A CRF-Based Stacking Model with Meta-features for Named Entity Recognition

Shifeng Liu(✉), Yifang Sun, Wei Wang, and Xiaoling Zhou

The University of New South Wales, Sydney, Australia
shifeng.liu@unsw.edu.au, {yifangs,weiw,xiaolingz}@cse.unsw.edu.au

Abstract. Named Entity Recognition (NER) is a challenging task in Natural Language Processing. Recently, machine learning based methods are widely used for the NER task and outperform traditional handcrafted rule based methods. As an alternative way to handle the NER task, stacking, which combines a set of classifiers into one classifier, has not been well explored for the NER task. In this paper, we propose a stacking model for the NER task. We extend the original stacking model from both model and feature aspects. We use Conditional Random Fields as the level-1 classifier, and we also apply meta-features from global aspect and local aspect of the level-0 classifiers and tokens in our model. In the experiments, our model achieves the state-of-the-art performance on the CoNLL 2003 Shared task.

Keywords: Named Entity Recognition · Stacking
Feature engineering

1 Introduction

Named Entity Recognition (NER) is a fundamental stage in Natural Language Processing (NLP). In the sentence "Trump dined at the Trump National Hotel.", NER aims to identify "Trump" as a person and "Trump National Hotel" as an organization. The identified entities can be then used in downstream applications (e.g., information extraction systems) and other NLP tasks (e.g., named entity disambiguation and relation extraction).

Early NER systems are based on handcrafted rules. The rules are defined by human experts, which makes them labour consuming to develop and, more importantly, impossible to cover all the cases. Recently, machine learning techniques become more popular and effective in solving the NER problem. Supervised learning is one of the most widely used learning approaches for the NER task, which takes a set of human labelled documents as the training data,

This work was partially supported by D2DCRC DC25002 and DC25003, and ARC DP 180103411.

D. Phung et al. (Eds.): PAKDD 2018, LNAI 10938, pp. 54–66, 2018.
https://doi.org/10.1007/978-3-319-93037-4_5

and predicts the named entities in the given documents (i.e., test data). Linear classifiers, such as Hidden Markov Model [8] and Conditional Random Fields (CRF) [7,10], are proved to be effective. However, the performance of these models is highly associated with carefully designed features, whose effectiveness is usually restricted within a specific domain. In recent years, an enormous amount of research effort has been put into neural network based methods [4,11,12,16], and these methods have achieved the state-of-the-art performance. Most neural network based methods simply use word embedding vectors as input [14,15]. However, to train the word embedding vectors, it requires a huge amount of unlabelled data in order to achieve high performance. These neural network based models are hard to adjust due to the non-trivial hyperparameter tuning process. The training stage is also much more time-consuming than linear classifiers, even with high performance GPUs.

Stacking [2,25] is an alternative way to improve the accuracy of a machine learning task. As a two-phrase method, the first step of stacking is to train a set of the level-0 classifiers (i.e., the base classifiers) using different models on the training dataset. In the second step, the level-1 classifier (e.g., a linear regression classifier) is trained on the training dataset with the predicted results from the level-0 classifiers as features. As such, the level-1 classifier is expected to achieve better performance than the level-0 classifiers.

Although traditional stacking method has shown its effectiveness on many machine learning tasks, there are still some potential improvements that can be explored, especially for the NER task. In this paper, we propose a CRF based stacking model with carefully designed meta-features to solve the NER problem. Comparing with the traditional stacking model, our model has two major differences. Firstly, we use CRF instead of linear regression as the level-1 classifier. The idea is inspired by the fact that CRF has shown its advantage as a sequential model in the NER task, while linear regression only works on independent instances. Secondly, we use a mix of meta-features and local features to improve the accuracy of the stacking model. Previous works either simply use the predicted results from the level-0 classifiers, or only extract features from the surface of tokens [7,10,18] (i.e., local features). We observe that the stacking model can also benefit from the non-local information. For example, the distribution of a token on different named entity types acts like the prior knowledge when we make the prediction. Moreover, our proposed model shows its robustness even when the performance of the level-0 classifiers is not good. This would be very helpful when users want to involve commercial NER systems (which usually cannot be tuned or re-trained).

The contributions of this paper are as follows:

- We proposed the CRF based stacking model named as SMEF, which took meta-features into consideration.
- We proposed a set of meta-features and local features and integrated these features in the stacking model to achieve better performance. We presented the details of the model and features in Sect. 3.

– We conducted extensive performance evaluation against the state-of-the-art NER systems. The proposed model outperforms all of them and achieves the overall F_1 score of 92.38% in the CoNLL 2003 Shared task. Moreover, the experiment results also show the effectiveness of each type of features and the robustness of the proposed model. Section 4 reports the experiment results and analyses.

