Instance-Based Learning of Span Representations: A Case Study through Named Entity Recognition

Hiroki Ouchi[1,2] **Jun Suzuki**[2,1] **Sosuke Kobayashi**[2,3]
Sho Yokoi[2,1] **Tatsuki Kuribayashi**[2,4] **Ryuto Konno**[2] **Kentaro Inui**[2,1]
[1] RIKEN [2] Tohoku University [3] Preferred Networks, Inc. [4] Langsmith, Inc.
hiroki.ouchi@riken.jp
{jun.suzuki,sosk,yokoi,kuribayashi,ryuto,inui}@ecei.tohoku.ac.jp

Abstract

Interpretable rationales for model predictions play a critical role in practical applications. In this study, we develop models possessing interpretable inference process for structured prediction. Specifically, we present a method of instance-based learning that learns similarities between spans. At inference time, each span is assigned a class label based on its similar spans in the training set, where it is easy to understand how much each training instance contributes to the predictions. Through empirical analysis on named entity recognition, we demonstrate that our method enables to build models that have high interpretability without sacrificing performance.

1 Introduction

Neural networks have contributed to performance improvements in structured prediction. Instead, the rationales underlying the model predictions are difficult for humans to understand (Lei et al., 2016). In practical applications, interpretable rationales play a critical role for driving human's decisions and promoting human-machine cooperation (Ribeiro et al., 2016). With this motivation, we aim to build models that have high interpretability without sacrificing performance. As an approach to this challenge, we focus on *instance-based learning*.

Instance-based learning (Aha et al., 1991) is a machine learning method that learns similarities between instances. At inference time, the class labels of the most similar training instances are assigned to the new instances. This transparent inference process provides an answer to the following question: *Which points in the training set most closely resemble a test point or influenced the prediction?* This is categorized into *example-based explanations* (Plumb et al., 2018; Baehrens et al., 2010). Recently, despite its preferable property, it has received little attention and been underexplored.

This study presents and investigates an instance-based learning method for *span representations*. A span is a unit that consists of one or more linguistically linked words. *Why do we focus on spans instead of tokens?* One reason is relevant to performance. Recent neural networks can induce good span feature representations and achieve high performance in structured prediction tasks, such as named entity recognition (NER) (Sohrab and Miwa, 2018; Xia et al., 2019), constituency parsing (Stern et al., 2017; Kitaev et al., 2019), semantic role labeling (SRL) (He et al., 2018; Ouchi et al., 2018) and coreference resolution (Lee et al., 2017). Another reason is relevant to interpretability. The tasks above require recognition of linguistic structure that consists of spans. Thus, directly classifying each span based on its representation is more interpretable than token-wise classification such as BIO tagging, which reconstructs each span label from the predicted token-wise BIO tags.

Our method builds a feature space where spans with the same class label are close to each other. At inference time, each span is assigned a class label based on its neighbor spans in the feature space. We can easily understand why the model assigned the label to the span by looking at its neighbors. Through quantitative and qualitative analysis on NER, we demonstrate that our instance-based method enables to build models that have high interpretability and performance. To sum up, our main contributions are as follows.

- This is the first work to investigate instance-based learning of span representations.[1]
- Through empirical analysis on NER, we demonstrate our instance-based method enables to build models that have high interpretability without sacrificing performance.

[1]Our code is publicly available at https://github.com/hiroki13/instance-based-ner.git.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 6452–6459

2 Related Work

Neural models generally have a common technical challenge: the black-box property. The rationales underlying the model predictions are opaque for humans to understand. Many recent studies have tried to look into classifier-based neural models (Ribeiro et al., 2016; Lundberg and Lee, 2017; Koh and Liang, 2017). In this paper, instead of looking into the black-box, we build interpretable models based on instance-based learning.

Before the current neural era, instance-based learning, sometimes called memory-based learning (Daelemans and Van den Bosch, 2005), was widely used for various NLP tasks, such as part-of-speech tagging (Daelemans et al., 1996), dependency parsing (Nivre et al., 2004) and machine translation (Nagao, 1984). For NER, some instance-based models have been proposed (Tjong Kim Sang, 2002; De Meulder and Daelemans, 2003; Hendrickx and van den Bosch, 2003). Recently, despite its high interpretability, this direction has not been explored.

