CopyNext: Explicit Span Copying and Alignment in Sequence to Sequence Models

Abhinav Singh[1,2]   **Patrick Xia[1]**   **Guanghui Qin[1]**
Mahsa Yarmohammadi[1]   **Benjamin Van Durme[1]**
[1]Johns Hopkins University   [2]Bloomberg L.P.
abhinavsingh282@gmail.com, {paxia,vandurme}@cs.jhu.edu,
{qin,mahsa}@jhu.edu

Abstract

Copy mechanisms are employed in sequence to sequence models (seq2seq) to generate reproductions of words from the input to the output. These frameworks, operating at the lexical *type* level, fail to provide an explicit alignment that records where each *token* was copied from. Further, they require contiguous token sequences from the input (spans) to be copied individually. We present a model with an explicit token-level copy operation and extend it to copying entire spans. Our model provides hard alignments between *spans* in the input and output, allowing for nontraditional applications of seq2seq, like information extraction. We demonstrate the approach on Nested Named Entity Recognition, achieving near state-of-the-art accuracy with an order of magnitude increase in decoding speed. [1]

1 Introduction

Sequence transduction converts a sequence of input tokens to a sequence of output tokens. It is a dominant framework for generation tasks, such as machine translation, dialogue, and summarization. Seq2seq can also be used for Information Extraction (IE), where the target structure is *decoded* as a linear output based on an encoded (linear) representation of the input.

As IE is traditionally considered a structured prediction task, it remains today that IE systems are assumed to produce an annotation on the input text. That is, predicting which specific tokens of an input string led to, e.g., the label of PERSON. This is in contrast to text generation which rarely, if ever, needs *hard alignments* between the input and the desired output. Our work explores a novel extension to seq2seq that provides such alignments.

---

[1]Our source code: https://github.com/abhinonymous/copynext

The luxury auto maker last year sold 1,214 cars in the U.S.	

(a) *last year* **DATE** *last* **REL** *year* **DUR** *<UNK>* **CARD** *U.S.* **CTRY**
(b) *last year* **DATE** *last* **REL** *year* **DUR** *1,214* **CARD** *U.S.* **CTRY**
(c) **4 5 DATE 4 REL 5 DUR 7 CARD 11 CTRY**
(d) **4 CN DATE 4 REL 5 DUR 7 CARD 11 CTRY**

Figure 1: Sequence transduction outputs for nested named entities in an example sentence using: (a) seq2seq, (b) pointer network, (c) Copy-only, and (d) CopyNext model. The numbers are predictions of *indices* corresponding to the tokens in the input sequence. CN refers to the CopyNext symbol, our proposed method of denoting the operation that copies the next token from the input. In (d), the next token from token 4 would be 5.

Specifically, we extend pointer (or copy) networks. Unlike the algorithmic tasks originally targeted by Vinyals et al. (2015), tasks in NLP tend to copy *spans* from the input rather than discontiguous tokens. This is prevalent for copying named entities in dialogue (Gu et al., 2016; Eric and Manning, 2017), entire sentences in summarization (See et al., 2017; Song et al., 2018), or even single words (if subtokenized). The need to efficiently copy spans motivates our introduction of an inductive bias that copies contiguous tokens. Like a pointer network, our model copies the first token of a span. However, for subsequent timesteps, our model generates a "CopyNext" symbol (CN) instead of copying another token from source. CopyNext represents the operation of copying the word following the last predicted word from the input sequence. Figure 1 highlights the difference between output sequences for several transductive models, including our CopyNext model.

