An Empirical Study on Finding Spans

Weiwei Gu[1*] **Boyuan Zheng**[2*]
Yunmo Chen[2] **Tongfei Chen**[3] **Benjamin Van Durme**[2]
[1] University of Rochester [2] Johns Hopkins University
[3] Microsoft Semantic Machines
wgu7@ur.rochester.edu, {bzheng12,yunmo,vandurme}@jhu.edu, tongfei@pm.me

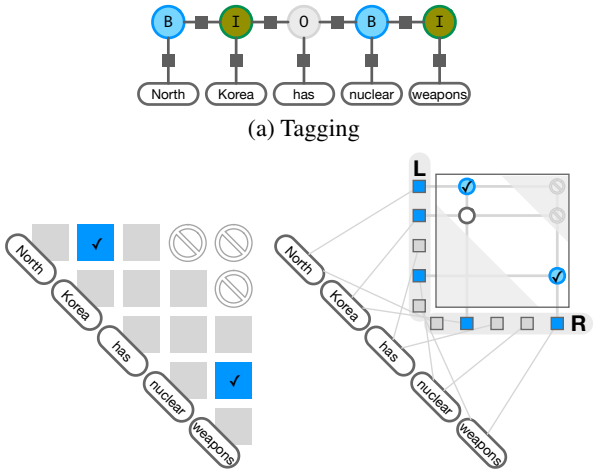
Abstract

We present an empirical study on methods for *span finding*, the selection of consecutive tokens in text for some downstream tasks. We focus on approaches that can be employed in training end-to-end information extraction systems, and find there is no definitive solution without considering task properties, and provide our observations to help with future design choices: 1) a tagging approach often yields higher precision while span enumeration and boundary prediction provide higher recall; 2) span type information can benefit a boundary prediction approach; 3) additional contextualization does not help span finding in most cases.

1 Introduction

Various information extraction (IE) tasks require a *span finding* component, which either directly yields the output or serves as an essential component of downstream linking. In named entity recognition (NER), spans in text are detected and typed; coreference resolution (RE) requires mention spans; mention spans are linked when performing relation extraction (RE), and in event extraction this also requires detection of *trigger spans*. In extractive question answering (QA), a span in a passage is detected to be presented as the answer to a given question.

Following the proliferation of large pre-trained models (Peters et al., 2018; Devlin et al., 2019; Raffel et al., 2020, *i.a.*), recent approaches to span finding can be roughly divided into three different types: as *tagging*, *span enumeration*, or *boundary prediction* (see §2, Table 1). In this paper, we present an empirical study of these methods and their influence on downstream tasks, hoping to shed light on how to build future NLP systems. Specifically, we discuss the design choice of span finding methods and examine two common tricks for improving performance. We answer the following questions:



(a) Tagging (b) Span enumeration (c) Boundary prediction

Figure 1: Three common methods for span finding, detecting the mention spans *North Korea* and *nuclear weapons* in the sentence. "⦸" denotes a candidate span that is too long to be considered.

1. **Q:** What is the best span finding method? Does this hold for various NLP tasks or different pretrained encoders?

   A: The choice depends on the downstream task: tagging generally has higher precision, but span enumeration or boundary prediction has higher recall. For most cases, boundary prediction is preferable to span enumeration. Tagging performs much better on a masked language model pretrained encoder (e.g., RoBERTa) than an encoder-decoder pretrained model (e.g., T5).

2. **Q:** Does inclusion of mention type information help (e.g., just `B-PERSON` or `B` in tagging)?

   A: For downstream IE tasks, tagging and span enumeration approaches prefer untyped tags, but boundary prediction heavily relies on type information to obtain good performance.

