IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 11, 2021, accepted March 2, 2021, date of publication March 4, 2021, date of current version March 16, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3064054

Learning Document-Level Label Propagation and Instance Selection by Deep Q-Network for Interactive Named Entity Annotation

TINGMING LU[1,2], YAOCHENG GUI[3], AND ZHIQIANG GAO[1,2]
[1]Key Laboratory of Computer Network and Information Integration, Ministry of Education, Southeast University, Nanjing 210096, China
[2]School of Computer Science and Engineering, Southeast University, Nanjing 210096, China
[3]School of Modern Posts and the Institute of Modern Posts, Nanjing University of Posts and Telecommunications, Nanjing 210003, China

Corresponding author: Zhiqiang Gao (zqgao@seu.edu.cn)

This work was supported by the National Science Foundation of China under Grant 61602260, Grant 61702279, and Grant 61502095.

ABSTRACT High quality annotated named entity corpora are essential language resources, but fully manual annotation is time-consuming. Interactive annotation offers an efficient alternative where humans and machines collaborate. Instances of named entity mentions tend to share the same label, when they co-occur in the same document and have similar surface forms. After selecting an instance in one sentence for manual annotation, the label of the instance can be propagated to instances in other sentences. This kind of document-level label propagation can be used to reduce human effort and improve annotation quality in interactive annotation. However, most existing literature assumes instances within different sentences are independent, and ignores document-level label propagation. This paper proposes a reinforcement learning-based approach, which learns to propagate labels among the instances within a document for interactive named entity annotation. In addition, our approach also learns instance selection for manual annotation. We optimize the objective which is a trade-off between human effort and annotation quality by training a deep Q-network. Our approach reduces human effort by more than 42% compared to baseline approaches, for achieving the same annotation quality (0.95 measured by F1 averaged on three datasets).

INDEX TERMS Interactive annotation, named entity recognition, reinforcement learning, deep Q-network.

I. INTRODUCTION

With the current success and widespread use of data-driven techniques for natural language processing, annotated corpora become essential resources [1]. Training of machine learning models, especially deep learning-based models, requires big annotated corpora [2], [3]. However, it is not easy to obtain high quality annotated corpora. Fully manual annotation is time-consuming and costly. Although recent developments in fully automatic annotation of named entity have improved prediction performance significantly [2]–[5], there are still some errors hard to be corrected automatically [6].

Interactive annotation offers an efficient alternative where humans and machines collaborate [7]–[9]. It is an iterative process of prediction and instance selection. Based on prediction confidences, some instances are selected for manual annotation, and then the manual annotation results are used to improve prediction of other instances. The goal of interactive annotation is to reduce human effort while maintaining annotation quality, or achieve better annotation quality under the same amount of human effort.

Document-level label propagation can be used to reduce human effort and improve annotation quality for interactive named entity (NE) annotation. Instances of NE mentions tend to share the same label, when they co-occur in the same document and have similar surface forms [10]. This kind of document-level label consistency is illustrated in a document in AKSW-News dataset [11] in Fig. 1. Four NE mentions rendered in red occur in three sentences and co-refer to the "Nintendo" company. If one of them has been manually annotated, then its label can be propagated to other ones. Labels can also be propagated at document-level among the four NE mentions rendered in blue (MICROSOFT, Microsoft Corp. etc.), and among the three NE mentions rendered in green (NOMURA, Nomura Research Institute Ltd. etc.).

The associate editor coordinating the review of this manuscript and approving it for publication was Arianna Dulizia.

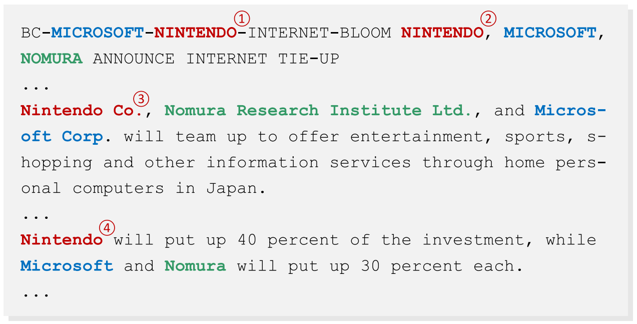

```
BC-MICROSOFT-NINTENDO①-INTERNET-BLOOM NINTENDO②, MICROSOFT,
NOMURA ANNOUNCE INTERNET TIE-UP
...
Nintendo Co.③, Nomura Research Institute Ltd., and Micros-
oft Corp. will team up to offer entertainment, sports, s-
hopping and other information services through home pers-
onal computers in Japan.
...
Nintendo④will put up 40 percent of the investment, while
Microsoft and Nomura will put up 30 percent each.
...
```

FIGURE 1. **Illustration of document-level label consistency among instances located in different sentences within a document.**

TABLE 1. **Comparison of our approach to previous approaches for interactive named entity annotation.**

Approach	Document-level label propagation	Instance selection
Constrained Inference (CI) [12]	N/A	Rule-based
Active Learning (AL) [13], [14]	N/A	Rule-based
k-best CI+AL [7]	N/A	Rule-based
Clustering+CI [8]	Rule-based	Rule-based
RL-DINEA (ours)	Learning-based	Learning-based

However, most previous interactive NE annotation approaches assume instances within different sentences are independent, with document-level label propagation ignored (see Table 1). The constrained inference-based (CI) approach [12] propagates labels in the inference phase at sentence level, but not at document-level. The active learning-based (AL) approach [13] uses manual annotation results in the training phase to learn a prediction model, and document-level label propagation is not considered. The k-best CI+AL [7] approach combines CI and AL, but also ignores document-level label propagation. To the best of our knowledge, the Clustering+CI approach [8] is the only previous work which propagates labels at document level for interactive NE annotation (see Table 1). In Clustering+CI, rule-based propagation strategies gather multiple instances into a cluster, and force the instances in the cluster to share the same label. Nevertheless, it is still challenging to design a "good" propagation strategy for interactive NE annotation.

To solve the above issues, this paper models the interactive NE annotation of a whole document as a Markov decision process (MDP), and proposes RL-DINEA, short for Reinforcement Learning-based Document-level Interactive Named Entity Annotation. Reinforcement learning is suitable for interactive annotation tasks due to its intrinsic sequentiality [15]. Document-level label propagation and instance selection are viewed as actions in the MDP. The objective of balancing human effort and annotation quality is reflected in rewards. The policy to select optimal actions is learned by training a deep Q-network [16].

In summary, this work makes the following contributions.

(1) We propose a reinforcement learning-based approach RL-DINEA to learn document-level label propagation and instance selection for interactive named entity annotation. By training a deep Q-network, RL-DINEA optimizes an objective function which is a trade-off between human effort and annotation quality. The source code is publicly available at GitHub.[1]

(2) Experimental results on AKSW-News, CoNLL'03, and Ontonotes 5.0 datasets demonstrate the effectiveness of RL-DINEA. To achieve the same annotation quality (0.95 measured by F1 averaged on the three datasets), RL-DINEA reduces human effort by more than 42% compared to the baseline approaches. Our learning-based document-level label propagation significantly outperforms rule-based propagation in previous studies, in terms of contribution to the improvement of annotation quality under the same amount of human effort.

The remaining part of this paper is organized as follows: Section II reviews related work. Section III introduces preliminary definitions. Section IV details our approach. The experiments are described in Section V, followed by the results and analysis in Section VI. Finally, we conclude in Section VII.

II. RELATED WORK

In this section, we first review the previous study on interactive NE annotation. Subsequently we introduce how document-level label propagation is leveraged to improve named entity recognition, which is closely related to interactive NE annotation. At last, reinforcement learning-based interactive annotation approaches are briefly introduced.

A. INTERACTIVE NAMED ENTITY ANNOTATION

Most previous interactive NE annotation approaches assume instances within different sentences are independent [12], [13], except Goldberg *et al.* [8] which adopt rule-based strategies to propagate manually annotated labels across sentences. Constrained inference-based approaches use manual annotation results in the inference phase [7], [8], [12], while active learning-based approaches use manual annotation results in the training phase [7], [13].

Kristjansson *et al.* [12] propose a constrained inference-based (CI) approach for interactive information extraction. The constrained Viterbi algorithm is presented, which modifies the traditional Viterbi algorithm by pruning from the search space those labelings that do not agree with the manual annotation results. In this way, when a label of a token is manually corrected, the labels of some other tokens will be automatically changed. They call this capability correction propagation.

Skeppstedt *et al.* [13] propose an active learning-based (AL) approach. Initially, a prediction model is trained on manually annotated sentences. The rest sentences are annotated automatically by the model. Then a batch of least confident sentences are selected for manual annotation, and are added to the training set to retrain the model. This procedure iterates until a stop condition is met.

[1]https://github.com/KrisWentaoWong/dqnner-c/

Zhang *et al.* [14] propose an active learning-based (AL) approach. They focus on annotation of entities that can often be expressed by regular expressions (regexes), such as bill date, email address, course number, phone number, etc. Humans write regexes, which are used to scan a corpus and produce weak labels to pretrain a neuron network. Then, based on the network, the most uncertain instances are selected for manual annotation. Finally, the manually annotated instances are used to fine-tune the network. They study the trade-off between spending human effort in writing regexes and spending human effort on manual annotation.

Culotta *et al.* [7] propose a hybrid approach that combines k-best constrained inference and active learning (k-best CI+AL). Based on a prediction model, k-best CI constrains the k-best Viterbi algorithm [17] to prune predictions that do not agree with the manual annotation results, and returns top k predictions. For any NE mention in these k predictions that is segmented correctly but classified incorrectly, humans are allowed to provide the correct label for this NE mention. In this way, the amount of human effort spent on segmentation is reduced. k-best CI and human correction are iteratively performed until all tokens within the current sentence are correctly labeled. Then the current sentence is added to the training set to retrain the prediction model.

Goldberg *et al.* [8] combine clustering and constrained inference (Clustering+CI), where rule-based clustering strategies are used to propagate manual annotation results across sentences. Multiple tokens are gathered into a cluster based on some rule-based strategies. One of the tokens in the cluster is selected for manual annotation, and the other ones are forced to share the same label. Then all these tokens are used as constraints in constrained inference.

It should be noted that unlike CI [12] and Clustering+CI [8], the AL approaches [13], [14] and the k-best CI+AL [7] approach suffer the following limitations. First, AL [13], [14] and k-best CI+AL [7] require humans to check and correct all tokens within a selected sentence. Manually annotating an entire sentence leads to unneeded redundancy, since a sentence usually contains only sparse labeling errors [8]. CI and Clustering+CI overcome this limitation by selecting tokens and requiring only one or part of the tokens within a sentence to be manually annotated. Second, AL cannot utilize manual annotation results immediately to improve prediction until the prediction model is retrained. Model retraining is often time-consuming. Hence, it is performed after a batch of sentences is manually annotated, while constrained inference can be performed immediately after a token is manually annotated.

In summary, Clustering+CI [8] is the only approach which makes document-level label propagation in the previous studies, and our RL-DINEA is different from Clustering+CI in terms of how to propagate labels at document level. Clustering+CI adopts rule-based strategies to force tokens with similar contexts to share a common label, while our RL-DINEA adopts a data-driven strategy which learns to propagate labels among candidate NE mentions with similar surface forms.

B. DOCUMENT-LEVEL LABEL PROPAGATION FOR NAMED ENTITY RECOGNITION

Document-level label propagation has been shown to improve performance of NER [18]–[23], which is a task closely related to interactive NE annotation. NER aims at extracting named entities from texts [24]. In AL-based interactive NE annotation, NER models are iteratively trained and make predictions. In CI-based interactive NE annotation, NER models make predictions under constraints of manual annotation results.

NER with document-level label propagation is usually approached using a two-stage framework [18]–[20]. In the first stage, draft labels for tokens or candidate NEs are predicted by a base model. In the second stage, the draft labels are refined by document-level label propagation.

Wang *et al.* [18] propose a document-level optimization approach to NER. Long-short-term memory (LSTM) is applied to perform token classification. Then, a global objective function is defined on the basis of the token classification results to achieve document-level optimization via Integer Linear Programming.

Gui *et al.* [20] introduce a label refinement approach to handle document-level label consistency. In this approach, a key-value memory network is first used to record draft labels predicted by the base model, and then a multi-channel Transformer refines these draft predictions based on the explicit co-occurrence relationship derived from the memory network.

Our previous work [19] proposes a reinforcement learning-based approach to document-level NER, short for RL-DNER. Candidate NEs with draft labels and estimated confidences are produced by ensemble of multiple existing NE taggers. Afterwards, document-level label propagation is modeled as actions in a framework of Markov decision process.

This paper extends our previous work RL-DNER [19] to interactive annotation setting, and proposes RL-DINEA. RL-DNER and RL-DINEA are approaches for different tasks–NER and interactive NE annotation, respectively. RL-DNER focuses only on maximizing performance of prediction, without interaction with humans. Whereas RL-DINEA is a human-in-the-loop approach. RL-DINEA attempts to solve a multi-objective problem, aiming at balancing human effort and annotation quality.

C. REINFORCEMENT LEARNING FOR INTERACTIVE ANNOTATION

Over the past few years, reinforcement learning has emerged as a powerful tool for solving complex sequential decision making problems. It is well known for its success in the area of game, such as Atari [16] and Go games [25]. RL is suitable for interactive annotation tasks due to its intrinsic sequentiality, and has been applied to interactive annotation of images and videos [15], [26], [27].

