Autoregressive Structured Prediction with Language Models

Tianyu Liu[ς] Yuchen Eleanor Jiang[ς]
Nicholas Monath[γ] Ryan Cotterell[ς] Mrinmaya Sachan[ς]
[ς]ETH Zürich [γ]Google Research
{tianyu.liu, yuchen.jiang}@inf.ethz.ch
nmonath@google.com {ryan.cotterell, mrinmaya.sachan}@inf.ethz.ch

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.

https://github.com/lyutyuh/ASP

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[1] 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.

Framing structured prediction as conditional language modeling is an increasingly common black-box technique for building structured predictors that has led to empirical success (Vinyals et al., 2015; Raffel et al., 2020; Athiwaratkun et al., 2020; De Cao et al., 2021; Paolini et al., 2021, *inter alia*). 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

[1] Typically, large means exponential in the size of the input.

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 993–1005
December 7-11, 2022 ©2022 Association for Computational Linguistics

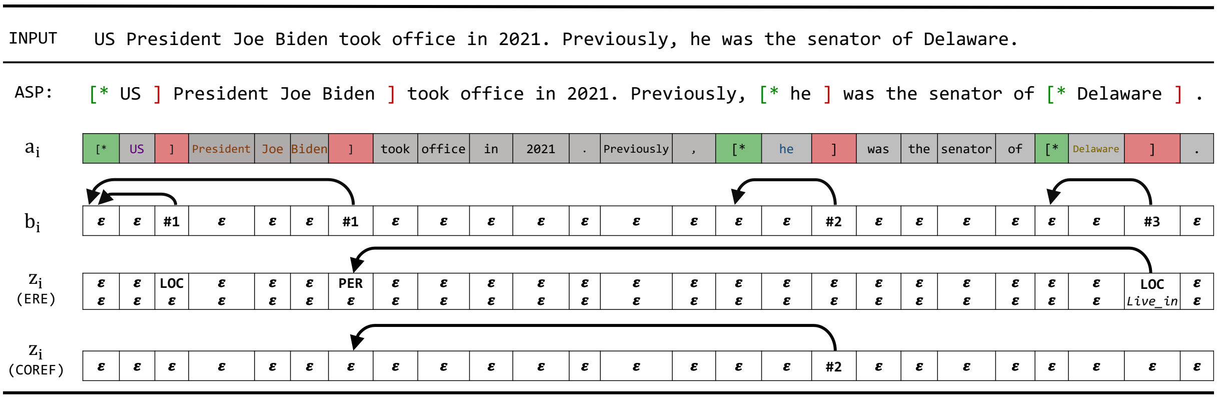
INPUT US President Joe Biden took office in 2021. Previously, he was the senator of Delaware.
ASP: [* US] President Joe Biden] took office in 2021. Previously, [* he] was the senator of [* Delaware] .
a_i
b_i
#1
#2
#3
z_i (ERE)
LOC
PER
Live_in
z_i (COREF)
ε

Figure 1: Illustration of the target outputs of our framework on coreference resolution (COREF) and end-to-end relation extraction (ERE). The lower part illustrates the decoding process of our model. The actions $\boldsymbol{y}_i$ are color-coded as], [* and copy. The structure random variables z_i are presented along with coreference links or relation links. We present words in the copy cells merely as an illustration.

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 $\boldsymbol{y} = y_1, \dots, y_N$, 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., $\mathcal{Y}_n \overset{\text{def}}{=} \mathcal{A} \times \mathcal{B}_n \times \mathcal{Z}_n$, 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., $y_n = \langle a_n, b_n, z_n \rangle$. We discuss each set in a separate paragraph below.

Structure-Building Actions. We first define a set of structure-building actions $\mathcal{A} = \left\{ \texttt{]}, \texttt{[}^*, \texttt{copy} \right\}$ that allow us to encode the span structure of a text, e.g., [**Delaware*] in Fig. 1 encodes that *Delaware* is a span of interest. More technically, the action] refers to a right bracket that marks the right-most part of a span. The action [* refers to a left bracket that marks the left-most part of a span. The superscript * on [* is inspired by the Kleene star and indicates that it is a placeholder for 0 or more consecutive left brackets[2]. Finally, 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, [**Delaware*], which is generated from a sequence of structure-building actions [*, 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.,

$$\mathcal{B}_n = \left\{ m \mid m < n \wedge a_m = \texttt{[}^* \right\} \qquad (1)$$