One exception is Wiseman and Stratos (2019), which used instance-based learning of token representations. Due to BIO tagging, it faces one technical challenge: inconsistent label prediction. For example, an entity candidate "World Health Organization" can be assigned inconsistent labels such as "`B-LOC I-ORG I-ORG`," whereas the ground-truth labels are "`B-ORG I-ORG I-ORG`." To remedy this issue, they presented a heuristic technique for encouraging contiguous token alignment. In contrast to such token-wise prediction, we adopt span-wise prediction, which can naturally avoid this issue because each span is assigned one label.

NER is generally solved as (i) sequence labeling or (ii) span classification.[2] In the first approach, token features are induced by using neural networks and fed into a classifier, such as conditional random fields (Lample et al., 2016; Ma and Hovy, 2016; Chiu and Nichols, 2016). One drawback of this approach is the difficulty dealing with nested entities.[3] By contrast, the span classification approach, adopted in this study, can straightforwardly solve nested NER (Finkel and Manning, 2009; Sohrab and Miwa, 2018; Xia et al., 2019).[4]

[2]Very recently, a hybrid model of these two approaches has been proposed by Liu et al. (2019).

[3]Some studies have sophisticated sequence labeling models for nested NER (Ju et al., 2018; Zheng et al., 2019).

[4]There is an approach specialized for nested NER using hypergraphs (Lu and Roth, 2015; Muis and Lu, 2017; Katiyar and Cardie, 2018; Wang and Lu, 2018).

3 Instance-Based Span Classification

3.1 NER as span classification

NER can be solved as multi-class classification, where each of possible spans in a sentence is assigned a class label. As we mentioned in Section 2, this approach can naturally avoid inconsistent label prediction and straightforwardly deal with nested entities. Because of these advantages over token-wise classification, span classification has been gaining a considerable attention (Sohrab and Miwa, 2018; Xia et al., 2019).

Formally, given an input sentence of T words $X = (w_1, w_2, \ldots, w_T)$, we first enumerate possible spans $\mathcal{S}(X)$, and then assign a class label $y \in \mathcal{Y}$ to each span $s \in \mathcal{S}(X)$. We will write each span as $s = (a, b)$, where a and b are word indices in the sentence: $1 \leq a \leq b \leq T$. Consider the following sentence.

$$\begin{array}{cccccc} \text{Franz}_1 & \text{Kafka}_2 & \text{is}_3 & \text{a}_4 & \text{novelist}_5 \\ [& \texttt{PER} &] & & \end{array}$$

Here, the possible spans in this sentence are $\mathcal{S}(X) = \{(1,1), (1,2), (1,3), \ldots, (4,5), (5,5)\}$. "Franz Kafka," denoted as $s = (1, 2)$, is assigned the person type entity label ($y = \texttt{PER}$). Note that the other non-entity spans are assigned the null label ($y = \texttt{NULL}$). For example, "a novelist," denoted as $s = (4, 5)$, is assigned `NULL`. In this way, the `NULL` label is assigned to non-entity spans, which is the same as the `O` tag in the BIO tag set.

The probability that each span s is assigned a class label y is modeled by using softmax function:

$$\mathrm{P}(y|s) = \frac{\exp(\mathrm{score}(s, y))}{\sum_{y' \in \mathcal{Y}} \exp(\mathrm{score}(s, y'))} \ .$$

Typically, as the scoring function, the inner product between each label weight vector $\mathbf{w}_y$ and span feature vector $\mathbf{h}_s$ is used:

$$\mathrm{score}(s, y) = \mathbf{w}_y \cdot \mathbf{h}_s \ .$$

The score for the `NULL` label is set to a constant, $\mathrm{score}(s, y = \texttt{NULL}) = 0$, similar to logistic regression (He et al., 2018). For training, the loss function we minimize is the negative log-likelihood:

$$\mathcal{L} = -\sum_{(X,Y) \in \mathcal{D}} \sum_{(s,y) \in \mathcal{S}(X,Y)} \log \mathrm{P}(y|s) \ ,$$

where $\mathcal{S}(X, Y)$ is a set of pairs of a span s and its ground-truth label y. We call this kind of models that use label weight vectors for classification *classifier-based span model*.

