Improving English Named Entity Recognition

Henry Chen, Johnny Tan
Emory University, Atlanta, Georgia
{henry.chen,johnny.tan}@emory.edu

Abstract

In recent years much of the work in named entity recognition has been focused on tackling entities in different languages or domains. However the task of English named entity recognition still remains to be solved. In this paper we explore more ways to improve the English named entity recognition system beyond just distributional semantics and the use of external gazetteer. These ways include pruning search space, increasing training data, chunked-based states, and active learning. Our experiments show we can still improve the English named entity recognition task and we find an improvement of at least .2% from our experiments.

1 Introduction

Named entity recognition is a common NLP task that is considered a sub task in relation extraction. The task is to recognize units like names, including person, organization and location names, and numeric expressions including time, date, money, and percent expressions. Standard entity recognition systems rely heavily on a statistical model using non-local features, distributional semantics, and an external knowledge gazetteer. Such systems have demonstrated to achieve up to 91% on the CoNLL 2003 test set. In recent years much of the work done in this task have been focused on twitter or web data, other languages such as Arabic, or other domains such as the medical field. This paper focuses back to improving the English named entity recognition task. In this paper we attempt to improve this task by exploring the following:

- Pruning the search space to enforce BIL tag agreement
- Aggregating more training data to improve the statistical model
- Using a chunked-based state generated from part-of-speech tagging
- Active learning experiments without using small amounts of training data

2 Related Work

Ratinov and Roth [2009] present the design challenges and fundamentals necessary in a named entity recognition system. This paper highlights the importance of non-local features and external knowledge. It achieves an improvement of 7% from the baseline of the CoNLL03 test set.

In recent years the use of neural nets to produce word embeddings have become a hot topic. Turian et al. [2010] and Zhai et al. [2016] show that the use of word representations in tasks such as named entity recognition yeilds improvements when handling unseen words.

Nothman et al. [2012] have automatically generated enormous multilingual silver standard training data by exploiting the structure of Wikipedia. They have stated that their approach performs 10% better than news wire-trained models on manually-annotated Wikipedia text and have released this corpus publicly.

Zhou and Su [2002] used a Hidden-Markov-Model that uses chunk information as features to do named entity classification. Chunk features are common in this task, and these chunk structures are usually derived from boundary categories (BIL), entity tags, or word features.

Laws and Schätze [2008] and Shen et al. [2004] have shown that the use of active learning with less hand annotated data yields nearly optimal results in NER. Laws and Schätze [2008] explores this by using stopping criterion called gradient stopping while requiring as much as 20% training data as exhaustive labeling.

3 Background and Approach

3.1 Increase Training Data

Nothman et al. [2012] provide three English corpus that consist of a gold corpus and two silver-standard [1].The gold corpus consists of a small sample of Wikipedia articles that are manually annotated. For the two generated silver standard corpus, one is called $wp2$ and it links the inference with disambiguation alias in their model. The other is $wp3$ and it links the inference with the text alias. The inference and aliases come from the structure the Wikipedia articles and the system used to generate the corpus. We experiment training each individual corpus above with our CoNLL03 training data and also training all of them together in hopes of improving our statistical model.

3.2 BILOU

The two most popular chunk or text segment schemes used for named entity recognition are BIO and BILOU. The BIO scheme suggests **B**eginning, **I**nside, and **O**utside. The BILOU scheme suggests **B**eginning, **I**nside, **L**ast, **O**utside, and **U**nit. Ratinov and Roth [2009] show that using the BILOU scheme yields better performance than the BIO scheme. They explain that by using BILOU scheme allows to learn a more expressive model with only a small increase of parameters to be learned. All our experiments use the BILOU scheme. The extra training data provided by Nothman et al. [2012] is in BIO format and requires conversion to BILOU scheme.

3.3 Word Clusters

Formed from embeddings generated by neural nets, word clusters provide valuable information to improve NLP tasks such as named entity recognition and part of speech tagging Turian et al. [2010], Zhai et al. [2016]. In our experiments we use word clusters created from Brown clustering on the New York Times corpus. The structure of these word clusters is a binary tree where each word is mapped to a bit string that represents the path to it from the root node. Brown cluster is an unsupervised learning algorithm that creates "class n-gram" models.

