ITER: Iterative Transformer-based Entity Recognition and Relation Extraction

Anonymous NAACL-HLT 2021 submission

Abstract

Entity Recognition and Relation Extraction are essential components in extracting structured information from text. Recent advances for both tasks generate a structured representation of the information in an autoregressive fashion, a time-intensive and computationally expensive approach. This raises the natural question whether autoregressive methods are necessary in order to achieve comparable results. In this work, we propose ITER, a functionally more expressive, non-autoregressive model, that unifies several improvements to a recent language modeling approach: ITER improves inference throughput by up to 23×, is capable of handling nested entities and effectively halves the number of required parameters in comparison. Furthermore, we achieve a SOTA result of 84.30 F1 for the relation extraction dataset ADE and demonstrate competitive performances for both named entity recognition with GENIA and CoNLL03 as well as for relation extraction with CoNLL04 and NYT.

1 Introduction

In recent years, there has been a shift towards using autoregressive methods in many common NLP tasks. Parallel to this development is an increasing focus on approaching NLP tasks such as relation extraction or (nested) named entity recognition as structured prediction problems. Given a sequence of text input, a given model autoregressively generates outputs that encode the structure contained within the input, which offers flexibility since the source and target vocabulary must not share any commonalities.

Flattening the output structure into a single string, preserving the information about the structure(s) in the input and using an autoregressive model to learn to generate this adapted target language (Cabot and Navigli, 2021; Paolini et al., 2021), is an *implicit* approach known to work well across task boundaries (Raffel et al., 2020). In this case, the target vocabulary typically contains the whole source language vocabulary. However, representing the structured output as a string introduces additional complexity when modeling intra-structure dependencies (Liu et al., 2022). More recently, Liu et al. proposed constraining the autoregressive model to *explicit* generation of the output structure. They define three types of basic actions to be performed at each generation step and use the T5 (Raffel et al., 2020) transformer to autoregressively generate the structure induced by said basic actions.

With this trend of using autoregressive methods for tasks such as relation extraction come however also several problems: As inference time scales linearly with the output sequence length, language modeling approaches are prone to low inference speed[1] especially with increasing model parameters (Pope et al., 2022). While scaling the model size from hundreds of millions of parameters to billions of parameters yields performance increments for Liu et al., this scaling can become infeasible, both in terms of compute required and the environmental impact when using those large scale models in production.

This raises the natural question whether a *non-autoregressive process* capable of generating such an output structure can achieve similar performance whilst addressing the aforementioned limitations of language modeling approaches. In this paper, we present ITER, an encoder-only transformer-based relation extraction model that addresses the limitations of state-of-the-art architectures and show that the structured prediction problem can be approached without a language modeling objective in mind.

To summarize, our key contributions are the following:

1. We present ITER, a transformer-based encoder-

[1] In terms of samples/second

only relation extraction model that addresses the limitations of autoregressive architectures. Instead of employing a language modeling objective, our model generates the structured output in three basic steps. We show that this encoder-based approach achieves performance similar to language modeling architectures whilst retaining only half of the number of parameters and increasing the inference throughput by a factor of up to 23×.

2. We identify several drawbacks with the state-of-the-art architecture ASP (Liu et al., 2022), which limit the model's expressivity for nested structures and generalization to test data. In our work, we address those limitations and translate the *autoregressive* approach into a three-stage process.

3. In our experiments, we observe that training ITER on NYT, CoNLL04, ADE (RE), or GENIA, CoNLL03 (NER) results in competitive performance across all datasets, while maintaining a significantly smaller size compared to SOTA models. We observe small average improvements of 0.5 F1 points on the ADE dataset.

4. We publish our implementation and checkpoints at GITHUB.COM/ANONYMOUS.

2 Related Work

The goal of relation extraction, sometimes also referred to as *end-to-end* relation extraction or *joint* entity and relation extraction, is to identify the names and types of *named entities*, inside a given text, as well as classify the *relationships* amongst these entities. (Grishman and Sundheim, 1996; Zhao and Grishman, 2005).

First approaches to relation extraction were to decompose the task into *named entity recognition* and *relation classification*, where the named entities are identified first, while the relationships between the found named entities are then classified in a second, separate stage that is being learned independently. This pipeline-based approach is known to be prone to error propagation (Zhong and Chen, 2021; Sui et al., 2020). Because of this known limitation, joint approaches modeling both tasks simultaneously have been introduced and have shown promising results (Gupta et al., 2016; Wang and Lu, 2020).