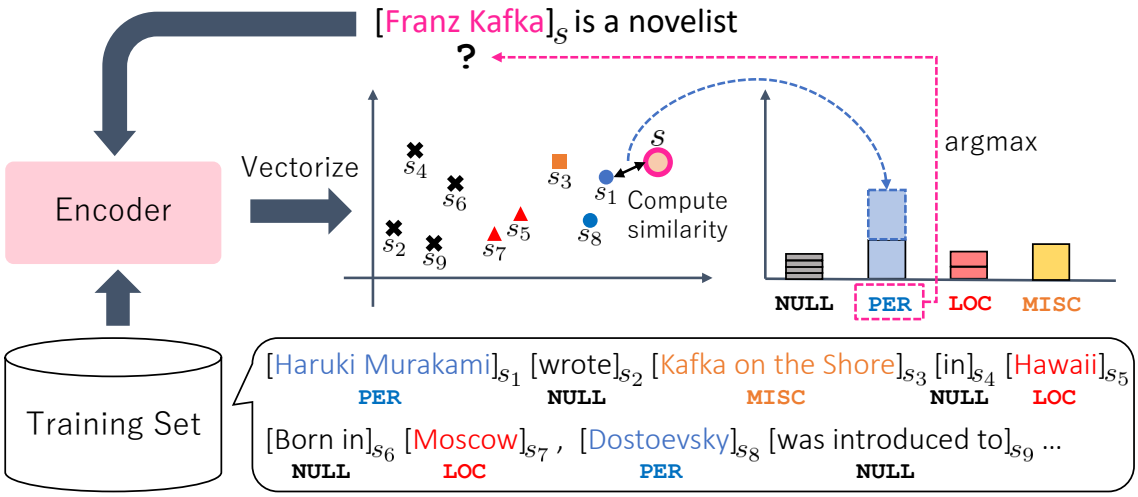


Figure 1: Illustration of our instance-based span model. An entity candidate "Franz Kafka" is used as a query and vectorized by an encoder. In the vector space, similarities between all pairs of the candidate (s) and the training instances ($s_1, s_2, \ldots, s_9$) are computed, respectively. Based on the similarities, the label probability (distribution) is computed, and the label with the highest probability PER is assigned to "Franz Kafka."

3.2 Instance-based span model

Our *instance-based span model* classifies each span based on similarities between spans. In Figure 1, an entity candidate "Franz Kafka" and the spans in the training set are mapped onto the feature vector space, and the label distribution is computed from the similarities between them. In this inference process, it is easy to understand how much each training instance contributes to the predictions. This property allows us to explain the predictions by specific training instances, which is categorized into *example-based explanations* (Plumb et al., 2018).

Formally, within the neighbourhood component analysis framework (Goldberger et al., 2005), we define the *neighbor span probability* that each span $s_i \in \mathcal{S}(X)$ will select another span s_j as its neighbor from candidate spans in the training set:

$$\mathrm{P}(s_j|s_i, \mathcal{D}') = \frac{\exp(\mathrm{score}(s_i, s_j))}{\sum_{s_k \in \mathcal{S}(D')} \exp(\mathrm{score}(s_i, s_k))} \ . \quad (1)$$

Here, we exclude the input sentence X and its ground-truth labels Y from the training set $\mathcal{D}$: $\mathcal{D}' = \mathcal{D} \setminus \{(X, Y)\}$, and regard all other spans as candidates: $\mathcal{S}(\mathcal{D}') = \{s \in \mathcal{S}(X')|(X', Y') \in \mathcal{D}'\}$. The scoring function returns a similarity between the spans s_i and s_j. Then we compute the probability that a span s_i will be assigned a label y_i:

$$\mathrm{P}(y_i|s_i) = \sum_{s_j \in \mathcal{S}(\mathcal{D}', y_i)} \mathrm{P}(s_j|s_i, \mathcal{D}') \ . \quad (2)$$

