GNNer: Reducing Overlapping in Span-based NER Using Graph Neural Networks

Urchade Zaratiana[*†], **Nadi Tomeh**[†], **Pierre Holat**[*†], **Thierry Charnois**[†]
[*] FI Group, Puteaux, France
[†] LIPN, Université Sorbonne Paris Nord - CNRS UMR 7030, Villetaneuse, France
{urchade.zaratiana,pierre.holah}@fi-group.com
{charnois,tomeh}@lipn.fr

Abstract

There are two main paradigms for Named Entity Recognition (NER): sequence labelling and span classification. Sequence labelling aims to assign a label to each word in an input text using, for example, BIO (Begin, Inside and Outside) tagging, while span classification involves enumerating all possible spans in a text and classifying them into their labels. In contrast to sequence labelling, unconstrained span-based methods tend to assign entity labels to overlapping spans, which is generally undesirable, especially for NER tasks without nested entities. Accordingly, we propose GNNer, a framework that uses Graph Neural Networks to enrich the span representation to reduce the number of overlapping spans during prediction. Our approach reduces the number of overlapping spans compared to strong baseline while maintaining competitive metric performance. Code is available at https://github.com/urchade/GNNer.

1 Introduction

Named Entity Recognition (NER) is an information extraction task that aims to identify named entities such as locations, organizations and person names from textual data. Frequently, NER is designed as a sequence labelling task where each word is classified into its respective label using an annotation scheme such as BIO (Huang et al., 2015; Lample et al., 2016). Such schemes are used to encode segment information on the token level. Recently, span-based NER has gained a lot of popularity by handling segments, instead of individual words, as the basic units for labelling (Luan et al., 2018; Wadden et al., 2019). Specifically, span-based NER enumerates every segment in a text and classifies them by their entity label, whereby non-entity segments are classified into an allocated `null` label. While this method has shown good empirical results, it often assigns entity labels to overlapping spans, which is not desirable, especially for flat NER tasks.

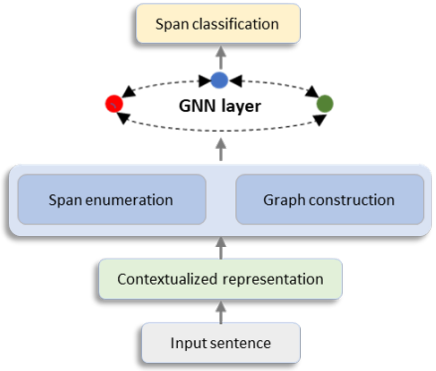

Figure 1: **The overall architecture of our framework**: GNNer

Therefore, to ensure that entities do not overlap, a constraint must be explicitly applied during decoding through, for example, Semi-Markov CRFs (Sarawagi and Cohen, 2005; Sato et al., 2017). Recent work by Fu et al. (2021) and Li et al. (2021) address overlapping entities using heuristic decoding: conflict between overlapping spans is resolved by retaining the span with the highest prediction probability, dropping the others. This approach has proven effective, however, the no-overlap constraint is not imposed during learning, which is sub-optimal. In this work, we consider that the no-overlap constraint could be optimized directly by injecting inductive biases into the model.

In this regard, we propose a new approach to reduce overlapping in span-based NERs without affecting the efficiency of heuristic-based decoding. The idea is to make the representation of each span directly influenced by other spans overlapping with it. Specifically, we encode overlapping information as a graph and feed it into the span representation using an equivariant graph neural network layer. In this way, we bias the model towards predictions that implicitly respect the constraints without explicitly modelling them. Our results

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Student Research Workshop, pages 97 - 103
May 22-27, 2022

demonstrate that injecting this graph during model training significantly reduces the number of overlaps compared to our baseline model while achieving better performance. We propose, in this paper, two variants of our model, `GNNer-Conv` based on the graph convolution network (Kipf and Welling, 2017) and `GNNer-AT` based on the graph attention network (Velickovic et al., 2018). We observe that `GNNer-AT` is best at preventing span overlaps at the cost of a low recall, while `GNNer-Conv` provides a better trade-off between the number of violated constraints and metric performance (precision, recall and F-score).