2 Related Work

Named Entity Recognition is a research topic with a long history. Most recent approaches to NER have focused on CRF model and neural network models. CRF is a sequence model which could be used to predict sequences of labels based on the handcrafted features [7,9,10,18]. Neural network takes word embedding vectors as input features and learns a dense score vector for each possible named entity types [4,11,12,16]. Moreover, CRF model can be used as the output layer in neural network based models. Some of the most recent works [11,12,16] have shown the effectiveness of this combination.

Stacking has been proposed for many years [2,25] as a way to combine multiple classifiers (the level-0 classifiers) into one model (the level-1 classifier) in order to achieve better accuracy. While the level-0 classifier can be any machine learning model, the level-1 classifier is usually linear regression [25], stacking trees and ridge regression [2]. FWLS [21] uses a linear combination of meta-features to formulate the weight of the level-0 classifiers in the level-1 classifier, and achieves good performance in the Netflix Prize competition. [17,27] add meta-features extracted from the level-0 classifiers to the level-1 classifier to improve the stacking performance.

Stacking has been applied to NLP tasks such as Part-of-Speech Tagging [3,22] and NER [6,23,24,26]. Tsukamoto et al. [23] and Wu et al. [26] apply an extension of AdaBoost to learn the level-1 classifier. They take the sequence label information into consideration, and use handcrafted features from tokens. [6,24] use CRF model as the level-1 classifier to solve biomedical NER tasks. They also make use of the handcrafted local features.

3 Model

Named Entity Recognition takes a token sequence $\mathbf{x}_i = (x_{i1}, \ldots, x_{is})$ as input, and predicts a corresponding label sequence $\mathbf{y}_i = (y_{i1}, \ldots, y_{is})$, where $\mathbf{x}_i$ is taken from the i-th sentence in the dataset $\mathcal{X}$ and s is the length of the token sequence. As a named entity could span several tokens, we do not directly use the named entity types as labels. Instead, we apply chunking encoding for these labels. There are two popular encoding methods: BIO (i.e., **B**egin, **I**nside, and **O**utside token of a named entity) and BIOES (i.e., **B**egin, **I**nside, **O**utside, **E**nd, and **S**ingle token of a named entity). Table 1 shows an example.

As there are various named entity types, we form the NER task as a multi-class classification problem. As such, for each x_{ij}, a classifier (either a level-0

Table 1. Example of chunking encoding

	x_{i1}	x_{i2}	x_{i3}	x_{i4}	x_{i5}	x_{i6}	x_{i7}	x_{i8}
Token	Trump	dined	at	the	Trump	National	Hotel	.
	y_{i1}	y_{i2}	y_{i3}	y_{i4}	y_{i5}	y_{i6}	y_{i7}	y_{i8}
BIO	B-PER	O	O	O	B-ORG	I-ORG	I-ORG	O
BIOES	S-PER	O	O	O	B-ORG	I-ORG	E-ORG	O

classifier or the level-1 classifier) returns a vector $\mathbf{c}(x_{ij})$ with m dimensions, where m is the number of classes. More specifically, in the NER task, m is the number of predefined named entity types. Generally, each dimension in $\mathbf{c}(x_{ij})$ is a binary value, i.e., 1 for the predicted class and 0 for the rest classes. Real numbers can be used in some classifiers to represent the probability or the confidence score of each class.

In this section, we will firstly give more detail about the stacking method [2,25]. Then we will introduce our model along with different meta-feature.

3.1 Stacking

Stacking is a two-phrase method. All the level-0 classifiers are trained on dataset $\mathcal{X}$. We denote the prediction of token sequence $\mathbf{x}_i$ by the k-th level-0 classifier as $\mathbf{c}_k(\mathbf{x}_i) = (\mathbf{c}_k(x_{i1}), \ldots, \mathbf{c}_k(x_{ij}), \ldots, \mathbf{c}_k(x_{is}))$, where $\mathbf{c}_k(x_{ij})$ is a m dimensional one-hot encoding vector. In this paper, all vectors are assumed to be column vectors unless noted. With predicted results from the level-0 classifiers, we generate a new dataset $\mathcal{Z}$, which consists of $\{(\mathbf{z}_i, \mathbf{y}_i), i = 1, \ldots, N\}$, where N is the size of dataset $\mathcal{X}$, $\mathbf{y}_i$ is the ground truth label sequence corresponding to the token sequence $\mathbf{x}_i$, and $\mathbf{z}_i$ is a vector sequence corresponding to $\mathbf{x}_i$. Each $\mathbf{z}_{ij}$ in $\mathbf{z}_i$ consists of $(x_{ij}, \mathbf{c}_1(x_{ij}), \ldots, \mathbf{c}_L(x_{ij}))$, where L is the number of the level-0 classifiers.

