Abstract

In recent years, NLP has moved towards the application of language models to a more diverse set of tasks. However, applying language models to structured prediction, e.g., predicting parse trees, taggings, and coreference chains, is not straightforward. Prior work on language model-based structured prediction typically flattens the target structure into a string to easily fit it into the language modeling framework. Such flattening limits the accessibility of structural information and can lead to inferior performance compared to approaches that overtly model the structure. In this work, we propose to construct a conditional language model over sequences of structure-building actions, rather than over strings in a way that makes it easier for the model to pick up on intra-structure dependencies. Our method sets the new state of the art on named entity recognition, end-to-end relation extraction, and coreference resolution.

1 Introduction

Many common NLP tasks, e.g., named entity recognition, relation extraction, and coreference resolution are naturally taxonomized as structured prediction, the supervised machine-learning task of predicting a structure from a large set. To generalize well to held-out data in a structured prediction problem, the received wisdom has been that it is necessary to correctly model complex dependencies between different pieces of the structure. However, a recent trend in structured prediction for language has been to forgo explicitly modeling such dependencies (Ma and Hovy, 2016; Lee et al., 2017; He et al., 2017, inter alia), and, instead, to apply an expressive black-box model, e.g., a neural network, with the hope that the model picks up on the dependencies without explicit instruction. The idea behind the framework is to encode the target structure as a string, flattening out the structure. Then, one uses a conditional language model to predict the flattened string encoding the structure. For instance, Vinyals et al. (2015) flatten parse trees into strings and predict the strings encoding the flattened trees from the sentence with a machine translation architecture. The hope is that the autoregressive nature of the language model allows it to learn to model the intra-structure dependencies and the necessary hard constraints that ensure the model even produces well-formed structures. Additionally, many modelers make use of pre-trained language models (Lewis et al., 2020; Raffel et al., 2020) to further improve the language models.

However, despite their empirical success, simply hoping that a black-box approach correctly models intricate intra-structure dependencies is often insufficient for highly structured tasks (Paolini et al., 2021, §1). Indeed, the act of flattening a structured object into a string makes properly modeling the intra-structure dependencies harder for many tasks, e.g., those that involve nested spans or long-distance dependencies. For instance, in coreference resolution, a conference link between two mentions can stretch across thousands of words, and a coreference chain can also contain over a hundred mentions (Pradhan et al., 2012). Flattening such a large amount of structured information into a string makes the task more difficult to model.

In this paper, we propose a simple framework that augments a conditional language model with explicit modeling of structure. Instead of modeling strings that encode a flattened representation of the target structure, we model a constrained set of actions that build the target structure step by step; see Fig. 1 for an example of our proposed
framework. Training a conditional language model to predict structure-building actions exposes the structure in a way that allows the model to pick up on the intra-structure dependencies more easily while still allowing the modeler to leverage pre-trained language models. We conduct experiments on three structured prediction tasks: named entity recognition, end-to-end relation extraction, and coreference resolution. On each task, we achieve state-of-the-art results without relying on data augmentation or task-specific feature engineering.

2 Autoregressive Structured Prediction

In this section, we describe our proposed approach, which we refer to as autoregressive structured prediction (ASP). Unlike previous approaches for structured prediction based on conditional language modeling, we represent structures as sequences of actions, which build pieces of the target structure step by step. For instance, in the task of coreference resolution, the actions build spans as well as the relations between the spans, contiguous sequences of tokens. We give an example in Fig. 1.

2.1 Representing Structures with Actions

While our approach to structured prediction, ASP, is quite general, our paper narrowly focuses on modeling structures that are expressible as a set of dependent spans, and we couch the technical exposition in terms of modeling spans and relationships among spans. Our goal is to predict an action sequence \( \mathbf{y} = y_1, \ldots, y_N \), where each action \( y_n \) is chosen from an action space \( \mathcal{Y}_n \). In this work, we take \( \mathcal{Y}_n \) to be factored, i.e., \( \mathcal{Y}_n \equiv \mathcal{A} \times \mathcal{B}_n \times \mathcal{Z}_n \), where \( \mathcal{A} \) is a set of structure-building actions, \( \mathcal{B}_n \) is the set of bracket-pairing actions, and \( \mathcal{Z}_n \) is a set of span-labeling actions. Thus, each \( y_n \) may be expressed as a triple, i.e., \( y_n = (a_n, b_n, z_n) \). We discuss each set in a separate paragraph below.