Liao *et al.* [15] model interactive image segmentation as a MDP and solve it with multi-agent RL. At each step, an RL model predicts the labels of all voxels, based on the previous predictions and human supervisions. After that, the model receives a feedback according to predefined measurement of segmentation. The above process is repeated until the maximum number of interactions is reached. Multi-agent RL is adopted to solve above MDP by finding the segmentation strategy to maximize the accumulated feedbacks received at each step.

Varga and Lorincz [26] learn instance selection in interactive video segmentation using a modified Dueling Deep Q-Network. The RL agent is trained to select optimal frames for manual annotation, by estimating the scale of potential discrepancies between the predicted labels and image-derived features. Once manual annotation results are provided by humans, label propagation algorithm re-estimates its label predictions for the whole video. The above two steps iterate until a given budget of human effort is exhausted.

Our RL-DINEA differs from previous RL-based interactive annotation approaches in two ways. First, previous approaches adopt RL either to learn prediction [15], or to learn instance selection [26], [27], while RL-DINEA jointly learns prediction and instance selection. Second, previous approaches apply RL in image and video annotation tasks, while RL-DINEA is for named entity annotation.

III. PRELIMINARY DEFINITIONS

This section presents preliminary definitions, including token sequence, document, named entity, instance, etc.

A. TOKEN SEQUENCE

Token sequence and three types of relations between token sequences are defined in *Definition 1–4*.

Definition 1 (Token Sequence): A token sequence

$$w = (w_1, w_2, \ldots, w_{|w|}), \quad w_i \in \mathcal{W},$$

is a sequence of tokens, where $|w|$ is the number of tokens in w; w_i is the i-th token in w, for $i = 1, \ldots, |w|$; $\mathcal{W}$ is the token space.

Definition 2 (Identical Token Sequences): A token sequence w and another token sequence $\tilde{w} = (\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_{|\tilde{w}|})$ are identical token sequences, if $|w| = |\tilde{w}|$ and $w_i = \tilde{w}_i$, for $i = 1, \ldots, |w|$.

Definition 3 (Sub-Token Sequence): A token sequence $\tilde{w}$ is a sub-token sequence of w, if $|\tilde{w}| < |w|$ and there exists an offset $b \in \{0, 1, \ldots, |w| - |\tilde{w}|\}$, such that $\tilde{w}_i = w_{b+i}$, for $i = 1, \ldots, |\tilde{w}|$.

Definition 4 (Super-Token Sequence): A token sequence $\tilde{w}$ is a super-token sequence of w, if w is a sub-token sequence of $|\tilde{w}|$.

B. DOCUMENT AND NAMED ENTITY

Document and named entity are defined in *Definition 5–6*.

Definition 5 (Document): A document d is a token sequence.

Definition 6 (Named Entity): A named entity (NE, entity) in a document d is a tuple

$$e = \langle w, b, y \rangle,$$

where w is a sub-token sequence of d; b is the offset of w in d; $y \in \mathcal{Y}$ is the entity label; $\mathcal{Y}$ is the entity label space consisting of possible entity types such as person, location and organization as well as a nil label ϕ indicating not-an-entity.

C. INSTANCE

We follow the convention of using the term "instance" to describe an occurrence of a token sequence for prediction and selection in interactive annotation. We formally define it in *Definition 7*.

Definition 7 (Instance): An instance x in a document d is a tuple

$$x = \langle w, b \rangle,$$

where w is a sub-token sequence of d; b is the offset of w in d. The instance space is denoted by $\mathcal{X}$.

According to Definition 7, an instance can refer to a sentence, a token (a token sequence consisting of only one token), or a candidate NE mention. Although this definition is general, the term "instance" in our approach is restricted to the meaning of candidate NE mention. So, according to *Definition 6* and *Definition 7*, a named entity e can also be represented by a tuple of an instance x and a label y:

$$e = \langle x, y \rangle.$$

Furthermore, five types of relations between instances are defined in *Definition 8–12*.

Definition 8 (Identical-Form Instance): $\tilde{x} = \langle \tilde{w}, \tilde{b} \rangle$ in a document d is an identical-form instance of $x = \langle w, b \rangle$ in d, if w and $\tilde{w}$ are identical token sequences, and $\tilde{b} \neq b$.

Definition 9 (Sub-Form Instance): $\tilde{x} = \langle \tilde{w}, \tilde{b} \rangle$ in a document d is a sub-form instance of $x = \langle w, b \rangle$ in d, if $\tilde{w}$ is a sub-token sequence of w.

Definition 10 (Super-Form Instance): $\tilde{x} = \langle \tilde{w}, \tilde{b} \rangle$ in a document d is a super-form instance of $x = \langle w, b \rangle$ in d, if $\tilde{w}$ is a super-token sequence of w.

For example, in Fig. 1, the first instance `NINTENDO` is an identical-form instance of the second instance `Nintendo`, and is a sub-form instance of `Nintendo Co.` `Nintendo Co.` is a super-form instance of the first instance `NINTENDO`. Case is ignored to improve performance, as suggested by Krishnan and Manning [10].

Definition 11 (Similar-Form Instance): $\tilde{x}$ is a similar-form instance of x, if $\tilde{x}$ is an identical-, super- or sub-form instance of x.

Definition 12 (Overlapping Instances): $x = \langle w, b \rangle$ and $\tilde{x} = \langle \tilde{w}, \tilde{b} \rangle$ are overlapping instances, if $[b, b + |w| - 1] \cap [\tilde{b}, \tilde{b} + |\tilde{w}| - 1] \neq \emptyset$.

IV. OUR APPROACH

This section describes our reinforcement learning-based approach RL-DINEA which learns document-level label propagation and instance selection for interactive NE annotation. An overview of RL-DINEA is given in Subsection IV-A. Preprocess is introduced in Subsection IV-B. The objective function of our approach is defined in Subsection IV-C. The RL model is detailed in IV-D.

A. OVERVIEW

An overview of RL-DINEA is presented in Fig. 2. RL-DINEA takes a document d as input, and outputs a sequence of named entities E. First, instances of candidate NE mentions with draft labels and confidences are extracted from d in a preprocessing step. Then, based on the extracted instances, the RL model consisting of an environment and a DQN agent performs document-level label propagation and instance selection iteratively in the interactive annotation setting. In this process, labels of the instances are refined. Finally, the RL model outputs the named entities in d by filtering out non-entity instances.

The time step t is initialized to 0 after preprocess. The index $i^{(t)}$ of the current instance is initialized to 1, and the index $j^{(t)}$ of the similar-form instance is chosen randomly. X, $Y^{(t)}$, $C^{(t)}$, $i^{(t)}$ and $j^{(t)}$ forms the current state $s^{(t)}$. Based on $s^{(t)}$, the DQN agent chooses an action which consists of a propagation sub-action and a selection sub-action. The propagation sub-action $a_{\mathrm{I}}^{(t)}$ determines whether to propagate the label $Y_{j^{(t)}}^{(t)}$ of the similar-form instance $X_{j^{(t)}}$ to the current instance $X_{i^{(t)}}$. The selection sub-action $a_{\mathrm{II}}^{(t)}$ requests the environment to respond in one of the following ways: a) select the current instance $X_{i^{(t)}}$ for manual annotation by humans, b) query another identical-, super-, or sub-form instance, or c) start to process next instance. The time step t is updated by $t \leftarrow t+1$ after the sub-actions are performed by the environment.

The above process iterates until all the instances in X have been processed. The terminal time step t is stored in τ, which will be used in Subsection IV-C.

B. PREPROCESS

This subsection introduces preprocess which extracts instances with draft label and estimated confidences from the input document. Firstly, multiple existing NE taggers which are called base taggers in this paper tag the input document, and output extracted instances of candidate NE mentions. Then, for each of the instances, an ensemble classifier assigns a draft label to it, and estimates confidence.

There are two reasons to perform preprocess. First, based on the instances produced in the preprocess, document-level label propagation among similar instances can be explicitly modeled in the interactive annotation setting. Second, after the instances which are candidate NE mentions have been extracted, manually annotating a selected instance is reduced to assigning a label of entity type to it. In this way, human do not have to determine the start and the end boundaries for any selected instance, which makes manual annotation faster and easier [7].

Resulted from decades of research on NER, there are many existing NE taggers, and they can be used to tag NEs as candidates for further processing. A base tagger can be any existing NE tagger, no matter it is based on a rule-based model [28] or a learning-based model (e.g., SVM [29], CRF [30], BERT [31], XLNet [32], GPT [33]).

Given a document d, instances are extracted by tagging d using K base taggers. The k-th base tagger outputs a sequence of named entities E^k, which can be viewed as a

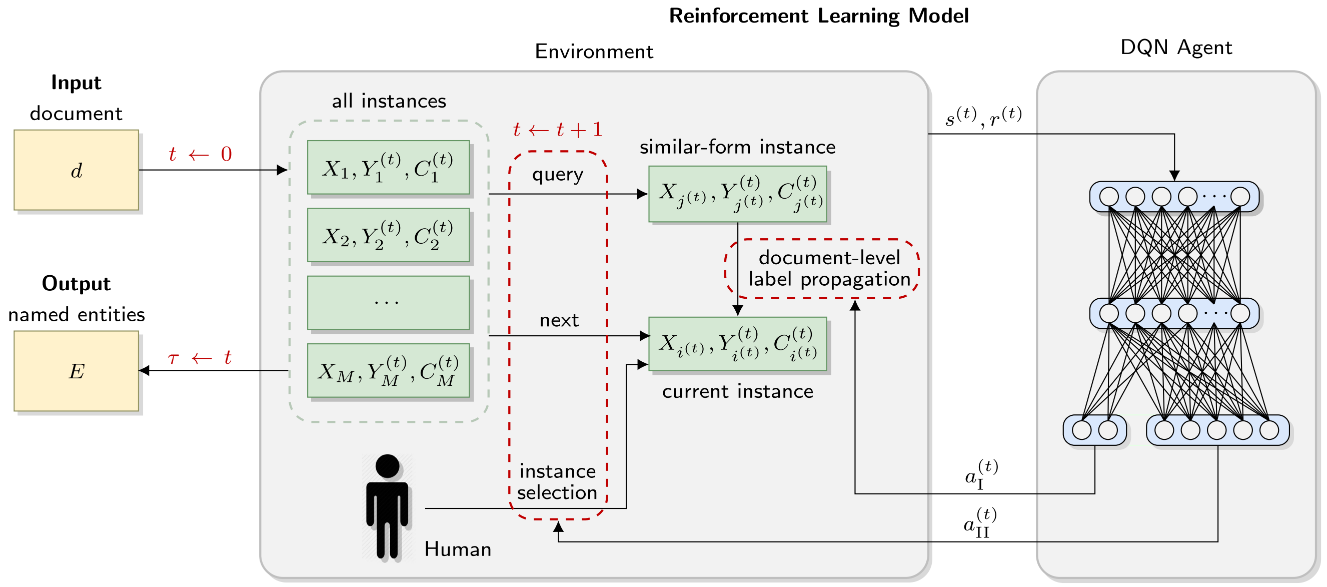


FIGURE 2. Overview of the reinforcement learning-based approach for document-level interactive named entity annotation.

tuple $\langle X^k, Y^k \rangle$, where X^k is a sequence of instances; Y^k is a sequence of labels. Instances in d can be obtained by uniting $X^1, X^2, \ldots, X^K$,

$$X = \bigcup_{k=1}^{K} X^k. \tag{1}$$

For each instance X_i in X (1), $i = 1, 2, \ldots, M$, where M is the size of X, the ensemble classifier assigns a draft label to X_i, and estimates the corresponding confidence, based on the K base taggers' output and local context of X_i. The ensemble classifier usually outperforms the best individual base tagger [11]. In addition, the ensemble classifier estimate confidences which are essential to our RL model, whereas not all base taggers can do so.

Features used by the ensemble classifier include base tagger features and context features.

- The base tagger features represent labels predicted by the K base taggers for a given instance X_i, defined as

$$f^k_{base} = \begin{cases} y & \text{if } \exists \langle X_i, y \rangle, \langle X_i, y \rangle \in E^k, \\ \phi & \text{otherwise,} \end{cases}$$

  where E^k is the sequence of the named entities tagged by the k-th base tagger on d, $k = 1, 2, \ldots, K$; $\langle X_i, y \rangle \in E^k$ denotes the relation that $\langle X_i, y \rangle$ is an element of the sequence E^k.
- The local context features are simply part-of-speech tags (postags). f^l_{postag} represents the postag of the token at location l, $l \in \{\mathsf{start}, \mathsf{end}, \mathsf{prev}, \mathsf{succ}\}$, indicating locations of the start token within X_i, end token within X_i, previous token before X_i, and succeeded token after X_i.

The ensemble classifier is implemented using LIBSVM [34]. LIBSVM applies the "one-against-one" [35] approach to solve the multiclass classification problem. $|\mathcal{Y}|(|\mathcal{Y}|-1)/2$ binary classifiers are constructed, where $|\mathcal{Y}|$ is the number of classes. In classification LIBSVM uses a voting strategy: each binary classification is considered to be a voting. The instance X_i is assigned with a label Y_i corresponding to the class with the maximum number of votes. Confidence C_i of X_i being assigned with Y_i is defined as the posterior probability $P(Y_i|X_i)$ estimated by LIBSVM.