Here, $\mathcal{S}(\mathcal{D}', y_i) = \{s_j \in \mathcal{D}' | \ y_i = y_j\}$, so the equation indicates that we sum up the probabilities of the neighbor spans that have the same label as the span s_i. The loss function we minimize is the negative log-likelihood:

$$\mathcal{L} = - \sum_{(X,Y) \in \mathcal{D}} \sum_{(s_i, y_i) \in \mathcal{S}(X,Y)} \log \mathrm{P}(y_i|s_i) \ ,$$

where $\mathcal{S}(X, Y)$ is a set of pairs of a span s_i and its ground-truth label y_i. At inference time, we predict $\hat{y}_i$ to be the class label with maximal marginal probability:

$$\hat{y}_i = \arg\max_{y \in \mathcal{Y}} \mathrm{P}(y|s_i) \ ,$$

where the probability $\mathrm{P}(y|s_i)$ is computed for each of the label set $y \in \mathcal{Y}$.

Efficient neighbor probability computation The neighbor span probability $\mathrm{P}(s_j|s_i, \mathcal{D}')$ in Equation 1 depends on the entire training set $\mathcal{D}'$, which leads to heavy computational cost. As a remedy, we use random sampling to retrieve K sentences $\mathcal{D}'' = \{(X'_k, Y'_k)\}_{k=0}^{K}$ from the training set $\mathcal{D}'$. At training time, we randomly sample K sentences for each mini-batch at each epoch. This simple technique realizes time and memory efficient training. In our experiments, it takes less than one day to train a model on a single GPU[5].

[5]NVIDIA DGX-1 with Tesla V100.

4 Experiments

4.1 Experimental setup

Data We evaluate the span models through two types of NER: (i) flat NER on the CoNLL-2003 dataset (Tjong Kim Sang and De Meulder, 2003) and (ii) nested NER on the GENIA dataset[6] (Kim et al., 2003). We follow the standard training-development-test splits.

Baseline We use a classifier-based span model (Section 3.1) as a baseline. Only the difference between the instance-based and classifier-based span models is whether to use softmax classifier or not.

Encoder and span representation We adopt the encoder architecture proposed by Ma and Hovy (2016), which encodes each token of the input sentence $w_t \in X$ with word embedding and character-level CNN. The encoded token representations $\mathbf{w}_{1:T} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_T)$ are fed to bidirectional LSTM for computing contextual ones $\overrightarrow{\mathbf{h}}_{1:T}$ and $\overleftarrow{\mathbf{h}}_{1:T}$. From them, we create $\mathbf{h}_s^{\text{lstm}}$ for each span $s = (a, b)$ based on LSTM-minus (Wang and Chang, 2016). For flat NER, we use the representation $\mathbf{h}_s^{\text{lstm}} = [\overrightarrow{\mathbf{h}}_b - \overrightarrow{\mathbf{h}}_{a-1}, \overleftarrow{\mathbf{h}}_a - \overleftarrow{\mathbf{h}}_{b+1}]$. For nested NER, we use $\mathbf{h}_s^{\text{lstm}} = [\overrightarrow{\mathbf{h}}_b - \overrightarrow{\mathbf{h}}_{a-1}, \overleftarrow{\mathbf{h}}_a - \overleftarrow{\mathbf{h}}_{b+1}, \overrightarrow{\mathbf{h}}_a + \overrightarrow{\mathbf{h}}_b, \overleftarrow{\mathbf{h}}_a + \overleftarrow{\mathbf{h}}_b]$.[7] We then multiply $\mathbf{h}_s^{\text{lstm}}$ with a weight matrix $\mathbf{W}$ and obtain the span representation: $\mathbf{h}_s = \mathbf{W}\, \mathbf{h}_s^{\text{lstm}}$. For the scoring function in Equation 1 in the instance-based span model, we use the inner product between a pair of span representations: $\text{score}(s_i, s_j) = \mathbf{h}_{s_i} \cdot \mathbf{h}_{s_j}$.