Thus, in general, $|\mathcal{B}_n|$ is $\mathcal{O}(n)$. 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 [* 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 $\mathcal{B}_n$. 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 $\mathcal{B}_n$ can be significantly smaller. When we decode action sequences $\boldsymbol{y}$ into a structure, unpaired `[*` and `]` 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

$$\mathcal{Z}_n = \Big\{ m \mid m < n \wedge a_m = \texttt{]} \Big\} \times \mathcal{L} \qquad (2)$$

where $\Big\{ m \mid m < n \wedge a_m = \texttt{]} \Big\}$ 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 $\mathcal{Z}_n$ 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 $D = \boldsymbol{w}_1, \ldots, \boldsymbol{w}_K$ be an input document of K sentences where $\boldsymbol{w}_k$ 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 $\boldsymbol{y}$ where $y_n \in \mathcal{Y}_n$. Now, we model the sequence of actions $\boldsymbol{y}$ as a conditional language model

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

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

$$p_{\boldsymbol{\theta}}(y_n \mid \boldsymbol{y}_{<n}, D) = \frac{\exp s_{\boldsymbol{\theta}}(y_n)}{\sum_{y'_n \in \mathcal{Y}_n} \exp s_{\boldsymbol{\theta}}(y'_n)} \qquad (4)$$

where $s_{\boldsymbol{\theta}}$ is a parameterized score function; we discuss several specific instantiations of $s_{\boldsymbol{\theta}}$ 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 $\boldsymbol{y}^*$ using a greedy decoding strategy. At decoding step n, we compute

$$y_n^* = \underset{y'_n}{\operatorname{argmax}}\, p_{\boldsymbol{\theta}}(y'_n \mid \boldsymbol{y}_{<n}, D) \qquad (5)$$

The chosen $y_n^* = \langle a_n^*, b_n^*, z_n^* \rangle$ will then be verbalized as a token as follows: If $a_n^* =$ `copy`, then we copy the next token from the input that is not present in the output. Otherwise, if $y_n^* =$ `[*` or $y_n^* =$ `]`, we insert `[*` or `]` 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 argmax $\boldsymbol{y}^*$.

Computational Complexity. Eq. (4) can be computed quite efficiently using our framework, as the cardinalities of $\mathcal{A}$ is $\mathcal{O}(1)$, and the size of $\mathcal{B}_n$ and $\mathcal{Z}_n$ are both $\mathcal{O}(n)$. A tighter analysis says the cardinalities of $\mathcal{B}_n$ and $\mathcal{Z}_n$ 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 $\Big\{ m \mid m < n \wedge a_m = \texttt{]} \Big\}$ 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 $\mathcal{E} = \{e_n\}_{n=1}^{|\mathcal{E}|}$ 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 $\mathcal{C}$. The function $s_{\boldsymbol{\theta}}(y_n)$ in Eq. (4) is implemented by a feed-forward network

$$\begin{aligned} & s_{\boldsymbol{\theta}}(y_n = \langle a_n, b_n, z_n \rangle) \\ & \overset{\text{def}}{=} \begin{cases} \mathrm{FFN}^{z_n}_{a_n}(\boldsymbol{m}_n) & \textbf{if } a_n = \boxed{]} \\ \mathrm{FFN}_{a_n}(\boldsymbol{h}_n) & \textbf{otherwise} \end{cases} \end{aligned} \tag{6}$$

where $\boldsymbol{h}_n$ is the decoder hidden state at step n, a column vector, and $\boldsymbol{m}_n = [\boldsymbol{h}_n^{\top}; \boldsymbol{h}_{b_n}^{\top}]^{\top}$ represents the mention that corresponds to y_n. Note that each $\mathrm{FFN}^{z_n}_{a_n}$ and FFN_{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 $\mathcal{C}$ and a set of pre-defined relations $\mathcal{R}$. 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 $\mathcal{C}$ and (ii) to identify all possible triples $\mathcal{T} = \{(e_n, r_n, e'_n)\}_{n=1}^{|\mathcal{T}|}$ in D where $e_n, e'_n \in \mathcal{E}$ are the head and tail entity and $r_n \in \mathcal{R}$ is the relation between e_n and e'_n. Here, the support of z_n takes the form of Eq. (2), where $\mathcal{L}$ is instantiated as $\mathcal{C} \times \mathcal{R}$. And $s_{\boldsymbol{\theta}}(y_n)$ 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

$$\mathcal{Z}_n = \left\{ m \mid m < n \wedge a_m = \boxed{]} \right\} \cup \{\epsilon\} \tag{7}$$

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 ϵ. Again, we define $s_{\boldsymbol{\theta}}(y_n = \langle a_n, b_n, z_n \rangle)$ as in Eq. (6) with the exception that, when $z_n = \epsilon$, we define $\mathrm{FFN}_{a_n}(\boldsymbol{m}_n)_{\epsilon} = \mathrm{FFN}_{a_n}(\boldsymbol{m}_n)$.