[*] Equal contribution.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 3976–3983
December 7-11, 2022

Task	Tagging	Span Enumeration	Boundary Prediction
NER	Features + decision trees (Sekine et al., 1998) Features + CRF (McCallum and Li, 2003) BiLSTM + CRF (Lample et al., 2016)	Instance-based (Ouchi et al., 2020) SpanNER (Fu et al., 2021)	As QA (Li et al., 2020)
EE	JointEE Features+CRF (Yang and Mitchell, 2016) As Cloze BERT+CRF (Chen et al., 2020)	DyGIE BiLSTM+GNN (Luan et al., 2019) DyGIE++ BERT+GNN (Wadden et al., 2019)	As QA (Du and Cardie, 2020)
RE	BiLSTM + CRF (Bekoulis et al., 2018)	PURE (Zhong and Chen, 2021)	CasRel (Wei et al., 2020)
CR	Features + MEMM (Florian et al., 2004)	Span Ranking BiLSTM (Lee et al., 2017)	s2e Coref (Kirstain et al., 2021)
QA	Features + CRF (Yao et al., 2013) BiLSTM + CRF (Du and Cardie, 2018)	RaSoR BiLSTM (Lee et al., 2016)	BiDAF (Seo et al., 2017) BERT on SQuAD (Devlin et al., 2019)

Table 1: An overview of various span detecting methods for various tasks in recent NLP literature.

3. **Q:** Is an additional contextualization with RNN layers on top of Transformers helpful? Some prior work put an LSTM layer on top of embeddings produced by Transformers (Straková et al., 2019; Shibuya and Hovy, 2020; Wang et al., 2021, i.a.). Are these necessary?

   A: Not for RoBERTa (a pretrained encoder-only), but we observe a slight benefit from BiLSTM layers atop T5 (an encoder-decoder).

2 Background

We assume a pre-trained base model is in place (e.g. BERT (Devlin et al., 2019), T5 (Raffel et al., 2020)). The encoding for each token x_i is denoted as $\mathbf{x}_i \in \mathbb{R}^d$. For further analyses, we focus on RoBERTa (Liu et al., 2019) and T5 since these represent two classes of pretrained models: one with an encoder trained with reconstruction loss; the other with both an encoder and a decoder.

Tagging Span selection can be reduced to a sequence tagging problem, usually under the BIO scheme. Such tagging problems can be modeled using a linear-chain *conditional random field* (CRF; Lafferty et al., 2001), with tags either typed or untyped (for example, in NER tags may be labeled with entity types `B-PERSON`, `I-LOCATION`, or without, `B`, `I`). In recent work, features input to CRF may be hand-crafted features or predominantly outputs of a neural network. Note that reducing span finding to a tagging problem does not allow the system to produce *overlapping* spans. There exist methods to address these (e.g., for the task of *nested* NER), but we leave the analysis of these methods for future work.

Span Enumeration Lee et al. (2017) first proposed to enumerate all spans (up to length k) and predict whether they are entity mentions for coreference resolution. A span embedding $\mathbf{s}_{ij}$ is derived for each span $x[i:j]$, usually a concatenation of the left and right boundary tokens $\mathbf{x}_i, \mathbf{x}_j$, a pooled version of all the tokens between x_i and x_j (usually an attention-weighted sum with a learned global query vector $\mathbf{q}$ (Lee et al., 2017; Lin and Ji, 2019)), and optionally some additional manual features $\boldsymbol{\phi}$:

$$a_{k\in\{i,\cdots,j\}} \propto \exp(\mathbf{q}\cdot\mathbf{x}_k)$$
$$\mathbf{s}_{ij} = \left[\mathbf{x}_i\ ;\ \mathbf{x}_j\ ;\ \textstyle\sum_{k=i}^{j} a_k\mathbf{x}_k\ ;\ \boldsymbol{\phi}\right]$$

Such span embedding can be used for both span detection (untyped) and span typing: for detection, a span is given a score indicating whether it is a span of interest by applying a feedforward network F on top of the span embedding:

$$\{x[i:j] \mid F(\mathbf{s}_{ij}) > 0, 0 \le j-i \le k\}$$

For typing, one can create a classifier with the span embedding as input, and the set of types plus an ε type (not a selected span) as the output label set.

