the knowledge hidden in the unlabeled data and it can greatly reduce the cost of human efforts.

5 RELATED WORK

Two threads of research are related to our work, i.e., entity linking and self-training. We will introduce them in details.

5.1 Entity Linking

Fully-supervised EL has been well studied by abundant literature [1], [2], where "fully-supervised" means not only the training of EL models is supervised by some documents annotated for EL specifically (e.g., AIDA-CoNLL [13]), but also the estimation of EL features for entities is supervised by a labeled corpus (e.g., Wikipedia). Specifically, prior probability provides the probability of the appearance of a candidate entity given the mention without considering the context where the mention appears. It is usually estimated using some structured information provided by Wikipedia such as hyperlinks in Wikipedia articles and pageview counts of the candidate entity. It plays a significant role in most EL methods [7], [10], [12], [13], [14], [15] due to its strong effectiveness. Relatedness score (e.g., WLM [36]) is

Fig. 4. Average F_1-score as iterations progress and different modules are removed from inter-slot optimization separately.
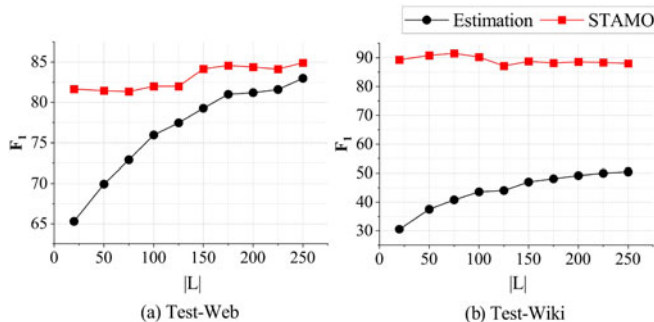

Fig. 5. Effect analysis of the labeled data size.

adopted by lots of EL works [11], [12], [13], [16], [17], [18] to calculate the topical coherence between candidate entities. WLM is defined under the assumption that two entities tend to be related if there are many Wikipedia articles that link to both.

Additionally, with the advent of deep learning, entity embedding that encodes each candidate entity as a dense and low-dimensional vector has been applied in many deep learning based EL methods [2], [7], [19], [20], [21], [22], [23], [24] to implicitly represent the semantic and syntactic properties of the candidate entity. Rich data existing in Wikipedia (e.g., textual entity description [7], [22], [23], entity context [20], [24], and hyperlink structure of entity pages [7], [19], [21]) are usually leveraged to learn entity embeddings. To sum up, the estimation of these EL features for entities depends on various structures existing in Wikipedia.

Since manual annotation is time-consuming, weakly-supervised setting for EL was considered in recent works [44], [45]. In this setting, KBs like Wikipedia are available while documents in the target domain are unlabeled (i.e., mentions are linked without any labeled examples). Specifically, Le and Titov [44] performed message passing based on the Wikipedia link subgraph corresponding to the document to construct a high recall list of candidates for each mention. What's more, Le and Titov [45] leveraged entity type information provided by Freebase [46] to generate entity embeddings for weakly-supervised linking.

In addition, zero-shot learning [28] has also been considered in the field of entity linking. The key idea of zero-shot is to train the model on a domain with rich labeled data resources and apply it to a new domain with minimal data [3]. In the setting of zero-shot EL [23], [29], [30], the entity being linked with has not been seen during training and is only defined by a textual entity description from its corresponding Wikipedia page. Specifically, Logeswaran et al. [30] and Wu et al. [23] utilized a BERT-based [42] cross-encoder to perform joint encoding of mentions and entities. The cross-encoder takes the concatenation of the mention context and the textual entity description to produce a scalar score for each candidate entity. Wu et al. [23] and Li et al. [29] leveraged a BERT-based [42] bi-encoder which uses two separate encoders to learn vectors of the mention context and the textual entity description respectively. The candidate entity ranking is performed by comparing dot-products of representation vectors.

In this paper, we consider a completely different setting from the settings introduced above. In our setting, EEs are newly discovered and have not be included in Wikipedia yet. Accordingly, no data from Wikipedia could be provided for EEs, which makes the existing EL models unable to link ambiguous mentions with EEs correctly as the absence of their EL features. Due to the fact that the world is constantly evolving and newly emerging entities are constantly being discovered, such a setting is essential and meaningful for real-world applications.