In [2], the loss function of stacking method is $\mathcal{L} = \sum_{i,j}(y_{ij} - \sum_k w_k c_k(x_{ij}))^2$, where w_k is the weight of $c_k(x_{ij})$, and both y_{ij} and $c_k(x_{ij})$ belong to $\mathcal{R}$.

However, this loss function is designed for regression problem thus can not be directly applied for a multi-class classification problem. Moreover, it does not consider the label dependency between sequential tokens.

In order to resolve the above issues, in this paper, we propose to use the Conditional Random Fields (CRF) model as the level-1 classifier instead of a linear regression model. The loss function of CRF is $\mathcal{L} = -\sum_i \log(p(\mathbf{y}_i \mid \mathbf{z}_i))$. Given each vector sequence $\mathbf{z_i}$, the probability of the corresponding label sequence $\mathbf{y_i}$ can be formulated as

$$p(\mathbf{y}_i \mid \mathbf{z}_i) = \frac{\exp(\sum_{j=0}^{s} U(\mathbf{z_{ij}}, y_{ij}) + \sum_{j=0}^{s-1} T_{y_{ij}, y_{i(j+1)}})}{\sum_{y\prime \in \mathbf{Y}_{\mathbf{z_i}}} \exp(\sum_{j=0}^{s} U(\mathbf{z_{ij}}, y\prime_{ij}) + \sum_{j=0}^{s-1} T_{y\prime_{ij}, y\prime_{i(j+1)}})}, \tag{1}$$

where $T_{y_{ij},y_{i(j+1)}}$ is the transition score between y_{ij} and $y_{i(j+1)}$, $\mathbf{Y}_{\mathbf{z_i}}$ is the set of all the possible label sequences, and $U(\mathbf{z}_{ij}, y_{ij})$ is the unary potential score of $\mathbf{z}_{ij}$ with the corresponding label y_{ij}. More specifically, $U(\mathbf{z}_{ij}, y_{ij})$ is defined as

$$U(\mathbf{z}_{ij}, y_{ij}) = \sum_{win} \sum_{k} \mathbf{w}^{\top}_{k,win,y_{ij}} \mathbf{c}_k(x_{i(j+win)}) + bias_{y_{ij}}, \tag{2}$$

where $\mathbf{w}_{k,win,y_{ij}}$ is the weight vector of the predicted result from the level-0 classifier c_k for token $x_{i(j+win)}$, and $bias_{y_{ij}}$ is the learned bias corresponding to label y_{ij}. win denotes the token offset in the context window. In this paper, we set two as the size of the context window, i.e., $win \in [0, \pm 1, \pm 2]$.

3.2 Stacking with Meta-features

We have observed that the characteristic of the level-0 classifiers is useful for the level-1 classifier. For example, if a classifier consistently recognizes "Jordan" as a location, we should not give such classifier too much trust when dealing with "Jordan" as "Jordan" could also be a person's name.

In this subsection, we propose several meta-features based on the statistic information of dataset $\mathcal{Z}$. More specifically, for each level-0 classifier, we will extract meta-features from its prediction results on dataset $\mathcal{X}$. Note that the values of meta-features vary from different datasets such as training set, development set, and testing set.

The unary potential part of the model is therefore modified as

$$\begin{aligned} U(\mathbf{z_{ij}}, y_{ij}) = & \sum_{win} \sum_{k} \mathbf{u}^{\top} \mathbf{w}^{\top}_{meta,k,win,y_{ij}} F_{meta}(x_{i(j+win)}, c_k) \mathbf{c}^{\top}_k(x_{i(j+win)}) \mathbf{u} \\ & + bias_{y_{ij}}, \end{aligned} \tag{3}$$

where $F_{meta}(x_{ij}, c_k)$ returns a vector of meta-features extracted from the level-0 classifier c_k for token x_{ij}, $\mathbf{w}_{meta,k,win,y_{ij}} \in \mathcal{R}^{|F_{meta}| \times m}$ is the weight matrix, and $\mathbf{u}$ is an all-ones vector with m dimension. Note that since there is only one nonzero element in $\mathbf{c}_k(x_{ij})$, only one column of weights in $\mathbf{w}_{meta,k,win,y_{ij}}$ are activated.

In our proposed model, $F_{meta}(x_{ij}, c_k)$ consists of four meta-features: constant, token label prior, token majority label, and token label entropy.