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 App. A.1 and App. A.2.

	Prec.	Rec.	F1
Ma and Hovy (2016)	91.4	91.1	91.2
Devlin et al.+BERT$_{\mathrm{L}}$	-	-	92.8
Ye et al.+ROBERTA$_{\mathrm{L}}$	-	-	94.0
Athiwaratkun et al.	-	-	91.5
Paolini et al.+T5$_{\mathrm{B}}$	-	-	91.7
ASP+T5$_{\mathrm{B}}$	91.4	92.2	91.8
ASP+T5$_{\mathrm{L}}$	92.1	93.4	92.8
ASP+T5$_{\mathrm{3B}}$	93.8	94.4	**94.1**

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

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.

[3] Ye 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.

[4] On 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.

	Ent	Rel
Eberts and Ulges (2020)	88.9	71.5
Zhao et al. (2020)	88.9	71.9
Wang and Lu+ALBERT$_{\text{XXL}}$	90.1	73.8
Paolini et al.+T5$_{\text{B}}$	89.4	71.4
ASP+T5$_{\text{B}}$	89.5	73.2
ASP+T0$_{\text{3B}}$	**90.3**	**76.3**

Table 2: **Micro** F1 scores of entity extraction and relation extraction on the CoNLL-04 joint entity relation extraction test set.

	Ent	Rel	Rel+
Wang and Lu+ALB$_{\text{XXL}}$	89.5	67.6	64.3
Zhong and Chen+ALB$_{\text{XXL}}$	90.9	69.4	67.0
Ye et al.+ALB$_{\text{XXL}}$[3]	91.1	72.4	70.3
Paolini et al.+T5$_{\text{B}}$	88.9	63.7	-
ASP+T5$_{\text{B}}$	90.7	71.1	68.6
ASP+T5$_{\text{L}}$	91.3	71.9	69.4
ASP+T5$_{\text{3B}}$[4]	**91.3**	**72.7**	**70.5**

Table 3: Test F1 scores of entity and relation extraction on the ACE-05 joint entity relation extraction task.

3.3 Coreference Resolution

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.

	MUC	B^3	CEAF$_{\phi_4}$	Avg. F1
Lee et al. (2017)	75.8	65.0	60.8	67.2
Joshi et al. (2020)	85.3	78.1	75.3	79.6
Joshi et al.+T5$_{\text{B}}$ †	79.8	70.2	66.8	72.3
Joshi et al.+T5$_{\text{L}}$ †	81.4	73.1	73.1	74.9
Urbizu et al.	64.9	66.5	65.3	65.6
Paolini et al.+T5$_{\text{B}}$	81.0	69.0	68.4	72.8
Dobrovolskii	86.3	79.9	76.6	81.0
ASP+T5$_{\text{B}}$	82.3	75.1	72.5	76.6
ASP+T5$_{\text{L}}$	84.7	77.7	75.2	79.3
ASP+T0$_{\text{3B}}$	86.9	81.5	78.4	82.3
ASP+FLAN-T5$_{\text{XXL}}$	87.2	81.7	78.6	**82.5**

Table 4: Results on the CoNLL-12 English test set. Avg. F1 denotes the average F1 of MUC, B^3, and CEAF$_{\phi_4}$. Models marked with † are our re-implementation. Other results are taken from their original papers. The full results are in Tab. 5.

4 Related Work

Most similar to our approach is the model of (Paolini et al., 2021), which also predicts structures in an iterative manner using conditional language models. Similar approaches exist for constituency parsing (Vinyals et al., 2015; Dyer et al., 2016), entity retrieval (De Cao et al., 2021), semantic parsing (Xiao et al., 2016), slot labeling, and intent classification (Athiwaratkun et al., 2020). Earlier work on search-based (Daumé et al., 2009; Doppa et al., 2014; Chang et al., 2015) and greedy-based approaches (Swayamdipta et al., 2016) applied to structured prediction also predict the structure in a sequential fashion as we do.Other work such as energy-based models (Belanger and McCallum, 2016; Tu and Gimpel, 2018, *inter alia*) and graphical models (Durrett and Klein, 2014; Ganea and Hofmann, 2017) predict structures more holistically.