**Structure-Building Actions.** We first define a set of structure-building actions \( \mathcal{A} = \{ \text{[ }, \text{]}, \text{copy} \} \) that allow us to encode the span structure of a text, e.g., \( \text{[ } \text{Delaware} \text{] } \) in Fig. 1 encodes that Delaware is a span of interest. More technically, the action \( \text{[ } \) refers to a right bracket that marks the right-most part of a span. The action \( \text{]} \) refers to a left bracket that marks the left-most part of a span. The superscript * on \( \text{[ } \) is inspired by the Kleene star and indicates that it is a placeholder for 0 or more consecutive left brackets. Finally, \text{copy} refers to copying a word from the input document. To see how these actions come together to form a span, consider the subsequence in Fig. 1, \( \text{[ } \text{Delaware} \text{] } \), which is generated from a sequence of structure-building actions \( \text{[ }, \text{ ], copy, and [ } \).

**Bracket-Pairing Actions.** Next, we develop the set of actions that allow the model to match left and right brackets; we term these bracket-pairing actions. The set of bracket-pairing actions consists of all previously constructed left brackets, i.e.,

\[
\mathcal{B}_n = \{ m \mid m < n \land a_m = \text{[ } \}
\]

Thus, in general, \( |\mathcal{B}_n| = \mathcal{O}(n) \). However, it is often the case that domain-specific knowledge can

\( ^{2}\)In our preliminary experiments, we observe unsatisfactory performance when the model has to generate consecutive left brackets. We leverage \( \text{[ } \) as an engineering workaround. We hypothesize that this phenomenon is due to the inability of transformers to recognize Dyck languages (Hahn, 2020; Hao et al., 2022).
be used to prune $B_n$. For instance, coreference mentions and named entities rarely cross sentence boundaries, which yields a linguistically motivated pruning strategy (Liu et al., 2022). Thus, in some cases, the cardinality of $B_n$ can be significantly smaller. When we decode action sequences $y$ into a structure, unpaired $[\mathbf{e}]$ and $[\mathbf{3}]$ can be removed ensuring that the output of the model will not contain unpaired brackets.

**Span-Labeling Actions.** Finally, we add additional symbols $z_n$ associated with each $y_n$ that encode a labeling of a single span or a relationship between two or more spans. For instance, see §2.3 for an example. We denote the set of all $z_n$ as

$$
Z_n = \left\{ m \mid m < n \land a_m = [\mathbf{3}] \right\} \times \mathcal{L} \quad (2)
$$

where $\left\{ m \mid m < n \land a_m = [\mathbf{3}] \right\}$ is the set of previous spans, which allows the model to capture intra-span relationships, and $\mathcal{L}$ denotes the set of possible labelings of the current span and the relationship between the adjoined spans. In general, designing $Z_n$ requires some task-specific knowledge in order to specify the label space. However, we contend it requires less effort than designing a flattened string output where different levels of structures may be intertwined (Paolini et al., 2021).

### 2.2 Model Parameterization

Let $D = w_1, \ldots, w_K$ be an input document of $K$ sentences where $w_k$ denotes the $k$th sentence in $D$. We first convert the structure to be built on top of $D$ into an action sequence, which we denote as $y$ where $y_n \in \mathcal{Y}_n$. Now, we model the sequence of actions $y$ as a conditional language model

$$
p_\theta(y \mid D) = \prod_{n=1}^N p_\theta(y_n \mid y_{<n}, D) \quad (3)
$$

The log-likelihood of the model is then given by $\log p_\theta(y \mid D) = \sum_{n=1}^N \log p_\theta(y_n \mid y_{<n}, D)$. We model the local conditional probabilities $p(y_n \mid y_{<n}, D)$ as a softmax over a dynamic set $\mathcal{Y}_n$ that changes as a function of the history $y_{<n}$, i.e.,

$$
p_\theta(y_n \mid y_{<n}, D) = \frac{\exp s_\theta(y_n)}{\sum_{y_{n}^\prime \in \mathcal{Y}_n} \exp s_\theta(y_{n}^\prime)} \quad (4)
$$

where $s_\theta$ is a parameterized score function; we discuss several specific instantiations of $s_\theta$ in §2.3. Finally, we note that the use of a dynamic vocabulary stands in contrast to most conditional language models where the vocabulary is held constant across time steps, e.g., Sutskever et al.’s (2014) approach to machine translation.

