Towards Open-Domain Named Entity Recognition via Neural Correction Models

Mengdi Zhu\(^1\), Zheyu Deng\(^1\), Wenhan Xiong\(^2\), Mo Yu\(^3\), Ming Zhang\(^1\), William Yang Wang\(^{1,2}\)

\(^1\)Peking University
\(^2\)University of California, Santa Barbara
\(^3\)IBM Research
\{1600012990.dzy97,mzhang\}@pku.edu.cn, \{xwhan,william\}@cs.ucsb.edu, yum@us.ibm.com

Abstract

Named Entity Recognition (NER) plays an important role in a wide range of natural language processing tasks, such as relation extraction, question answering, etc. However, previous studies on NER are limited to a particular genre, using small manually-annotated or large but low-quality datasets. In this work, we propose a semi-supervised annotation framework to make full use of abstracts from Wikipedia and obtain a large and high-quality dataset called AnchorNER. We assume anchored strings in abstracts are named entities and annotate them with entity types mentioned in DBpedia. To improve the coverage, we design a neural correction model trained with a human-annotated NER dataset, DocRED, to correct the false-negative entity labels, and then train a BERT model with the corrected dataset. We evaluate our trained model on six NER datasets and our experimental results show that we have obtained state-of-the-art open-domain performances — on top of the strong baselines BERT-base and BERT-large, we achieve relative improvements of 4.66% and 3.07% respectively.

1 Introduction

Named entity recognition (NER) aims to identify named entities, such as person, location, organization, etc, from texts (Yadav and Bethard 2018). As a key component of many natural language processing (NLP) tasks such as data mining, summarization, and information extraction (Chen et al. 2004; Banko et al. 2007; Aramaki et al. 2009), NER has drawn much attention and many studies have been conducted in this field. In practice, NER could be applied to various genres of texts. Therefore, a NER model, especially a NER toolkit, should get high enough results in almost every case. However, open-domain is a great challenge for NER due to the limited annotated NER datasets.

Previously, there are two main ways to address the problem. The first is using external knowledge and pre-training. Some researchers exploit features (Ghaddar and Langlais 2018; Wu, Liu, and Cohn 2018) and some other studies utilize dictionaries or gazetteers (Nadeau, Turney, and Matwin 2006; Shang et al. 2018). However, with this method, external resources are usually domain-specific and building up such resources is time-consuming. Besides, dictionaries are finite and introduction of the dictionary also suffers from mismatching and missing matching problems. Moreover, some other studies utilize word and character embeddings. Pre-trained word embeddings are widely used in long short-term memory (LSTM) models (Huang, Xu, and Yu 2015) and convolutional neural networks (CNN) are used to implement character embeddings (Ma and Hovy 2016; Chiu and Nichols 2016). Some researchers also come up with contextual embeddings (Akbik, Bergmann, and Vollgraf 2019) to replace fixed embeddings and obtain promising results. Recently, with the development of pre-training language models (Devlin et al. 2019), they are also implemented in NER to significantly improve the performances. These strong models, nonetheless, only learn word representations without information regarding named entities. After the pre-training process, they still need to predict NER labels from word representations, which means that many improvements still can be made based on these models.

The second way is increasing the training data with semi-labeled or auto-labeled data. Nothman et al. (2013) implement a semi-supervised method to classify Wikipedia articles into named entity types and link anchor texts to the corresponding articles. Ghadjar and Langlais (2017) exploit out-links and out-links of out-links to annotate named entities in the target article and build an auto-labeled dataset called WiNER. However, these methods suffer from noise and many false labels are generated during the annotation.

In this work, our solution is along the direction of combining the best of both worlds. We propose a neural correction model and we use the correction model to correct the false-negative entity labels of a large-scale but low-quality NER dataset. As correction is an easier task comparing to labeling from scratch, a relatively smaller training data can already help and address the lack of NER dataset as well. We first make use of abstracts from Wikipedia and leverage anchored strings in Wikipedia and a well-built knowledge base, DBpedia, to annotate named entities in the abstracts. Then, in or-
order to correct the false-negative entity labels, we implement a neural correction model trained with a manually-annotated and high-precision dataset, DocRED. In order to train the correction model better, we utilize curriculum learning to assist the correction. We use the correction model to build a large-scale and high-quality dataset called AnchorNER. After that, we train BERT models with our corrected dataset. Experimental results demonstrate that we have obtained state-of-the-art open-domain performances. We will release our OpenNER toolkit for open-domain named entity recognition.\footnote{Our code and data are released at https://github.com/zmd971202/OpenNER.}