Constant. A constant 1 is used to maintain the predicted label of token x_{ij} from c_k. This feature helps us improve the model without losing the original information. Note that if we only apply this feature, then Eq. 3 falls back to Eq. 2, which is the standard CRF model.

Token Label Prior. Each token has a prior probability of being a named entity type $type_t$. For example, if token x_{ij} appears 11 times in the dataset and 9 of them are predicted as a person by classifier c_1, then we can approximate its token label prior of being a person as $\frac{9}{11}$. For each $type_t$, we use $F_{meta,prior,type_t}(x_{ij}, c_k, type_t)$ to denote the token label prior.

Token Majority Label. We define $F_{meta,major}(x_{ij}, c_k)$ to denote whether $\mathbf{c}_k(x_{ij})$ is consistent with the majority label of token x_{ij} for classifier c_k. For example, assume the surface of token x_{ij} is "Jordan" and the majority label of "Jordan" under the prediction of c_1 is person. Then we set $F_{meta,major}(x_{ij}, c_1) = 1$ for $\mathbf{c}_k(x_{ij} =$ "Jordan") indicates person and $F_{meta,major}(x_{ij}, c_1) = 0$ otherwise. We use $F_{meta,major}(x_{ij}, c_k)$ to enhance the impact of $\mathbf{c}_k(x_{ij})$.

Token Label Entropy. The token majority label feature may not be effective when the majority prediction is not distinguishable. Therefore, we apply the entropy of named entity types for token x_{ij}, denoted as $F_{meta,entropy}(x_{ij}, c_k)$, to further improve the performance of our model.

3.3 Stacking with Joint Meta-Features

We also observe that for a given token, the predicted labels over all the level-0 classifiers are helpful for the final decision. For example, if four out of five level-0 classifiers recognize token x_{ij} as a person, then most likely it is a person. However, this can not be captured by meta-features as they consider information from the level-0 classifiers independently.

We propose a set of meta-features which consider the joint information from the predicted labels of the given token from each level-0 classifier, and name them joint meta-features.

Similarly, we add the joint meta-features into the unary potential part and change it to

$$\begin{aligned} U(\mathbf{z_{ij}}, y_{ij}) = &\sum_{win} \sum_{k} \mathbf{u}^\top \mathbf{w}^\top_{meta,k,win,y_{ij}} F_{meta}(x_{i(j+win)}, c_k) \mathbf{c}_k^\top(x_{i(j+win)}) \mathbf{u} \\ &+ \sum_{win} \mathbf{w}^\top_{joint,win,y_{ij}} F_{joint}(\mathbf{c}_1(x_{i(j+win)}), \ldots, \mathbf{c}_L(x_{i(j+win)})) + bias_{y_{ij}}, \end{aligned} \tag{4}$$

where $F_{joint}(\mathbf{c}_1(x_{ij}), \ldots, \mathbf{c}_L(x_{ij}))$ is a vector of the joint meta-features extracted from the predicted labels of the level-0 classifiers, and $\mathbf{w}_{joint,win,y_{ij}}$ is the weight vector.

We propose the following two joint meta-features: context prior, and joint label.

Context Prior. Under different local contexts, the probability of a token being $type_t$ should be different. For example, in sentence "Jordan is in Asia.", "Jordan" is more likely to be a location. We use the portion of the number of labels predicted as $type_t$ labels over the number of the level-0 classifiers to approximate the local context prior, denoted as $F_{joint,prior,type_t}(\mathbf{c}_1(x_{ij}), \ldots, \mathbf{c}_L(x_{ij}))$. For example, the context prior of "Jordan" being predicted as a location is $\frac{2}{3}$ when two of three level-0 classifiers predict "Jordan" as a location.

Joint Label. Following the sequence of the level-0 classifiers i.e., c_1, ..., c_k, we connect their predicted labels as the joint label (i.e., PER-LOC-PER). We denote this joint label feature as $F_{joint,joint}(\mathbf{c}_1(x_{ij}), \ldots, \mathbf{c}_L(x_{ij}))$. It reserves all the information in the predicted labels rather than only keeps the majority label.

3.4 Stacking with Local Embedding Features

In addition, we also apply the local embedding features to enhance our model. Note that even if the level-0 classifiers have applied these local embedding features, there is no duplicated usage of local embedding features as we use a different model.