5.2 Self-Training

Self-training, or broadly speaking self-labeled, is an intuitive but successful methodology to tackle the semi-supervised

classification problem. An overview of self-labeled techniques is presented in [32]. In general, traditional self-labeled methods aim at finding accurate confidence measures. For example, co-training [47] is a classic multi-view self-labeled technique. In co-training, each instance is represented by two sets of features, which are also called two views, and then two classifiers are separately trained on different views. If these two views are both redundant and conditionally independent, co-training can effectively reduce generalization errors. Recently, class-balanced self-training [48] and confidence regularized self-training [49] achieved state-of-the-art results in semantic segmentation. Since instance selection based on the confidence score could be impractical in some scenarios, some recent works [50], [51] also began to focus on self-training without instance selection. In this paper, we creatively regard self-training as a multiple optimization process with respect to the EL features of EEs, and propose both intra-slot and inter-slot optimizations to alleviate the error reinforcement problem caused by the mislabeled data implicitly, rather than throw the mislabeled data away explicitly.

6 CONCLUSION

In this paper, we study a new task of learning EL features for emerging entities in a general way. We propose a novel approach *STAMO* to learn high-quality EL features for EEs automatically, which needs just a small number of labeled documents for each EE collected from the Web, as it could further leverage the knowledge hidden in the unlabeled data. *STAMO* is mainly based on self-training, which makes it flexibly integrated with any numerical EL feature or EL model. We regard self-training as a multiple optimization process and propose both intra-slot optimization and inter-slot optimization to alleviate the error reinforcement problem implicitly. In intra-slot optimization, we consider that the EL features of EEs should also minimize the objective function which is used to train the given EL model. In inter-slot optimization, we propose a hypothetical optimization process which makes a direct connection between the EL features of EEs in the current slot and the historical slots, and this novel perspective enables us to leverage the information provided by the historical slots to improve the future learning process. Finally, extensive experiments demonstrate the effectiveness of our approach.

REFERENCES

[1] W. Shen, J. Wang, and J. Han, "Entity linking with a knowledge base: Issues, techniques, and solutions," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 2, pp. 443–460, Feb. 2015.

[2] W. Shen, Y. Li, Y. Liu, J. Han, J. Wang, and X. Yuan, "Entity linking meets deep learning: Techniques and solutions," *IEEE Trans. Knowl. Data Eng.*, to be published, doi: 10.1109/TKDE.2021.3117715.

[3] O. Sevgili, A. Shelmanov, M. Arkhipov, A. Panchenko, and C. Biemann, "Neural entity linking: A survey of models based on deep learning," 2020, *arXiv:2006.00575*.

[4] W. Ma et al., "Jointly learning explainable rules for recommendation with knowledge graph," in *Proc. World Wide Web Conf.*, 2019, pp. 1210–1221.

[5] Y. Hao et al., "An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, 2017, pp. 221–231.

[6] R. Blanco, G. Ottaviano, and E. Meij, "Fast and space-efficient entity linking for queries," in *Proc. 8th ACM Int. Conf. Web Search Data Mining*, 2015, pp. 179–188.

[7] O.-E. Ganea and T. Hofmann, "Deep joint entity disambiguation with local neural attention," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 2619–2629.

[8] Y. Cao, L. Hou, J. Li, and Z. Liu, "Neural collective entity linking," in *Proc. 29th Int. Conf. Comput. Linguistics*, 2018, pp. 675–686.

[9] J. Wu, R. Zhang, Y. Mao, H. Guo, M. Soflaei, and J. Huai, "Dynamic graph convolutional networks for entity linking," in *Proc. Web Conf.*, 2020, pp. 1149–1159.

[10] Z. Fang, Y. Cao, R. Li, Z. Zhang, Y. Liu, and S. Wang, "High quality candidate generation and sequential graph attention network for entity linking," in *Proc. Web Conf.*, 2020, pp. 640–650.

[11] M. Xue et al., "Neural collective entity linking based on recurrent random walk network learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 5327–5333.

[12] L. Ratinov, D. Roth, D. Downey, and M. Anderson, "Local and global algorithms for disambiguation to wikipedia," in *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics Hum. Lang. Technol.*, 2011, pp. 1375–1384.