C. OBJECTIVE FUNCTION

Given a set of documents D, interactive NE annotation aims to maximize the following objective function:

$$\mathcal{J}(D) \doteq \mathcal{Q}(D) - \lambda \mathcal{E}(D), \tag{2}$$

where $\mathcal{Q}$ is the function of annotation quality (Definition 13); $\mathcal{E}$ is the function of human effort (Definition 14); λ is a trade-off parameter balancing human effort and annotation quality, $\lambda \in \mathbb{R}_{\geq 0}$. Interactive annotation reduces to fully manual annotation if λ is close to zero, and to pure prediction if λ is sufficient large.[2]

[2]The exact value of λ depends on how $\mathcal{Q}(D)$ and $\mathcal{E}(D)$ are defined.

Definition 13 (Annotation Quality): Given a set of documents D, annotation quality is defined as the accuracy, which is the number of correctly labeled instances over the total number of instances:

$$\mathcal{Q}(D) \doteq \frac{1}{Z(D)} \sum_{d \in D} \sum_{i=1}^{M^{(d)}} \mathbf{1}(Y_i^{(d,\tau^{(d)})} = G_i^{(d)}), \tag{3}$$

where $Z(D)$ is the total number of instances in D; $M^{(d)}$ is the number of instances in document d; $Y^{(d,\tau^{(d)})}$ denotes labels for instances in d at the terminal time step $\tau^{(d)}$; $G_i^{(d)}$ denotes the ground truth label for the i-th instance in d; $\mathbf{1}$ denotes the indicator function.

We use the d superscript in $M^{(d)}$ because a set of documents D are involved in (3), and each document in D contains different numbers of instances. The d superscript appears in $Y_i^{(d,\tau^{(d)})}$ and $G_i^{(d)}$ for the same reason.

Annotation quality $\mathcal{Q}(D)$ is defined as accuracy in (3) but not F1, because the increase of accuracy always leads to the increase of F1 in our problem setting, and accuracy is simpler than F1. F1 is the harmonic mean of precision and recall [24]. Accuracy increases with the increase of true positive or true negative instances. When true positive instances increase, both precision and recall increase. When true negative instances increase, false positive instances must decrease, so precision increases and recall remains the same. When accuracy reaches 1, precision reaches 1, recall reaches an upper bound determined by the base taggers which produce the instances, and F1 reaches an upper bound determined by recall since precision is 1.

Definition 14 (Human Effort): Given a set of documents D, human effort is defined as the number of manually annotated instances over the total number of instances:

$$\mathcal{E}(D) \doteq \frac{1}{Z(D)} \sum_{d \in D} \sum_{i=1}^{M^{(d)}} \mathcal{M}(X_i^{(d)}), \tag{4}$$

where $\mathcal{M}(X_i^{(d)})$ returns 1 if $X_i^{(d)}$ is a manually annotated instance, or 0 otherwise.

The objective function can be rewritten as (5), according to (2), (3) and (4).

$$\mathcal{J}(D) \doteq \frac{1}{Z(D)} \sum_{d \in D} \sum_{i=1}^{M^{(d)}} \left(\mathbf{1}(Y_i^{(d,\tau^{(d)})} = G_i^{(d)}) - \lambda \mathcal{M}(X_i^{(d)})\right). \tag{5}$$

Before casting learning document-level label propagation and instance selection as an optimization problem, we formally define them in Definition 15 and Definition 16.

Definition 15 (Instance Selection): Instance selection is a function

$$\mathcal{S} : \mathcal{X}^M \times \mathcal{Y}^M \times [0,1]^M \times \mathbb{N}_{1:M} \mapsto \{0,1\},$$

$\mathcal{S}(X, Y^{(t)}, C^{(t)}, i^{(t)})$ determines whether to select the current instance $X_{i^{(t)}}$ for manual annotation, where $\mathcal{X}$, $\mathcal{Y}$ and $[0,1]$

are instance space, label space and confidence space, respectively; M is the number of instances in the given document; $\mathbb{N}_{1:M}$ is the range of instance indexes in X. $\mathcal{S}$ returns 1 if yes, or 0 otherwise.

Definition 16 (Document-Level Label Propagation): Document-level label propagation is a function

$$\mathcal{P} : \mathcal{X}^M \times \mathcal{Y}^M \times [0, 1]^M \times \mathbb{N}_{1:M} \times \mathbb{N}_{1:M} \mapsto \{0, 1\},$$

where $\mathcal{P}(X, Y^{(t)}, C^{(t)}, i^{(t)}, j^{(t)})$ determines whether to update $Y^{(t)}_{i^{(t)}}$ and $C^{(t)}_{i^{(t)}}$ using $Y^{(t)}_{j^{(t)}}$ and $C^{(t)}_{j^{(t)}}$; $i^{(t)}$ and $j^{(t)}$ are the indexes of the current instance and the similar-form instance at time step t, respectively. $\mathcal{P}$ returns 1 if yes, and 0 otherwise.

In our approach, human effort is determined directly by the instance selection function $\mathcal{S}$ (Definition 15). Annotation quality is determined directly by both the document-level label propagation function $\mathcal{P}$ and $\mathcal{S}$. In fact, $\mathcal{P}$ and $\mathcal{S}$ influence each other in the iterative process of interactive annotation. So, human effort and annotation quality are jointly determined by $\mathcal{P}$ and $\mathcal{S}$.

Given a set of documents D, the problem of learning document-level label propagation and instance selection to maximize the objective function $\mathcal{J}$ in (2) can be solved by determining θ^*, such that:

$$\theta^* = \arg\max_\theta \mathcal{J}(D), \tag{6}$$

where $\theta = \langle \theta_\mathcal{P}, \theta_\mathcal{S} \rangle$ represents learnable parameters of the document-level label propagation function $\mathcal{P}$ and the instance selection function $\mathcal{S}$.

D. REINFORCEMENT LEARNING MODEL

The reinforcement learning model consists of an environment providing states and rewards, and a DQN agent learning to choose actions to maximize accumulated reward it receives [36]. Interactive annotation of a whole document is framed as an MDP. Based on the MDP, learning document-level label propagation and instance selection to maximize the objective function in (5) can be casted as a reinforcement learning problem.

The MDP can be represented as a tuple $\langle S, A, T, R, \gamma \rangle$, where S is the state space; A is the action space; T is the transition function; R is the reward function; γ is the discount rate, $0 \leq \gamma \leq 1$. This is an episodic MDP. In each episode, the agent handles one document d only. Each episode starts by processing the first instance in d, and is terminated when all instances in d have been processed.

In the above MDP, actions determine document-level label propagation and instance selection. Rewards reflect the objective function $\mathcal{J}$ in (5). The parameters θ in (6) can be optimized by training the DQN.

The MDP is detailed in Subsubsections IV-D1 to IV-D4, and the DQN is described in Subsubsection IV-D5, followed by the training algorithm in IV-D6.

1) STATES

The state of the environment at time step t is a tuple

$$s^{(t)} \doteq \langle X, Y^{(t)}, C^{(t)}, i^{(t)}, j^{(t)} \rangle, \tag{7}$$

where X is the instances in the input document d; $Y^{(t)}$ and $C^{(t)}$ are labels and confidences at time step t, respectively; $i^{(t)}$ and $j^{(t)}$ are indexes of the current instance and the similar-form instance, respectively.

When an episode starts at time step $t = 0$, X, $Y^{(0)}$ and $C^{(0)}$ are produced in preprocess, $i^{(0)}$ is initialized to 1, and $j^{(0)}$ is chosen randomly. $Y^{(t)}$, $C^{(t)}$, $i^{(t)}$ and $j^{(t)}$ are updated in transitions, which will be described in Subsubsection IV-D3.

When the state $s^{(t)}$ is sent to the DQN agent, it is mapped to a feature vector, which consists of label features, confidence features, local context features, form features and overlapping features.

- The label features include $Y^{(t)}_{i^{(t)}}$, $Y^{(t)}_{j^{(t)}}$, and match of them $\mathbf{1}(Y^{(t)}_{i^{(t)}} = Y^{(t)}_{j^{(t)}})$. Nominal values such as $Y^{(t)}_{i^{(t)}}$ and $Y^{(t)}_{j^{(t)}}$ are represented using one-hot encoding.
- The confidence features include $C^{(t)}_{i^{(t)}}$, $C^{(t)}_{j^{(t)}}$ and difference between them $C^{(t)}_{i^{(t)}} - C^{(t)}_{j^{(t)}}$.
- The local context features are postags of the tokens within and around $X_{i^{(t)}}$ and $X_{j^{(t)}}$, as described in Subsection IV-B.
- The form features represent whether $X_{j^{(t)}}$ is a *identical-*, *super-*, or *sub-form instance* of $X_{i^{(t)}}$, according to Definition 8, Definition 9 and Definition 10.
- The overlapping features indicate whether there exists other instances overlapping with $X_{i^{(t)}}$ and $X_{j^{(t)}}$, according to Definition 12.

2) ACTIONS

Based on the current state $s^{(t)}$, the DQN agent chooses the current action $a^{(t)}$ consisting of two sub-actions

$$a^{(t)} = \langle a^{(t)}_{\mathrm{I}}, a^{(t)}_{\mathrm{II}} \rangle,$$

where $a^{(t)}_{\mathrm{I}}$ is the propagation sub-action; $a^{(t)}_{\mathrm{II}}$ is the selection sub-action.

The propagation sub-action $a^{(t)}_{\mathrm{I}}$ corresponds to the document-level label propagation, $a^{(t)}_{\mathrm{I}} \in \{0, 1\}$. When $a^{(t)}_{\mathrm{I}} = 1$, the environment propagates the label and the confidence of the similar-form instance $X_{j^{(t)}}$ to the current instance $X_{i^{(t)}}$:

$$Y^{(t+1)}_{i^{(t)}} \leftarrow Y^{(t)}_{j^{(t)}}, C^{(t+1)}_{i^{(t)}} \leftarrow C^{(t)}_{j^{(t)}}.$$

Otherwise no propagation is performed.

The selection sub-action $a^{(t)}_{\mathrm{II}}$ determines not only instance selection, but also whether to query another similar-form instance, or to start to process next instance. Possible values of $a^{(t)}_{\mathrm{II}}$ are human, identical, sub, super, and next. In the case of $a^{(t)}_{\mathrm{II}} =$ human, $X_{i^{(t)}}$ will be manually annotated by humans. The cases identical, super, and sub indicate that the environment is requested to query a identical-, sub- and super-form instance of $X_{i^{(t)}}$, respectively. In the last case of next, the environment is requested to switch to process next instance $X_{i^{(t)}+1}$ if there is any unprocessed instance in d.

3) TRANSITIONS

After the agent has chosen the current action $a^{(t)}$ based on the current state $s^{(t)}$, the environment responds with a transition to the next state $s^{(t+1)}$ using the transition function:

$$s^{(t+1)} \leftarrow T(s^{(t)}, a^{(t)}).$$

Calculation of the transition function is presented in Algorithm 1. It takes $s^{(t)}$ and $a^{(t)}$ as input, and outputs $s^{(t+1)}$.

Algorithm 1 Calculation of the Transition Function $T(s^{(t)}, a^{(t)})$

```
Input: current state $s^{(t)} = \langle X, Y^{(t)}, C^{(t)}, i^{(t)}, j^{(t)} \rangle$,
       current action $a^{(t)} = \langle a_{\mathrm{I}}^{(t)}, a_{\mathrm{II}}^{(t)} \rangle$
Output: next state $s^{(t+1)}$
1  $Y^{(t+1)} \leftarrow Y^{(t)}$, $C^{(t+1)} \leftarrow C^{(t)}$;
2  if $a_I^{(t)} = 1$ then
3      $Y_{i^{(t)}}^{(t+1)} \leftarrow Y_{j^{(t)}}^{(t)}$, $C_{i^{(t)}}^{(t+1)} \leftarrow C_{j^{(t)}}^{(t)}$;
4  $i^{(t+1)} \leftarrow i^{(t)}$, $j^{(t+1)} \leftarrow j^{(t)}$;
5  if $a_{II}^{(t)}$ = human then
6      $Y_{i^{(t+1)}}^{(t+1)} \leftarrow$ manual label of $X_{i^{(t+1)}}$, $C_{i^{(t+1)}}^{(t+1)} \leftarrow 1.0$;
7      $i^{(t+1)} \leftarrow i^{(t)} + 1$;
8      $j^{(t+1)} \leftarrow$ index of a randomly chosen similar-form
       instance of $X_{i^{(t+1)}}$;
9  else if $a_{II}^{(t)} \in$ {identical, sub, super} then
10     $j^{(t+1)} \leftarrow$ index of a similar-form instance chosen
       according to $a_{\mathrm{II}}^{(t)}$;
11 else if $a_{II}^{(t)}$ = next then
12     $i^{(t+1)} \leftarrow i^{(t)} + 1$;
13     $j^{(t+1)} \leftarrow$ index of a randomly chosen similar-form
       instance of $X_{i^{(t+1)}}$;
14 $s^{(t+1)} \leftarrow \langle X, Y^{(t+1)}, C^{(t+1)}, i^{(t+1)}, j^{(t+1)} \rangle$;
15 return $s^{(t+1)}$;
```

First, $Y^{(t+1)}$ and $C^{(t+1)}$ are initialized using $Y^{(t)}$ and $C^{(t)}$, separately (line 1). If $a_{\mathrm{I}}^{(t)} = 1$, perform document-level label propagation (line 2 and line 3).