The main contributions of our work are three-fold:

- We build a large-scale and high-quality NER dataset called AnchorNER with Wikipedia and DBpedia.
- We implement a semi-supervised correction model with curriculum learning to correct the false-negative entity labels.
- BERT models trained with AnchorNER yield state-of-art results in the open-domain setting and we will release our model as a NER toolkit called OpenNER.

### 2 Related Work

**Statistical Models for Named Entity Recognition** : Previously, many prior research studies have been conducted in NER. Before deep learning models are widely used in NER, there are many common machine learning systems, such as Hidden Markov Models (HMM), Support Vector Machines (SVM), Conditional Random Fields (CRF) and decision trees, proposed in NER. Zhou and Su (2002) propose a Hidden Markov Model (HMM) and an HMM-based chunk tagger based on four types of internal and external features. Li, Bontcheva, and Cunningham (2004) implement an SVM model with uneven margins and experiment with different window sizes to obtain different combinations of feature vectors. Krishnan and Manning (2006) utilize two CRF models in which one is used to make predictions based on local features and another CRF is trained using both the outputs of the first CRF model and local information. These feature-engineered NER systems must rely on manually-designed features and they only focus on the shallow information. Meanwhile, the features are usually specific for the task and the models are not general enough.

**Neural Models for Named Entity Recognition** : Recently, neural networks are becoming more and more popular and the implementation of neural networks improves NER performances a lot. Collobert et al. (2011) propose the first word-level neural network model using CNN and CRF. Some other studies exploit character-level architectures. In (Kuru, Can, and Yuret 2016), Kuru, Can, and Yuret (2016) propose CharNER which utilizes a stacked biLSTM and a Viterbi decoder. Gillick et al. (2016) describe an LSTM-based model called BTS which encodes each character into bytes and achieves high performances on CoNLL 2003 dataset without feature engineering. Many models also combine word-level and character-level embeddings. Ma and Hovy (2016) implement a CNN to get character embeddings and they are input into a biLSTM-CRF model with word embeddings to make predictions. Chiu and Nichols (2016) also add character and word features to embeddings and achieve a 91.62% $F_1$ score on CoNLL 2003 English dataset. Limsopatham and Collier (2016) focus on NER on twitter and concatenated representations obtained from features and embeddings in both character and word level are input into a biLSTM-CRF model. Lample et al. (2016) use biLSTM to obtain character-level embeddings and concatenate them with word embeddings to be the inputs of biLSTM-CRF model. Yadav, Sharp, and Bethard (2018) incorporate affix features with character+word NN architecture and show that affix embeddings can capture complementary information. These neural models are limited to the small NER datasets and hard to obtain high results in the open-domain setting if they are trained with specific datasets.

**Unsupervised Pre-training Methods for Named Entity Recognition** : In recent years, unsupervised pre-training is becoming more and more popular and a lot of strong models are proposed. Peters et al. (2018) introduce ELMo representation which is a new type of deep contextualized word representation learned from a pre-trained bidirectional language model. BERT is proposed in (Devlin et al. 2019) and it is pre-trained using the masked language model task and the next sentence prediction task. Both BERT-base and BERT-large achieve very high performances on English CoNLL 2003 dataset. Akbik, Blythe, and Vollgraf (2018) propose contextual string embeddings which are produced with a pre-trained character language model and the biLSTM-CRF model using the contextual string embeddings obtains state-of-art 93.09% $F_1$ score on English CoNLL 2003 dataset. Baevski et al. (2019) present a pre-trained bi-directional transformer model using cloze-style word reconstruction task. The model outperforms all the previous models and achieves 93.5% on English CoNLL 2003 dataset. Although pre-trained language models are proved very helpful for many natural language understanding tasks including NER, these strong models are not fully exploited due to the lack of large-scale and high-quality NER dataset. Only a little NER information from small NER datasets is input into the model during the fine-tuning process. From this point of view, the amount of NER information still needs to be increased when much semantic and syntactic information has been learned.