We apply our model for the Nested Named Entity Recognition (NNER) task (Ringland et al., 2019). Unlike traditional named entity recognition, named entity mentions in NNER may be subsequences of
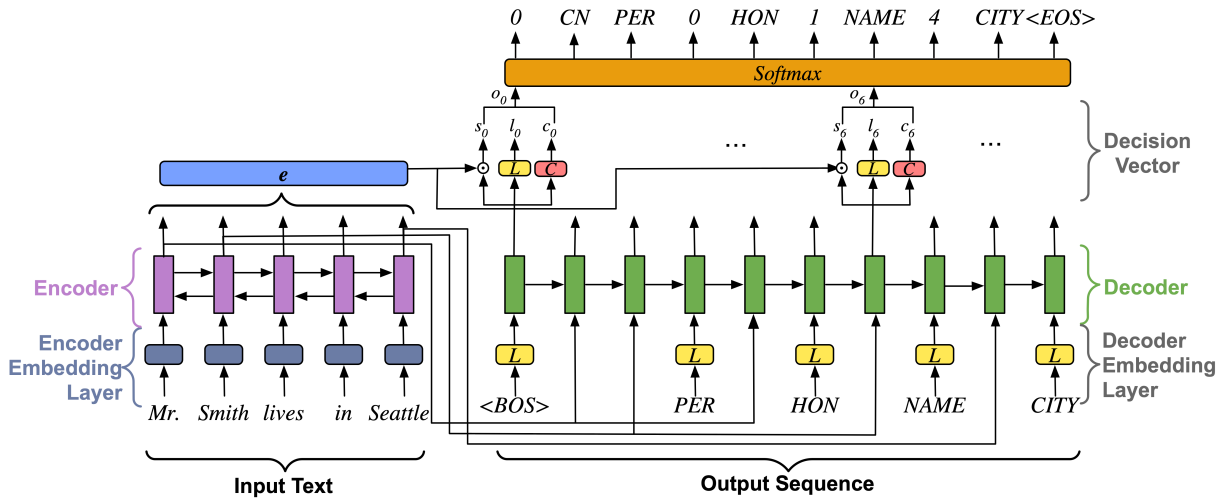
Figure 2: For each decoder timestep a decision vector chooses between labeling, a CopyNext operation, or pointing to an input token. The decoder input comes from either an encoder state or a label embedding.

other named entity mentions (such as *[[last] year]* in Figure 1). We find that both explicit copying and CopyNext lead to a system faster than prior work and better than a simple seq2seq baseline. It is, however, outperformed by a much slower model that performs an exhaustive search over the space of potential labels, a solution that does not scale to large complex label sets.

2 Related Work

Pointer networks (Vinyals et al., 2015; Jia and Liang, 2016; Merity et al., 2016) are seq2seq models that employ a soft attention distribution (Bahdanau et al., 2014) to produce an output sequence consisting of values from the input sequence. Pointer-generator networks (Miao and Blunsom, 2016; Gulcehre et al., 2016, *inter alia*) extend the range of output types by combining the distribution from the pointer with a vocabulary distribution from a generator. Thus, these models operate on the *type* level. In contrast, our model operates at the *token* level. Instead of using soft attention distribution of the encoder states, we use hard attention, resulting in a single encoder state, or a single token, to feed to the decoder. This enables explicit copying of span offsets.

Closest to our work, Zhou et al. (2018) and Pan-thaplackel et al. (2020) have tackled span copying by extending pointer-generator networks and predicting both start and end indices of entire spans that need to be copied. Using those offsets, they perform a forced decoding of the predicted tokens within the span. These works focus on text generation tasks, like sentence summarization, question generation, and editing. In contrast, we are concerned with information extraction tasks as transduction, where hard alignments to the input sentence are crucial and output sequences must represent a valid linearized structure. Specifically, we study nested named entity recognition (NNER).

Prior work uses several approaches to model NNER: machine reading comprehension (Li et al., 2019), transition-based methods (Wang et al., 2018), mention hypergraphs (Lu and Roth, 2015; Wang and Lu, 2018; Katiyar and Cardie, 2018), and seq2seq models (Straková et al., 2019).

3 Model Description

We formulate the task as transforming the input sentence X to a linearized sequence Y which represents the gold structure: labeled spans. Specifically, Y contains input word indices, CopyNext symbols, and labels from a label set $\mathcal{L}$.

As described earlier, the model (Figure 2) is reminiscent of pointer networks. We extend its capabilities by introducing the notion of a "Copy Next" operation where the network predicts to copy the word sequentially after the previous prediction.

3.1 Encoder

Embedding Layer This layer embeds a sequence of tokens $X = \langle x_1, x_2, ..., x_{N'} \rangle$ into a sequence of vectors $\mathbf{x} = \langle \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N \rangle$ by using (possibly contextualized) word embeddings. The gold labels are adjusted to account for tokenization.