2 Model

Given an input sequence, our task involves enumerating and classifying every span. The architecture of our model, summarized in Figure 1, includes the following components: token representation layer, span representation layer, GNN layer and span classification layer. Our model is similar to the vanilla span-based NER models (Lee et al., 2017; Luan et al., 2019), to which we add the GNN layer.

2.1 Word Representation

The primary component of our architecture is the word representation layer. The purpose of this layer is to return a set of embedding vectors $\{\boldsymbol{h}^0, \boldsymbol{h}^1, \ldots, \boldsymbol{h}^L\}$ from a sequence of tokens $\{w^0, w^1, \ldots, w^L\}$. For this part, we employ pre-trained Transformer models such as BERT (Devlin et al., 2019). However, since pre-trained Transformer models produce sub-word instead of word representations, we retain for each word its first sub-word representation. This choice works well in practice for token classification tasks (Devlin et al., 2019; Beltagy et al., 2019).

2.2 Span Representation

After representing words with their contextualized embeddings, we enumerate all the spans of the sentence up to a maximum span width, which we set to 6 in all our experiments, following prior works (Sarawagi and Cohen, 2005; Xia et al., 2019). Next, we compute the representation of a span as the concatenation of word embeddings of its left and right extremities, along with a learned embedding of the span width. Specifically, a span (i, j) of width k is represented by the vector $s_{ij} = \boldsymbol{h}^i \otimes \boldsymbol{h}^j \otimes \boldsymbol{z}_k$ where $\boldsymbol{h}^i$ and $\boldsymbol{h}^j$ are respectively the representation of the words at indexes i and j, and $\boldsymbol{z}_k$ corresponds to the embedding vector for spans of width k; the $\otimes$ symbol denotes the concatenation operation.

2.3 Graph construction

Given two spans s_1 and s_2, our graph as represented by the adjacency matrix $\boldsymbol{A}$ is defined as follows:

$$\boldsymbol{A}[s_1, s_2] = \begin{cases} 1, & \text{if } s_1 = s_2 \\ 0, & \text{if } |s_1 \cap s_2| = 0 \\ -1, & \text{otherwise} \end{cases} \quad (1)$$

In the adjacency matrix, the edge weight 1 corresponds to self-connection, 0 to non-overlapping nodes, and -1 to overlapping spans. The choice of -1 for the overlap case is supposed to bias the model to learn dissimilar representations for overlapping spans. However, we believe that there may be a better choice to achieve this objective, which would require more in-depth investigation. The addition of the span graph information to the model before the classification layer gives each span information about the spans connected to it and thus allows them to make predictions in a collaborative way, i.e. to make their predictions according to the predictions of their neighbours in the graph.

2.4 Span refinement with GNN

After the initial BERT-based representations of all spans are obtained, we refine them using a GNN layer exploiting the previously constructed graph. We propose two versions of the GNN layer: `GNNer-CONV`, based on graph convolution; and `GNNer-AT` based on attention mechanisms. By exploiting the graph information, we expect the model to implicitly learn that two overlapping spans should not be predicted as a named entity at the same time by learning dissimilar representations for them.