**Greedy Decoding.** We determine the approximate best sequence $y^*$ using a greedy decoding strategy. At decoding step $n$, we compute

$$
y^*_n = \operatorname{argmax}_{y_n} p_\theta(y_n \mid y_{<n}, D) \quad (5)
$$

The chosen $y^*_n = (a^*_n, b^*_n, z^*_n)$ will then be verbalized as a token as follows: If $a^*_n = \text{copy}$, then we copy the next token from the input that is not present in the output. Otherwise, if $y^*_n = [\mathbf{e}]$ or $y^*_n = [\mathbf{3}]$, we insert $[\mathbf{e}]$ or $[\mathbf{3}]$ into the output sequence, respectively. The verbalized token is then fed into the conditional language model at the next step. The decoding process terminates when the model copies a distinguished symbol EOS symbol from the input. The end of the procedure yields an approximate $\text{argmax} y^*$.

**Computational Complexity.** Eq. (4) can be computed quite efficiently using our framework, as the cardinalities of $A$ is $O(1)$, and the size of $B_n$ and $Z_n$ are both $O(n)$. A tighter analysis says the cardinalities of $B_n$ and $Z_n$ are roughly linear in the number of spans predicted. In practice, we have $n \ll |V|$ where $|V|$ is the size of vocabulary, which is the step-wise complexity of (Paolini et al., 2021). A quantitative analysis of the number of mentions in coreference can be found in App. B.

**Generality.** Despite our exposition’s focus on tasks that involve assigning labels to span or span pairs, our method is quite general. Indeed, almost any structured prediction task can be encoded by a series of structure-building actions. For tasks that involve labeling tuples of spans, e.g., semantic role labeling makes use of tree-tuples that consist of the subject, predicate, and object, Eq. (2) can be easily extended with a new space of categorical variables $\left\{ m \mid m < n \land a_m = [\mathbf{3}] \right\}$ to model the extra item.

### 2.3 Task-specific Parameterizations

We now demonstrate how to apply ASP to three language structured prediction tasks: named entity recognition, coreference resolution, and end-to-end relation extraction.

**Named Entity Recognition.** Named entity recognition is the task of labeling all mention spans $\mathcal{E} = \{e_n\}_{n=1}^K$ in a document $D$ that refers to named...
entities. Since named entity recognition only requires labeling spans (and not linking them), we only need our task-specific $z_n$ to encode the entity type, which is canonically taken from a set of pre-defined categories $C$. The function $s_{\theta}(y_n)$ in Eq. (4) is implemented by a feed-forward network

$$s_{\theta}(y_n) = \langle \alpha_n, b_n, z_n \rangle$$

\[
\begin{cases} 
\text{FFN}_{a_n}^{\pi_n}(m_n) & \text{if } a_n = \mathbb{1} \\
\text{FFN}_{a_n}^{\pi_n}(h_n) & \text{otherwise}
\end{cases}
\]

where $h_n$ is the decoder hidden state at step $n$, a column vector, and $m_n = [h_n^T; h_{b_n}^T]$ represents the mention that corresponds to $y_n$. Note that each FFN$_{\pi_n}^{z_n}$ and FFN$_{\pi_n}^{a_n}$ represent independent feed-forward networks with no shared parameters.

**End-to-End Relation Extraction.** End-to-end relation extraction is the task of jointly extracting a set of entities alongside a set of relations between pairs of extracted entities. Formally, given a set of pre-defined entity categories $C$ and a set of pre-defined relations $R$. The goal is (i) to identify all possible entities $\mathcal{E} = \{e_n\}_{n=1}^{\mathcal{E}}$ in $D$ that could be associated with one of the entity types $c$ in $C$ and (ii) to identify all possible triples $\mathcal{T} = \{(e_n, r_n, r'_n)\}_{n=1}^{\mathcal{T}}$ in $D$ where $e_n, r'_n \in \mathcal{E}$ are the head and tail entity and $r_n \in \mathcal{R}$ is the relation between $e_n$ and $r'_n$. Here, the support of $z_n$ takes the form of Eq. (2), where $\mathcal{C}$ is instantiated as $C \times R$. And $s_{\theta}(y_n)$ kept the same as in Eq. (6).