Model configuration We train instance-based models by using $K = 50$ training sentences randomly retrieved for each mini-batch. At test time, we use $K = 50$ nearest training sentences for each sentence based on the cosine similarities between their sentence vectors[8]. For the word embeddings, we use the GloVe 100-dimensional embeddings (Pennington et al., 2014) and the BERT embeddings (Devlin et al., 2019).[9]

[6] We use the same one pre-processed by Zheng et al. (2019) at https://github.com/thecharm/boundary-aware-nested-ner

[7] We use the different span representation from the one used for flat NER because concatenating the addition features, $\overrightarrow{\mathbf{h}}_a + \overrightarrow{\mathbf{h}}_b$ and $\overleftarrow{\mathbf{h}}_a + \overleftarrow{\mathbf{h}}_b$, to the subtraction features improves performance in our preliminary experiments.

[8] For each sentence $X = (w_1, w_2, \ldots, w_T)$, its sentence vector is defined as the vector averaged over the word embeddings (GloVe) within the sentence: $\frac{1}{T}\sum_t \mathbf{w}_t^{\text{emb}}$.

[9] Details on the experimental setup are described in Appendices A.1.

	Classifier-based	Instance-based
	GloVe	
Flat NER	90.68 ±0.25	90.73 ±0.07
Nested NER	73.76 ±0.35	74.20 ±0.16
	BERT	
Flat NER	90.48 ±0.18	90.48 ±0.07
Nested NER	73.27 ±0.19	73.92 ±0.20

Table 1: Comparison between classifier-based and instance-based span models. Cells show the F_1 scores and standard deviations on each test set.

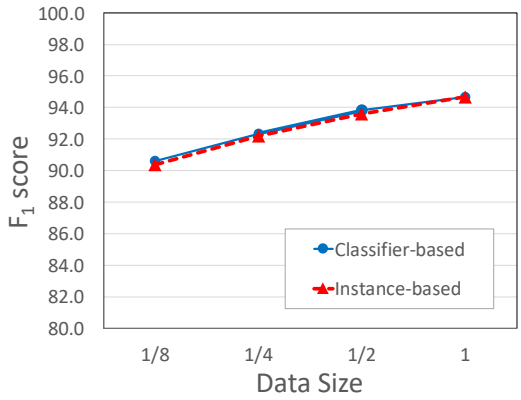


Figure 2: Performance on the CoNLL-2003 development set for different amounts of the training set.

4.2 Quantitative analysis

We report averaged F_1 scores across five different runs of the model training with random seeds.

Overall F_1 scores We investigate whether or not our instance-based span model can achieve competitive performance with the classifier-based span model. Table 1 shows F_1 scores on each test set.[10] Consistently, the instance-based span model yielded comparable results to the classifier-based span model. This indicates that our instance-based learning method enables to build NER models without sacrificing performance.

Effects of training data size Figure 2 shows F_1 scores on the CoNLL-2003 development set by the models trained on full-size, $1/2$, $1/4$ and $1/8$ of the training set. We found that (i) performance of both models gradually degrades when the size of the training set is smaller and (ii) both models yield very competitive performance curves.

[10] The models using GloVe yielded slightly better results than those using BERT. One possible explanation is that subword segmentation is not so good for NER. In particular, tokens in upper case are segmented into too small elements, e.g., "LEICESTERSHIRE" → "L," "##EI," "##CE," "##ST," "##ER," "##S," "##H," "##IR," "##E."

	QUERY	... [Tom Moody] took six for 82 but ...
	Classifier-based	
1	PER	... [Billy Mayfair] and Paul Goydos and ...
2	NULL	... [Billy Mayfair and Paul Goydos] and ...
3	NULL	... [Billy Mayfair and Paul Goydos and] ...
4	NULL	... [Billy] Mayfair and Paul Goydos and ...
5	NULL	... [Ducati rider Troy Corser] , last year ...
	Instance-based	
1	PER	[Ian Botham] began his test career ...
2	PER	... [Billy Mayfair] and Paul Goydos and ...
3	PER	... [Mark Hutton] scattered four hits ...
4	PER	... [Steve Stricker] , who had a 68 , and ...
3	PER	... [Darren Gough] polishing off ...

Table 2: Example of span retrieval. An entity candidate "Tom Moody" in the CoNLL-2003 development set used as a query for retrieving five nearest neighbors from the training set.