**Open-domain Named Entity Recognition** : Previous NER models are trained with different datasets in different text genres. The most common datasets are CoNLL 2003 dataset (Sang and Meulder 2003), Ontonote5 dataset (Pradhan et al. 2012), etc. At the same time, some models also focus on NER in twitter (Ritter et al. 2011; Li et al. 2012). However, few studies are focusing on open-domain NER task. Although there are some NER toolkits, such as StanfordNLP (Manning et al. 2014), spaCy\footnote{https://github.com/explosion/spaCy.} and NLTK (Wagner...
Clive Anderson
Clive Stuart Anderson (born 10 December 1952 in Stanmore, Middlesex) is an English television and radio presenter, comedy writer and former barrister. Winner of a British Comedy Award in 1999, Anderson began experimenting with comedy and writing comedic scripts during his 15-year legal career, before starring in Whose Line Is It Anyway? on BBC Radio 4, then later Channel 4. He has also hosted a number of radio programmes, and made guest appearances on Have I Got News for You, Mock the Week and QI.

Anchor String | DBpedia Type | Entity Type
--- | --- | ---
Clive Anderson | Person | PER
Stanmore | Place | LOC
BBC Radio 4 | Organisation | ORG
Channel 4 | Organisation | ORG
Whose Line Is It Anyway? | TelevisionShow | MISC
Have I Got News for You | TelevisionShow | MISC
Mock the Week | TelevisionShow | MISC
QI | TelevisionShow | MISC

Step 1: Extract anchor strings.
Step 2: Search in DBpedia.
Step 3: Use these entities to annotate text.

3 AnchorNER Dataset
We apply the pipeline described hereafter to a dump of abstracts of English Wikipedia from 2017 and obtain AnchorNER. This dataset is built out of 5.2M abstracts of Wikipedia articles, consisting of 268M tokens accounting for 12M sentences. The pipeline used to annotate named-entity from abstracts of Wikipedia is illustrated in Figure 1. Our strategy for annotations is a three-step process:

1. We consider the title and anchored strings of hyperlinks in the abstract of articles most likely to be named entities.

2. We search for the types of these potential entities in DBpedia and map them to entity types. For instance, we map Person, Place, Organisation to PER, LOC and ORG, respectively. If an entry does not belong to any of the previous classes, we tag it as MISC.

3. We make an exact match in the original text using the entities we find in DBpedia.

Through this process, we get an initial version of AnchorNER. The coverage, or the ratio of annotated tokens, of the dataset is 10.22%, which is lower than the coverage of the manually-annotated CoNLL-2003 dataset which is 16.64%. The main reason for our relatively low coverage is that we may miss the entity which does not appear in any anchored string. The missing entities, for example, Middlesex (LOC), British Comedy Award (ORG), will be captured by our correction model mentioned in the next section. The importance of the correction step will be illustrated in detail in the ablation study section.

After the false-negative entity labels are corrected by the correction model, the coverage of AnchorNER raises to 22.26%. We further assess the annotation quality of a random subset of 1000 tokens and we measure an accuracy of 98% for labels, which is better than the 92% accuracy of WiNER dataset.

4 Our Method
In this section, we introduce our methods. First, we briefly give a definition of the NER task in Section 4.1. Then we
build up a correction dataset in Section 4.2 and propose a semi-supervised correction model in Section 4.3. Section 4.4 describes how we use the idea of curriculum learning to learn the correction model.

4.1 Task Definition
Before introducing our method, we first give a formal formulation of the NER task.

NER is the process of locating and classifying named entities in text into predefined entity categories. In this paper, we define four types of entities: person (PER), organizations (ORG), locations (LOC) and miscellaneous names (MISC).

Formally, we define a sentence as a sequence of tokens \( s = \langle w_1, w_2, \ldots, w_n \rangle \). NER is to annotate each token with a tag. Tokens tagged with O are outside of named entities. The B-X tag indicates that this token is the first word of a named entity of type \( X \) while the I-X tag is used for words inside a named entity of type \( X \).