2.1 Span-based Techniques

Table-filling or span-based strategies were and still are viable approaches to modeling relation extraction and related tasks (Gupta et al., 2016; Wang and Lu, 2020; Joshi et al., 2020; Tang et al., 2022). Recent examples of this include DiffusionNER (Shen et al., 2023) and UniRel (Tang et al., 2022), both are models that do not emit a time complexity scaling linearly with the output size, which in turn enables fast inference. Instead, a constant time-complexity is achieved in both cases, as a diffusion based approach solely depends on the number of diffusion steps and a single forward-pass is required to detect relationships in UniRel. This strongly differs from *autoregressive* techniques, where the inference time is scaling linearly with the length of the output sequence (Shen et al., 2023).

2.2 Autoregressive Techniques

Modeling the task as a *seq2seq* problem has established itself as the state-of-the-art for relation extraction in the last couple of years (Cabot and Navigli, 2021; Wang et al., 2022; Paolini et al., 2021; Liu et al., 2022). Enabled by the Transformer (Vaswani et al., 2017), the task is then formulated as a translation objective: Given an example sentence, the model translates the input into a flattened string that encodes the structural information contained within the source text (Liu et al., 2022).

Both (m)REBEL (Cabot and Navigli, 2021; Cabot et al., 2023) and TANL (Paolini et al., 2021) translate the input sequence into a flattened output string, that, in the (m)REBEL case, also no longer resembles natural language. Paolini et al. augment the target output with information about entity types and relations to other named entities. Both models are finetuned to produce a target language specific to the task. A comparison of different model outputs is available in Table 1. Either model can also deal with nested entities, which is crucial when dealing with real-world data, as for example data from the biomedical domain is known to often contain nested entities (Finkel and Manning, 2009).

2.3 Limitations of Autoregressive Structured Prediction (ASP)

ASP (Liu et al., 2022) has shown that autoregressively generating a sequence of *actions* instead of generating (augmented) natural language yields

Model	Output
REBEL	<triplet> Barack Obama <subj> Honolulu, Hawaii <obj> place of birth
TANL	[Barack Obama | *person* | *place of birth* = Honolulu, Hawaii] was born in [Honolulu, Hawaii | *location*]
ASP	[* Barack Obama] was born in [* Honolulu, Hawaii]
ITER *(ours)*	*Barack Obama* was born in *Honolulu, Hawaii* [ ] [ ]

Table 1: Comparison of different (autoregressive) relation extraction system outputs: REBEL (Cabot and Navigli, 2021), TANL (Paolini et al., 2021) and ASP (Liu et al., 2022). Input into all models was the sentence "Barack Obama was born in Honolulu, Hawaii". Comparing ASP and our method, one can observe that ASP requires [* and] actions to be placed alongside the input to model the same structure of the input.

model performance benefits not only for relation extraction, but for named entity recognition and co-reference resolution as well. At every generation step, their model can perform three distinct types of actions, *structure-building* actions, *bracket-pairing* and *span-labeling* actions. For the *structure-building actions* (Eq. 1), the model can either perform [* or] actions at the current generation position or `copy` the next token from the input into the output.

$$\mathcal{A}^{ASP} = \{\texttt{[*}, \texttt{]}, \texttt{copy}\} \tag{1}$$

The generation completes when all input tokens have been copied into the output. *Bracket-pairing* actions (Eq. 2) aim to connect the current position with a previously performed [* action, resulting in a span.

$$\mathcal{B}^{ASP} = \{m \mid m < n \wedge a_m = \texttt{[*}\} \tag{2}$$

Span-labeling actions allow both the labeling of individual spans and the linking of the current formed span to an earlier found span in the output sequence, modeling relationships between named entities (Eq. 3). For relation extraction, $\mathcal{L}$ is instantiated as the cartesian product of the named entity and relation types: $\mathcal{L} = \mathbf{T}_E \times \mathbf{T}_R$.

$$\mathcal{Z}_n = \{m \mid m < n \wedge a_m = \texttt{]}\} \times \mathcal{L} \tag{3}$$

The authors of ASP employ a conditional language model to learn to produce the optimal output structure. At every time-step their model will perform three basic actions sourced from their respective defined sets for *structure-building* actions $\mathcal{A}$, *bracket-pairing* actions $\mathcal{B}_n$ and *span-labeling* actions $\mathcal{Z}_n$.

Said approach however is not capable of capturing *nested entities*. At every time-step, ASP can only complete one span with one preceding [* action due to the definition of $\mathcal{B}_n$.