Boundary Prediction BiDAF (Seo et al., 2017) introduced a method to select one span from a sequence of text that we term *boundary prediction*. This has been widely adopted following the proliferation of work based on pretrained models such as BERT for QA.

In boundary prediction, two vectors $\mathbf{l}, \mathbf{r}$ over tokens are computed to indicate whether a token is a left or right boundary of a span:

$$l_i \propto \exp(\mathbf{q}_\mathrm{L}\cdot\mathbf{x}_i); \quad r_j \propto \exp(\mathbf{q}_\mathrm{R}\cdot\mathbf{x}_j)$$

To determine the most likely span, one selects $[\arg\max_i l_i, \arg\max_j r_j]$.

This method can be extended to the case where the model does not select any span (Devlin et al., 2019). A special [CLS] token may be prepended to

Encoder	Method	Entities		EE				RE	CR	QA	
		Ent-I	Ent-C	Trig-I	Trig-C	Arg-I	Arg-C	F1	Avg. F1	EM	F1
$\text{RoBERTa}_{\text{base}}$	Tagging	94.6	**90.0**	**73.3**	68.9	**53.4**	**50.8**	**59.5**	54.5	74.4	80.8
	Span Enumeration	94.8	89.8	72.7	**69.0**	22.6	16.7	13.1	**72.5**	68.8	73.8
	Boundary Prediction	**94.9**	89.9	72.3	67.7	41.6	37.4	39.6	71.7	**77.6**	**85.0**
$\text{T5}^{\text{enc}}_{\text{base}}$	Tagging	94.5	88.6	60.2	39.0	24.2	22.4	18.6	55.1	58.1	67.3
	Span Enumeration	92.0	86.2	59.2	47.8	20.2	14.1	10.2	70.4	62.3	66.1
	Boundary Prediction	**95.5**	**90.5**	**70.7**	**68.3**	**39.4**	**35.5**	**37.3**	**70.5**	**70.3**	**76.1**

Table 2: Basic experimental results on downstream tasks that involve mention detection. We report the results of entity extraction and event extraction from ACE05-E$^+$ dataset (F-score, %), relation extraction from ACE05-R dataset (F-score, %), coreference resolution from OntoNotes dataset, and QA task from SQuAD 2.0 dataset.

the sequence of tokens, taking index 0, and the model is trained to select the placeholder span $[0, 0]$ if no span should be selected.

Boundary prediction can also be extended to select more than one span. In CASREL (Wei et al., 2020), instead of selecting the most likely left and right indices, they select multiple left and right index if their score surpasses a threshold:

$$L = \{i \mid l_i \geq \theta\}; \quad R = \{j \mid r_i \geq \theta\}$$

Then a heuristic is used to match these candidate left/right boundaries to select spans. Li et al. (2020) extended the idea: instead of a heuristic, a model F (can be a 2-layer feedforward network, as is used in our experiments) is used to score all candidate spans selected by the threshold (up to length k):

$$\{x[i:j] \mid F(\mathbf{s}_{ij}) > 0, i \in L, j \in R, 0 \leq j - i \leq k\}$$

Since it is the most flexible and heuristic-free, we will focus on the last method in Li et al. (2020). To modify this span detector into a typed classifier, one can apply the same trick in span enumeration.

3 Experimental Setup

We perform all our experiments on $\text{RoBERTa}_{\text{base}}$ and the encoder part of T5_{base} ($\text{T5}^{\text{enc}}_{\text{base}}$). Each number reported is an average of 3 runs with different random seeds. Each run is trained with a single Quadro RTX 6000 GPU with 24GB memory.

NER, EE, and RE We use the ACE 2005 dataset (Walker et al., 2006) to evaluate model performance on NER, EE, and RE tasks. We follow OneIE (Lin et al., 2020) to compile dataset splits and establish the baseline using their released codebase.

For comparable setups across tasks, we disable all the global features which involve complicated cross-subtask interactions and cross-instance interactions that are hard to adapt to other span finding methods. We also disable the additional biaffine entity classifier and event type classifier and use the typing from the span finding module directly in inference time for the experiments in Table 2.