**Coreference Resolution.** The task of coreference resolution involves identifying all mention spans $\mathcal{E} = \{e_n\}_{n=1}^{\mathcal{E}}$ in $D$ and then clustering them. However, in addition to identifying the mention spans, the task of coreference resolution requires us to assign an antecedent to every possible mention in $D$. To encode coreference resolution in our framework, we consider the task-specific $z_n$ from the set

$$Z_n = \{m \mid m < n \wedge a_m = \mathbb{1}\} \cup \{e\} \tag{7}$$

where we follow the convention set in Lee et al. (2017) that the antecedent of the first mention in each coreference chain is defined to be $e$. Again, we define $s_{\theta}(y_n = \langle a_n, b_n, z_n \rangle)$ as in Eq. (6) with the exception that, when $z_n = e$, we define FFN$_{a_n}^{\pi_n}(m_n)e = \text{FFN}_{a_n}^{z_n}(m_n)$.

### 3 Experiments

We experiment on three NLP structured prediction tasks: named entity recognition, end-to-end relation extraction, and coreference resolution. We are primarily interested in understanding whether ASP provides advantages over two existing formalisms: (i) conditional language models (Athiwaratkun et al., 2020; Paolini et al., 2021) that flatten the structure into a string (augmented language models), and (ii) the classic discriminative models whose autoregressivity is bounded. We experiment with three pre-trained language models, T5 (Raffel et al., 2020), T0 (Sanh et al., 2021), and Flan-T5 (Chung et al., 2022) for the three tasks under consideration. Additional experimental details are given in Appendix A.1 and A.2.

#### 3.1 Named Entity Recognition

First, we evaluate our model on the CoNLL-03 English NER task. Following previous work, we report the micro precision, recall, and F1 score. As shown in Tab. 1, our model using T0-3B backbone outperforms all other models without data augmentation or ensembling.

#### 3.2 End-to-End Relation Extraction

We compare ASP on the CoNLL-04 and ACE-05 English end-to-end relation extraction datasets. The results are shown in Tab. 2 and Tab. 3. Our proposed approach achieves state-of-the-art results on both datasets using T5-3B as the backbone. In particular, it outperforms the flattened-string model of Paolini et al. (2021) by a large margin (> 0.9 F1). We hypothesize that this is due to relations requiring higher-order dependencies between spans.

### Table 1: Test F1 scores of named entity recognition on the CoNLL-03 test set.

<table>
<thead>
<tr>
<th>Model</th>
<th>Prec.</th>
<th>Rec.</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ma and Hovy (2016)</td>
<td>91.4</td>
<td>91.1</td>
<td>91.2</td>
</tr>
<tr>
<td>Devlin et al.+BERTL</td>
<td>-</td>
<td>-</td>
<td>92.8</td>
</tr>
<tr>
<td>Ye et al.+ROBERTAL</td>
<td>-</td>
<td>94.0</td>
<td></td>
</tr>
<tr>
<td>Athiwaratkun et al.</td>
<td>-</td>
<td>91.5</td>
<td></td>
</tr>
<tr>
<td>Paolini et al.+T5B</td>
<td>-</td>
<td>91.7</td>
<td></td>
</tr>
<tr>
<td>ASP+T5B</td>
<td>91.4</td>
<td>92.2</td>
<td>91.8</td>
</tr>
<tr>
<td>ASP+T5L</td>
<td>92.1</td>
<td>93.4</td>
<td>92.8</td>
</tr>
<tr>
<td>ASP+T53B</td>
<td>93.8</td>
<td>94.4</td>
<td>94.1</td>
</tr>
</tbody>
</table>

3Ye et al. (2022) counts symmetric relations twice for evaluation, which is inconsistent with previous work. We report the re-evaluated scores under the standard metric.

3On ACE-05, we observe inferior performance using T0-3B instead of T5-3B. We suspect this is due to systematic deficiencies in dataset preprocessing, e.g., errors during sentencization and tokenization as well as inconsistent capitalization.
We then conduct experiments on the standard OntoNotes benchmark in the CoNLL-12 English shared task dataset (Pradhan et al., 2012). Tab. 4 reports the results. Again, our model achieves state-of-the-art performance among systems without any data augmentation \(^5\), outperforming the previous state of the art by 1.5 F1 score. We also observe that our ASP models substantially outperform discriminative models that make use of the same PLM. Further analysis is provided in App. B.