	QUERY	... spokesman for [Air France] 's ... Pred: LOC Gold: ORG
1	LOC	... [Colombia] turned down American 's ...
2	LOC	... involving [Scotland] , Wales , ...
3	LOC	... signed in [Nigeria] 's capital Abuja ...
4	LOC	... in the West Bank and [Gaza] .
5	LOC	... on its way to [Romania] ...

Table 3: Example of an error by the instance-based span model. Although the gold label is ORG (Organization), the wrong label LOC (Location) is assigned.

4.3 Qualitative analysis

To better understand model behavior, we analyze the instance-based model using GloVe in detail.

Examples of retrieved spans The span feature space learned by our method can be applied to various downstream tasks. In particular, it can be used as a span retrieval system. Table 2 shows five nearest neighbor spans of an entity candidate "Tom Moody." In the classifier-based span model, person-related but non-entity spans were retrieved. By contrast, in the instance-based span model, person (PER) entities were consistently retrieved.[11] This tendency was observed in many other cases, and we confirmed that our method can build preferable feature spaces for applications.

Errors analysis The instance-based span model tends to wrongly label spans that includes location or organization names. For example, in Table 3, the wrong label LOC (Location) is assigned to "Air France" whose gold label is ORG (Organization).

[11]The query span "Tom moody" was a cricketer at that time, and some neighbors, "Ian Botham" and "Darren Gough," were also cricketers.

	Classifier-based	Instance-based
GloVe	94.91 $\pm$0.11	94.96 $\pm$0.06
BERT	96.20 $\pm$0.03	96.24 $\pm$0.04

Table 4: Comparison in syntactic chunking. Cells show F_1 and standard deviations on the CoNLL-2000 test set.

Note that by looking at the neighbors, we can understand that country or district entities confused the model. This implies that prediction errors are easier to analyze because the neighbors are the rationales of the predictions.

4.4 Discussion

Generalizability *Are our findings in NER generalizable to other tasks?* To investigate it, we perform an additional experiment on the CoNLL-2000 dataset (Tjong Kim Sang and Buchholz, 2000) for syntactic chunking.[12] While this task is similar to NER in terms of short-span classification, the class labels are based on syntax, not (entity) semantics. In Table 4, the instance-based span model achieved competitive F_1 scores with the classifier-based one, which is consistent with the NER results. This suggests that our findings in NER are likely to generalizable to other short-span classification tasks.

Future work One interesting line of future work is an extension of our method to span-to-span relation classification, such as SRL and coreference resolution. Another potential direction is to apply and evaluate learned span features to downstream tasks requiring entity knowledge, such as entity linking and question answering.

5 Conclusion

We presented and investigated an instance-based learning method that learns similarity between spans. Through NER experiments, we demonstrated that the models build by our method have (i) competitive performance with a classifier-based span model and (ii) interpretable inference process where it is easy to understand how much each training instance contributes to the predictions.

Acknowledgments

This work was partially supported by JSPS KAKENHI Grant Number JP19H04162 and JP19K20351. We would like to thank the members of Tohoku NLP Laboratory and the anonymous reviewers for their insightful comments.

[12]The models are trained in the same way as in nested NER.

References

David W Aha, Dennis Kibler, and Marc K Albert. 1991. Instance-based learning algorithms. *Machine learning*, 6(1):37–66.

David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert MÃžller. 2010. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Walter Daelemans and Antal Van den Bosch. 2005. *Memory-based language processing*. Cambridge University Press.

Walter Daelemans, Jakub Zavrel, Peter Berck, and Steven Gillis. 1996. MBT: A memory-based part of speech tagger-generator. In *Proceedings of Fourth Workshop on Very Large Corpora*.

Fien De Meulder and Walter Daelemans. 2003. Memory-based named entity recognition using unannotated data. In *Proceedings of HLT-NAACL*, pages 208–211.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Jenny Rose Finkel and Christopher D. Manning. 2009. Nested named entity recognition. In *Proceedings of EMNLP*, pages 141–150.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AISTATS*, pages 249–256.

Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Ruslan R Salakhutdinov. 2005. Neighbourhood components analysis. In *Proceedings of NIPS*, pages 513–520.