Architecture The input embedding is further encoded by a stacked bidirectional LSTM (Hochre-

iter and Schmidhuber, 1997) into encoder states $\mathbf{e} = \langle \mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_N \rangle$ where each state is a concatenation of the forward ($\overrightarrow{f}$) and backward ($\overleftarrow{f}$) outputs of the last layer of the LSTM and $\mathbf{e}_i \in \mathbb{R}^D$:

$$\mathbf{e}_i^j = [\overrightarrow{f}^j(\mathbf{e}_i^{j-1}, \mathbf{e}_{i-1}^j); \overleftarrow{f}^j(\mathbf{e}_i^{j-1}, \mathbf{e}_{i+1}^j)], \quad (1)$$

where $\mathbf{e}_i^j$ is the j-th layer encoder hidden state at timestep i and D is the hidden size of the LSTM.

3.2 Decoder

The target for the transducer is the linearized representation of the nested named entity spans and labels. We generate a decision y that either points to (a) a timestep in the encoder sequence, marking the starting index of a span, or (b) the CopyNext symbol, which operates by advancing the right boundary of the span to include the next (sub)word of the input sequence, or (c) a label $l \in \mathcal{L}$, signifying both the end of the span and classifying the span.

Input Embeddings We learn D-dimensional embeddings for each label $l \in \mathcal{L}$. The vectors corresponding to the start index of a span and the Copy-Next operation are the encoder outputs $\mathbf{e}_i$ where i is equal to the start index or index pointed to by CopyNext and are fed directly to the decoder.[2]

Architecture The decoder is a stacked LSTM taking as input either an encoder state $\mathbf{e}_i$ or a label embedding and produces decoder state $\mathbf{d}_t \in \mathbb{R}^D$.

Decision Vector We predict scores for making a labeling decision, a CopyNext operation, or pointing to a token in the input. At each decoding step t, for labels, we train a linear layer $W_L \in \mathbb{R}^{D \times |\mathcal{L}|}$ with input $\mathbf{d}_t$ and output scores $\mathbf{l}_t$. Likewise, we do the same for the CopyNext symbol using a linear layer $W_C \in \mathbb{R}^{D \times 1}$ with input $\mathbf{d}_t$ and output score $\mathbf{c}_t$. The score of pointing to an index i in the input sequence is calculated by dot product: $s_t^i = \mathbf{e}_i \cdot \mathbf{d}_t$. The decision distribution $\mathbf{y}_t$ is then:

$$\mathbf{y}_t = \text{softmax}([\mathbf{s}_t; \mathbf{l}_t; \mathbf{c}_t]), \ \mathbf{y}_t \in \mathbb{R}^{N+|\mathcal{L}|+1}. \quad (2)$$

3.3 Training and Prediction

Our training objective is the cross-entropy loss:

$$\ell = \sum_t \sum_k \delta_{y_t^k, y_t^\star} \log(y_t^k) \quad (3)$$

where $y^\star$ is the gold decision, $k \in [0, N + |\mathcal{L}| + 1)$ (representing all three kinds of possible decisions:

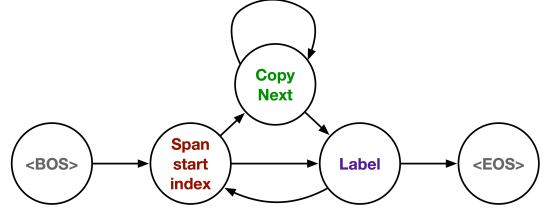[2]We will use $\mathbf{e}_i$ to refer to $\mathbf{e}_i^{(-1)}$.

Figure 3: State machine of a well-formed predicted sequence for masking of the decision vector at inference.

Embedding	P	R	F1
RoBERTa	**90.6**	**82.6**	**86.4**
BERT large cased	90.4	78.2	83.9
XLNet large cased	79.1	52.6	63.2
SpanBERT large cased	89.1	80.2	84.4
ELMo	82.1	73.3	77.4
GloVe	76.5	67.8	71.9

Table 1: A comparison of networks used for embedding input tokens before feeding them into the encoder LSTM. These are evaluated on NNER.

index, label or CopyNext) and $\delta_{y_t^k, y_t^\star}$ is 1 if $y_t^k = y_t^\star$ and 0 otherwise. The summation over index t covers the whole dataset.