[13] J. Hoffart et al., "Robust disambiguation of named entities in text," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2011, pp. 782–792.

[14] P. Le and I. Titov, "Improving entity linking by modeling latent relations between mentions," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 1595–1604.

[15] C. Ran, W. Shen, and J. Wang, "An attention factor graph model for tweet entity linking," in *Proc. World Wide Web Conf.*, 2018, pp. 1135–1144.

[16] W. Shen, J. Wang, P. Luo, and M. Wang, "Linden: Linking named entities with knowledge base via semantic knowledge," in *Proc. World Wide Web Conf.*, 2012, pp. 449–458.

[17] M. C. Phan, A. Sun, Y. Tay, J. Han, and C. Li, "Pair-linking for collective entity disambiguation: Two could be better than all," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 7, pp. 1383–1396, Jul. 2018.

[18] J. Hoffart, S. Seufert, D. B. Nguyen, M. Theobald, and G. Weikum, "KORE: Keyphrase overlap relatedness for entity disambiguation," in *Proc. 21st ACM Int. Conf. Informat. Knowl. Manage.*, 2012, pp. 545–554.

[19] I. Yamada, H. Shindo, H. Takeda, and Y. Takefuji, "Joint learning of the embedding of words and entities for named entity disambiguation," in *Proc. 20th SIGNLL Conf. Comput. Natural Lang. Learn.*, 2016, pp. 250–259.

[20] Y. Cao, L. Huang, H. Ji, X. Chen, and J. Li, "Bridge text and knowledge by learning multi-prototype entity mention embedding," in *Proc. ACL*, 2017, pp. 1623–1633.

[21] Ö. Sevgili, A. Panchenko, and C. Biemann, "Improving neural entity disambiguation with graph embeddings," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics Student Res. Workshop*, 2019, pp. 315–322.

[22] Z. Fang, Y. Cao, Q. Li, D. Zhang, Z. Zhang, and Y. Liu, "Joint entity linking with deep reinforcement learning," in *Proc. World Wide Web Conf.*, 2019, pp. 438–447.

[23] L. Wu, F. Petroni, M. Josifoski, S. Riedel, and L. Zettlemoyer, "Scalable zero-shot entity linking with dense entity retrieval," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2020, pp. 6397–6407.

[24] S. Zwicklbauer, C. Seifert, and M. Granitzer, "Robust and collective entity disambiguation through semantic embeddings," in *Proc. 39th Int. ACM SIGIR Conf. on Res. Develop. Informat. Retrieval*, 2016, pp. 425–434.

[25] B. Fetahu, A. Anand, and A. Anand, "How much is wikipedia lagging behind news?," in *Proc. ACM Web Sci. Conf.*, 2015, Art. no. 28.

[26] J. Hoffart, Y. Altun, and G. Weikum, "Discovering emerging entities with ambiguous names," in *Proc. 23rd Int. Conf. World Wide Web*, 2014, pp. 385–396.

[27] J. Singh, J. Hoffart, and A. Anand, "Discovering entities with just a little help from you," in *Proc. 25th ACM Int. Conf. Informat. Knowl. Manage.*, 2016, pp. 1331–1340.

[28] B. Romera-Paredes and P. Torr, "An embarrassingly simple approach to zero-shot learning," in *Proc. 32nd Int. Conf. Int. Conf. Mach. Learn.*, 2015, pp. 2152–2161.

[29] B. Z. Li, S. Min, S. Iyer, Y. Mehdad, and W.-T. Yih, "Efficient one-pass end-to-end entity linking for questions," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2020, pp. 6433–6441.

[30] L. Logeswaran, M.-W. Chang, K. Lee, K. Toutanova, J. Devlin, and H. Lee, "Zero-shot entity linking by reading entity descriptions," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 3449–3460.

[31] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synth. Lectures Artif. Intell. Mach. Learn.*, vol. 3, no. 1, pp. 1–130, 2009.

[32] I. Triguero, S. García, and F. Herrera, "Self-labeled techniques for semi-supervised learning: Taxonomy, software and empirical study," *Knowl. Informat. Syst.*, vol. 42, no. 2, pp. 245–284, 2015.