Alan Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *Proceedings of Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop*.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of ACL*, pages 364–369.

Iris Hendrickx and Antal van den Bosch. 2003. Memory-based one-step named-entity recognition: Effects of seed list features, classifier stacking, and unannotated data. In *Proceedings of CoNLL*, pages 176–179.

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A neural layered model for nested named entity recognition. In *Proceedings of NAACL-HLT*, pages 1446–1459.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. In *Proceedings of NAACL-HLT*, pages 861–871.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Junichi Tsujii. 2003. Genia corpusa semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182.

D.P. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv: 1412.6980*.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of ACL*, pages 3499–3505.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of ICML*, pages 1885–1894.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of EMNLP*, pages 188–197.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of EMNLP*, pages 107–117.

Tianyu Liu, Jin-Ge Yao, and Chin-Yew Lin. 2019. Towards improving neural named entity recognition with gazetteers. In *Proceedings of ACL*, pages 5301–5307.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of EMNLP*, pages 857–867.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of NIPS*, pages 4765–4774.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of ACL*, pages 1064–1074.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Aldrian Obaja Muis and Wei Lu. 2017. Labeling gaps between words: Recognizing overlapping mentions with mention separators. In *Proceedings of EMNLP*, pages 2608–2618.

Makoto Nagao. 1984. *A framework of a mechanical translation between Japanese and English by analogy principle*. Elsevier Science Publishers.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of CoNLL*, pages 49–56, Boston, Massachusetts, USA.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2018. A span selection model for semantic role labeling. In *Proceedings of EMNLP*, pages 1630–1642.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of ICML*, pages 1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP)*, pages 1532–1543.

Gregory Plumb, Denali Molitor, and Ameet S Talwalkar. 2018. Model agnostic supervised local explanations. In *Proceedings of NIPS*, pages 2515–2524.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of KDD*, pages 1135–1144.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Mohammad Golam Sohrab and Makoto Miwa. 2018. Deep exhaustive model for nested named entity recognition. In *Proceedings of EMNLP*, pages 2843–2849.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of ACL*, pages 818–827.

Erik F. Tjong Kim Sang. 2002. Memory-based named entity recognition. In *Proceedings of CoNLL)*.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task chunking. In *Proceedings of CoNLL*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL*, pages 142–147.

Bailin Wang and Wei Lu. 2018. Neural segmental hypergraphs for overlapping mention recognition. In *Proceedings of EMNLP*, pages 204–214.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional LSTM. In *Proceedings of ACL*, pages 2306–2315.

Sam Wiseman and Karl Stratos. 2019. Label-agnostic sequence labeling by copying nearest neighbors. In *Proceedings of ACL*, pages 5363–5369.

Congying Xia, Chenwei Zhang, Tao Yang, Yaliang Li, Nan Du, Xian Wu, Wei Fan, Fenglong Ma, and Philip Yu. 2019. Multi-grained named entity recognition. In *Proceedings of ACL*, pages 1430–1440.

Changmeng Zheng, Yi Cai, Jingyun Xu, Ho-fung Leung, and Guandong Xu. 2019. A boundary-aware neural model for nested named entity recognition. In *Proceedings of EMNLP-IJCNLP*, pages 357–366.

A Appendices

A.1 Experimental setup

Name	Value
CNN window size	3
CNN filters	30
BiLSTM layers	2
BiLSTM hidden units	100 dimensions
Mini-batch size	8
Optimization	Adam
Learning rate	0.001
Dropout ratio	{0.1, 0.3, 0.5}

Table 5: Hyperparameters used in the experiments.