At prediction time we find the decision $\overline{y}_t$ with the greatest probability ($\overline{y}_t = \arg\max_i(y_t^i)$) at decoder step t.[3] The input to the decoder at $t + 1$ timestamp can be one of three things: (1) the output $\mathbf{e}_i$ of the encoder when $\overline{y}_t$ points to the index i of the input sequence, (2) the embedding of the label l predicted at t when $\overline{y}_t$ points to the label $l \in \mathcal{L}$, or (3) the output $\mathbf{e}_{i+1}$ of the encoder where i was the input to the decoder at t when $\overline{y}_t$ points to the CopyNext operation. The decoder halts when the $\langle EOS \rangle$ label is predicted or the maximum output sequence length is reached.

To ensure well-formed target output sequences, we use a state machine (Figure 3) to mask parts of $\overline{y}_t$ that would lead to an illegal sequence at $t + 1$.

4 Experiments and Results

Our experiments analyze the effects of various choices in different components of the system. We use the NNE dataset and splits from Ringland et al. (2019), resulting in 43,457, 1,989, and 3,762 sentences in the training, development, and test splits. Experiments for model development and analysis use the development set.

Text Representation We first establish the best performing text embeddings which we fix for the

---

[3]Initial experiments with beam search suggest an expensive tradeoff between time and performance (Appendix A.1).

13

Model	P	R	F1	Speed
Hypergraph	91.8	91.0	91.4	1.0x
Transition	77.4	70.1	73.6	6.3x
Seq2seq	86.6	63.6	75.4	(*)
Copy	85.8	81.3	83.5	16.7x
CopyNext	88.7	84.7	86.7	16.7x

Table 2: NNER accuracy and speed on the test set for external baselines and our models. *Seq2seq is based on a reference implementation to ensure correctness, but not efficiency: it has the same asymptotics as the Copy and CopyNext models, and can be considered similar in speed.
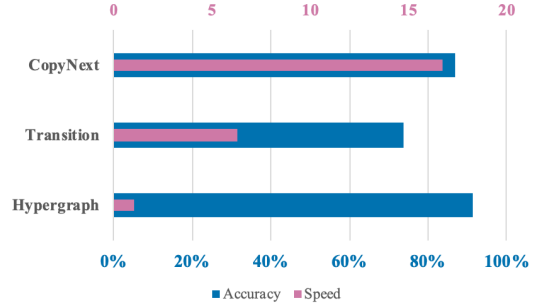

Figure 4: Performance in terms of Accuracy (%F1) and Speed (relative to Hypergraph). The CopyNext model is nearly as accurate as Hypergraph while over 16 times faster.

rest of the experiments. The rationale is that the embedding will provide an orthogonal boost in accuracy to the network with respect to the other changes in the network structure. We find in Table 1 that RoBERTa large (Liu et al., 2019) is best.[4]

Linearization Strategy Previous work (Zhang et al., 2019) has shown that linearization scheme affects model performance. We experiment with several variants of sorting spans in ascending order based on their start index. We also try sorting based on end index and copying the *previous* token instead. We find sorting based on end index performs poorly, while sorting by start all perform similarly. Our final linearization strategy sorts by start index, then span length (longer spans first). Additional ties (in span label) are broken randomly.

RoBERTa Embedding Layer Recent work suggests that NER information may be stored in the lower layers of an encoder (Hewitt and Manning, 2019; Tenney et al., 2019). We found using the 15th layer of RoBERTa rather than the final one (24th), is slightly helpful (see Appendix A.1).