Then, $i^{(t+1)}$ and $j^{(t+1)}$ are initialized using $i^{(t)}$ and $j^{(t)}$, separately (line 3). If the value of $a_{\mathrm{II}}^{(t)}$ is human, which means $X_{i^{(t+1)}}$ is selected for manual annotation, update $Y_{i^{(t+1)}}^{(t+1)}$ using the manual label of $X_{i^{(t+1)}}$, set $C_{i^{(t+1)}}^{(t+1)}$ to 1.0, and switch to next instance (line 5 to line 8). If the value of $a_{\mathrm{II}}^{(t+1)}$ is identical, sub or super, query an identical-, sub- or super-form instance, and update $j^{(t+1)}$ as the index of the chosen instance (line 9 and line 10). Otherwise, switch to next instance (line 11 to line 13).

Finally, construct $s^{(t+1)}$ and output it (line 14 and line 15).

4) REWARDS

In reinforcement learning, the agent's goal is to maximize the total reward it receives over the long run. To maximize the objective function $\mathcal{J}$ in (5) which is a weighted sum of human effort and annotation quality, we define the reward function as

$$R_\lambda(s^{(t)}, a^{(t)}, s^{(t+1)}) \doteq \Delta_{\mathcal{Q}}(t, t+1) - \lambda \Delta_{\mathcal{E}}(t, t+1), \quad (8)$$

where $\Delta_{\mathcal{Q}}(t, t+1)$ and $\Delta_{\mathcal{E}}(t, t+1)$ represent the changes of annotation quality and human effort from time step t to $t+1$, respectively; λ is the trade-off parameter.

The change of annotation quality $\Delta_{\mathcal{Q}}(t, t+1)$ in (8) is defined as the change of correctness of the current instance $X_{i^{(t)}}$'s label:

$$\Delta_{\mathcal{Q}}(t, t+1) \doteq \mathbf{1}(Y_{i^{(t)}}^{(t+1)} = G_{i^{(t)}}) - \mathbf{1}(Y_{i^{(t)}}^{(t)} = G_{i^{(t)}}), \quad (9)$$

where $Y_{i^{(t)}}^{(t+1)}$ and $Y_{i^{(t)}}^{(t)}$ are labels of $X_{i^{(t)}}$ at time step $t+1$ and t, respectively; $G_{i^{(t)}}$ is the ground truth label of $X_{i^{(t)}}$.

The increase of human effort $\Delta_{\mathcal{E}}(t, t+1)$ is determined by whether the current instance is selected for manual annotation at time step t:

$$\Delta_{\mathcal{E}}(t, t+1) \doteq \mathbf{1}(a_{\mathrm{II}}^{(t)} = \text{human}). \quad (10)$$

The normalization factor $Z(D)$ in (5) is eliminated in (9) and (10), because $Z(D)$ is a constant for a given set of documents D in our approach.

In each episode, a given document d is processed. According to (8), (9) and (10), maximizing $\sum_{t=1}^{\tau} r^{(t)}$ is equivalent to maximize $\mathcal{J}(\{d\})$, where $r^{(t)}$ is the reward at time step t, $r^{(t)} \doteq R_\lambda(s^{(t-1)}, a^{(t-1)}, s^{(t)})$; τ is the terminal time step; $\sum_{t=1}^{\tau} r^{(t)}$ represents the accumulated reward in the episode; $\mathcal{J}(\{d\})$ represents the objective value on d. In reinforcement learning, the agent is trained to maximize the total reward it receives over the long run. By training the agent on D, $\mathcal{J}(D)$ in (5) can be optimized, and the parameters θ^* of document-level label propagation and instance selection in (6) can be solved.

5) DEEP Q-NETWORK

The deep Q-network [16] is used in the RL model to approximate the optimal action-value function, from which the optimal policy is derived. The DQN is trained by adjusting its parameters to reduce the mean-squared error in the Bellman equation. By introducing a target network, the stability of learning is improved. In addition, experience replay speeds up the learning process.

Reinforcement learning aims to learn an optimal policy which maximizes the expected accumulated reward [36]. Formally, a policy is a function $\pi : S \mapsto A$, which maps a state s to an action a. The accumulated reward after time step t is referred to as return, denoted by $G^{(t)}$:

$$G^{(t)} \doteq r^{(t+1)} + \gamma r^{(t+2)} + \gamma^2 r^{(t+3)} + \cdots + \gamma^{\tau-t-1} r^{(\tau)},$$

where $r^{(t+1)}$ is the reward at time step $t+1$; τ is the terminal time step; γ is the discount rate. The optimal action-value function Q^* is defined as the maximum expected return achievable by following any policy, after observing a state s and taking an action a:

$$Q^*(s,a) \doteq \max_{\pi} \mathbb{E}[G^{(t)} | s^{(t)} = s, a^{(t)} = a, \pi].$$

Once Q^* has been solved, the greedy policy under Q^* is an optimal policy:

$$\pi^*(s) = \arg\max_{a} Q^*(s,a).$$

The Q-learning algorithm [37] estimates Q^* by using the Bellman equation as an iterative update,

$$Q_{n+1}(s,a) \leftarrow Q_n(s,a) + \alpha\Big[r + \gamma \max_{a'} Q_n(s',a') - Q_n(s,a)\Big],$$

where $\alpha \in (0, 1]$ is a learning rate. The algorithm converges to the optimal action-value function, $Q_n \to Q^*$, as $n \to \infty$. This basic approach estimate the action-value function separately for each state-action pair, leading to lack of generalization. Our problem involves a continuous state space, so a function is needed to approximate Q^*.

Deep Q-learning approach [16] approximates Q^* using a deep Q-network with parameters θ, $Q(s, a; \theta) \approx Q^*(s, a)$. The DQN can be trained by minimizing a sequence of loss functions:

$$\mathcal{L}(\theta_n) = \mathbb{E}_{s,a,r}\Big[\big(r + \gamma \max_{a'} Q(s', a'; \theta_n^-) - Q(s, a; \theta_n)\big)^2\Big], \tag{11}$$

where θ_n are the parameters of the DQN at iteration n; θ_n^- are the parameters used to compute the target at iteration n. As the error decreases, the approximated function $Q(s, a; \theta)$ converges to $Q^*(s, a)$.

By introducing a target network separately from the online network, the stability of learning is improved. Parameters θ_n^- of the target network are periodically updated with θ_n of the online network. Thereby correlations between the target network and the online network are reduced.

Experience replay allows the DQN agent to learn from earlier memories, and can speed up learning and break undesirable temporal correlations. The replay memory is a buffer which stores transitions (s, a, r, s'). During the training process, updates are applied on a batch of samples drawn randomly from the buffer.

The deep Q-network consists of two linear layers followed by rectified linear units, along with two separate output layers. Each of the two linear layers has 20 hidden units. The first output layer has two units, and outputs state-action value for the propagation sub-action $a_{\mathrm{I}}^{(t)}$. The second output layer has five units, and outputs state-action value for the selection sub-action $a_{\mathrm{II}}^{(t)}$.

6) TRAINING ALGORITHM

Training algorithm of the deep Q-network for interactive NE annotation is presented in Algorithm 2. Input of the algorithm includes a set of training documents D, number of training epochs N, the parameter ε balancing exploration and exploitation, the trade-off parameter λ balancing human effort and annotation quality, and the discount rate γ which is used in the loss function defined in (11). The algorithm outputs the learned parameters θ.

Algorithm 2 Training Algorithm of the Deep Q-Network for Interactive Named Entity Annotation

```
Input: a set of documents $D$, number of training epochs
       $N$, exploration parameter $\varepsilon$, trade-off parameter
       $\lambda$, discount rate $\gamma$
Output: parameters of the deep Q-network $\theta$
1  Initialize $\theta$ randomly;
2  for epoch = 1 to $N$ do
3      for $d \in D$ do
4          Obtain $X$, $Y^{(0)}$, $C^{(0)}$ by preprocess;
5          $t \leftarrow 0$;
6          $i^{(t)} \leftarrow 1$;
7          $j^{(t)} \leftarrow$ index of a similar-form instance chosen
           randomly;
8          $s^{(t)} \leftarrow \langle X, Y^{(t)}, C^{(t)}, i^{(t)}, j^{(t)} \rangle$;
9          do
10             if $Random(\mathbb{R}_{[0,1]}) < \varepsilon$ then
11                 $a^{(t)} \leftarrow Random(A)$;
12             else
13                 $a^{(t)} \leftarrow \arg\max_{a \in A} Q(s^{(t)}, a; \theta)$;
14             $s^{(t+1)} \leftarrow T(s^{(t)}, a^{(t)})$;
15             $r^{(t+1)} \leftarrow R_\lambda(s^{(t)}, a^{(t)}, s^{(t+1)})$;
16             Update $\theta$ by performing gradient descent on
               $\mathcal{L}$ with respect to $\theta$;
17             $t \leftarrow t + 1$;
18         while $i^{(t)} < |X|$;
19 return $\theta$;
```

First, initialize θ randomly (line 1). Then, run N epochs (line 2 to 18). In each epoch, iterate over D. In each iteration, an episode is started and terminated for a document d in D (line 4 to 18). During each episode, θ is updated. Finally, after $N \times |D|$ episodes, output θ.

After starting an episode for a document d, preform preprocess and obtain X, $Y^{(0)}$, $C^{(0)}$ (line 4). Then set the time step t to 0, set the index of the current instance $i^{(t)}$ to 1, set $j^{(t)}$ to the index of an instance chosen randomly, and construct the initial state (line 5 to line 8).

Then loop until all instances in X have been processed (line 9 to line 18). At each time step t, with probability ε select a random action $a^{(t)}$, otherwise select $a^{(t)}$ with the maximal Q-value (line 10 to 13). Then transition to the next state $s^{(t+1)}$ by T using $s^{(t)}$ and $a^{(t)}$ (see Algorithm 1). Notice that $Y^{(t+1)}$, $C^{(t+1)}$, $i^{(t+1)}$ and $j^{(t+1)}$ are returned by T using $s^{(t+1)}$. Obtain a reward using R_λ defined in (8). At last, perform a gradient descent step on the loss function $\mathcal{L}$ in (11) with respect to θ (line 16), and increase the time step t (line 17).

The parameters of the training algorithm are set as the following:

- The number of training epochs N is empirically set to 50.
- The parameter ε in ε-greedy exploration is annealed from 1 to 0.1 over 500k transitions.
- The parameter λ is set to 0, 0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.64 and 1, separately, to control the trade-off between human effort and annotation quality in interactive annotation.
- The discount rate γ is set to 1, to establish the equivalence between maximization of the accumulated reward and maximization of the objective function in (5).

Early stopping is applied to prevent overfitting. We first train with a constant learning rate $\alpha = 2.5\times 10^{-5}$ on the training set and monitor the performance on the development set at each epoch. Then, at the epoch with the highest development performance, we start a learning rate annealing schedule: decrease α an order of magnitude, train for 5 epochs, and stop. The final performance reported are averaged over the last 5 epochs before stopping.

V. EXPERIMENTS

This section describes datasets used in our experiments, base taggers for preprocess, baseline approaches, and evaluation metrics.

A. DATASETS

Three publicly available datasets are used, which are AKSW-News, CoNLL'03 and Ontonotes 5.0. Table 2 gives detailed information of the datasets, including number of documents, average number of sentences per document, and textual genre. Each document in the datasets contains more than one sentences, so that we can evaluate whether labels can be propagated across sentences in interactive NE annotation.

TABLE 2. Detailed information of the datasets.

Dataset	Documents	Avg. sentences	Genre
AKSW-News	325	35.7	news
CoNLL'03	447	16.5	news
Ontonotes 5.0	719	32.5	news, transcript, web log, etc.

The AKSW-News dataset consists of 325 newspaper articles [11], most of which are reports from the aerospace domain. The CoNLL-2003 shared task [38] dataset (CoNLL'03) is a subset of Reuters 1996 news corpus, and is widely used in NER task. Since one of our base taggers (StanfordNER) is trained on the training part of CoNLL'03, we only use the testing part. The Ontonotes 5.0 [39] dataset contains 3,145 annotated documents, which come from a wide range of sources including newswire, bible, transcripts, magazines, and web blogs. 719 documents are sampled randomly from the dataset and are used in our experiments.

Each dataset is divided equally into four parts: 1) TRAIN-E for training the ensemble classifier combining multiple base taggers, 2) TRAIN-Q for training the Q-network, 3) DEV for tuning parameters, and 4) TEST for evaluation.

B. BASE TAGGERS

Seven existing NE taggers are used as base taggers for instance extraction in preprocess. Three classes are considered in the experiments, namely person, location and organization. F1s of the taggers on the datasets in our experiments are listed in Table 3. All the taggers perform significantly worse on Ontonotes 5.0 than on AKSW-News and CoNLL'03, due to that Ontonotes 5.0 is a mixed-genre corpus (see Table 2).