### 5 Conclusion

In this paper, we propose a novel framework for structured prediction that encodes a structure as a series of structure-building actions that obtains state-of-the-art performance across three tasks. In contrast to past approaches for structured prediction, our approach is compatible with pre-trained large language models. This allows us to reduce structured prediction to the problem of fine-tuning pre-trained language models over an enlarged alphabet. We show empirically that ASP outperforms previous structured prediction models by a large margin. Indeed, we set the new state of the art on three tasks: named entity recognition, end-to-end relation extraction, and coreference resolution.

#### Acknowledgements

We acknowledge support from an ETH Zürich Research grant (ETH-19 21-1) and a grant from the Swiss National Science Foundation (project #201009) for this work. We also thank Zeerak Talat and Peter Szemraj for their feedback on this manuscript.

---

\(^{5}\)We achieve 82.9 F1 score on the development set, outperforming the result without pretraining on additional data reported by Wu et al. (2020, Tab. 5). In addition, training our model does not require the usage of TPUs.
Ethical Considerations

To consider the ethical implications of our work, we consider the tasks and models used and our proposed approach. The tasks considered, named entity recognition, relation extraction, and coreference resolution are often used in a pipeline of approaches (say for automatically building knowledge bases). Understanding the biases, errors, and failure cases of these tasks and their models and how they affect downstream use cases of the knowledge base would be important to consider. That said, to our knowledge the proposed approach does not exacerbate (or lessen) or introduce new considerations to the ones known about tasks/models more generally.

Limitations

Autoregressive Modeling Assumption. The decoder model, which is autoregressive, introduces an inductive bias on the structured prediction approach. Specifically, the left-to-right approach requires the model to model dependencies in a specific order. This could account for some of the reduction in performance compared to task-specific discriminative models. Understanding the implications of the autoregressive decision is indeed an interesting question, but one that we felt was out of scope for this short paper.

Efficiency. In our experiments, we reduce the burden of finding many mention spans in two-stage approaches. On sentence-level tasks, e.g., entity and relation extraction, the number of decoding steps is relatively small. For instance, the average number of words in an input sentence is ≈20. Our system has a lighter memory trace as opposed to discriminative models. This extra time cost can be partially compensated with larger batch sizes. However, on document-level tasks, e.g., coreference resolution, the number of decoding steps is too large to be compensated with parallelism. More efficient methods for inference such as non-autoregressive decoding (Gu et al., 2018) remain to be explored in future work.

Decoding Algorithms. In this work, we use greedy decoding in all the experiments. Alternative decoding algorithms might further improve the quality of the generated sequences, e.g., beam search (Zhang and Clark, 2008; Goldberg et al., 2013).

Choice of Pretrained Language Models. In this work, the choice of T5 and its variants as the conditional language model backbone of our model is largely motivated by their ability to handle arbitrarily long sequences. Unlike BART and GPT, T5 uses relative position encoding. On document-level tasks such as coreference resolution, the ability to process long sequences is extremely important. However, other pretrained conditional language models, either with encoder–decoder structures or decoder-only structures, can be used as a backbone. It might be interesting to explore techniques that generalize fixed-length position encoding to longer sequences.

References


Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of


Markus Eberts and Adrian Ulges. 2020. Span-based joint entity and relation extraction with transformer pre-training. 24th European Conference on Artificial Intelligence.


A Experimental Details

A.1 Experimental Settings
In our experiments, the \{T5,T0,flan-T5\}-base, \{T5,T0,flan-T5\}-large, \{T5,T0,flan-T5\}-{3B,XL}, \{T5,T0,flan-T5\}-{11B,XXL} have 220 million, 770 million, 3 billion, and 11 billion parameters respectively\(^6\). The feedforward neural networks described in §2.3 have one hidden layer of size 150 for ACE-05, 4096 for CoNLL-03, CoNLL-04, and CoNLL-12.