4.2 Correction Dataset
The goal of the neural correction model is to use carefully annotated high-quality dataset to correct the false negatives of large open-domain Wikipedia text. In order to implement the correction model, we first build a correction dataset with our AnchorNER dataset and DocRED dataset (Yao et al. 2019). DocRED is the largest human-annotated dataset constructed from Wikipedia and Wikidata with named entities annotated. Since DocRED is obtained from Wikipedia, as well as AnchorNER, there are 2,937 articles consisting of 8,882 sentences that appear in both datasets. Though some of the articles are not the same, most of the entities manually annotated in articles from DocRED have appeared in abstracts from AnchorNER. We believe that the initial labels in AnchorNER can help us learn how to make annotations in DocRED, in other words, learn a pattern of manual annotations. To get ground truth for each token in the articles which appear in both datasets, we find out all the entities marked in DocRED and use them to make exact matches in abstracts from AnchorNER, in descending order of length of entities. If we match a phrase that has not been matched, we will consider this tag as the ground truth of all the tokens in this phrase. For those sentences that do not obtain any ground truth, we remove them from our dataset because the difference between them and the corresponding ones in DocRED may be too big, which means they can not help with the learning of the correction model. As a result, the dataset comprises 121,627 tokens accounting for 4,288 sentences, containing one word per line with empty lines representing sentence boundaries. Each word is followed by two tags. The first is the initial label in AnchorNER, and the second is the ground truth obtained from DocRED.

4.3 Correction Model
We define our correction dataset as \( \mathcal{D} = \{ S, L, L' \} \) where \( S \) is the sentences in the correction dataset, \( L \) is the entity labels from AnchorNER dataset and \( L' \) is the entity labels from DocRED dataset. We define \( S = \{ s_1, s_2, \ldots, s_n \} \) where \( s_i = \{ w_1, w_2, \ldots, w_m \} \) is a sentence and \( w_i \) is a word. For entity labels, \( L = \{ l_1, l_2, \ldots, l_n \} \) and \( L' = \{ l'_1, l'_2, \ldots, l'_n \} \) where \( l_i = \{ l_i^1, l_i^2, \ldots, l_i^n \} \) and \( l'_i = \{ l'_i^1, l'_i^2, \ldots, l'_i^n \} \).

For a sentence \( s_i \in S \), we first input it into a BERT model to obtain its representations \( r_i = \{ r_i^1, r_i^2, \ldots, r_i^m \} \) where \( m \) denotes the length of the sentence. Before the last classification layer, we embed the entity labels \( l_i \) from AnchorNER dataset and concatenate the entity label embeddings and the sentence representations in the corresponding positions. We denote \( e_i = \text{Embed}(l_i) \) and \( c_i = e_i \oplus r_i \) where \( \oplus \) means concatenation. Therefore, the last classification layer can be defined as \( \text{CLS}(c_i) = p(l_i'|c_i; \theta) \) where \( \theta \) denotes the parameters of the classification layer. The objective function for our correction model can be defined as

\[
\mathcal{J} = - \sum_i p(l_i'|c_i; \theta) \tag{1}
\]

4.4 Curriculum Learning
Curriculum learning (Bengio et al. 2009) is a training strategy proposed by Bengio et al. in the context of machine learning. They demonstrate that models can be better trained when the inputs are not randomly presented but organized in a meaningful order, such as from easy to hard.

Inspired by this thought, we rank all sentences in the correction dataset from easy to hard and split the correction dataset into three sets which are input into the correction model in order. Specifically, we calculate an \( F_1 \) score \( f_i \) for each sentence in the correction dataset with the corresponding entity label \( l_i \) and \( l'_i \) (see Figure 3 for the distribution). We remove the sentences whose \( F_1 \) scores are lower than 0.1. Then, we rank all the sentences in the correction dataset according to their \( F_1 \) score from high to low and split the correction dataset into three sets \( D_1, D_2 \) and \( D_3 \). That means in \( D_1 \), the sentence has more similar labels \( l_i \) and \( l'_i \) and \( D_1 \) is easier for the correction model to learn. Similarly, \( D_3 \) is more difficult for the model to learn. We input the three sets from \( D_1 \) to \( D_3 \) and train our correction model with each set for five epochs.
In this section, we evaluate the effectiveness of the proposed method using several different NER data sets.

5.1 Experiment Setup

Data Sets We conduct our experiments on six open-domain NER data sets.