TABLE 3. F1s of the seven base taggers on the datasets in our experiments.

Tagger	AKSW News	CoNLL'03	Ontonotes 5.0	Avg.
StanfordNER	0.913	0.914	0.846	0.891
BiLSTM-CNN-CRF	0.806	0.881	0.753	0.813
IllinoisNET	0.728	0.856	0.643	0.742
GATE	0.770	0.638	0.588	0.665
OpenNLP	0.576	0.787	0.555	0.640
Balie	0.648	0.556	0.511	0.572
LinePipe	0.550	0.538	0.525	0.538
Avg.	0.713	0.739	0.631	0.694

The Stanford Named Entity Recognizer[3] (StanfordNER) is a CRF-based tagger [30]. According to its introduction[3], this implementation does not precisely correspond to the paper [30] published in 2005. New distributional features and a mixture of CoNLL, MUC-6, MUC-7 and ACE named entity corpora for training make it fairly robust across domains.

BiLSTM-CNN-CRF [40], [41] is a neural network architecture that benefits from both word- and character-level representations automatically, by using combination of bidirectional LSTM, CNN and CRF. It requires no feature engineering or data preprocessing, thus making it applicable to a wide range of sequence labeling tasks.

Base taggers also include Illinois Named Entity Tagger (Illinois) [42], General Architecture for Text Engineering (GATE) [28], Apache OpenNLP Name Finder (OpenNLP), Ottawa Baseline Information Extraction (Balie) [43] and LingPipe.[4] Although their performance is lower compared to StanfordNER and BiLSTM-CNN-CRF, we believe that more base taggers are beneficial because they produce more instances as candidate NEs and hence increase the potential performance of our approach.

C. BASELINE APPROACHES

Baseline approaches are CI [12], and two Clustering+CI approaches (TokenTrigram+CI and LabelTrigram+CI) [8]. Constrained inference is implemented based on off-the-shelf StanfordNER as Goldberg *et al.* [8] do in their experiments. All of them adopt rule-based instance selection.

[3]http://nlp.stanford.edu/software/CRF-NER.shtml (version 3.6.0).
[4]http://alias-i.com/lingpipe/ (version 4.1.0).

- The **CI** baseline approach is a pure constrained inference-based approach proposed by Kristjansson *et al.* [12]. Labels of the manual annotated tokens are utilized as constraints, and prediction of other tokens around the manual annotated tokens is influenced. CI simply selects the least confident tokens for manual annotation.
- The **TokenTrigram+CI** baseline approach [8] applies rule-based clustering for document-level label propagation before constrained inference and instance selection. In each iteration, token w_i and token w_j are gathered into a cluster if

$$(w_{i-1}, w_i, w_{i+1}) = (w_{j-1}, w_j, w_{j+1}),$$

where w_{i-1} and w_{i+1} are the tokens before and after w_i. Then, one token is selected randomly from the least confident cluster for manual annotation. Other tokens in the cluster are forced to share the same label with the selected token. In this way, manually annotated labels are propagated across sentences within a document.
- The **LabelTrigram+CI** baseline approach [8] adopts rule-based strategy for document-level label propagation too, like TokenTrigram+CI. w_i and w_j are clustered together if

$$(y_{i-1}, w_i, y_{i+1}) = (y_{j-1}, w_j, y_{j+1}),$$

where y_{i-1} and y_{i+1} are the predicted labels for w_{i-1} and w_{i+1}.

AL [13], [14] and k-best CI+AL [7] approaches are not included in our experiments because they require humans to check and correct labels for all tokens in a selected sentence. Besides, Zhang *et al.* [14] require humans to write regexes. Whereas our RL-DINEA and the above-mentioned baseline approaches require humans to assign one label to one NE mention or one token in each iteration.

D. EVALUATION METRICS

Metrics of human effort and annotation quality are used to measure the performance of RL-DINEA and the baseline approaches.

- **Precision**, **recall** and **F1** [24] are used to measure annotation quality for comparing RL-DINEA with the baseline approaches. F1 is the harmonic mean of precision and recall. Although annotation quality is defined in (3) for RL-DINEA as accuracy, which is the ratio of correctly labeled instances over total instances extracted by the base taggers, accuracy is not applicable to the baseline approaches, because they are not approaches that predict labels for instances extracted by the base taggers.
- **Human effort** is defined in (4) as the number of instances selected for manual annotation, normalized by the total number of instances extracted in the preprocess.

VI. RESULTS AND ANALYSIS

This section presents experimental results and analysis. Precisions, recalls and F1s under different amounts of human effort of RL-DINEA and the baseline approaches are compared in Subsection VI-A. Document-level label propagation strategies based on learning and rules are compared in Subsection VI-B. Trade-off between human effort and annotation quality using the parameter λ in RL-DINEA is described in Subsection VI-C. Contribution of F1 by propagation and selection to RL-DINEA is analyzed in Subsection VI-D. Ablation studies of the base taggers and the state features are presented in Subsection VI-E and Subsection VI-F, respectively. At last, convergence of the training algorithm is discussed in Subsection VI-G.

A. RESULTS

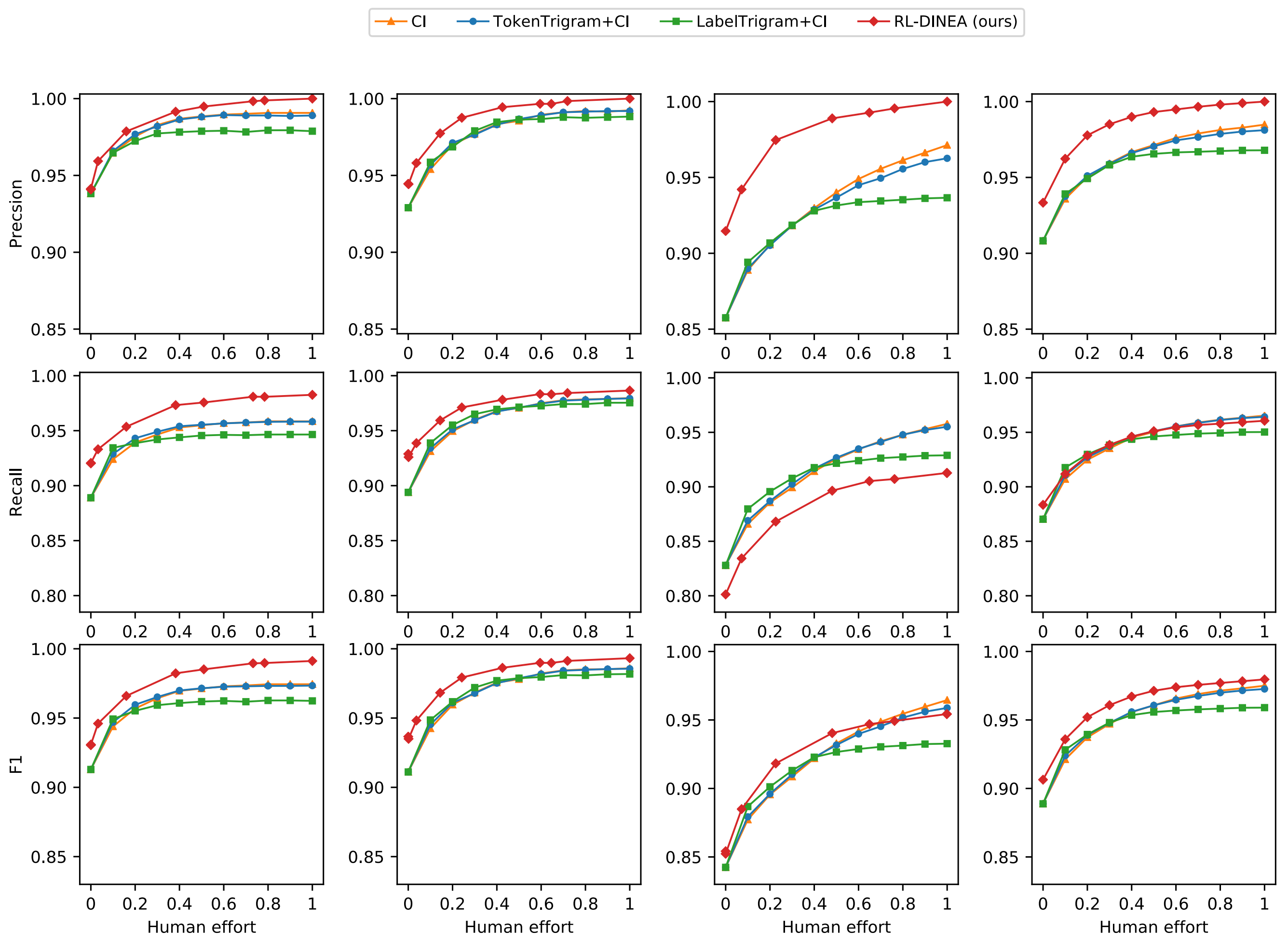
Averaged on the three datasets,[5] RL-DINEA outperforms all the baseline approaches both in terms of precision and F1 under the same amount of human effort (see the top and bottom parts of Fig. 3 (D)), and performs slightly worse in some cases in terms of recall (see the middle part of Fig. 3 (D)). The results will be described in detail below.

In terms of precision, RL-DINEA outperforms all the baseline approaches on all the datasets (see the upper part of Fig. 3). Initially, when no human effort is spent, RL-DINEA outperforms the baseline approaches due to ensemble of multiple base taggers and document-level label propagation. With the increase of human effort, RL-DINEA maintains the advantage. At last, when human effort reaches 1, precision reaches 1 because all instances have been selected for manual annotation.

In terms of recall, RL-DINEA performs best on AKSW-News and CoNLL'03, but is worst on Ontonotes 5.0 (see the middle part of Fig. 3). Ontonotes 5.0 is a mixed-genre corpus, and the base taggers have relatively poor performance on it (see Table 3). This results in a relatively low upper bound of recall that RL-DINEA can achieve on Ontonotes 5.0, which is 0.913, while the upper bounds of recall on AKSW-News and CoNLL'03 are 0.983 and 0.986, respectively. RL-DINEA does not select or predict named entities which have not been recognized by the base taggers.

Nevertheless, the precision of RL-DINEA on Ontonotes 5.0 is not affected by the relatively poor performance of the base taggers (see the upper part of Fig. 3 (C)). A large portion of the instances produced by the base taggers with poor performance on Ontonotes 5.0 should be classified as negative. In fact, the ensemble classifier trained on this kind of data may tend to achieve a high precision but a low recall, which are 0.917 and 0.777, respectively. Based on the ensemble classifier, RL-DINEA achieves much higher precisions than its recalls under different amounts of human effort.

[5]For each dataset, linear interpolation is used to estimate the precisions under the amount of human effort of 0.1, 0.2, ..., and 0.9. Then for each amount of human effort, the estimated precisions on the three datasets are averaged. The averaged recalls and F1s are obtained similarly.



(A) AKSW-News (B) CoNLL'03 (C) Ontonotes 5.0 (D) Average

FIGURE 3. Human effort/precision, recall and F1 curves of our approach and the baseline approaches.

In terms of F1, RL-DINEA performs the best on the three datasets in most cases (see the bottom part of Fig. 3). The only exception is that RL-DINEA fails to beat CI and TokenTrigram+CI on Ontonotes 5.0 when the amount of human effort is more than 0.6. This is due to the low recall of RL-DINEA on Ontonotes 5.0.

Averaged on the three datasets, in terms of human effort required to achieve the same F1, RL-DINEA significantly reduces human effort compared to the baseline approaches (see Table 4). Taking F1 of 0.95 as an example, which is the best F1 that LabelTrigram+CI can achieve in our experiments, RL-DINEA requires only human effort of 0.19, while the baseline approaches require at least human effort of 0.33.

The above results demonstrate the effectiveness of RL-DINEA. RL-DINEA which learns document-level label propagation and instance selection, outperforms the three baseline approaches, including CI which do not propagate any label at document level, and TokenTrigram+CI and LabelTrigram+CI which adopt rule-based document-level label propagation.

TABLE 4. Human effort required by our approach and the baseline approaches to achieve different annotation qualities (F1s).