NNER Results In Table 2, we evaluate our best-performing (dev.) model on the test set. We compare our approach against the previous best approaches reported in Ringland et al. (2019): hypergraph-based (Hypergraph, Wang and Lu, 2018) and transition-based (Transition, Wang et al., 2018) models proposed to recognize nested mentions. We also contrast the CopyNext model against a baseline seq2seq model and one with only a hard copy operation (see (a) and (c) in Figure 1). Prior

work (Wang and Lu, 2018) has given an analysis of the run-time of their approach. Based on their concern about asymptotic speed we also provide the following analysis and practical speed efficiency of the systems and their accuracies.[5]

We find that Hypergraph outperforms the Copy-Next model by 4.7 F1 with most of the difference in recall. This is likely due to the exhaustive search used by Hypergraph, as our model is 16.7 times faster. An analysis of their code and algorithm reveals that their lower bound time complexity $\Omega(mn)$ is higher than ours $\Omega(n)$, n is length of input sequence and m is number of mention types. Since the average decoder length is low, the best case scenario often occurs. The Transition system has 6.3 times faster prediction speed compared to Hypergraph, however, it comes with 17.8% absolute drop in F1 accuracy. Our model is substantially faster than both. Furthermore, we show that both an explicit Copy and the CopyNext operation are useful, resulting in gains of 8.1 F1 and 11.3 F1 over a seq2seq baseline.

NNER Error Analysis The errors made by the model on the development set can be clustered broadly into four main types: (1) correct span detection but mislabeled, (2) correct label but incorrect span detection (either subset or superset of correct span), (3) both span and label were incorrectly predicted, and (4) missing spans entirely. Table 7 in Appendix A.1 provides examples.

5   Conclusion and Future Work

We propose adopting pointer and copy networks with hard attention and extending these models

---

[4]We experimented with mean pooling RoBERTa vectors for subwords (Zhang et al., 2019) to maintain the same span lengths as input. Pooling subword units led to poorer performance (see Appendix A.1 Table 3).

[5]Efficiency is measured using wall clock time for the entire test set, performed with Intel Xeon 2.10GHz CPU and a single GeForce GTX 1080 TI GPU.

with a CopyNext operation, enabling sequential copying of spans given just the start index of the span. On a traditionally structured prediction task of NNER, we use a sequence transduction model with the CopyNext operation, leading to a competitive model that provides a 16.7x speedup relative to current state of the art (which performs an exhaustive search), at a cost of 4.7% loss in F1, largely due to lower recall.

Our model is a step forward in structured prediction as sequence transduction. We have found in initial experiments on event extraction similar relative improvements to that discussed here: future work will investigate applications to richer transductive semantic parsing models (Zhang et al., 2019; Cai and Lam, 2019).

Acknowledgments

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Deng Cai and Wai Lam. 2019. Core semantic first: A top-down approach for AMR parsing. 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 3799–3809, Hong Kong, China. 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.

Mihail Eric and Christopher Manning. 2017. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 468–473, Valencia, Spain. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149, Berlin, Germany. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. 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 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8(0):64–77.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. 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 861–871.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2019. A unified mrc framework for named entity recognition. *arXiv preprint arXiv:1910.11476*.

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.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 857–867.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 319–328.

Sheena Panthaplackel, Miltiadis Allamanis, and Marc Brockschmidt. 2020. Copy that! editing sequences by copying spans.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew 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.

Nicky Ringland, Xiang Dai, Ben Hachey, Sarvnaz Karimi, Cecile Paris, and James R. Curran. 2019. NNE: A dataset for nested named entity recognition in English newswire. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5176–5181, Florence, Italy. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Kaiqiang Song, Lin Zhao, and Fei Liu. 2018. Structure-infused copy mechanisms for abstractive summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1717–1729, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Jana Straková, Milan Straka, and Jan Hajič. 2019. Neural architectures for nested ner through linearization. *arXiv preprint arXiv:1908.06926*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks.

Bailin Wang and Wei Lu. 2018. Neural segmental hypergraphs for overlapping mention recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 204–214, Brussels, Belgium. Association for Computational Linguistics.

Bailin Wang, Wei Lu, Yu Wang, and Hongxia Jin. 2018. A neural transition-based model for nested mention recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1011–1017, Brussels, Belgium. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32*, pages 5753–5763. Curran Associates, Inc.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. AMR parsing as sequence-to-graph transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 80–94, Florence, Italy. Association for Computational Linguistics.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2018. Sequential copying networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.