3.4 Gazetteer and Ambiguity Classes

A named entity gazetteer (dictionary) is extracted from DBpedia ontology with undigitalized (all numbers are replaced by 0's) regular word forms. The gazetteer is implemented in the form of a prefix tree that stores the word sequences, query frequency, potential named entity classes, and the majority tag the system has predicted. The ambiguity classes feature are created for each process state initialized. The algorithms below illustrates how the features are populated:

[1]http://schwa.org/projects/resources/wiki/Wikiner

Algorithm 1 Ambiguity class features generation

```
Require: $w_i$: word form of the $i^{th}$ token
Require: $e_i : \{(s_j, endIndex)\}$ such that $s_j$ are
  possible named entity tags that of a word sequence $w_{i \to k}$, $endIndex = k + 1$ is the exclusive index where $w_k$ is the last word of the sequence.

  $ambi_i$ sets hold all ambiguity classes of $w_i$
  for $\forall e_i$ do
    if $e_i \neq \emptyset$ then
      for $\forall (s_j, endIndex)$ in $e_i$ do
        if $endIndex == i$ then
          add all $"U-" + s_j$ to $ambi_i$
        else
          add all $"B-" + s_j$ to $ambi_i$
          add all $"I-" + s_j$ to $ambi_m$ where
          $i < m < endIndex - 1$
          add all $"L-" + s_j$ to $ambi_n$ where
          $n = endIndex - 1$
        end if
      end for
    end if
  end for
  return $ambi$
```

3.5 Adaptive Gradient Descent Learning

Our system uses a stochastic adaptive subgradient algorithm called Adagrad Duchi et al. [2011]. We train with Adagrad in mini batch and on-line fashion.

3.6 Non-Local Features

Ratinov and Roth [2009] have shown a 3% improvement from baseline using non-local features and Mao et al. have explained the importance of non-local features to help low recall from the domination of the empty class. We use two non-local features recommended by Mao et al.. They are the following:

- **Word sequence co-occurrence frequency**: Occurrence information of word sequences which capture relationships between identical candidate entities

- **Majority named entity tag prediction**: Majority label assigned to a sequence which capture relationships between identical entities and their classes

3.7 Chunk-Based States and Features

Instead of processing the word tokens in a linear, left-to-right fashion, we introduce the concept of "chunk-based state". Rather than taking individual words as an unit and extract features of itself and its surroundings, the system pre-processes a given sentence and creates chunks of words that has similar part-of-speech tags, such as (NN, NNP). The following algorithm illustrates the steps that we take to acquire the chunks:

Algorithm 2 Word sequence chunking for chunk-based state initialization

```
Require: $pos_i$: part-of-speech tag of the $i^{\text{th}}$ token

  $chunk_i$ an array holds the chunk ID of $i^{\text{th}}$ token
  $chunk_0 \leftarrow 0$
  for $\forall pos_i, i > 0$ do
    if $pos_i \approx pos_{i-1} \,||\, pos_{i-1} \approx pos_i$ then
      $chunk_i \leftarrow chunk_{i-1}$
    else
      $chunk_i \leftarrow chunk_{i-1} + 1$
    end if
  end for
  return $chunk$
```

After the chunk-based state has been initialized. The chunks provides additional collective feature of a word sequence. The following features are available from this concept:

- Chunked word forms
- Chunked part-of-speech tags
- Ambiguity class of chunked word sequence

3.8 BIL enforcement

In order to ensure complete predictions for the outputs of our system, we suppress certain predictions based on the previous prediction. By doing so, the system forces the decode output to have either I or L as the prefix of a prediction label when the previous prediction has prefix of B or I. By the same token, when the previous prediction label has either prefix of U or O, the system suppress the prediction of all labels that begin with the prefix of I or L. Such enforcement is implemented for all stages (training, development, and decode) of the system for consistency. The following algorithm demonstrates how BIL enforcement is implemented:

Algorithm 3 BIL enforcement for predicted labels

```
Require: $tag_{0 \to i}$: predicted labels of the $0^{\text{th}}$ to $i^{\text{th}}$
    where $i$ is the index of current token
Require: $score_j$: prediction scores that corresponds to $label_j$

  if $tag_{i-1}.charAt(0) \in \{'B', 'I'\}$ then
    for all $score_j$ where $laebl_j.charAt(0) \in \{'B', 'O', 'U'\}$ do
      $score_j \leftarrow -\infty$
    end for
  else if $tag_{i-1}.charAt(0) \in \{'O', 'U'\}$ then
    for all $score_j$ where $laebl_j.charAt(0) \in \{'I', 'L'\}$ do
      $score_j \leftarrow -\infty$
    end for
  end if
  return $score$
```

3.9 System Prediction Aggregation

The system automatically learns and augments its named entity gazetteer in order to capture possible unseen named entities. The system selectively learns the output of its prediction labels based on the batch size. By doing so, we hope to reduce the amount of noise that the system might potentially learn from wrong predictions. Each complete named entity prediction is served as additional input to the gazetteer. If the entry already exists, the co-occurrence frequency and the majority tag of the sequence will be updated. If not, the entry will be generated and memorized in the gazetteer.