In order to apply the local embedding features, the unary potential part is modified to

$$\begin{aligned} U(\mathbf{z_{ij}}, y_{ij}) = & \sum_{win} \sum_{k} \mathbf{u}^\top \mathbf{w}^\top_{meta,k,win,y_{ij}} F_{meta}(x_{i(j+win)}, c_k) \mathbf{c}_k^\top (x_{i(j+win)}) \mathbf{u} \\ & + \sum_{win} \mathbf{w}^\top_{joint,win,y_{ij}} F_{joint}(\mathbf{c}_1(x_{i(j+win)}), \ldots, \mathbf{c}_L(x_{i(j+win)})) \\ & + \sum_{win} \mathbf{w}^\top_{local,win} F_{local}(x_{i(j+win)}) + bias_{y_{ij}}, \end{aligned} \tag{5}$$

where $F_{local}(x_{ij})$ extracts the local embedding features from token x_{ij} and $\mathbf{w}_{local,win}$ is the weight vector.

Following [9], we cluster word embeddings [15] by using the batch k-means clustering algorithm [20] with different numbers of clusters. For token x_{ij} and the number of clusters, we use the clustering id as one local embedding feature in our model. The number of clusters is set as 500, 1000, 1500, 2500 and 3000.

4 Experiment

We evaluate our model on two public NER benchmarks: the CoNLL 2003 Shared task [19] and the ACE 2005 dataset. Our model achieves the state-of-the-art performance on both benchmarks. In this section, we will firstly give the details about the datasets and the evaluation metrics. Then we will describe our training process. After that, we will show the overall performance of our model, and feature effectiveness results. In the end, we will show the robustness of our model when using existing low-performance classifiers as the level-0 classifiers.

4.1 Dataset and Evaluation

The CoNLL 2003 Shared Task. (CoNLL03) consists of news articles from the Reuters RCV corpus. There are four predefined named entity types: `PER` (Person), `LOC` (Location), `ORG` (Organization), and `MISC` (Miscellaneous). It includes standard tokenized training, development and test sets. We use the English data of the shared task. The details of the dataset can be found in Table 2.

The ACE 2005 Dataset. (ACE05) consists of articles from diverse sources including Broadcast News, Broadcast Conversations, Newswire, Weblog, Usenet, and Conversational Telephone Speech. Seven named entity types are predefined

Table 2. Statistics of The CoNLL 2003 Shared task [19]

	Articles	Sentences	Tokens	LOC	MISC	ORG	PER
Train	946	14,987	203,621	7,140	3,438	6,321	6,600
Dev.	216	3,466	51,362	1,837	922	1,341	1,842
Test	231	3,684	46,435	1,668	702	1,661	1,617

in the dataset, including `FAC` (Facility), `GPE` (Geo-Political Entity), `LOC` (Location), `ORG` (Organization), `PER` (Person), `VEH` (Vehicle), and `WEA` (Weapon). As we only have the full training set, we split the dataset into training (56%), development (24%), and test (20%) sets following [1]. Texts in the dataset are tokenized using the spaCy tokenizer. We work on the English data of the ACE 2005 dataset. The details of the ACE05 dataset can be found in Table 3.

Table 3. Statistics of The ACE 2005 dataset

	Articles	Sentences	Tokens	FAC	GPE	LOC	ORG	PER	VEH	WEA
Train	337	12,965	164,539	130	3,175	161	1,546	4,506	66	16
Dev.	145	5,142	73,411	81	1,565	83	713	1,804	20	11
Test	117	4,348	60,186	52	1,393	70	674	1,518	23	7

We evaluate the performance of different models by comparing the predicted results on the test set using *Precision*, *Recall*, and F_1 score. The predicted results can be classified into true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), according to the ground truth. *Precision* (P) is defined as $\frac{TP}{TP+FP}$, *Recall* is defined as $\frac{TP}{TP+FN}$, and F_1 score is the harmonic average of P and R. Following the previous work [4,11,16,18], named entities are evaluated in phrase level.

4.2 Training

We use the BIOES chunking encoding as we find it bringing slightly better performance than the BIO encoding. This is also consistent with the observation in the previous work [18]. We pre-process the text by lowercasing all the tokens and replacing all the digits with 0 following [4].

We use the following NER classifiers as our level-0 classifiers:

- spaCy[1] is based on CNN with Glove word embedding vectors [15] as input;
- CoreNLP [13] is based on CRF with handcrafted features;
- UIUC [18] is based on handcrafted features and a set of gazetteers by using a regularized averaged perceptron;

[1] https://spacy.io/.

– MITIE[2] makes use of structural SVM and word embedding vectors;
– NeuroNER [16] has a BLSTM-CRF architecture with Glove word embedding vectors [15] as input.