2.4.1 GNNer-CONV

The first variant of our model uses a GCN (Kipf and Welling, 2017) layer, but since GCN is not well suited in the presence of negative edges (Derr et al., 2018), we run two independent 1-layer GCNs over the span representations S: a first GCN, GCN_+ using only positive edges E^+ and another GCN GCN_- using only negative edges E^- for which we concatenate the two representations to get the final

	Architecture	Precision	Recall	F1	Num. Ov.
Conll 2003	Baseline	89.83±0.48	90.31±0.26	90.06±0.15	83±27
	GNNer-CONV	90.12±0.32	89.88±0.36	**90.16±0.52**	52±1
	GNNer-AT	89.54±0.84	79.32±0.04	84.12±0.37	24±11
SciERC	Baseline	66.69±0.49	69.89±0.45	68.25±0.33	87±4
	GNNer-CONV	66.89±1.59	70.34±0.50	**68.57±0.96**	35±3
	GNNer-AT	63.21±0.51	58.06±0.86	60.53±0.69	13±2
NCBI	Baseline	85.30±0.45	89.59±0.74	87.39±0.13	43±12
	GNNer-CONV	85.98±0.45	88.93±0.45	**87.43±0.45**	16±5
	GNNer-AT	84.78±0.18	79.41±0.61	81.98±0.38	10±4

Table 1: **The results of the experiments on the test datasets**. We report the micro-averaged precision, recall and F1-score as well as Num. OV., the total number of overlapping spans on all the test set (without normalization). The numbers are the result of averaging across 3 different/independent runs using different random seeds.

span representation:

$$\begin{aligned} \boldsymbol{S}^{+} &= GCN_{+}(\boldsymbol{S}, E^{+}) \\ \boldsymbol{S}^{-} &= GCN_{-}(\boldsymbol{S}, E^{-}) \\ \boldsymbol{S}^{final} &= \boldsymbol{S}^{+} \otimes \boldsymbol{S}^{-} \end{aligned} \tag{2}$$

Note that running a 1-layer GCN on the positive edges is equivalent to a linear layer since the positive edges are self-connections.

2.4.2 GNNer-AT

The second variant of our method uses a graph attention network (Velickovic et al., 2018) but instead of using additive attention, we employ a dot product attention which is much faster and more space-efficient in practice, according to Vaswani et al. (2017). More specifically, we project the span representation into keys $\boldsymbol{K}$, queries $\boldsymbol{Q}$, and values $\boldsymbol{V}$ using a two-layer feed-forward network, and compute the attention score as the dot product of the queries and all keys. We further include the scaling factor $\frac{1}{\sqrt{d_{model}}}$ following (Vaswani et al., 2017) to prevent saturation. We then multiply this attention score by the weighted adjacency matrix. We compute the final span representation as follows:

$$\boldsymbol{S}^{final} = (\frac{\boldsymbol{Q}\boldsymbol{K}^{T}}{\sqrt{d_{model}}} \odot \boldsymbol{A})\boldsymbol{V} \tag{3}$$

In the above equation, $\odot$ denotes element-wise multiplication or Hadamard product which is used to mask the attention for null edges. One downside to this approach is that the self-attention mechanism has a quadratic complexity in the number of spans.

2.5 Span classification

Lastly, the final representation of the spans is passed to a linear layer with softmax activation to predict the span labels. Remember that for non-entity spans, we allocate a `null` label.

$$Y = \text{softmax}(\boldsymbol{S}^{final}\boldsymbol{W}^{(f)}) \tag{4}$$

Here, $\boldsymbol{W}^{(f)}$ is a weight matrix that project the span representations into the label space and the softmax activation function is applied to the label dimension.

3 Experiments

3.1 Experimental Setup

Datasets We evaluate our approach on three benchmark datasets: Conll-2003 (Tjong Kim Sang and De Meulder, 2003), SciERC NER (Luan et al., 2018) and NCBI (Doğan et al., 2014). Conll-2003 is a general domain NER dataset that extracts person, organization and location entity mentions from text. SciERC is a dataset for scientific information extraction that consists of article abstracts extracted from Artificial Intelligence related articles. NCBI is a NER dataset that is designed to identify disease mentions in biomedical texts. For all the datasets, we employed the standard train, test and validation splits.