Network setup Basically, we follow the encoder architecture proposed by Ma and Hovy (2016). First, the token-encoding layer encodes each token of the input sentence $w_t \in (w_1, w_2, \ldots, w_T)$ to a sequence of the vector representations $\mathbf{w}_{1:T} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_T)$. For the models using GloVe, we use the GloVe 100-dimensional embeddings[13] (Pennington et al., 2014) and character-level CNN. For the models using BERT, we use the BERT-Base, Cased[14] (Devlin et al., 2019), where we use the first subword embeddings within each token in the last layer of BERT. During training, we fix the word embeddings (except the CNN). Then, the encoded token representations $\mathbf{w}_{1:T} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_T)$ are fed to bidirectional LSTM (BiLSTM) (Graves et al., 2013) for computing contextual ones $\overrightarrow{\mathbf{h}}_{1:T}$ and $\overleftarrow{\mathbf{h}}_{1:T}$. We use 2 layers of the stacked BiLSTMs (2 forward and 2 backward LSTMs) with 100-dimensional hidden units. From $\overrightarrow{\mathbf{h}}_{1:T}$ and $\overleftarrow{\mathbf{h}}_{1:T}$, we create $\mathbf{h}^{\text{lstm}}_s$ for each span $s = (a, b)$ based on LSTM-minus (Wang and Chang, 2016). For flat NER, we use the representation $\mathbf{h}^{\text{lstm}}_s = [\overrightarrow{\mathbf{h}}_b - \overrightarrow{\mathbf{h}}_{a-1}, \overleftarrow{\mathbf{h}}_a - \overleftarrow{\mathbf{h}}_{b+1}]$. For nested NER, we use $\mathbf{h}^{\text{lstm}}_s = [\overrightarrow{\mathbf{h}}_b - \overrightarrow{\mathbf{h}}_{a-1}, \overleftarrow{\mathbf{h}}_a - \overleftarrow{\mathbf{h}}_{b+1}, \overrightarrow{\mathbf{h}}_a + \overrightarrow{\mathbf{h}}_b, \overleftarrow{\mathbf{h}}_a + \overleftarrow{\mathbf{h}}_b]$. We then multiply $\mathbf{h}^{\text{lstm}}_s$ with a weight matrix $\mathbf{W}$ and obtain the span representation: $\mathbf{h}_s = \mathbf{W}\,\mathbf{h}^{\text{lstm}}_s$. Finally, we use the span representation $\mathbf{h}_s$ for computing the label distribution in each model. For efficient computation, following Sohrab and Miwa (2018), we enumerate all possible spans in a sentence with the sizes less than or equal to the maximum span size L, i.e., each span $s = (a, b)$ is satisfied with the condition $b - a < L$. We set L as 6.

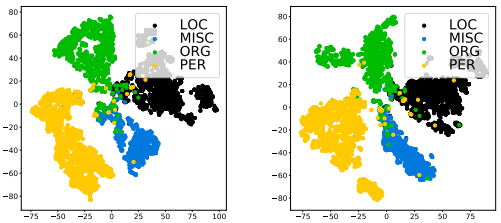
[13] https://nlp.stanford.edu/projects/glove/

[14] https://github.com/google-research/bert

Hyperparameters Table 5 lists the hyperparameters used in the experiments. We initialize all the parameter matrices in BiLSTMs with random orthonormal matrices (Saxe et al., 2013). Other parameters are initialized following Glorot and Bengio (2010). We apply dropout (Srivastava et al., 2014) to the token-encoding layer and the input vectors of each LSTM with dropout ratio of $\{0.1, 0.3, 0.5\}$.

Optimization To optimize the parameters, we use Adam (Kingma and Ba, 2014) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The initial learning rate is set to $\eta_0 = 0.001$. The learning rate is updated on each epoch as $\eta_t = \eta_0/(1 + \rho t)$, where the decay rate is $\rho = 0.05$ and t is the number of epoch completed. A gradient clipping value is set to 5.0 (Pascanu et al., 2013). Parameter updates are performed in mini-batches of 8. The number of training epochs is set to 100. We save the parameters that achieve the best F1 score on each development set and evaluated them on each test set. Training the models takes less than one day on a single GPU, NVIDIA DGX-1 with Tesla V100.

A.2 Feature space visualization



(a) Classifier-based (b) Instance-based

Figure 3: Visualization of entity span features computed by classifier-based and instance-based models.

To better understand span representations learned by our method, we observe the feature space. Specifically, we visualize the span representations $\mathbf{h}_s$ on the CoNLL-2003 development set. Figure 3 visualizes two-dimensional entity span representations by t-distributed Stochastic Neighbor Embedding (t-SNE) (Maaten and Hinton, 2008). Both models successfully learned feature spaces where the instances with the same label come close each other.