For NER, in addition to the standard entity classification F1 (Ent-C), we also report entity identification F1 (Ent-I) to measure how models detect spans. For EE, we use the standard {trigger (Trig) / argument (Arg)}-{identification (I) / classification (C)} F1 scores. For RE, the standard F1 is used.

Coreference Resolution We use the higher-order coreference resolution model (Lee et al., 2018) as implemented in AllenNLP (Gardner et al., 2018) as the baseline for coreference resolution.

We report the average F1 (Avg. F1) of the three common metrics, namely MUC, B^3, and CEAF_{ϕ_4} on the OntoNotes dataset (Weischedel et al., 2013).

Extractive QA We use the Transformer QA model in BERT as the baseline. We evaluate on the dev set of SQuAD 2.0 (Rajpurkar et al., 2018), a large-scale reading comprehension dataset containing both answerable and unanswerable questions. We keep the first span in the input sequence for the questions with multiple answer spans and discard the others. We use exact match (EM) and token overlap F1 for QA.

4 Discussions

4.1 Which Span Finder to Use?

From the results presented in Table 2, we find that boundary prediction is potentially preferable to span enumeration. Although span enumeration outperforms boundary prediction by a small margin in coreference resolution, boundary prediction outperforms span enumeration in other downstream tasks.

Approach	P	R
Tagging	**81.7**	72.8
Span Enumeration	25.5	**96.1**
Boundary Prediction	25.5	**96.0**

Table 3: Breakdown of mention score of each span finding method on OntoNotes dataset. We present the results from the models using $\text{RoBERTa}_{\text{base}}$ encoder.

Method	Entities		EE				RE
	Ent-I	Ent-C	Trig-I	Trig-C	Arg-I	Arg-C	F1
Tagging	94.6	89.6	72.7	69.1	54.5	52.1	56.5
- w/o Typing	+0.4	+0.5	+0.4	−1.1	+3.2	+3.0	+9.6
Span Enumeration	94.7	52.6	72.3	51.5	28.1	17.6	10.7
- w/o Typing	+0.1	−0.2	−0.1	+0.8	+1.5	+1.3	+1.0
Boundary Prediction	95.1	70.1	72.4	69.2	42.2	35.9	37.6
- w/o Typing	−0.0	−16.9	+2.1	−9.2	−12.6	−16.1	−27.3

Table 4: Experiment results of typing on IE tasks. Positive impact on model performance is shown in green while negative in red.

Encoder	Method	Entities		EE				RE	CR	QA	
		Ent-I	Ent-C	Trig-I	Trig-C	Arg-I	Arg-C	F1	Avg. F1	EM	F1
$\text{RoBERTa}_{\text{base}}$	Tagging	−0.0	−1.0	−1.3	−2.3	−2.9	−2.2	−1.2	+0.9	+0.3	+0.5
	Span Enumeration	+0.3	+0.1	+0.1	−0.6	+0.8	−0.8	−1.3	−0.1	−0.1	+0.1
	Boundary Prediction	+0.5	+0.6	−0.3	+0.6	−0.5	−0.2	−0.8	+0.6	+0.5	+0.4
$\text{T5}^{\text{enc}}_{\text{base}}$	Tagging	−0.4	+1.0	−7.0	+9.1	+5.2	+6.5	+4.0	+0.1	+4.0	+3.4
	Span Enumeration	+1.4	+1.7	+1.3	+5.1	−1.6	−0.8	−0.3	+0.9	+1.3	+1.0
	Boundary Prediction	−0.5	−0.8	−3.0	−4.0	−1.6	−1.9	−1.4	+1.1	+3.6	+3.8

Table 5: Experiment results adding BiLSTM contextual layer to our baseline models. The table shows the performance gaps compared to counterparts in Table 2 that do not have an additional contextualization. Positive impact on model performance is shown in green while negative in red.