3.10 Active Learning

Active learning is another approach we experimented as a potential area of improvements for the system, although it does not account of the official evaluation results of system performance. Instead of learning the auto prediction labels that the system output, we assume human validation for every prediction that the system makes. The system will learn the predictions that are correct by consulting the oracle. For each correct prediction, it will create additional learning instances, in which it expands its named entity dictionary if the word sequence is unseen and keeps track of the frequency count of certain labels associated to a word sequence.

4 Experiments and Results

4.1 Setup and Evaluation

We trained on CONLL'03 NER shared task data set and the corpus from Nothman et al. [2012], and the training configuration is the following: (1) Learning algorithm: adagrad-mini-batch (2) L1 Regularization: 2e-6 (3) Learning rate α : 0.02 (4) Max. epochs: 30 (5) Batch size: 5 (6) Label size: 20 (7) Feature hashing: true [size: 1,500,000] (8) Roll-in: 0.95 (9) Bias: 0. A standard F1 measure is used as the evaluation metric of our tasks. A harmonic score of both precision and recall of decode output will be used to measure the system performance.

4.2 Development

For the development sets, we compare the performances of adding named entity gazetteer (prefix tree), additional data for training, auto tag learning, and active learning. For all tests, we also develop our model with the options of BIL enforcement in additional to both left-to-right (L2R) state and Chunk-based state. The F1 measure results, see Table 1 and 2, of our trained model gives F1 scores of 93.38% and 93.41% without BIL enforcement and 93.31% and 93.42% with BIL enforcement respectively for left-to-right and chunk-based states. In general, the models when both named entity gazetteer and additional training data are used performs the best.

4.3 Evaluation

The best models from development (L2R and chunk states with prefix tree and WikiGold training data) are used for evaluation on the CoNLL'03 NER evaluation set data. The results are shown in Table 4. The best performance is given by the model using left-to-right state without BIL enforcement, in which it yields 91.57% in F1 measure. In terms of speed test, as shown in Table 5, the same model performs the best as well and gives the speed of 57,119 tokens/second and 4509 sentences/second.

4.4 Error Analysis

Table 6 shows the error analysis of forcing the BIL to be complete as opposed to not forcing it. By forcing the tags to be complete, the accuracy doesn't necessarily go up.

5 Conclusion and Error Analysis

5.1 Overall Performance

The overall performance of our system gives around a F1 measure of 93.40% (P: 94.22%, R: 92.59%) and 91.53%(P: 92.39%, R: 90.70%) for development and evaluation sets. The speed gradually decreases as we incorporate our improvements and we lose about 500 sentences per second.

5.1.1 L2R State vs. Chunk State

The left-to-right state generally gives 0.1-0.2% lower performance compared to that of chunk-based state when developing the models. Surprisingly, it outperformed the chunk-based model during evaluation. This might attribute to the fact that the features given from the chunked state over fits to the development set or the test set contains sequences that forms very different chunks from the training set.

5.1.2 BIL enforcement

By enforcing BIL tag agreement, the performance of the system actually decreases by 0.1% F1 measure in average. This is particularly interesting since intuitively by ensuring the completeness of predicted named entity chunk, the system should theoretically provide better results. We further investigate on this phenomena. In table 6, we see that through BIL enforcement, the system does have higher percentages of predicted tag completeness. There is nearly no prediction that are not completed. On the contrary, the percentages of completed tags are slightly lower compared to that of ones with BIL enforcement. This can be attributed to two main factors: (1) Although BIL enforcement suppresses the predictions and prunes the search space of undesirable labels, the enforcement is simply the a rule-based approach, thus the suppressed labels are not learned by the optimizer. (2) Since the pruned search space is not learned by the optimizer, there is a high chance that a narrower list of valid prediction might cause confusion for the recognizer and results in a wrong classification. Ultimately, the enforcement yields worse performance in general.

5.2 Active Learning

Instead of using less training data and showing that our model improves from active learning as it goes, we try to simulate what would happen if our model learned "human" corrected annotations. By learning these corrections, these results can't be compared to the evaluation of CoNLL03 task therefore we analyze them separately. The results from our active learning show an improvement of 2-3% on top of the highest scores received for each data set. Although we now see that active learning does help, it is unclear if the model is just overfitting to our test and development sets.