We follow the same preprocessing procedure and train/dev/test split of previous work on all datasets. For all the experiments, we use the AdamW optimizer (Kingma and Ba, 2015). We train 40 epochs on CoNLL-12 for coreference resolution with batch size 1. For end-to-end relation extraction on CoNLL-04 and ACE-05, we train 100 epochs with batch size 8. The initial learning rates are set to 5e-5 for \{T5,T0,flan-T5\}-base and \{T5,T0,flan-T5\}-large models, 3e-5 for \{T5,T0,flan-T5\}-{3B,XL,11B,XXL} models.

We apply bfloat16 training in our experiments. One single A100-40GB GPU is used for training models that use \{T5,T0,flan-T5\}-base and \{T5,T0,flan-T5\}-large models. Two A100-40GB GPUs are required to train models that use 3B or XL. Six A100-80GB GPUs are required to train models that use 11B or XXL models. It takes around 0.1 seconds for base-scale models and 1 second per updating step for \{3B,11B,XXL\} models.

A.2 Datasets

A.2.1 Named Entity Recognition
CoNLL-03. We use the CoNLL-03 dataset (Tjong Kim Sang and De Meulder, 2003) to evaluate our model on named entity recognition. This dataset consists of 946 training articles, 216 development articles, and 231 test sentences. We evaluate under the document-level settings, which means we feed the entire document into the model instead of the individual sentences.

A.2.2 End-to-End Relation Extraction
CoNLL-04. The CoNLL-04 dataset contains four types of entities (location, organization, person, other) and five types of relations (work for, kill, organization based in, live in, located in). We split the dataset as the training (922 sentences), validation (231 sentences), and test (288 sentences) as in previous work. For the ACE-05 dataset, we follow the train/dev/test split of previous work (Zhong and Chen, 2021).

ACE-05. The ACE-05 dataset (Walker and Consortium, 2005) contains 511 documents in total collected from multiple domains including newswire, broadcast, discussion forums, etc. We follow Luan et al. (2019)’s preprocessing script\(^7\) and split the dataset into train/dev/test set. ACE-05 contains inconsistently capitalized data. The newswire portion collected from CNN are entirely lowercased, which involves around 20 documents. Previous works (Zhong and Chen, 2021; Ye et al., 2022) that use case-insensitive encoders such as ALBERT are not affected by this deficiency. However, the T5 model and its variants are case-sensitive. We use the python truecase package\(^8\) to restore the correct capitalization during preprocessing.

A.2.3 Coreference Resolution
CoNLL-12. The CoNLL-12 English shared task dataset for coreference resolution (Pradhan et al., 2012) contains 2802 documents for training, 343 for validation, and 348 for testing. During training, we chunk the documents into segments of 2048 maximum words. In total, there are 2830 segments for training. During the evaluation, we use the entire document as the input to the model.