We also hypothesize that the structured prediction process for ASP suffers from suboptimal training due to the nature of the *span-labeling* actions $\mathcal{Z}_n$. Linking the span formed at the current position to another span in the sequence is constrained by the fact that only links to spans that have been completed in the past (i.e. earlier in the sequence) are valid. As it is impossible to link to spans that will be found in the future (i.e. spans that come after the span ending at the current position), the authors of ASP introduce a directionality parameter to counteract the asymmetric property of the relations in the dataset.

This prevents the two tuples (*Barack Obama*, `work_for`, *the american people*) and (*the american people*, `work_for`, *Barack Obama*) from being indistinguishable.

This is important as those two examples encode drastically different information. The directionality parameter however effectively doubles the number of relations ($\mathbf{T}'_R = \mathbf{T}_R \times \mathbb{B}$), leading to fewer training examples per relation type, and we hypothesize that this yields subpar training results.

Aside from those architectural issues, ASP and similar *seq2seq* Transformer models such as TANL or REBEL all suffer from the linearly scaling time-complexity of generative architectures, significantly impacting their inference speed (Paolini et al., 2021). This raises the question whether *eliminating* the requirement for a conditional language model from ASP can retain the same model performance whilst crucially reducing inference time, allowing for the identification of nested entities and addressing the generalization issues.

3 Approach

We base our approach for ITER on the work(s) of Liu et al.. In order to translate their autoregressive approach into a constant-in-time approach, several adjustments to the structured prediction process are necessary.

The overall objective remains the same: producing a sequence of structured actions $\mathbf{y} \in \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_N$ that encodes the named entities and their relations between each other, given a sequence of N tokens $\mathbf{x} = \langle x_1, x_2, \ldots, x_N \rangle \in \mathcal{V}^N$ from the vocabulary $\mathcal{V}$. However, as we restrain from the autoregressive approach, we require $|\mathbf{x}| = |\mathbf{y}|$, i.e. the length of the structured actions sequence must match the length of the input, such that there exists exactly one structured action y_n for each input token x_n. This restriction allows to use non-autoregressive Transformer approaches, because we no longer have to deal with output sequences $\mathbf{y}$ longer than the input $\mathbf{x}$, on the contrary to ASP (Theorem 1).

The first necessary change is to transition to a smaller subset of the *structure-building* actions $\mathcal{A}$ (Eq. 4).

$$\mathcal{A} = \{ [\, , \,] \} \tag{4}$$

Our model must be allowed to perform both [and] actions at the same time, to not lose model expressiveness, otherwise it will not be able to correctly classify single-token spans[2]. Therefore, the *structure-building* actions A_n performed at every position n must now be a subset of $\mathcal{A}$, to allow for this (behavior|functionality). This is reflected in the definition for our structured output $\mathbf{y}$ (Eq. 5)[3].

$$\forall n : y_n \in \wp(\mathcal{A}) \times \wp(\mathcal{B}) \tag{5}$$

optional: The changes for (the *structure-building* actions/$\mathcal{A}$) have another simplifying consequence: we must no longer `copy` tokens from the input, as every token has its own corresponding set of actions $A_n \subseteq \mathcal{A}$/ as the original inputs $\mathbf{x}$ can be maintained for decoding.

To be able to properly handle nested entities, or, more specifically, two or more entities ending at the same position, the *bracket-pairing* actions are also present in $\mathcal{Y}_n$, as a subset B_n of all possible such actions $\mathcal{B}_n$. This change comes in combination with two adjustments to the definition of $\mathcal{B}_n$ itself.

For position n, ITER will be allowed to pair with [actions up until that position n, circumventing single-token named entity issues (1, Eq. 6) and each pairing is allowed to hold its own named entity type $t \in \mathbf{T}_E$ (2, Eq. 6).

$$\mathcal{B} = \{m \mid m \overset{(1)}{\leq} n \wedge [\in A_m\} \overset{(2)}{\times} \mathbf{T}_E \tag{6}$$

3.1 Identifying Named Entities

Remember that the predicate for relation extraction is the necessity to locate the named entities in the input, before relationships amongst those can be determined. Spans can be uniquely identified by their starting and end positions in combination with the type of the named entity in the input sequence.

3.1.1 Determining Starting Positions of Named Entities

To identify said spans, as a first step, the model learns to predict the positions where the spans of named entities in the input $\mathbf{x}$ are beginning. This task is modeled by the function *is_left* (Eq. 7), which receives a sequence of tokens $\mathbf{x}$ as input and outputs an equally sized sequence of Boolean values $\langle b_1 \ldots b_N \rangle \in \mathbb{B}^N$:

$$\textit{is\_left} : \mathcal{V}^N \to \mathbb{B}^N. \tag{7}$$

At all positions where $b_n = \top$ holds true, ITER performs the *left bracket* action [, and as such the corresponding action is included in the set of actions performed at position n: $\textit{is\_left}(\mathbf{x})_n = \top \implies [\in A_n$.