5.3 Additional Training Data

We find that the silver standard corpus generated by Nothman et al. [2012] do not help our model and actually hurt it. Although it can be considered to be near optimal performance of using manually annotated data the increase in training time due to the size of the silver corpus is not a good trade-off. However the gold corpus or the small sample of manually annotated provided do not slow down the training time due its size and actually improves the performance in both the test and development sets. The addition of more training data regardless of size increases training time and doesn't always necessarily improve performance.

Acknowledgments

We acknowledge Dr. Jinho D. Choi for providing valuable training, development, and test data for our experiments in this paper. Also, we acknowledge the fact that our parser is built on top of the infrastructure of EmoryNLP, founded and maintained by Dr. Jinho D. Choi at Emory University.

References

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

Florian Laws and Hinrich Schätze. Stopping criteria for active learning of named entity recognition. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 465–472. Association for Computational Linguistics, 2008.

Xinnian Mao, Wei Xu, Yuan Dong, Saike He, and Haila Wang. Using non-local features to improve named entity recognition recall.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. Learning multilingual named entity recognition from Wikipedia. *Artificial Intelligence*, 194:151–175, 2012. doi: 10.1016/j.artint.2012.03.006. URL `http://dx.doi.org/10.1016/j.artint.2012.03.006`.

Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics, 2009.

Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew-Lim Tan. Multi-criteria-based active learning for named entity recognition. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 589. Association for Computational Linguistics, 2004.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

Mike Zhai, Johnny Tan, and Jinho Choi. Intrinsic and extrinsic evaluations of word embeddings. Association for Advancement of Artifcial Intelligence,2016, 2016.

GuoDong Zhou and Jian Su. Named entity recognition using an hmm-based chunk tagger. In *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 473–480. Association for Computational Linguistics, 2002.

Approaches	L2R State			Chunk State		
	P	R	F1	P	R	F1
Baseline	93.54	91.82	92.67	93.65	92.06	92.85
+Prefix tree	93.96	92.28	93.16	94.09	92.51	93.30
+WikiGold data	94.26	92.53	**93.38**	94.18	92.65	**93.41**
+Auto tag learning	94.15	92.44	93.29	94.02	92.41	93.21
+Active learning	96.66	95.84	96.25	96.68	95.96	96.32

Table 1: Development result (without BIL-enforced predictions)

Approaches	L2R State			Chunk State		
	P	R	F1	P	R	F1
Baseline	93.13	91.96	92.54	93.32	92.17	92.74
+Prefix tree	93.73	92.54	93.13	92.54	92.54	93.23
+WikiGold data	93.97	92.65	**93.31**	94.05	92.80	**93.42**
+Auto tag learning	93.96	92.70	93.32	93.83	92.65	93.23
+Active learning	95.27	94.48	94.87	95.38	94.85	95.11

Table 2: Development result (with BIL-enforced predictions)

Approaches	L2R State			Chunk State		
	P	R	F1	P	R	F1
Baseline	92.07	90.31	91.18	92.15	90.39	91.26
+Wp2	90.40	88.58	89.48	90.47	88.72	89.59
+Wp3	90.47	88.76	89.61	90.38	88.70	89.53
+WikiGold(2415)	92.39	90.75	**91.56**	92.34	90.65	**91.49**
+All	89.90	87.89	88.83	89.84	87.95	88.88

Table 3: Development result (with only additional training data)

Approaches	L2R State			Chunk State		
	P	R	F1	P	R	F1
Regular	92.45	90.70	**91.57**	92.33	90.71	**91.51**
BIL-enforced	91.98	90.80	91.39	92.03	90.77	91.39

Table 4: Evaluation result on evaluation set

Approaches	L2R State		Chunk State	
	Token/sec	Sent./sec	Token/sec	Sent./sec
Regular	**57119**	**4509**	51679	4080
BIL-enforced	40685	3212	34770	2755

Table 5: Speed test comparison

Approaches	Non-enforced predictions		BIL-enforced predictions	
	Complete	Total	Complete	Total
Baseline	5854	5880	5867	5868
+Prefix tree	5860	5876	5867	5868
+WikiGold data	5852	5872	5862	5863
+Auto tag learning	5852	5872	5862	5863
+Active learning	5903	5919	5935	5936

Table 6: Result Comparison of BIL-enforced Prediction Agreement