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 $\approx$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

Ben Athiwaratkun, Cicero Nogueira dos Santos, Jason Krone, and Bing Xiang. 2020. Augmented natural language for generative sequence labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 375–385, Online. Association for Computational Linguistics.

David Belanger and Andrew McCallum. 2016. Structured prediction energy networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, page 983–992.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé, and John Langford. 2015. Learning to search better than your teacher. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, volume 37, pages 2058–2066.

Hyung Won Chung, Le Hou, S. Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Wei Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *ArXiv*, abs/2210.11416.

Hal Daumé, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*, 75(3):297–325.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *International Conference on Learning Representations*.

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.

Vladimir Dobrovolskii. 2021. Word-level coreference resolution. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7670–7675, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. 2014. Structured prediction via output space search. *Journal of Machine Learning Research*, 15(38):1317–1350.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics*, 2.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

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

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2619–2629, Copenhagen, Denmark. Association for Computational Linguistics.

Yoav Goldberg, Kai Zhao, and Liang Huang. 2013. Efficient implementation of beam-search incremental parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 628–633, Sofia, Bulgaria. Association for Computational Linguistics.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Michael Hahn. 2020. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171.

Yiding Hao, Dana Angluin, and Robert Frank. 2022. Formal language recognition by hard attention transformers: Perspectives from circuit complexity. *Transactions of the Association for Computational Linguistics*, 10:800–810.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada. Association for Computational Linguistics.

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

Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel Weld. 2019. BERT for coreference resolution: Baselines and analysis. 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 5803–5808, Hong Kong, China. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 2(1):1–127.

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.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Tianyu Liu, Yuchen Jiang, Ryan Cotterell, and Mrinmaya Sachan. 2022. A structured span selector. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2629–2641, Seattle, United States. 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. 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.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages. In *9th International Conference on Learning Representations, ICLR 2021*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea. 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. *Journal of Machine Learning Research*, 21(140):1–67.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. 2021. Multitask prompted training enables zero-shot task generalization.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Greedy, joint syntactic-semantic parsing with stack LSTMs. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 187–197, Berlin, Germany. 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.

Lifu Tu and Kevin Gimpel. 2018. Learning approximate inference networks for structured prediction. In *International Conference on Learning Representations*.

Gorka Urbizu, Ander Soraluze, and Olatz Arregi. 2020. Sequence to sequence coreference resolution. In *Proceedings of the Third Workshop on Computational Models of Reference, Anaphora and Coreference*, pages 39–46, Barcelona, Spain (online). Association for Computational Linguistics.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

C. Walker and Linguistic Data Consortium. 2005. *ACE 2005 Multilingual Training Corpus*. LDC corpora. Linguistic Data Consortium.

Jue Wang and Wei Lu. 2020. Two are better than one: Joint entity and relation extraction with table-sequence encoders. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1706–1721, Online. Association for Computational Linguistics.

Shuly Wintner. 2010. *Formal Language Theory*, chapter 1. John Wiley & Sons, Ltd.

Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. 2020. CorefQA: Coreference resolution as query-based span prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6953–6963, Online. Association for Computational Linguistics.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1341–1350, Berlin, Germany. Association for Computational Linguistics.

Deming Ye, Yankai Lin, Peng Li, and Maosong Sun. 2022. Packed levitated marker for entity and relation extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*

(Volume 1: Long Papers), pages 4904–4917, Dublin, Ireland. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 562–571, Honolulu, Hawaii. Association for Computational Linguistics.

Tianyang Zhao, Zhao Yan, Yunbo Cao, and Zhoujun Li. 2020. Asking effective and diverse questions: A machine reading comprehension based framework for joint entity-relation extraction. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3948–3954. International Joint Conferences on Artificial Intelligence Organization. Main track.

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.

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. 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.