Following the standard procedure of stacking [25], we generate the training set (i.e. $\mathcal{Z}_{train}$) for the level-1 classifier as follows. Firstly, we separate the original training set (i.e. $\mathcal{X}_{train}$) evenly into five parts (i.e. $\mathcal{X}^1_{train}, \ldots, \mathcal{X}^5_{train}$). Then we train each level-0 classifier with four of them (e.g. $\mathcal{X}^1_{train}, \ldots, \mathcal{X}^4_{train}$) and get the predicted results on the left part (e.g. $\mathcal{Z}^5_{train}$). These predicted results (i.e. $\mathcal{Z}^1_{train}, \ldots, \mathcal{Z}^5_{train}$) are combined together to form $\mathcal{Z}_{train}$. For the development set and test set, we get the predicted results of the level-0 classifiers trained on the original training set (i.e. $\mathcal{X}_{train}$).

We use binary values in $\mathbf{c}(\mathbf{x}_i)$ as described in Sect. 3. For those level-0 classifiers which provide scores for their predicted named entities, we use the score to replace the binary values.

In all the experiments, our model is optimized using stochastic gradient descent with l_2 regularization. We train the proposed model with different C (the coefficient for l_2 regularization) and select the one with the best performance on the development set as the final C of the model. Following [4,16], we train our model ten times and report the average value for each metric. In addition, we also report the standard deviation to show the robustness of model.

4.3 Overall Results

In Table 4, we compare the performance of our model with the following state-of-the-art models:

– NeuroNER (2017) [5] has a neural network architecture of BLSTM-CRF with Glove word embedding vectors [15] as input;
– UIUC (2009) [18] is based on handcrafted features and a set of gazetteers by using a regularized averaged perceptron;
– Lample et al. (2016) [11] combines BLSTM and CRF with input word embedding vectors trained on different corpora from Glove [15];
– Ma and Hovy (2016) [12] is a neural network architecture with combination of BLSTM, CNN and CRF;
– Chiu and Nichols (2016) [4] is a hybrid of BLSTM and CNNs along with a set of gazetteers;
– TagLM (2017) [16] combines GRUs and CRF as well as an external bidirectional neural language model trained on a one billion token corpus.

For the first four models, we show the best results of the reported results in the original papers and the results from our experiments. For the last two models, we list the reported results as we do not have the corresponding models and source codes. In the CoNLL03, our model achieves the best average F_1 score of 92.38%. It shows a significant improvement compared with the previous best result of $91.93\% \pm 0.19\%$ from TagLM [16]. Moreover, our model is more stable than previous methods.

[2] https://github.com/mit-nlp/MITIE.

Table 4. Test set performance comparison in The CoNLL 2003 Shared task

Model	$P \pm std$	$R \pm std$	$F_1 \pm std$
NeuroNER (2017) [5]	90.54%	90.78%	90.66%
UIUC (2009) [18]	91.20%	90.50%	90.80%
Lample et al. (2016) [11]	-	-	90.94%
Ma and Hovy (2016) [12]	91.35%	91.06%	91.21%
Chiu and Nichols (2016) [4]	91.39%	91.85%	91.62% ± 0.33%
TagLM (2017) [16]	-	-	91.93% ± 0.19%
SMEF	**92.95%** ± 0.08%	91.83% ± 0.04%	**92.38%** ± 0.03%

4.4 Effectiveness of Our Model and Meta-features

Table 5 reports the performance of the level-0 classifiers and SMEF in the CoNLL03 and the ACE05 datasets. The best level-0 classifier in the CoNLL03 scores 90.66% F_1. Our model has an increase of 1.72% in F_1 score compared with it. In the ACE05, our model increases the F_1 score by 2.43% compared with the best level-0 classifier.

In SMEF, we use three types of features including meta-features, joint meta-features, and local embedding features. Table 6 shows the results with different types of features on both datasets.

In order to justify our choice of using CRF as the level-1 classifier, we also implement a stacking model with logistic regression classifier as the level-1 classifier (i.e., LR in Table 6). Our model achieves better performance thanks to the sequence inference ability of CRF.

We show that all types of features are effective. Generally speaking, one can always achieve better F_1 score by applying more features. When using all the features, our model achieves the best performance in both datasets.

Meta-features are the most effective. All the models with meta-features consistently outperform those without meta-features. The model with only meta-features (i.e., meta in Table 6) shows a decent result which surpasses the previous best result of 91.93% from TagLM [16] in the CoNLL03. According to our analysis, the most effective meta-feature is the **token label prior**. The other two