While tagging and span enumeration suffer a considerable performance drop in all downstream tasks when using $\text{T5}^{\text{enc}}_{\text{base}}$ as encoder, boundary prediction only suffers a slight performance drop.

From the breakdown of the mention scores of each model on the OntoNotes dataset (coreference resolution) in Table 3, we can see that although span enumeration focuses significantly on recall,[1] boundary prediction can reach a comparable level of recall, whereas in a downstream task like QA where precision is needed, span enumeration cannot reach a comparable level of performance to boundary prediction.

The choice between tagging and boundary prediction depends on various factors, including but not limited to the language model, downstream task, training strategy, etc. Overall, tagging excels at precision; in contrast, boundary prediction and enumeration have better recall.

[1] In the evaluation of coreference resolution, singleton mention clusters (i.e., clusters that have only 1 mention) are ignored in computing the evaluation scores. This practice weeds out lots of spans that should not be selected as mentions. This is the reason that a coreference model can achieve state-of-the-art results with low mention detection precision.

4.2 Does Typing Help Span Finders?

As can be seen from the results in Table 4, tagging with untyped labels outperforms tagging with typed labels in all tasks except trigger classification; the margin is even more significant for the tasks of event argument extraction and relation extraction. We hypothesize that under joint training, the types in the labels might hinder the model from learning other objectives. Therefore, if it is not necessary to have typed tags, it is recommended to use plain `BIO` labels for tagging.

As for boundary prediction, we found that performing classification with types is crucial in the joint training with downstream tasks. This is possibly due to the nature of the two-step process of the method, where the first step can be seen as a coarse classifier to select potential mention candidates while the second step double-checks such candidates (or classifies candidates with more fine-grained types). However in span enumeration we observed that it is not much impacted by the inclusion of types. We hypothesize this results from label imbalance. Under span enumeration, there are a considerable number of spans that should not be labelled as valid mention spans. When downstream tasks further require classifying spans to more fine-

grained types, the label distribution would be seriously imbalanced and dominated by the ε (null type) label, making the learning ineffective.

4.3 Additional Contextualization?

We further examine a commonly seen practice of having an additional contextualization with RNN layers atop Transformer encoders. Following (Lee et al., 2017) as implemented in AllenNLP (Gardner et al., 2018), we use a 1-layer BiLSTM with hidden dimension of 200 for each direction.

We stack the BiLSTM contextual layer atop the Transformer encoders and report the experimental results in Table 5. We observe that, for IE tasks, adding additional contextualization does not affect model performance; for extractive QA, it improves model performance. Also, when using encoder-decoder architecture models (e.g., T5), the additional contextualization would lead to higher variance in downstream tasks compared to using encoder-only models (e.g., BERT). We hypothesize that the difference might come from the underlying architecture in T5, in which the encoder learns to specialize its representation to support the decoder. Therefore, when the encoder is being used alone, an additional contextualization might serve a similar purpose as a decoder and have to learn to utilize the representation to some extent. Even with such exceptions, from the design choice perspective, the merit of this trick is limited as it is not helpful in most cases while introducing training variance and additional parameters to the model.

5 Conclusions

We identified and investigated three common span finding methods in the NLP community: tagging, span enumeration, and boundary prediction. Through extensive experiments, we found that there is not a single recipe that is best for all scenarios. The design choices on downstream tasks rely on specific task properties, specifically the trade-off between precision and recall. We suggest that precision-focused tasks consider tagging or boundary prediction, and recall-focused (such as coreference resolution) tasks consider span enumeration or boundary prediction.

We further examined two commonly used tricks to improve span finding performance, i.e., adding span type information during the training and adding additional contextualization with an RNN on top. We observed that boundary prediction on IE tasks heavily relies on type information, and adding additional contextualization mostly does not help span finding.

Architectures will continue to evolve and models will continue to grow in size, which may lead to different conclusions on the relative benefits of approaches. Still, the fundamental task of isolating informative spans of text will remain. We hope this study helps inform system designers today with existing models and still in the future as a starting point for further inquiry.