- **CoNLL03** (Sang and Meulder 2003) the CoNLL 2003 Shared Task dataset is a well known NER dataset built up with Reuters newswire articles. It is annotated with four entity types (PER, LOC, ORG and MISC).
- **DocRED** (Yao et al. 2019) DocRED is a human-annotated dataset constructed from Wikipedia and Wiki-data. Yao et al. (2019) annotate 5,053 Wikipedia documents containing 1,022k words with named entities and their relations. Entity types include person, location, organization, time and number which are mapped to CoNLL 2003 named entity (NE) classes.
- **Ontonote5** (Pradhan et al. 2012) the OntoNote 5.0 dataset contains newswire, magazine articles, broadcast news, broadcast conversations, web data and conversational speech data. The dataset has about 1.6M words and is annotated with 18 named entity types. We follow (Nothman 2008) to map annotations to CoNLL 2003 tag set.
- **Tweet** (Ritter et al. 2011) Ritter et al. (2011) annotate 2,400 tweets (34k tokens) with 10 entity types and we map the entity types to CoNLL 2003 tag set.
- **Webpage** Ratinov and Roth (2009) manually annotated a collection of 20 webpages (8k tokens) on different topics with the CoNLL 2003 NE classes.
- **WikiGold** (Balasuriya et al. 2009) Balasuriya et al. (2009) manually annotate a set of Wikipedia articles comprising 40k tokens with the CoNLL 2003 tag set.

Competing Methods We compare the performance of our model against the following approaches.

- **Bi-LSTM-CRF** (Huang, Xu, and Yu 2015) A bidirectional Long Short-Term Memory (LSTM) network with a Conditional Random Field (CRF) layer.
- **CVT** (Clark et al. 2018) A semi-supervised learning algorithm that improves the representations of a Bi-LSTM sentence encoder using a mix of labeled and unlabeled data.
- **ELMo** (Peters et al. 2018) A type of deep contextualized word representation which models both complex characteristics of word use and how these uses vary across linguistic contexts.
- **Flair** (Akbik, Bergmann, and Vollgraf 2019; Akbik, Blythe, and Vollgraf 2018) A type of contextual word representation which is distilled from all contextualized instances using a pooling operation.
- **BERT** (Devlin et al. 2019) A language representation model which is designed to pre-train deep bidirectional representations from the unlabeled text by jointly conditioning on both left and right context in all layers.

5.2 Implementation Details

We first split AnchorNER dataset into training set, development set and testing set according to the categories of different titles. Considering of the limitation of computing resources, we randomly sample 70,000 abstracts and 20,000 abstracts for training phase and testing phase respectively, accounting for three million tokens from AnchorNER datasets.

All the BERT models in our model use default parameters. We use the cased BERT model and the maximum sequence length is 128. In the correction model, we utilize 12-dimension one-hot vectors for embeddings of entity labels from AnchorNER. The optimizer is BERTAdam (Devlin et al. 2019) and learning rate is 1e-5 for both BERT-base and BERT-large. We set batch size as 32 for BERT-base and 8 for BERT-large. Warming-up proportion is 0.4. We use our AnchorNER training set to fine-tune both the BERT-base and BERT-large for 5 epochs. After that, both BERT-base and BERT-large are fine-tuned with CoNLL dataset for 50 epochs. For the competing methods, the BERT models are fine-tuned and other models are trained with CoNLL 2003 English dataset only.
Table 1: Comparing our methods with state-of-the-art methods on various open-domain NER datasets.

| Model             | CoNLL03 | DocRED | Ontonote5 | Tweet | Webpage | Wikigold | Avg. |
|-------------------|---------|--------|-----------|-------|---------|----------|------|
| **Public NER toolkits** |         |        |           |       |         |          |      |
| NLTK              | 48.91   | 45.19  | 39.00     | 21.18 | 28.17   | 44.22    | 37.78|
| SpaCy             | 65.14   | 51.32  | **76.66** | 31.87 | 36.39   | 58.86    | 53.37|
| StandordNER       | 87.95   | 53.37  | 60.64     | 35.74 | 44.34   | 62.00    | 57.34|
| **State-of-the-art neural models** |         |        |           |       |         |          |      |
| Bi-LSTM-CRF       | 87.00   | 52.30  | 56.33     | 33.66 | 46.45   | 61.81    | 56.26|
| CVT               | 92.09   | 67.43  | 62.41     | 40.49 | 51.76   | 73.57    | 64.63|
| ELMo              | 92.51   | 69.84  | 62.35     | 33.82 | **52.33** | 75.63    | 64.41|
| Flair             | 91.95   | 71.82  | 66.63     | 47.79 | 45.58   | 78.21    | 67.00|
| BERT<sub>base</sub> | 92.10   | 73.67  | 66.31     | 50.00 | 49.85   | 80.44    | 68.72|
| BERT<sub>large</sub> |         |        |           |       |         |          |      |
| **Our models**    |         |        |           |       |         |          |      |
| OpenNER<sub>base</sub> | 92.02  | 79.80  | 66.71     | 50.22 | 48.75   | **82.37** | 70.12|
| OpenNER<sub>large</sub> | 92.15  | **80.20** | 68.19    | **51.45** | 50.81 | 82.16   | **70.83** |