Table 5. Test set performance of the level-0 classifiers and SMEF

Dataset	CoNLL03			ACE05		
Model	$P \pm std$	$R \pm std$	$F_1 \pm std$	$P \pm std$	$R \pm std$	$F_1 \pm std$
spaCy	82.62%	80.44%	81.51%	84.30%	74.55%	79.13%
CoreNLP [13]	87.41%	79.32%	83.17%	75.52%	33.34%	46.26%
UIUC [18]	90.10%	80.42%	84.98%	85.06%	84.56%	84.81%
MITIE	88.72%	86.88%	87.79%	80.55%	77.58%	79.03%
NeuroNER [5]	90.54%	90.78%	90.66%	85.80%	**87.82%**	86.80%
SMEF	**92.95%** ± 0.08%	**91.83%** ± 0.04%	**92.38%** ± 0.03%	**91.01%** ± 0.18%	87.52% ± 0.19%	**89.23%** ± 0.07%

Table 6. Effectiveness of meta-features

Dataset	CoNLL03			ACE05		
Model	$P \pm std$	$R \pm std$	$F_1 \pm std$	$P \pm std$	$R \pm std$	$F_1 \pm std$
LR	91.71% ± 0.00%	90.63% ± 0.00%	91.17% ± 0.00%	89.29% ± 0.00%	86.54% ± 0.00%	87.89% ± 0.00%
base_CRF	92.39% ± 0.08%	90.77% ± 0.04%	91.57% ± 0.04%	90.52% ± 0.37%	86.83% ± 0.41%	88.63% ± 0.13%
joint	92.53% ± 0.08%	90.79% ± 0.05%	91.65% ± 0.02%	90.14% ± 0.64%	87.10% ± 0.44%	88.59% ± 0.20%
local	92.47% ± 0.10%	90.95% ± 0.07%	91.75% ± 0.04%	90.80% ± 0.23%	86.92% ± 0.20%	88.82% ± 0.05%
joint + local	92.58% ± 0.08%	91.03% ± 0.04%	91.80% ± 0.05%	90.84% ± 0.32%	86.80% ± 0.21%	88.77% ± 0.13%
meta	92.73% ± 0.08%	91.49% ± 0.07%	92.11% ± 0.05%	90.77% ± 0.21%	87.36% ± 0.38%	89.03% ± 0.10%
meta + joint	92.69% ± 0.04%	91.54% ± 0.06%	92.11% ± 0.03%	90.65% ± 0.48%	87.49% ± 0.38%	89.04% ± 0.06%
meta + local	92.82% ± 0.10%	91.77% ± 0.05%	92.29% ± 0.05%	**91.11%** ± 0.31%	87.40% ± 0.23%	89.22% ± 0.08%
all	**92.95%** ± 0.08%	**91.83%** ± 0.04%	**92.38%** ± 0.03%	91.01% ± 0.18%	**87.52%** ± 0.19%	**89.23%** ± 0.07%

meta-features also show their effectiveness especially for `ORG` and `MISC` in the CoNLL03, and `PER` and `ORG` in the ACE05.

4.5 Our Model with the Existing Level-0 Classifiers

Some NER systems, especially commercial NER systems, cannot be tuned or retrained on a specified dataset. Thus they may not be able to present satisfactory results. Our model offers a solution to deal with new data by only using these existing low-performance classifiers as the level-0 classifiers.

Since these existing classifiers are not trained on the specified dataset, our model is essentially an ensemble model without changing the loss function of CRF. These existing classifiers are usually trained on different datasets, even with different predefined named entity types, which could also be different from the specified dataset. For example, CoreNLP, MITIE, and NeuroNER are trained on the CoNLL03 and have four named entity types; spaCy and UIUC are trained on OntoNotes 5.0 with eighteen named entity types.

Table 7 shows the performance in the ACE05 with the existing level-0 classifiers. Our model achieves F_1 score of 88.87%, which is much better than any of the existing level-0 classifiers. It also outperforms the best trained classifier NeuroNER (whose F_1 score is 86.80%). Comparing with the SMEF model

Table 7. Performance in The ACE 2005 dataset using the existing level-0 classifiers

	Model	$P \pm std$	$R \pm std$	$F_1 \pm std$
Existing	CoreNLP [13]	37.16%	18.62%	24.81%
	NeuroNER [5]	43.40%	41.80%	42.58%
	MITIE	47.07%	40.86%	43.75%
	spaCy	44.40%	53.12%	48.37%
	UIUC [18]	44.61%	52.42%	48.20%
Our model	SMEF	**90.73%** ± 0.54%	**87.10%** ± 0.41%	**88.87%** ± 0.08%

with the trained level-0 classifiers, the one with the existing level-0 classifiers is just slightly worse (e.g., 0.46% lower F_1 score). Moreover, SMEF takes less than 5 min to train the model on the ACE05, which is much faster than a neural network (e.g., NeuroNER needs more than 2 h on the same dataset).