	Domain	Train	Dev	Test
Conll 2003	News	14,987	3,466	3,684
NCBI	Bio	5432	923	940
SciERC	CS	350	50	50

Table 2: **The statistics of the datasets**

Evaluation We evaluate our models on the test splits of the corresponding datasets. Our evaluation is based on the exact match between true and gold entities by discarding non-entity spans. We report

the micro-averaged precision, recall and F1. In addition, we also measure the ability of each model to avoid entity overlaps during classification by reporting the number of entity overlaps (Num. Ov.) across all the test set, where a lower number is better.

Implementation details For all our experiments, we used either pre-trained BERT (Devlin et al., 2019) or SciBERT (Beltagy et al., 2019) as the word encoder depending on the dataset used i.e. BERT for conll-2003, and SciBERT for SciERC and NCBI. We employed a span width embedding of 128 dimensions, and down-projected the span representation (768 * 2 + 128) into 128 units before the GNN layer, using a linear layer. We used only one layer for all GNN variants, which resulted in the best performance on the dev set. In fact, we noticed in our preliminary experiments that adding more layers resulted in decreased performance and slower convergence during training. For all experiments, we set our learning rate to 1e-5 and used Adam (Kingma and Ba, 2017) as our optimizer. We ran all our models for up to 50 epochs and kept the checkpoint with the best validation performance for testing. All our models are implemented in the PyTorch (Paszke et al., 2019) and we used the heavily tested GCN layer provided by PyTorch Geometric library (Fey and Lenssen, 2019).

Baseline We used the same architecture without the GNN layer as our baseline. For fair comparisons, we increased the size of the baseline layers to obtain a comparable number of parameters to our proposed models.

3.2 Results

Table 1 summarizes the results of our experiments by reporting the performance measures (micro-averaged Precision, Recall and F1-score) and the Num. Ov. on the test set. The numbers are the result of averaging across 3 independent runs using different random seeds.

Main results From the table 1 we can draw several conclusions. First, `GNNer-AT` outperforms every approach at reducing Num. Ov. On average. It produces 4 times fewer overlaps than the baseline model and 2 times fewer than the `GNNer-CONV` model. However, it has low recall (-11 absolute points compared to the baseline on conll-2003) but can maintain a comparable precision score. The problem of low recall could be caused by overly restricting the span representation through the use of negative edges in our span graph, which could prevent the model from predicting many entities. Second, `GNNer-CONV` gets competitive results while maintaining a low Num. Ov. compared to the baseline model, making it the best balance between Num. Ov. and metric performance.

Learning curves Figure 2 shows the evolution of precision, recall, and Num. Ov. during model training. The plot is shown for training on the SciERC dataset, we obtained similar curves on Conll-2003 and NCBI datasets. We observe that the baseline model trains faster than the GNN-based method, which can be explained by the non-overlap constraint induced by the GNN that favours low recall. On the other hand, the Num. Ov. of the graph-based approach remains low during training, especially for the GNNer-AT approach, while the baseline model increases at the first stage of training before gradually decreasing.

4 Limitations

There are several limitations to our approach. First, the addition of GNN does not completely remove the overlapping spans in contrast to heuristic approches. Moreover, the inclusion of GNN layer bring more comptation to the model which result into a slower model than the baseline span-based NER. In fact since, the overlaping span graph is dense (contains many egde), the model does not really benefit of efficient sparse operations of GNN layers.

5 Related works

Approaches for NER NER is an important tasks in Natural Language Processing and is used in many downstream information extraction applications. Usually, NER tasks are designed as sequence labelling (Chiu and Nichols, 2016; Huang et al., 2015; Ma and Hovy, 2016; Lample et al., 2016; Akbik et al., 2018; Zaratiana et al., 2022). The goal is to predict BIO tags in which a word is labelled as B-tag if it is the beginning of an entity, I-tag if it is within but not the first in the entity and O for non-entity words. Recently, different approaches have been proposed to perform NER tasks that go beyond traditional sequence labelling. One approach that has been widely adopted is the span-based approach (Luan et al., 2018, 2019; Wadden et al., 2019; Xue et al., 2020) where the representation of