6 Limitations

As an empirical study, we provide observations under different combination of design choices for building an end-to-end trained IE systems with a span finding component. We hope such observations could provide insights for future work, but we have to admit that we are also bounded by the limitations of empirical studies, and that a theoretical analysis is out of scope of this paper. As a result, we only make our claims based on the experiment results with the baseline models that we evaluated, and hope that it could generalize well to other architectures.

Acknowledgments

We thank Elias Stengel-Eskin, William Gantt, and Reno Kritz for helpful comments and feedback. This work was supported in part by DARPA AIDA (FA8750-18-2-0015) and IARPA BETTER (2019-19051600005). The views and conclusions contained in this work are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, or endorsements of DARPA or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018. Joint entity recognition and relation extraction as a multi-head selection problem. *Expert Syst. Appl.*, 114:34–45.

Yunmo Chen, Tongfei Chen, Seth Ebner, Aaron Steven White, and Benjamin Van Durme. 2020. Reading the manual: Event extraction as definition comprehension. In *Proceedings of the Fourth Workshop on*

Structured Prediction for NLP, pages 74–83, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Xinya Du and Claire Cardie. 2018. Harvesting paragraph-level question-answer pairs from Wikipedia. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1907–1917, Melbourne, Australia. Association for Computational Linguistics.

Xinya Du and Claire Cardie. 2020. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 671–683, Online. Association for Computational Linguistics.

R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N. Nicolov, and S. Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 1–8, Boston, Massachusetts, USA. Association for Computational Linguistics.

Jinlan Fu, Xuanjing Huang, and Pengfei Liu. 2021. SpanNER: Named entity re-/recognition as span prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7183–7195, Online. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Yuval Kirstain, Ori Ram, and Omer Levy. 2021. Coreference resolution without span representations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 14–19, Online. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML 2001*, pages 282–289. Morgan Kaufmann.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 687–692, New Orleans, Louisiana. Association for Computational Linguistics.

Kenton Lee, Tom Kwiatkowski, Ankur P. Parikh, and Dipanjan Das. 2016. Learning recurrent span representations for extractive question answering. *CoRR*, abs/1611.01436.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859, Online. Association for Computational Linguistics.

Ying Lin and Heng Ji. 2019. An attentive fine-grained entity typing model with latent type representation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6197–6202, Hong Kong, China. Association for Computational Linguistics.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general

framework for information extraction using dynamic span graphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3036–3046, Minneapolis, Minnesota. Association for Computational Linguistics.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 188–191.

Hiroki Ouchi, Jun Suzuki, Sosuke Kobayashi, Sho Yokoi, Tatsuki Kuribayashi, Ryuto Konno, and Kentaro Inui. 2020. Instance-based learning of span representations: A case study through named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6452–6459, Online. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Satoshi Sekine, Ralph Grishman, and Hiroyuki Shinnou. 1998. A decision tree method for finding and classifying names in Japanese texts. In *Sixth Workshop on Very Large Corpora*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Takashi Shibuya and Eduard Hovy. 2020. Nested named entity recognition via second-best sequence learning and decoding. *Transactions of the Association for Computational Linguistics*, 8:605–620.

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested NER through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331, Florence, Italy. Association for Computational Linguistics.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5784–5789, Hong Kong, China. Association for Computational Linguistics.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 Multilingual Training Corpus LDC2006T06. *Linguistic Data Consortium*.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021. Automated concatenation of embeddings for structured prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2643–2660, Online. Association for Computational Linguistics.

Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. 2020. A novel cascade binary tagging framework for relational triple extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1476–1488, Online. Association for Computational Linguistics.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2013. OntoNotes Release 5.0 LDC2013T19. *LDC*.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–299, San Diego, California. Association for Computational Linguistics.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 858–867, Atlanta, Georgia. Association for Computational Linguistics.

Zexuan Zhong and Danqi Chen. 2021. A frustratingly easy approach for entity and relation extraction. In

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 50–61, Online. Association for Computational Linguistics.