Effort \ F1 / Approach	0.91	0.92	0.93	0.94	0.95	0.96	0.97
CI	0.07	0.10	0.15	0.23	0.33	0.49	0.74
TokenTrigram+CI	0.06	0.09	0.14	0.21	0.33	0.49	0.81
LabelTrigram+CI	0.05	0.08	0.12	0.21	0.34	N/A	N/A
RL-DINEA (ours)	**0.01**	**0.05**	**0.08**	**0.13**	**0.19**	**0.29**	**0.47**

B. COMPARISON BETWEEN LEARNING- AND RULE-BASED DOCUMENT-LEVEL LABEL PROPAGATION

Learning-based document-level label propagation in RL-DINEA outperforms rule-based propagation in baseline approaches (TokenTrigram+CI and LabelTrigram+CI) significantly, in terms of contribution to the improvement of annotation quality under the same amount of human effort (see Fig. 4). Our learning-based propagation consistently improves annotation quality under different amount of human effort on all datasets, while rule-based propagation may

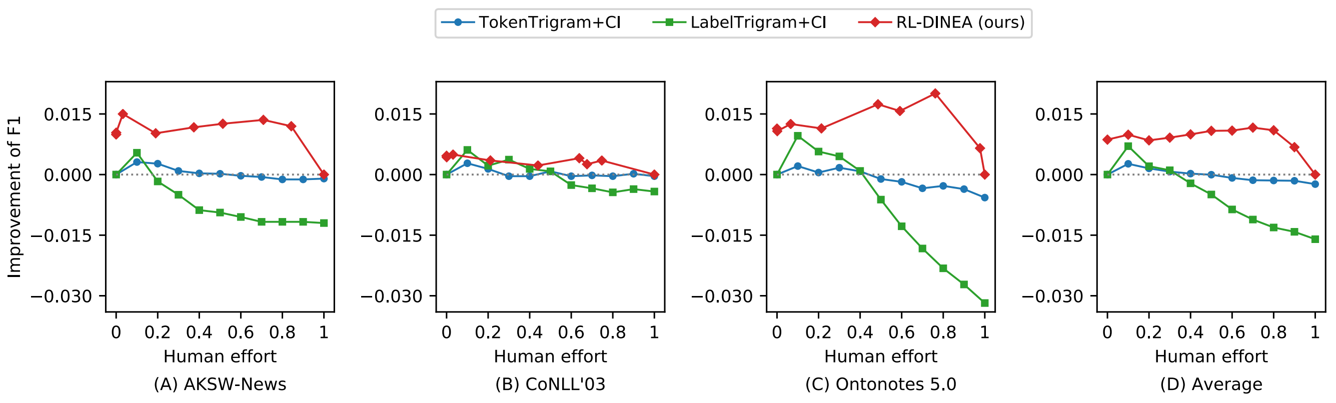


FIGURE 4. **Improvement of F1 contributed by learning- and rule-based document-level label propagation.**

deteriorate annotation quality when human effort is increased to a certain degree.

When the amount of human effort is zero, RL-DINEA improves annotation quality by propagation, while TokenTrigram+CI and LabelTrigram+CI do not. This is because RL-DINEA can propagate a label from any instance, regardless of whether it is manually annotated or not, while TokenTrigram+CI and LabelTrigram+CI only make propagation from manually annotated instances.

After some instances have been selected for manual annotation, LabelTrigram+CI achieves more improvement of annotation quality by propagation than RL-DINEA on CoNLL'03 and Ontonotes 5.0 (see Fig. 4 (B) and (C)). LabelTrigram+CI also outperforms TokenTrigram+CI, because LabelTrigram+CI adopts a more aggressive propagation strategy, which propagates a label to more instances.

As human effort increases, RL-DINEA consistently improves annotation quality by propagation, while TokenTrigram+CI and LabelTrigram+CI which adopt rule-based propagation start to deteriorate annotation quality, especially the latter with more aggressive propagation strategy. TokenTrigram+CI and LabelTrigram+CI may introduce some errors, which means that some tokens are assigned with wrong labels by propagation [8]. This kind of errors will further cause more errors, because the wrong labels are used as constraints in the inference. In fact, deterioration of annotation quality by the rule-based propagation in TokenTrigram+CI and LabelTrigram+CI is also shown in the bottom part of Fig. 3 (D), in which CI without any document-level label propagation outperforms TokenTrigram+CI and LabelTrigram+CI when human effort is greater than 0.5. From the results in Fig. 4, RL-DINEA suffers much less from errors introduced by propagation, compared to TokenTrigram+CI and LabelTrigram+CI.

When human effort reaches 1, RL-DINEA makes no improvement of annotation quality by propagation, while TokenTrigram+CI and LabelTrigram+CI continue to deteriorate annotation quality. For RL-DINEA, all candidate NE mentions have been manually annotated, so there is no propagation, and hence no improvement of annotation quality. While for TokenTrigram+CI and LabelTrigram+CI, more tokens are selected for manual annotation, some more errors may be introduced by propagation and further constrained inference.

Our learning-based document-level propagation shows good adaptability to various situations, due to its data-driven nature. Propagation in RL-DINEA is modeled as actions in the MDP, and the reward function is designed to penalize incorrect propagation. Propagation is learned by training the deep Q-network with different value of the trade-off parameter λ on different datasets.

C. TRADE-OFF BETWEEN HUMAN EFFORT AND ANNOTATION QUALITY

Trade-off between human effort and annotation quality in RL-DINEA is controlled using the parameter λ. Lower value of λ leads to more human effort and higher annotation quality; and vice versa. The results with different values of λ on AKSW-News are presented in the 3D space in Fig. 5. Each round dot represents a tuple of human effort, annotation quality and objective value evaluated on the TEST set after each training epoch. The early results during the training process appear in green, and the late results appear in red. To make it clear, projections of the results onto the plane of human effort and annotation quality are also presented in Fig. 5.

When $\lambda = 0$, RL-DINEA immediately achieves the annotation quality (accuracy) of 1.0 under the human effort of 1.0. This means RL-DINEA learns to select all instances for manual annotation, and reduces interactive annotation to fully manual annotation where no document-level label propagation is needed.

In the case of $\lambda = 0.04$, RL-DINEA starts from fully manual annotation with the annotation quality of 1.0, the human effort of 1.0, and the objective value of 0.9600. Aiming to maximize the objective value, RL-DINEA gradually learns to select less instances for manual annotation, and to propagate labels among instances. Finally, the annotation quality decreases to 0.9944, the human effort decreases to 0.5220, and the objective value increases to 0.9735.

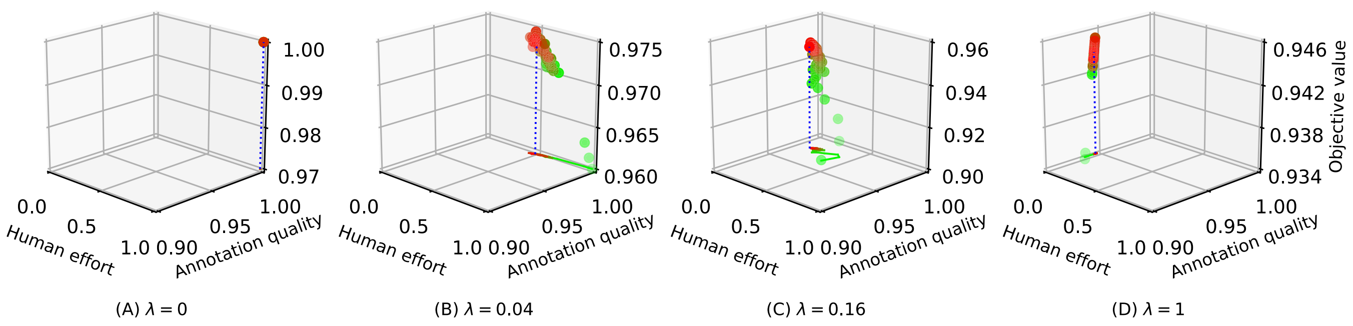


FIGURE 5. Results with different values of the trade-off parameter λ in the 3D space of human effort, annotation quality and objective value during the training process.

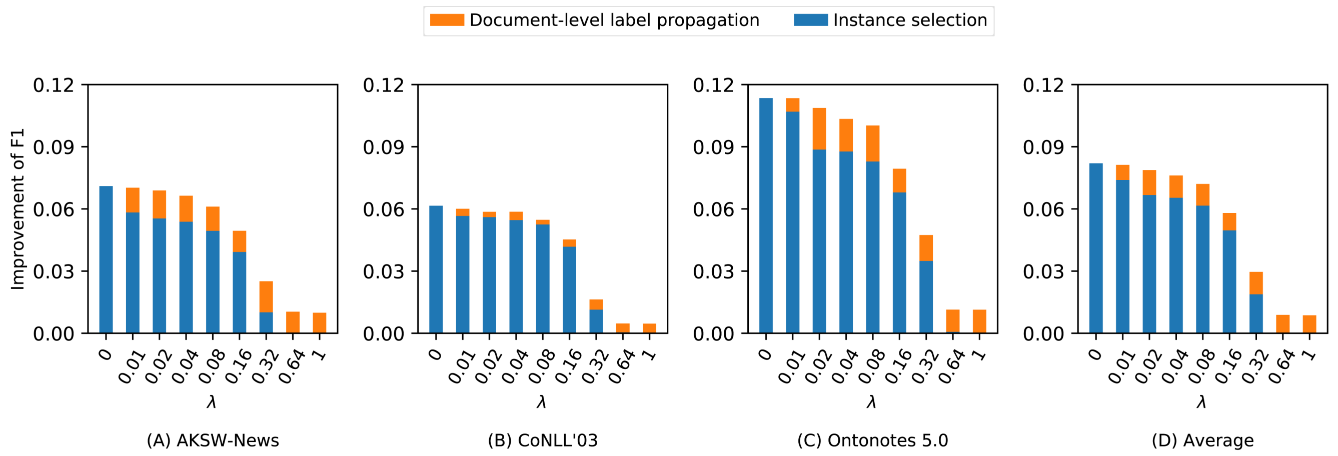


FIGURE 6. Improvement of F1 contributed by document-level label propagation and instance selection in our approach.

In the case of λ = 0.16, RL-DINEA does not start from fully manual annotation, due to higher value of λ. It finally achieves lower annotation quality of 0.9743 under lower human effort of 0.1523. The objective value increases from 0.9003 to 0.9500.

When the value of λ increases to 1, RL-DINEA reduces interactive annotation to pure prediction without any human effort. During the process, the annotation quality increases from 0.9338 to 0.9439, and the objective value is always equal to the annotation quality since the human effort is always 0. In this case, improvement of the annotation quality is all due to document-level label propagation.

D. ANALYSIS OF DOCUMENT-LEVEL LABEL PROPAGATION AND INSTANCE SELECTION

Analysis of the results shows that both document-level label propagation and instance selection contribute to improving F1 of RL-DINEA, compared to the ensemble classifier described in Subsection IV-B (see Fig. 6). When λ is 0, all improvement of F1 is contributed by instance selection, because all instances of candidate NE mentions have been manually annotated and there is no document-level label propagation. In cases of λ from 0.01 to 0.64, both document-level label propagation and instance selection contribute to the improvement of F1. When λ increases to 1, no instance is selected for manual annotation, and hence no improvement of annotation quality is contributed by instance selection.

Improvement of F1 contributed by document-level label propagation and instance selection is related to performance of the ensemble classifier. Lower performance of the ensemble classifier means more space of improvement of F1 for both document-level label propagation and instance selection. F1s of the ensemble classifier on AKSW-News, CoNLL'03 and Ontonotes 5.0 are 0.920, 0.932 and 0.840, respectively. Correspondingly, document-level label propagation and instance selection improve F1 more significantly on Ontonotes 5.0 than on AKSW-News and CoNLL'03 (see Fig. 6).

Besides, improvement of F1 contributed by document-level label propagation is also related to number of sentences per document. A document containing more sentences often contains more similar instances, so there are often more document-level label propagations. This explains why document-level label propagation improves F1 more significantly on AKSW-News than on CoNLL'03. The average

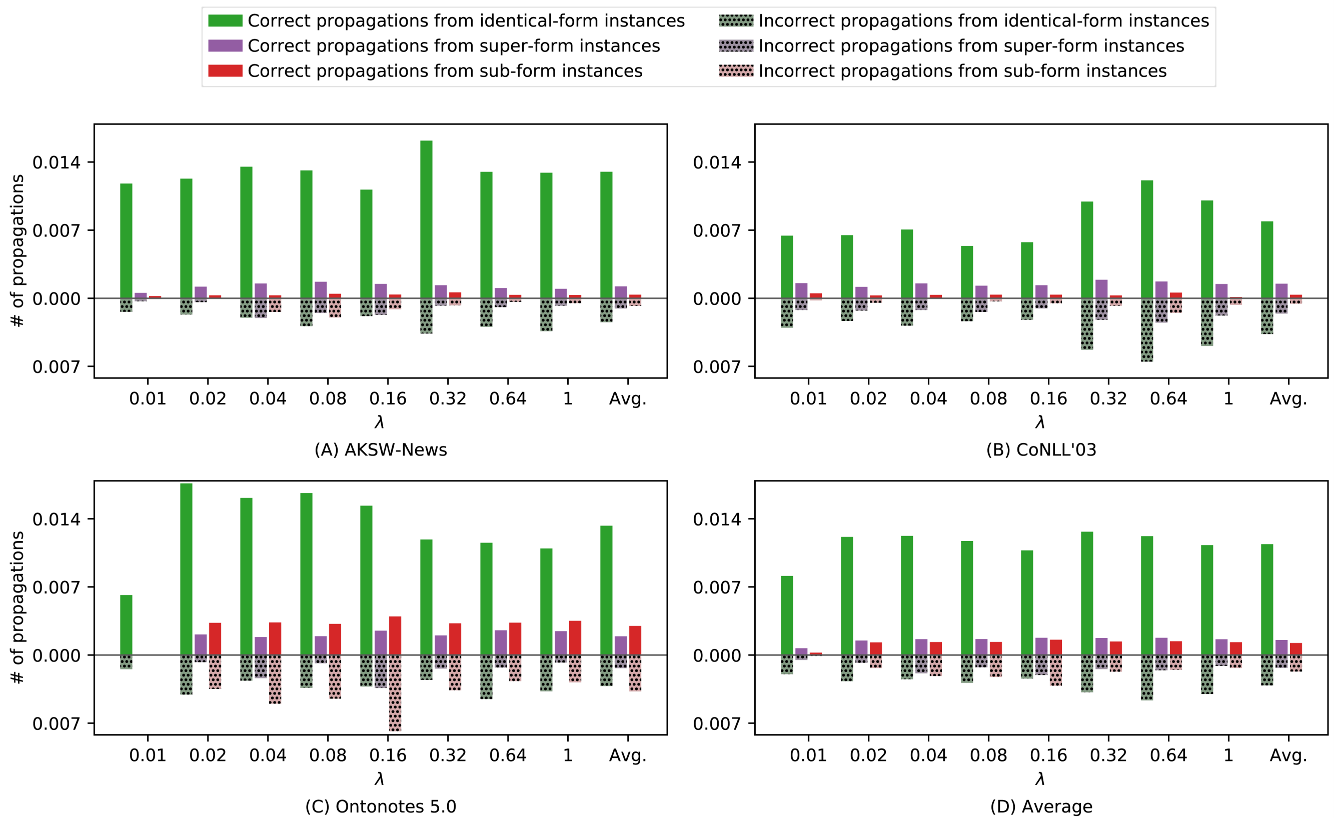


FIGURE 7. **Numbers of correct and incorrect propagations from identical-, super- and sub-form instances.**

number of sentences per document in AKSW-News is more than twice that in CoNLL'03 (see Table 2), although the F1s on these two dataset are not significantly different.