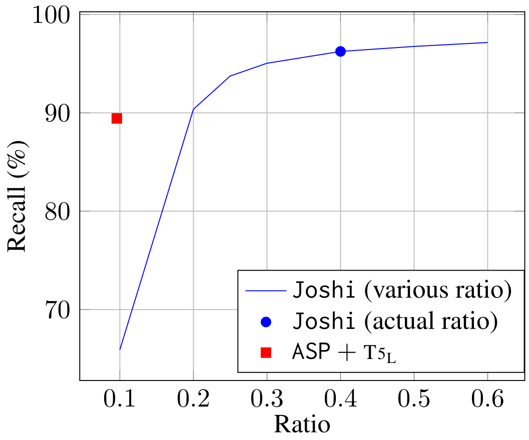

Figure 2: Recall rate of gold mentions. The ratio on the x-axis refers to the number of predicted mentions divided by $|D|$. Joshi refers to the two-stage model of (Joshi et al., 2020).

[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/

	MUC			B^3			CEAF$_{\phi_4}$			
	P	R	F	P	R	F	P	R	F	Avg. F1
Lee et al. (2017)	78.4	73.4	75.8	68.6	61.8	65.0	62.7	59.0	60.8	67.2
Lee et al. (2018)	81.4	79.5	80.4	72.2	69.5	70.8	68.2	67.1	67.6	73.0
Joshi et al. (2019)	84.7	82.4	83.5	76.5	74.0	75.3	74.1	69.8	71.9	76.9
Joshi et al. (2020)	85.8	84.8	85.3	78.3	77.9	78.1	76.4	74.2	75.3	79.6
Joshi et al.+T5$_\text{B}$ †	82.4	77.4	79.8	72.3	68.2	70.2	70.5	63.5	66.8	72.3
Joshi et al.+T5$_\text{L}$ †	85.5	77.7	81.4	78.3	68.5	73.1	75.0	65.9	73.1	74.9
Dobrovolskii	84.9	87.9	86.3	77.4	82.6	79.9	76.1	77.1	76.6	81.0
Urbizu et al.	-	-	64.9	-	-	66.5	-	-	65.3	65.6
Paolini et al.+T5$_\text{B}$	-	-	81.0	-	-	69.0	-	-	68.4	72.8
ASP+T5$_\text{B}$	81.7	82.8	82.3	74.2	76.1	75.1	74.5	70.6	72.5	76.6
ASP+T5$_\text{L}$	83.3	86.2	84.7	75.9	79.5	77.7	75.8	74.5	75.2	79.3
ASP+FLAN-T5$_\text{L}$	83.5	87.6	85.5	76.3	81.8	79.0	76.0	76.2	76.1	80.2
ASP+T0$_\text{3B}$	85.8	88.3	86.9	79.6	83.3	81.5	78.3	78.5	78.4	82.3
ASP+FLAN-T5$_\text{XL}$	84.9	88.7	86.7	78.5	83.8	81.1	78.4	78.5	78.4	82.2
ASP+FLAN-T5$_\text{XXL}$	86.1	88.4	87.2	80.2	83.2	81.7	78.9	78.3	78.6	**82.5**

Table 5: Full results on the CoNLL-12 English test set. Avg. F1 denotes the average F1 of MUC, B^3, and CEAF$_{\phi_4}$. 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 during decoding. 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 $\mathcal{Z}_i$ 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.

	Prec.	Rec.	F1
ASP+T5$_{\text{B}}$	91.4	92.2	91.8
ASP+FLAN-T5$_{\text{B}}$	92.7	93.8	93.3
ASP+T5$_{\text{L}}$	92.1	93.4	92.8
ASP+FLAN-T5$_{\text{L}}$	93.3	94.3	93.8
ASP+T5$_{\text{3B}}$	93.8	94.4	**94.1**

Table 6: Test F1 scores of named entity recognition on the CoNLL-03 test set.

	Ent	Rel
ASP+T5$_{\text{B}}$	89.5	73.2
ASP+FLAN-T5$_{\text{B}}$	89.4	73.8
ASP+FLAN-T5$_{\text{L}}$	90.5	76.2

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

E Decoding Examples

We provide decoding examples from the tasks we experiment on in Tab. 8. The `copy` actions are verbalized into tokens.

named entity recognition	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.</s>
end-to-end relation extraction	And this final story: retired [* Senator] [* Strom Thurmond] has never made a secret about [* his] fondness for young pretty [* women] .</s>
coreference resolution	<speaker> - </speaker> [* 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.</s>
	<speaker> - </speaker> [* 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.</s>

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