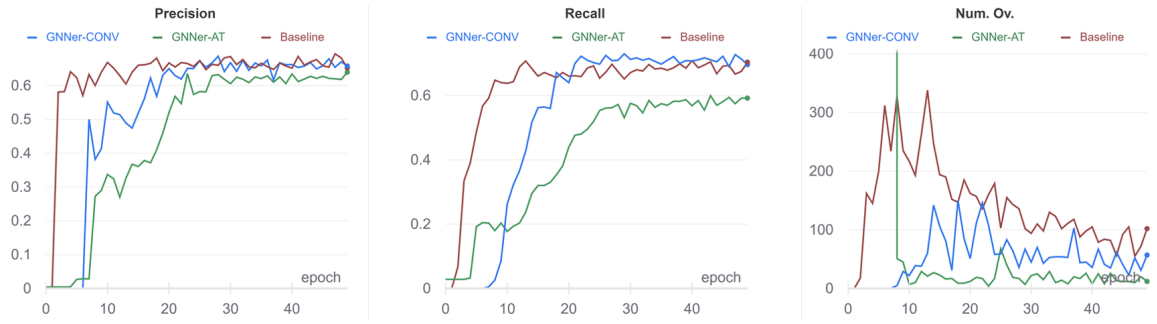

Figure 2: **Evolution of precision, recall and number of overlaps (Num. Ov.) on the SciERC validation set**.

each segment is computed using a neural network, then fed to a classifier. To prevent overlapping span, priors works either used heuristic decoding (Fu et al., 2021; Li et al., 2021; Xia et al., 2019) or structured decoding using semi-CRFs (Sato et al., 2017; Ye and Ling, 2018). However, to the best of our knowledge, no work have used GNN for the purpose of reducing span overlap for NER. Some work (Li et al., 2020) has also approached NER as a question answering task in which named entities are extracted by retrieving answer spans. In addition, with the growing popularity of prompt-based learning, recent work such as (Cui et al., 2021) considers NER as template filling by fine-tuning a BART (Lewis et al., 2019) encoder-decoder model. In contrast we focus on learning appropriate span representations.

GNN for NLP GNNs have gained a lot of popularity recently due to their powerful ability to represent arbitrary shapes of data (Hamilton et al., 2018; Wu et al., 2019; Hamilton, 2020). Specifically, GNNs provide a way to inject prior knowledge into NLP systems through, for example, dependency graphs (Liu et al., 2018; Zhang et al., 2019), constituency graphs (Marcheggiani and Titov, 2020) or knowledge graphs (Sun et al., 2018; Lin et al., 2021). As a result, GNNs have been widely applied to different NLP tasks such as Neural Machine Translation (Bastings et al., 2017; Beck et al., 2018), Semantic Parsing (Xu et al., 2018; Shao et al., 2020), Information Extraction (Fu et al., 2019; Sun et al., 2019) and text classification (Yao et al., 2018; Liu et al., 2020). More relevant to our work, DyGiE (Luan et al., 2019; Wadden et al., 2019) used GNNs to refine the span representation for joint NER and RE extraction, but in contrast, they learn their graph dynamically during training while we used a static span graph. For a detailed review of GNNs for NLP, please refer to Wu et al. (2021).

6 Conclusion

In this work, we investigated new span-based NER method using Graph Neural Networks. Our best approach, built on a Graph Convolution Network, significantly reduces the number of overlapping spans compared to a strong baseline (up to 2 times less) while maintaining competitive metric performance. In future work, we will explore ways to integrate GNN-enhanced representations into architectures for joint named entity recognition and relation extraction tasks.

Ethical considerations

There are ethical considerations to take into account when using NER technology. For example, the technology may disproportionately work worse for some populations with uncommon name structure. This could have a negative impact on these groups, as their names may not be accurately recognized and classified by the software. It is important that we are aware of potential biases in our data and algorithms, so that we can avoid unfairly discriminating against certain groups of people.