The numbers of correct and incorrect propagations from identical-, super- and sub-form instances are shown in see Fig. 7. For each dataset, numbers are normalized by total number of instances in the dataset. The case of $\lambda = 0$ is omitted because there is no propagation in this case.

RL-DINEA benefits much more from propagations from identical-form instances than super- and sub-form instances. Averaged on all datasets and different values of λ, the proportions of propagations from identical, super- and sub-form instances over the total propagations are 71.4%, 14.1% and 14.5%, respectively. The proportions of correct propagations from identical, super- and sub-form instances over the total ones are 78.5%, 53.8% and 42.2%, respectively. Propagations from sub-form instances do not always have positive effect because the base taggers and the ensemble classifier would usually be much more certain about labels assigned to super-form instances, since they are longer and have more contextual information [10].

E. ANALYSIS OF BASE TAGGERS

A series of experiments are conducted to better understand how the base taggers contribute to precision, recall and F1 of our approach. The base taggers are added one by one to the experiments according to their F1s in ascending order (see Table 3). In the first experiment, only LingPipe and Balie are used. In each subsequent experiment, one more base tagger is added.

The precision improves significantly as the number of base taggers integrated increases when no or little amount of human effort is spent (see the upper part of Fig. 8). As the amount of human effort increases, the gaps gradually narrow. The experiments which use less base taggers achieve the precision of 1 under less amount of human effort, because there are less instances in these experiments (see (1)).

The recall always improves as the number of base taggers integrated increases (see the middle part of Fig. 8). This is intuitive, since more base taggers extract more instances of candidate NE mentions from documents (see (1)).

The F1 always improves as the number of base taggers integrated increases (see the bottom part of Fig. 8). This means that RL-DINEA can benefit from the development of NER techniques, since any NE tagger can be used as a base tagger in RL-DINEA.

F. ANALYSIS OF STATE FEATURES

Two ablation studies are conducted to show the contribution of each kind of state features to the overall performance of our approach. In the first study, each kind of features is used

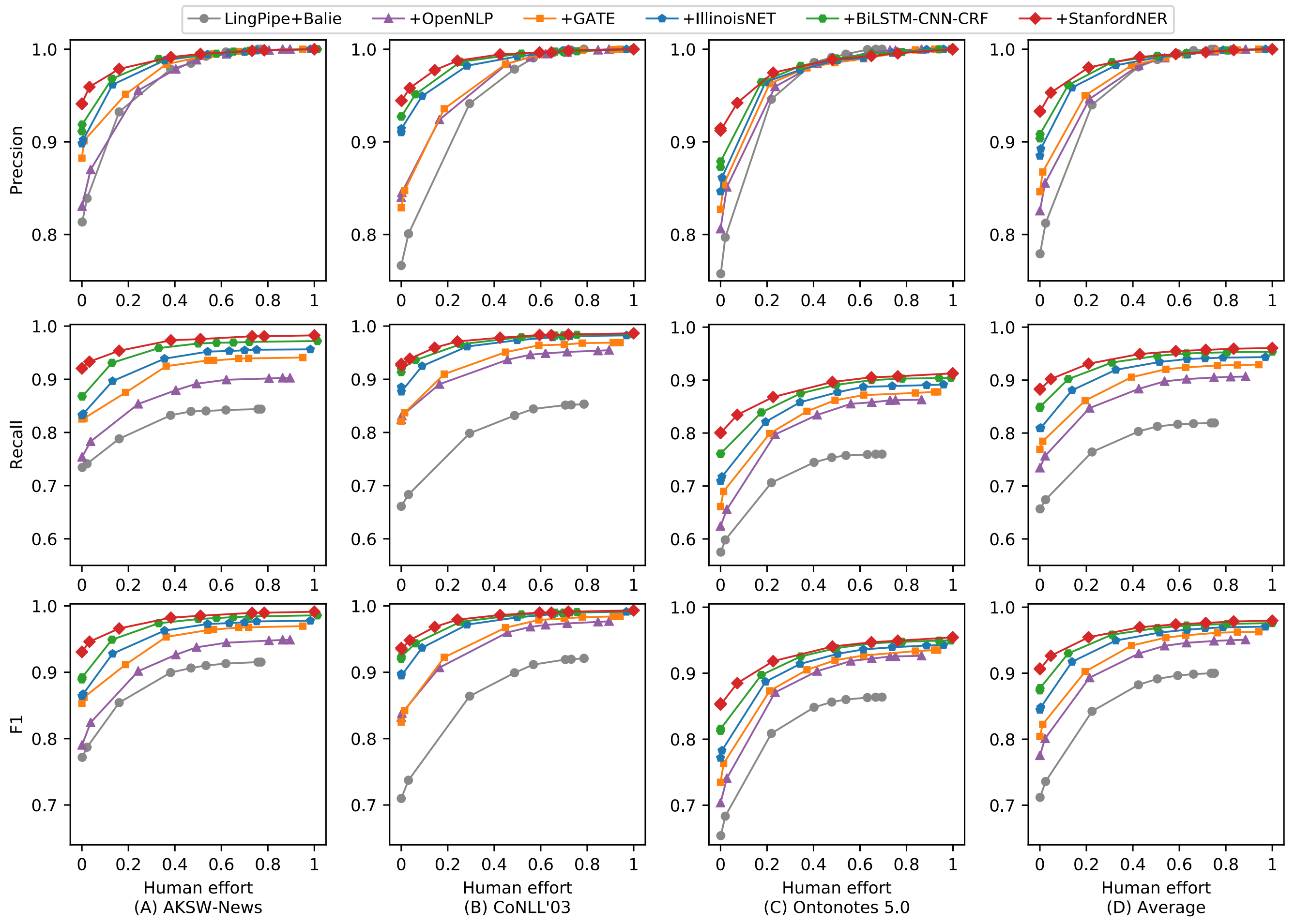


FIGURE 8. Results of ablation experiments of base taggers.

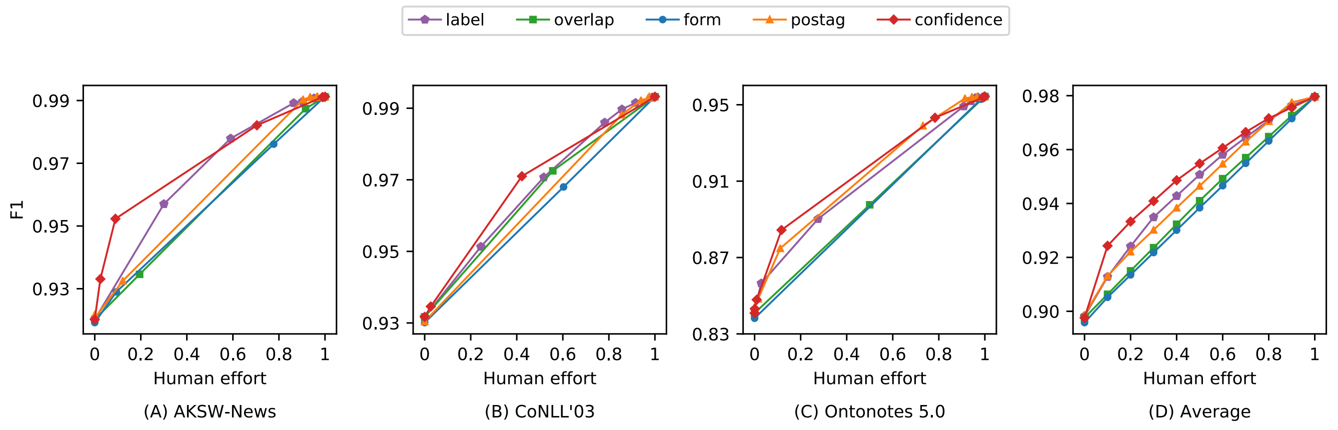


FIGURE 9. Results of ablation experiments using state features individually.

individually to train the DQN agent. In the second study, features are aggregated to train the DQN agent.

Among the five kinds of state features described in Subsection IV-D, confidence, label and postag features have a significant effect when being used individually, whereas overlapping and form features do not (see Fig. 9). Nevertheless, when these features are aggregated, all of them contribute to the performance (see Fig. 10).

G. ANALYSIS OF CONVERGENCE

Objective value curves in Fig. 11 show good convergence of the training algorithm for RL-DINEA. The objective value

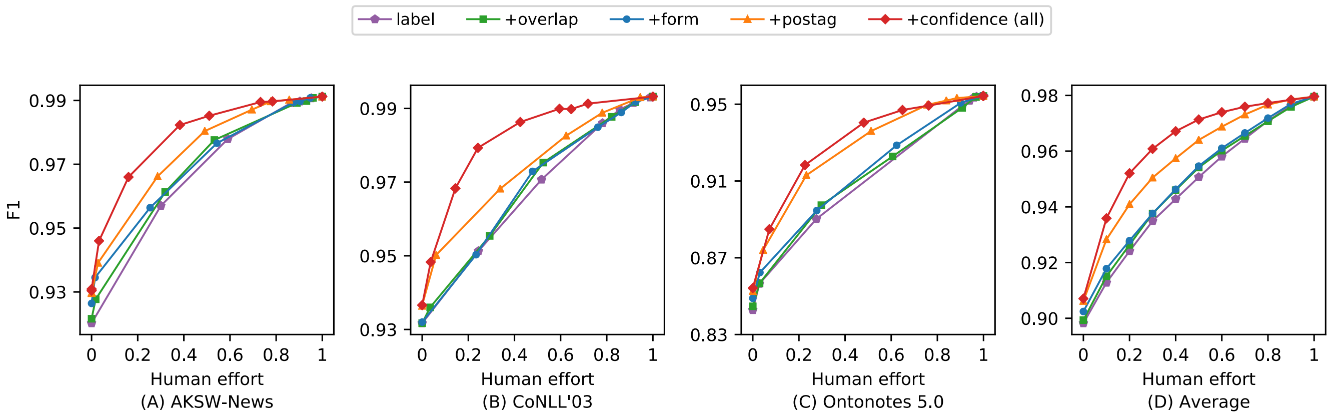


FIGURE 10. Results of ablation experiments using aggregated state features.

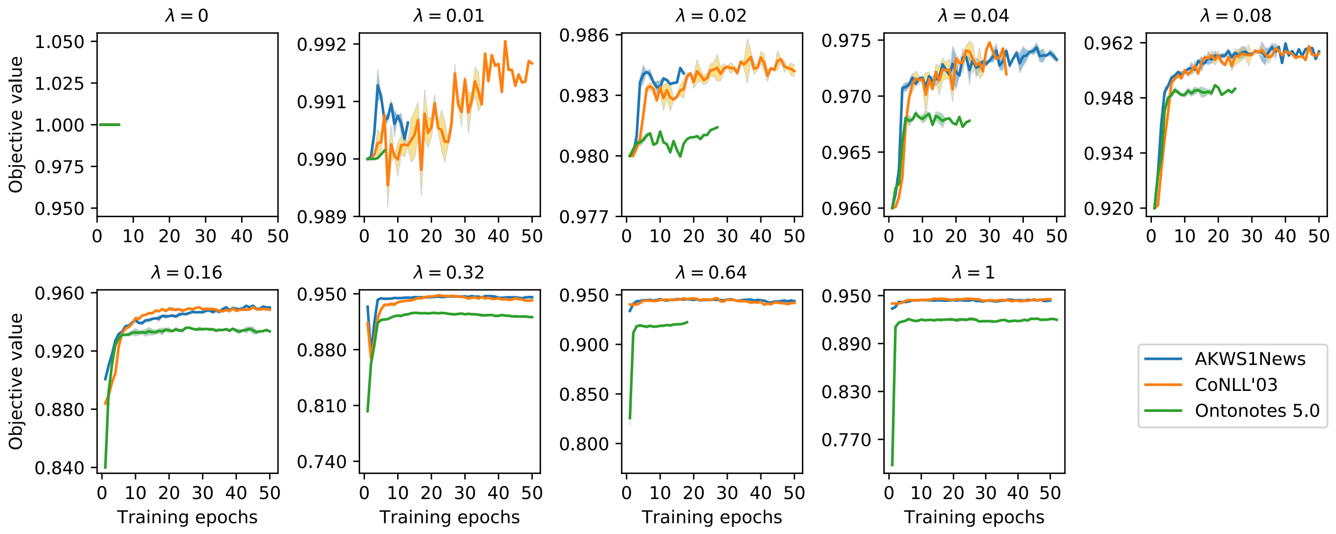


FIGURE 11. Objective value curves during the training process with different values of the trade-off parameter λ.

is a weighted sum of human effort and annotation quality, defined in (5), and is measured on the TEST set after each training epoch. With the values of λ ranging from 0 to 1, the objective values increase gradually with respect to the number of training epochs. This demonstrates that the training algorithm is insensitive to the choice of λ.