[33] J. Li, P. Han, X. Ren, J. Hu, L. Chen, and S. Shang, "Sequence labeling with meta-learning," *IEEE Trans. Knowl. Data Eng.*, to be published, doi: 10.1109/TKDE.2021.3118469.

[34] J. Li, B. Chiu, S. Feng, and H. Wang, "Few-shot named entity recognition via meta-learning," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 9, pp. 4245–4256, Sep. 2022.

[35] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 50–70, Jan. 2020.

[36] D. M. I. H. Witten, "An effective, low-cost measure of semantic relatedness obtained from wikipedia links," *Wikipedia Artif. Intell. An Evolving Synergy*, pp. 25–30, 2008.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.

[38] H. Ji and R. Grishman, "Knowledge base population: Successful approaches and challenges," in *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics Hum. Lang. Technol.*, 2011, pp. 1148–1158.

[39] S. Cucerzan, "Large-scale named entity disambiguation based on wikipedia data," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2007, pp. 708–716.

[40] D. Milne and I. H. Witten, "Learning to link with wikipedia," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2008, pp. 509–518.

[41] E. Gabrilovich, M. Ringgaard, and A. Subramanya, "FACC1: Freebase annotation of clueweb corpora, version 1 (release date 2013–06-26, format version 1, correction level 0)," 2013.

[42] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Annu. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2019, pp. 4171–4186.

[43] H. Tang, X. Sun, B. Jin, and F. Zhang, "A bidirectional multi-paragraph reading model for zero-shot entity linking," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 13 889–13 897.

[44] P. Le and I. Titov, "Boosting entity linking performance by leveraging unlabeled documents," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 1935–1945.

[45] P. Le and I. Titov, "Distant learning for entity linking with automatic noise detection," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 4081–4090.

[46] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2008, pp. 1247–1250.

[47] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proc. 11th Annu. Conf. Comput. Learn. Theory*, 1998, pp. 92–100.

[48] Y. Zou, Z. Yu, B. Vijaya Kumar, and J. Wang, "Unsupervised domain adaptation for semantic segmentation via class-balanced self-training," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 289–305.

[49] Y. Zou, Z. Yu, X. Liu, B. Kumar, and J. Wang, "Confidence regularized self-training," in *Proc. Eur. Conf. Comput. Vis.*, 2019, pp. 5982–5991.

[50] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imageNet classification," in *Proc. IEEE/ CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10 687–10 698.

[51] J. He, J. Gu, J. Shen, and M. Ranzato, "Revisiting self-training for neural sequence generation," in *Proc. Int. Conf. Learn. Representations*, 2019.

Chenwei Ran received the BS degree and the PhD degree in computer science from Tsinghua University, China, in 2014 and 2020 respectively. His research interests include entity linking, Web search, and natural language processing.

Wei Shen received the PhD degree in computer science from Tsinghua University, China, in 2014. He is currently an associate professor in the College of Computer Science, Nankai University, China. His research interests include entity linking, knowledge base population, and text mining. He was a recipient of ACM China Rising Star Award (Honorable Mention), CCF-Intel Young Faculty Researcher Program, and CAAI Outstanding Doctoral Dissertation Award.

Jianbo Gao received his BS degree in computer science from the Hefei University of Technology, China, in 2020. He is currently a master candidate with Nankai University. His research interests include data mining and knowledge graph.

Yuhan Li received the BS degree from Northeast Forestry University, China, in 2020. He is currently working toward the master candidate with Nankai University. His research interests include knowledge graph, entity linking and data mining.

Jianyong Wang (Fellow, IEEE) received the PhD degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, in 1999. He is currently a professor in the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests mainly include data mining and Web information management. He has co-authored more than 60 papers in some leading international conferences and some top international journals. He is serving or ever served as a PC member for some leading international conferences, such as SIGKDD, VLDB, ICDE, WWW, and an associate editor of *IEEE Transactions on Knowledge and Data Engineering* and *ACM Transactions on Knowledge Discovery from Data*. He is a member of the ACM.

Yantao Jia received the PhD degree in mathematics. He is a technical expert in Huawei Technologies Co. Ltd. His main research interests include search, recommendation, open knowledge computing, social computing, and combinatorial algorithms.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.