---

\(^6\)https://github.com/google-research/text-to-text-transfer-transformer
\(^7\)https://github.com/luanyi/DyGIE/tree/master/preprocessing
\(^8\)https://pypi.org/project/truecase/
<table>
<thead>
<tr>
<th>Models</th>
<th>P</th>
<th>R</th>
<th>F</th>
<th>P</th>
<th>R</th>
<th>F</th>
<th>P</th>
<th>R</th>
<th>F</th>
<th>Avg. F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lee et al. (2017)</td>
<td>78.4</td>
<td>73.4</td>
<td>75.8</td>
<td>68.6</td>
<td>61.8</td>
<td>65.0</td>
<td>62.7</td>
<td>59.0</td>
<td>60.8</td>
<td>67.2</td>
</tr>
<tr>
<td>Lee et al. (2018)</td>
<td>81.4</td>
<td>79.5</td>
<td>80.4</td>
<td>72.2</td>
<td>69.5</td>
<td>70.8</td>
<td>68.2</td>
<td>67.1</td>
<td>67.6</td>
<td>73.0</td>
</tr>
<tr>
<td>Joshi et al. (2019)</td>
<td>84.7</td>
<td>82.4</td>
<td>83.5</td>
<td>76.5</td>
<td>74.0</td>
<td>75.3</td>
<td>74.1</td>
<td>69.8</td>
<td>71.9</td>
<td>76.9</td>
</tr>
<tr>
<td>Joshi et al. (2020)</td>
<td>85.8</td>
<td>84.8</td>
<td>85.3</td>
<td>78.3</td>
<td>77.9</td>
<td>78.1</td>
<td>76.4</td>
<td>74.2</td>
<td>75.3</td>
<td>79.6</td>
</tr>
<tr>
<td>Joshi et al. +T5b†</td>
<td>82.4</td>
<td>77.4</td>
<td>79.8</td>
<td>72.3</td>
<td>68.2</td>
<td>70.2</td>
<td>70.5</td>
<td>63.5</td>
<td>66.8</td>
<td>72.3</td>
</tr>
<tr>
<td>Joshi et al. +T5l†</td>
<td>85.5</td>
<td>77.7</td>
<td>81.4</td>
<td>78.3</td>
<td>68.5</td>
<td>73.1</td>
<td>75.0</td>
<td>65.9</td>
<td>73.1</td>
<td>74.9</td>
</tr>
<tr>
<td>Dobrovolskii</td>
<td>84.9</td>
<td>87.9</td>
<td>86.3</td>
<td>77.4</td>
<td>74.1</td>
<td>76.1</td>
<td>77.1</td>
<td>76.6</td>
<td>76.6</td>
<td>81.0</td>
</tr>
<tr>
<td>Urbizu et al.</td>
<td>-</td>
<td>-</td>
<td>64.9</td>
<td>-</td>
<td>-</td>
<td>66.5</td>
<td>-</td>
<td>-</td>
<td>65.3</td>
<td>65.6</td>
</tr>
<tr>
<td>Paolini et al. +T5b†</td>
<td>-</td>
<td>81.0</td>
<td>-</td>
<td>69.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>68.4</td>
<td>72.8</td>
<td></td>
</tr>
<tr>
<td>ASP +T5b</td>
<td>81.7</td>
<td>82.8</td>
<td>82.3</td>
<td>74.2</td>
<td>76.1</td>
<td>75.1</td>
<td>74.5</td>
<td>70.6</td>
<td>72.5</td>
<td>76.6</td>
</tr>
<tr>
<td>ASP +T5s</td>
<td>83.3</td>
<td>86.2</td>
<td>84.7</td>
<td>75.9</td>
<td>79.5</td>
<td>77.7</td>
<td>75.8</td>
<td>74.5</td>
<td>75.2</td>
<td>79.3</td>
</tr>
<tr>
<td>ASP +FLAN-T5L</td>
<td>83.5</td>
<td>87.6</td>
<td>85.5</td>
<td>76.3</td>
<td>81.8</td>
<td>79.0</td>
<td>76.0</td>
<td>76.2</td>
<td>76.1</td>
<td>80.2</td>
</tr>
<tr>
<td>ASP +T03b</td>
<td>85.8</td>
<td>88.3</td>
<td>86.9</td>
<td>79.6</td>
<td>83.3</td>
<td>81.5</td>
<td>78.3</td>
<td>78.5</td>
<td>78.4</td>
<td>82.3</td>
</tr>
<tr>
<td>ASP +FLAN-T5XL</td>
<td>84.9</td>
<td>88.7</td>
<td>86.7</td>
<td>78.5</td>
<td>83.8</td>
<td>81.1</td>
<td>78.4</td>
<td>78.5</td>
<td>78.4</td>
<td>82.2</td>
</tr>
<tr>
<td>ASP +FLAN-T5XXL</td>
<td>86.1</td>
<td>88.4</td>
<td>87.2</td>
<td>80.2</td>
<td>83.2</td>
<td>81.7</td>
<td>78.9</td>
<td>78.3</td>
<td>78.6</td>
<td>82.5</td>
</tr>
</tbody>
</table>

Table 5: Full results on the CoNLL-12 English test set. Avg. F1 denotes the average F1 of MUC, B³, and CEAFφ₄. Models marked with † are our re-implementation. Other results are taken from their original papers.

B Coreference Resolution

In this section, we analyze the performance of mention detection for coreference resolution of our model in Fig. 2. This analysis casts light on how our model plans globally in an autoregressive manner. In the task of coreference resolution, only the entities that are mentioned more than once in a document are annotated as mentions. This is to say, an utterance of an entity should only be labeled if that entity is referred to again afterward. Thus, in previous coreference resolution models, a dedicated mention detection module that enumerates candidate textual spans (e.g., noun phrases and pronouns) for mentions is indispensable. However, our model is able to directly predict the exact set of mentions that we require, even if the target sequence is generated from left to right. We conclude that this results from the cross-attention mechanism which enables the model to look at relevant parts in the input document. Given an input document of |D| words, our model predicts only 0.096 |D| mentions with a 89.6% recall rate of gold mentions. This refrained mention detection strategy imposes a limit on the cardinality of Zᵢ in Eq. (2). As a result, this relatively small constant factor (compared to 0.4 used in most previous work) keeps our model tractable without the need for pruning strategies as in the models based on (Lee et al., 2017).