In some cases, early stopping is triggered by the results on the DEV set to prevent overfitting. The typical case is when the λ is set to 0. After the first training epoch, the training algorithm immediately learns that the optimal policy is to select all instances for manual annotation. Then the learning rate α is decreased by an order of magnitude, and the training algorithm stops after five training epochs.

VII. CONCLUSION

This paper proposes a reinforcement learning-based approach RL-DINEA to learn document-level label propagation and instance selection for interactive named entity annotation. The policy to select optimal actions is learned by training the deep Q-network. Experiments are conducted on AKSW-News, CoNLL'03, and Ontonotes 5.0 datasets. To achieve the same annotation quality (0.95 measured by F1 averaged on the three datasets), RL-DINEA reduces human effort by more than 42% compared to the baseline approaches, including one approach without document-level label propagation, and two approaches with rule-based document-level label propagation. RL-DINEA outperforms all the baseline approaches, in terms of annotation quality (averaged on the three datasets) under the same amount of human effort. In addition, any named entity tagger can be integrated as a base tagger into RL-DINEA, and the precision, recall and F1 of RL-DINEA improve significantly as strong base taggers are integrated. So, in future work, we will integrate strong base taggers based on BERT, XLNet, GPT, etc. We will explore cross-document label propagation, and extend our approach to other interactive annotation tasks, such as image segmentation and video segmentation.

REFERENCES

[1] J. Pustejovsky and A. Stubbs, ''The basics,'' in *Natural Language Annotation for Machine Learning: A Guide to Corpus-Building for Applications*, 1st ed. Sebastopol, CA, USA: O'Reilly Media, 2012, ch. 1, pp. 1–29.

[2] V. Yadav and S. Bethard, ''A survey on recent advances in named entity recognition from deep learning models,'' in *Proc. COLING*, 2018, pp. 2145–2158.

[3] J. Li, A. Sun, J. Han, and C. Li, ''A survey on deep learning for named entity recognition,'' *IEEE Trans. Knowl. Data Eng.*, early access, Mar. 17, 2020, doi: 10.1109/TKDE.2020.2981314.

[4] T. Al-Moslmi, M. G. Ocaña, A. L. Opdahl, and C. Veres, ''Named entity extraction for knowledge graphs: A literature overview,'' *IEEE Access*, vol. 8, pp. 32862–32881, 2020.

[5] H. Zhu, C. He, Y. Fang, and W. Xiao, ''Fine grained named entity recognition via seq2seq framework,'' *IEEE Access*, vol. 8, pp. 53953–53961, 2020.

[6] T. Stanislawek, A. Wróblewska, A. Wójcicka, D. Ziembicki, and P. Biecek, ''Named entity recognition—Is there a glass ceiling?'' in *Proc. CoNLL*, 2019, pp. 624–633.

[7] A. Culotta, T. Kristjansson, A. McCallum, and P. Viola, ''Corrective feedback and persistent learning for information extraction,'' *Artif. Intell.*, vol. 170, nos. 14–15, pp. 1101–1122, Oct. 2006.

[8] S. Goldberg, D. Z. Wang, and C. Grant, ''A probabilistically integrated system for crowd-assisted text labeling and extraction,'' *J. Data Inf. Qual.*, vol. 8, no. 2, pp. 1–23, Feb. 2017.

[9] D. Wang, G. Hu, Q. Liu, C. Lyu, and M. M. Islam, ''Region-based nonparametric model for interactive image segmentation,'' *IEEE Access*, vol. 7, pp. 111124–111134, 2019.

[10] V. Krishnan and C. D. Manning, ''An effective two-stage model for exploiting non-local dependencies in named entity recognition,'' in *Proc. COLING-ACL*, 2006, pp. 1121–1128.

[11] R. Speck and A.-C. N. Ngomo, ''Ensemble learning for named entity recognition,'' in *Proc. ISWC*, 2014, pp. 519–534.

[12] T. Kristjansson, A. Culotta, P. Viola, and A. McCallum, ''Interactive information extraction with constrained conditional random fields,'' in *Proc. AAAI*, 2004, pp. 412–418.

[13] M. Skeppstedt, C. Paradis, and A. Kerren, ''PAL, a tool for pre-annotation and active learning,'' *J. Lang. Technol. Comput. Linguistics*, vol. 31, no. 1, pp. 91–110, 2017.

[14] S. Zhang, L. He, E. Dragut, and S. Vucetic, ''How to invest my time: Lessons from human-in-the-loop entity extraction,'' in *Proc. SIGKDD*, Jul. 2019, pp. 2305–2313.

[15] X. Liao, W. Li, Q. Xu, X. Wang, B. Jin, X. Zhang, Y. Wang, and Y. Zhang, ''Iteratively-refined interactive 3D medical image segmentation with multi-agent reinforcement learning,'' in *Proc. CVPR*, Jun. 2020, pp. 9394–9402.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, ''Human-level control through deep reinforcement learning,'' *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[17] N. Seshadri and C.-E.-W. Sundberg, ''List Viterbi decoding algorithms with applications,'' *IEEE Trans. Commun.*, vol. 42, no. 234, pp. 313–323, Apr. 1994.

[18] L. Wang, S. Li, Q. Yan, and G. Zhou, ''Domain-specific named entity recognition with document-level optimization,'' *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 17, no. 4, pp. 1–15, Aug. 2018.

[19] T. Lu, Y. Gui, and Z. Gao, ''Document-level named entity recognition with *Q*-network,'' in *Proc. PRICAI*, 2019, pp. 164–178.

[20] T. Gui, J. Ye, Q. Zhang, Y. Zhou, Y. Gong, and X. Huang, ''Leveraging document-level label consistency for named entity recognition,'' in *Proc. IJCAI*, Jul. 2020, pp. 3976–3982.

[21] B. Zhang, S. Whitehead, L. Huang, and H. Ji, ''Global attention for name tagging,'' in *Proc. CoNLL*, 2018, pp. 86–96.

[22] Z. Dai, H. Fei, and P. Li, ''Coreference aware representation learning for neural named entity recognition,'' in *Proc. IJCAI*, Aug. 2019, pp. 4946–4953.

[23] A. Hu, Z. Dou, J. Y. Nie, and J. R. Wen, ''Leveraging multi-token entities in document-level named entity recognition,'' in *Proc. AAAI*, 2020, pp. 7961–7968.

[24] A. Goyal, V. Gupta, and M. Kumar, ''Recent named entity recognition and classification techniques: A systematic review,'' *Comput. Sci. Rev.*, vol. 29, pp. 21–43, Aug. 2018.

[25] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, ''Mastering the game of go with deep neural networks and tree search,'' *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[26] V. Varga and A. Lőrincz, ''Reducing human efforts in video segmentation annotation with reinforcement learning,'' *Neurocomputing*, vol. 405, no. 10, pp. 247–258, Sep. 2020.

[27] K. M. Lee, H. Myeong, and G. Song, ''SeedNet: Automatic seed generation with deep reinforcement learning for robust interactive segmentation,'' in *Proc. CVPR*, Jun. 2018, pp. 1760–1768.

[28] H. Cunningham, Y. Wilks, and R. J. Gaizauskas, ''GATE: A general architecture for text engineering,'' in *Proc. COLING*, vol. 2, 1996, pp. 1057–1060.

[29] P. McNamee and J. Mayfield, ''Entity extraction without language-specific resources,'' in *Proc. COLING*, 2002, pp. 1–4.

[30] J. R. Finkel, T. Grenager, and C. Manning, ''Incorporating non-local information into information extraction systems by Gibbs sampling,'' in *Proc. ACL*, 2005, pp. 363–370.

[31] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, ''BERT: Pre-training of deep bidirectional transformers for language understanding,'' in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.

[32] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, ''XLNet: Generalized autoregressive pretraining for language understanding,'' in *Proc. NeurIPS*, 2019, pp. 5753–5763.

[33] T. B. Brown *et al.*, ''Language models are few-shot learners,'' in *Proc. NeurIPS*, 2020, pp. 1877–1901.

[34] C.-C. Chang and C.-J. Lin, ''LIBSVM: A library for support vector machines,'' *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, 2011.

[35] S. Knerr, L. Personnaz, and G. Dreyfus, ''Single-layer learning revisited: A stepwise procedure for building and training a neural network,'' in *Neurocomputing*. Berlin, Germany: Springer, 1990, pp. 41–50.

[36] R. S. Sutton and A. G. Barto, ''Introduction,'' in *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018, ch. 1, sec. 1, pp. 1–4.

[37] C. J. C. H. Watkins and P. Dayan, ''*Q*-learning,'' *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[38] E. F. T. K. Sang and F. De Meulder, ''Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition,'' in *Proc. CoNLL*, vol. 4, 2003, pp. 142–147.

[39] S. Pradhan, A. Moschitti, N. Xue, O. Uryupina, and Y. Zhang, ''CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes,'' in *Proc. EMNLP CoNLL*, 2012, pp. 1–40.

[40] X. Ma and E. Hovy, ''End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF,'' in *Proc. ACL*, vol. 1, 2016, pp. 1064–1074.

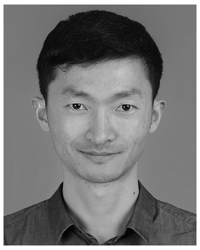
[41] N. Reimers and I. Gurevych, ''Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging,'' in *Proc. EMNLP*, 2017, pp. 338–348.

[42] L. Ratinov and D. Roth, ''Design challenges and misconceptions in named entity recognition,'' in *Proc. CoNLL*, 2009, pp. 147–155.

[43] D. Nadeau, ''Balie–baseline information extraction: Multilingual information extraction from text with machine learning and natural language techniques,'' School Inf. Technol. Eng., Univ. Ottawa, Ottawa, ON, Canada, Tech. Rep., 2005.

TINGMING LU received the B.S. degree in computer science and technology from Anhui University, Hefei, China, in 2004, and the M.S. degree in computer software and theory from Southeast University, Nanjing, China, in 2012, where he is currently pursuing the Ph.D. degree.

From 2004 to 2008, he was a Software Engineer with Neusoft, Nanjing, and with Huawei, Nanjing. His current research interests include reinforcement learning and natural language processing.



YAOCHENG GUI received the B.S. degree in computer science and technology from Hohai University, Nanjing, China, in 2009, and the M.S. degree in computer software and theory and the Ph.D. degree in software engineering from Southeast University, Nanjing, in 2012 and 2020, respectively.

He is currently a Lecturer with the School of Modern Posts and the Institute of Modern Posts, Nanjing University of Posts and Telecommunications, Nanjing. His current research interests include reinforcement learning and natural language processing.

ZHIQIANG GAO received the B.A. and M.D. degrees from Southeast University and the Ph.D. degree from the Department of Mechanical Engineering, Tsinghua University, in 1995.

He was with the National CAD Application and Training Network-Nanjing Center and the Department of Mechanical Engineering, Southeast University. In 2000, he held a postdoctoral position with the Department of Social Informatics, Kyoto University, Japan, where he was involved in digital city and artificial intelligence. In 2002, he was an Associate Professor with the Department of Computer Science and Engineering, Southeast University. In 2007, he joined the Institute of Web Science, to study ontology learning and natural language processing (NLP) and large scale reasoning. His current research interests include material science and manufacturing, numerical simulation of heat transfer and fluid flow, CAD, CG, visualization, VR, game theory, agent oriented programming, semantic web, knowledge capture, web mining, military simulation, philosophy, traveling, basketball, and Taijiquan. He was concentrated in human-like and conversational agent, multi-agent system, agent oriented programming, and machine learning. In 2003, a project was launched about interactive multi-agent system used for shooter training simulation, in which agents are hosted by augmented reality. He has to construct 3-D buildings and terrains, design and debug scenarios, and recognize behavior of shooters in the real world. Although this project is more complicated than designing ordinary CG games, it is much more interesting and challenging. He is also involved in 973 Project for semantic grid. Recently, he is responsible for two National Science Foundation of China projects, which are related to ontology learning and inductive logic programming. Speech recognition, text understanding, and image recognition are faced with the same puzzles, and natural language processing is one of the best test beds for various innovative AI ideas. Machine Learning may play a key role in these fields in the future. In addition, semantic annotation of 3-D environments for multi-agents to plan and act ''rationally'' is an important branch of semantic web. He firmly believes that a new era for search engine based on deep NLP, machine learning, and large scale reasoning is coming.