Acknowledgments

This work was performed using HPC resources from GENCI-IDRIS (Grant 20XX-AD011013096).

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, Copenhagen, Denmark. Association for Computational Linguistics.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. 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 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Jason P. C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using bart.

Tyler Derr, Yao Ma, and Jiliang Tang. 2018. Signed graph convolutional network.

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.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. Ncbi disease corpus: A resource for disease name recognition and concept normalization. *Journal of Biomedical Informatics*, 47:1–10.

Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with pytorch geometric.

Jinlan Fu, Xuanjing Huang, and Pengfei Liu. 2021. Spanner: Named entity re-/recognition as span prediction.

Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun Ma. 2019. GraphRel: Modeling text as relational graphs for joint entity and relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1409–1418, Florence, Italy. Association for Computational Linguistics.

William L. Hamilton. 2020. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2018. Representation learning on graphs: Methods and applications.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Thomas Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907.

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.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified mrc framework for named entity recognition.

Yangming Li, lemao liu, and Shuming Shi. 2021. Empirical analysis of unlabeled entity problem in named entity recognition. In *International Conference on Learning Representations*.

Yuxiao Lin, Yuxian Meng, Xiaofei Sun, Qinghong Han, Kun Kuang, Jiwei Li, and Fei Wu. 2021. Bertgcn: Transductive text classification by combining gcn and bert.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, Brussels, Belgium. Association for Computational Linguistics.

Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. 2020. Tensor graph convolutional networks for text classification.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf.

Diego Marcheggiani and Ivan Titov. 2020. Graph convolutions over constituent trees for syntax-aware semantic role labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3915–3928, Online. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library.

Sunita Sarawagi and William W Cohen. 2005. Semi-markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems*, volume 17. MIT Press.

Motoki Sato, Hiroyuki Shindo, Ikuya Yamada, and Yuji Matsumoto. 2017. Segment-level neural conditional random fields for named entity recognition. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 97–102, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Bo Shao, Yeyun Gong, Weizhen Qi, Guihong Cao, Jianshu Ji, and Xiaola Lin. 2020. Graph-based transformer with cross-candidate verification for semantic parsing. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8807–8814.

Changzhi Sun, Yeyun Gong, Yuanbin Wu, Ming Gong, Daxin Jiang, Man Lan, Shiliang Sun, and Nan Duan. 2019. Joint type inference on entities and relations via graph convolutional networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1361–1370, Florence, Italy. Association for Computational Linguistics.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242, Brussels, Belgium. Association for Computational Linguistics.

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 the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio', and Yoshua Bengio. 2018. Graph attention networks. *ArXiv*, abs/1710.10903.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. *ArXiv*, abs/1909.03546.

Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, and Bo Long. 2021. Graph neural networks for natural language processing: A survey.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A comprehensive survey on graph neural networks.

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 the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1430–1440, Florence, Italy. Association for Computational Linguistics.

Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. 2018. Exploiting rich syntactic information for semantic parsing with graph-to-sequence model.

Mengge Xue, Bowen Yu, Zhenyu Zhang, Tingwen Liu, Yue Zhang, and Bin Wang. 2020. Coarse-to-fine pre-training for named entity recognition.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2018. Graph convolutional networks for text classification.

Zhixiu Ye and Zhen-Hua Ling. 2018. Hybrid semi-Markov CRF for neural sequence labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 235–240, Melbourne, Australia. Association for Computational Linguistics.

Urchade Zaratiana, Pierre Holat, Nadi Tomeh, and Thierry Charnois. 2022. Hierarchical transformer model for scientific named entity recognition. *ArXiv*, abs/2203.14710.

Chen Zhang, Qiuchi Li, and Dawei Song. 2019. Aspect-based sentiment classification with aspect-specific graph convolutional networks. 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 4568–4578, Hong Kong, China. Association for Computational Linguistics.