C Modeling More Restricted Structures

In this work, we tackled three tasks that are traditionally considered structured prediction problems. Named entity recognition and relation extraction consider labeling spans with a set of given types. Coreference resolution has long-range dependencies and has to model relationships between spans. However, there are structured prediction problems that require more restricted outputs. For instance, in dependency parsing, a spanning tree connecting every word in the input sentence is the desired output (Kübler et al., 2009). While in constituency parsing, a parse tree in Chomsky Normal Form is supposed to be a complete binary tree except for the leaf nodes (Wintner, 2010). Modeling such types of structures requires a more specified definition of task-specific actions. In future work, we aim to explore the abilities and limitations of our method.

D Experiments with Flan-T5

We conduct additional experiments with the latest pretrained language model Flan-T5 (Chung et al., 2022). Flan-T5 is pretrained on more supervised tasks and achieves better performance than the original T5 on multiple NLU tasks. The results are shown in Tab. 5, Tab. 6, and Tab. 7. We find that with the same size of the model, Flan-T5 performs better than T5 in general.
Table 6: Test F1 scores of named entity recognition on the CoNLL-03 test set.

<table>
<thead>
<tr>
<th></th>
<th>Prec.</th>
<th>Rec.</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASP + T5B</td>
<td>91.4</td>
<td>92.2</td>
<td>91.8</td>
</tr>
<tr>
<td>ASP + FLAN - T5B</td>
<td>92.7</td>
<td>93.8</td>
<td>93.3</td>
</tr>
<tr>
<td>ASP + T5L</td>
<td>92.1</td>
<td>93.4</td>
<td>92.8</td>
</tr>
<tr>
<td>ASP + FLAN - T5L</td>
<td>93.3</td>
<td>94.3</td>
<td>93.8</td>
</tr>
<tr>
<td>ASP + T53B</td>
<td>93.8</td>
<td>94.4</td>
<td>94.1</td>
</tr>
</tbody>
</table>

Table 7: Test F1 scores of named entity recognition on the CoNLL-04 test set.

<table>
<thead>
<tr>
<th></th>
<th>Ent</th>
<th>Rel</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASP + T5B</td>
<td>89.5</td>
<td>73.2</td>
</tr>
<tr>
<td>ASP + FLAN - T5B</td>
<td>89.4</td>
<td>73.8</td>
</tr>
<tr>
<td>ASP + FLAN - T5L</td>
<td>90.5</td>
<td>76.2</td>
</tr>
</tbody>
</table>

E Decoding Examples

We provide decoding examples from the tasks we experiment on in Tab. 8. The copy actions are verbalized into tokens.
GUNMEN WOUND TWO MANCHESTER UNITED FANS IN AUSTRIA. VIENNA 1996-12-06 Two Manchester United soccer fans were wounded by unidentified gunmen on Friday and taken to hospital in the Austrian capital, police said. "The four Britons were shot at from a Mercedes car at around 1 a.m.," a spokeswoman told Reuters. The two men were hit in the pelvis and leg. Police said their lives were not in danger. The fans, in Austria to watch their team play Rapid Vienna last Wednesday, may have been involved in a pub brawl earlier, the spokeswoman said. Manchester United won 2-0.<br>

And this final story: retired Senator Strom Thurmond has never made a secret about his fondness for young pretty women. Al Gore won't be the next U.S. President, but he has a slim chance of becoming the next President at Harvard. Gore holds a degree from the university, and is one of about 500 people nominated for the job. A school official talked about the Vice President's chances during an interview with the Boston Globe. He says it's unlikely Gore will be selected, because he doesn't have enough experience in the academic world.<br>

Violence between Israelis and Palestinians continued in its third month, though at a slightly reduced level overall. Israeli and Palestinian negotiators met separately at the White House with President Bill Clinton in hopes of restarting direct negotiations between them for a final settlement.<br>

Table 8: Predicted sequences from CoNLL-03, ACE-05, and CoNLL-12 dataset.