There are mainly three reasons for this. The first reason is that SMEF makes use of consistent and correlated named entity types between the level-0 classifiers and the ACE05. The second reason is that the level-0 classifiers provide prior distributions on corresponding named entity types for each token even though they have different predefined named entity types. Thus, meta-features would be effective in this scenario. The last reason is that local embedding features provide additional information beyond the named entity labels predicted by the level-0 classifiers.

5 Conclusion

In this paper, we propose a new stacking method with CRF model and meta-features for the NER task. These meta-features extract non-local information over the dataset for each level-0 classifier, and local information of the level-0 classifiers and tokens. Our approach, SMEF, achieves the state-of-the-art performance on the benchmark CoNLL 2003 Shared task. Besides, even with existing low-performance classifiers as the level-0 classifiers, our model can still achieve robust performance on the evaluated dataset.

References

1. Bansal, M., Klein, D.: Coreference semantics from web features. In: ACL (1), pp. 389–398. The Association for Computer Linguistics (2012)
2. Breiman, L.: Stacked regressions. Mach. Learn. **24**(1), 49–64 (1996)
3. Chen, H., Zhang, Y., Liu, Q.: Neural network for heterogeneous annotations. In: EMNLP, pp. 731–741. The Association for Computational Linguistics (2016)
4. Chiu, J.P.C., Nichols, E.: Named entity recognition with bidirectional LSTM-CNNs. TACL **4**, 357–370 (2016)
5. Dernoncourt, F., Lee, J.Y., Szolovits, P.: NeuroNER: an easy-to-use program for named-entity recognition based on neural networks. In: EMNLP (System Demonstrations), pp. 97–102. Association for Computational Linguistics (2017)
6. Ekbal, A., Saha, S.: Stacked ensemble coupled with feature selection for biomedical entity extraction. Knowl.-Based Syst. **46**, 22–32 (2013)
7. Finkel, J.R., Grenager, T., Manning, C.D.: Incorporating non-local information into information extraction systems by Gibbs sampling. In: ACL (2005)
8. Florian, R., Ittycheriah, A., Jing, H., Zhang, T.: Named entity recognition through classifier combination. In: CoNLL, pp. 168–171. ACL (2003)
9. Guo, J., Che, W., Wang, H., Liu, T.: Revisiting embedding features for simple semi-supervised learning. In: EMNLP, pp. 110–120. ACL (2014)
10. Krishnan, V., Manning, C.D.: An effective two-stage model for exploiting non-local dependencies in named entity recognition. In: ACL (2006)
11. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. In: HLT-NAACL (2016)

12. Ma, X., Hovy, E.H.: End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In: ACL (1). The Association for Computer Linguistics (2016)
13. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: ACL (System Demonstrations), pp. 55–60. The Association for Computer Linguistics (2014)
14. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS (2013)
15. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: EMNLP, pp. 1532–1543. ACL (2014)
16. Peters, M.E., Ammar, W., Bhagavatula, C., Power, R.: Semi-supervised sequence tagging with bidirectional language models. In: ACL (1) (2017)
17. Rajani, N.F., Mooney, R.J.: Stacking with auxiliary features. In: IJCAI (2017)
18. Ratinov, L., Roth, D.: Design challenges and misconceptions in named entity recognition. In: CoNLL, pp. 147–155. ACL (2009)
19. Sang, E.F.T.K., Meulder, F.D.: Introduction to the CoNLL-2003 shared task: language-independent named entity recognition (2003)
20. Sculley, D.: Combined regression and ranking. In: KDD, pp. 979–988. ACM (2010)
21. Sill, J., Takács, G., Mackey, L.W., Lin, D.: Feature-weighted linear stacking. CoRR abs/0911.0460 (2009)
22. Sun, W., Peng, X., Wan, X.: Capturing long-distance dependencies in sequence models: a case study of Chinese part-of-speech tagging. In: IJCNLP, pp. 180–188. Asian Federation of Natural Language Processing/ACL (2013)
23. Tsukamoto, K., Mitsuishi, Y., Sassano, M.: Learning with multiple stacking for named entity recognition. In: CoNLL. ACL (2002)
24. Wang, H., Zhao, T., Tan, H., Zhang, S.: Biomedical named entity recognition based on classifiers ensemble. IJCSA **5**(2), 1–11 (2008)
25. Wolpert, D.H.: Stacked generalization. Neural Netw. **5**(2), 241–259 (1992)
26. Wu, D., Ngai, G., Carpuat, M.: A stacked, voted, stacked model for named entity recognition. In: CoNLL, pp. 200–203. ACL (2003)
27. Zenko, B., Dzeroski, S.: Stacking with an extended set of meta-level attributes and MLR. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) ECML 2002. LNCS (LNAI), vol. 2430, pp. 493–504. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36755-1_41