Table 2: Ablation Study Results.

| Model                | CoNLL03 | DocRED | Ontonote5 | Tweet | Webpage | Wikigold | Avg. |
|----------------------|---------|--------|-----------|-------|---------|----------|------|
| Our<sub>base</sub>   | 92.02   | 79.80  | 66.71     | 50.22 | 48.75   | **82.37** | 70.12|
| w/o Wiki label       | 92.11   | 78.03  | **67.29** | 49.50 | 48.56   | 81.20    | 69.45|
| smaller correction dataset | 92.13 | 77.56  | **67.29** | 48.57 | 47.09   | 81.90    | 69.09|
| w/o curriculum learning | 91.58 | 77.59  | 66.94     | 46.98 | 47.21   | 81.05    | 68.56|
| w/o correction       | 91.86   | 73.44  | 66.44     | 46.75 | 46.29   | 78.41    | 67.20|

**Ablation on fine-tuning with labeled data**

| Model                | CoNLL (i.e.,B<sub>base</sub>) | DocRED | Ontonote5 | Tweet | Webpage | Wikigold | Avg. |
|----------------------|--------------------------------|--------|-----------|-------|---------|----------|------|
| CoNLL (i.e.,B<sub>base</sub>) | 91.95 | 71.82 | 66.63 | 47.79 | 45.58 | 78.21 | 67.00 |
| DocRED               | 70.25 | **88.18** | 56.82 | 47.03 | **50.68** | 76.76 | 64.95 |
| DocRED+CoNLL (mixed) | 90.63 | 88.02 | 64.89 | 43.25 | 49.30 | 77.44 | 68.92 |
| DocRED+CoNLL (sequential) | 91.69 | 75.81 | 66.24 | 47.86 | 48.23 | 79.79 | 68.27 |

5.3 Main Results

**Comparison with competing methods** The performance of each model is presented in Table 5.2. From the results, we have the following observations: (1) Our approach is comparable to the state-of-the-art result on the benchmark dataset CoNLL03. (2) Our approach outperforms existing methods for open-domain NER, increasing average \(F_1\) score by 4.66% and 3.07% with BERT base and large respectively.

**Comparison with existing NER toolkits** As shown in Table 5.2, our toolkit OpenNER significantly and consistently outperforms existing NER toolkits on six NER data sets.

5.4 Ablation Studies

Now we systematically look into some important components of our method, and we conduct some ablation studies to analyze the effectiveness of each component.

**Effectiveness of AnchorNER dataset** We evaluate the effectiveness of AnchorNER dataset by not using AnchorNER dataset. Instead, we train a BERT model directly with DocRED dataset and CoNLL dataset. We try two training methods: (1) First use DocRED dataset then use CoNLL dataset to fine-tune BERT model twice. (2) Use the mix of DocRED dataset and CoNLL dataset to fine-tune BERT model. These two methods reduce the performance by 2.64% and 1.71% as shown in Table 5.2. Although DocRED is a high-quality NER dataset, its size is still limited. Our models are exposed to more named entities if we use AnchorNER dataset and our models can obtain much more information.

**Effectiveness of AnchorNER label** We remove the AnchorNER label from the correction model to evaluate its effectiveness. In this way, the correction model becomes an original BERT classifier trained with DocRED dataset. Results show that removing these labels slightly hurts the performance of the model, reducing the average score by 0.96%. Although our uncorrected dataset has a low recall rate, it still has precise labels which can add more information to DocRED dataset.

**Effectiveness of the size of the correction dataset** We try to restrict the collection of the correction dataset, leaving only the same sentences that appear in both AnchorNER and
DocRED. A total of 2,587 sentences consisting of 61,343 tokens meets this requirement, which is almost half the size of our correction dataset. The results in the third row of Table 5.2 illustrate that the overall performance using this smaller correction dataset is decreased by 1.47%. As the correction model is exposed to a larger correction dataset, more information about corrected named entities is learned by our correction model, which improves the performances of the model.

Effectiveness of correction We study the effectiveness of correction by removing the process of correction and only using the initial label from AnchorNER. As we mentioned in Section 3, we will miss some entities in this way, resulting in a lower recall rate. As shown in the fifth row of Table 5.2 show that removing the correction part leads to a reduction of 4.16% in average score. This result illustrates that correction process is the most important part in our approach, and the improvement of the quality of AnchorNER dataset can greatly improve the final performance of the model.

Effectiveness of curriculum learning During the training of the correction model, we take advantage of the idea of curriculum learning. We compare this training method with a variant, in which we directly train the correction model with the whole correction dataset. From Table 5.2 we can see that if we do not use this idea, the performance is decreased by 2.22%. By using curriculum learning, we input our correction dataset into the correction model in order and it is easier for the correction model to learn the pattern of the correction process.

5.5 Qualitative Study of the Correction Model

Corrected v.s. uncorrected AnchorNER dataset We compare the labels for the same 1000 tokens in corrected and uncorrected AnchorNER dataset and 61 entities are retrieved after correction. This result clearly illustrates the effectiveness and necessity of the correction process.

Case study Figure 4 shows two examples of correction. The sentence in Figure 4(a) first appears in Figure 1, where the entity British Comedy Award is not recognized during the three-step process mentioned in Section 3 because it does not appear in any hyperlink. The second example in Figure 4(b) shows that entity CANDU and Canada Deuterium Uranium are not recognized. In the uncorrected dataset, Canada is mislabeled as B-LOC. This is because Canada appears in one of the hyperlinks, but Canada Deuterium Uranium does not. With the help of the correction model, all the labels are corrected.

However, we also identify some errors during the correction process as shown in Figure 4(c). Two Flint should be marked as MISC, but the correction model fails to recognize it. Instead, it only tags Flint as I-MISC. Another type of error is that the model modifies the correct label to the wrong label. Maybe our model should be further improved to address these issues.

5.6 Discussion

Our work is similar to the application of distant supervision on relation extraction (Mintz et al. 2009). As they extract relations from Freebase, we extract entities from anchored strings and search them in DBpedia. For each entity that appears in DBpedia, we find all the locations where it appears in the sentence and annotate them with the entity type mentioned in DBpedia.

When building large dataset, instead of being supervised by knowledge bases in their work, our algorithm is supervised by the correction model trained with a relatively smaller but high-quality dataset. We use the correction model to correct the false negatives of Wikipedia text so as to increase the recall rate of AnchorNER dataset. Therefore, our approach takes advantage of semi-supervised methods, leading to state-of-the-art performances.

6 Conclusion and Future Work

In this paper, we propose a semi-supervised annotation framework to make full use of abstracts from Wikipedia and obtain a large and high-quality dataset called AnchorNER. We utilize anchored strings in abstracts and DBpedia to annotate the dataset. We also design a neural correction model trained with DocRED to correct the false-negative entity labels and then train a BERT model with the corrected dataset. Our trained BERT model has obtained state-of-the-art open-domain performances — average $F_1$ score is increased by 4.66% and 3.07% with BERT base and large respectively. In the future, we can also use our AnchorNER dataset during the pre-training process and come up with more pre-training methods leveraging NER information.

References

Akbik, A.; Bergmann, T.; and Vollgraf, R. 2019. Pooled contextualized embeddings for named entity recognition. In NAACL-HLT (1).

Akbik, A.; Blythe, D.; and Vollgraf, R. 2018. Contextual string embeddings for sequence labeling. In COLING.
Huang, L.; Zhou, J.; and Sun, M. 2019. Docred: A large-scale document-level relation extraction dataset. In ACL (1).
Zhou, G., and Su, J. 2002. Named entity recognition using an hmm-based chunk tagger. In ACL.