3.1.2 Pairing Left and Right Brackets

After determining where spans of named entities start in the input $\mathbf{x}$, the next step is to identify which positions $\mathbf{x}_m$ $(m \geq n)$ following $\mathbf{x}_n$ in the input form a span of named entity type $t \in \mathbf{T}_E$, for any n, m where $b_n = \top$ and $m > n$.

$$\textit{is\_span} : \mathcal{V}^N \times \mathbb{B}^N \to \wp(\mathbb{N} \times \mathbf{T}_E)^N \tag{8}$$

The model learns a projection *is_span*, that maps the input $\mathbf{x}$ and positions m (where $[\in A_m \implies b_m = \checkmark$) to a sequence of tuples of indices and entity types (m, t). For each position n, the output of $B_n = \textit{is\_span}(\mathbf{x}, \mathbf{b})_n$ determines whether or not the respective positions $\langle x_m \ldots x_n \rangle$ form a span of type t with a preceding left bracket [at position m iff. the left bracket at position m is paired with the right bracket at position n and with type $t \in \mathbf{T}_E$, i.e. $(m, t) \in B_n$. If $|B_n| > 0$, then ITER performs action] at position n, i.e. $] \in A_n$. A visualization of the first two stages is shown in Figure 1.

[2] Think of a single-token named entity x_i = BERLIN: the model must be able to determine the span of this entity ends at the same position it started., so a_i must now be a set: $a_i = \{ [,] \}$.

[3] Here, $\wp$ defines the powerset operation

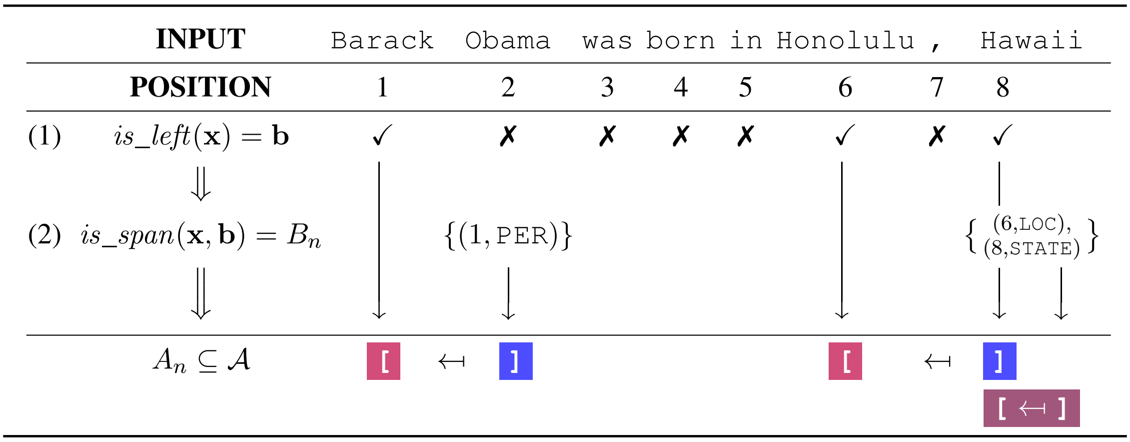

Figure 1: Visualization of stages one (*is_left*) and two (*is_span*) of the model. *is_left* yields three positions where spans are beginning: 1,6 and 8 (Stage 1). *is_span* then creates pairings of types *person* between position 2 and 1, *location* between position 8 and 6 and *state* for position 8, a 1-token span (indicated by [↤]).

3.2 Identifying Relationships amongst Named Entities

While the first two steps are concerned/dealing with identifying named entities in the input $\mathbf{x}$, the third step now tests pairs of identified named entities for their relationship with each other. For the non-nested case, is_link projects the input $\mathbf{x}$ alongside two indices n_1, n_2, where two spans in the input $\mathbf{x}$ are ending, i.e.] $\in A_{n_1}, A_{n_2}$, to a vector of non-normalized logits, resembling probabilities after applying the sigmoid function (Eq. 9).

$$is\_link : \mathcal{V}^N \times \mathbb{N} \times \mathbb{N} \to \mathbb{R}^{|\mathbf{T}_R|} \tag{9}$$

This now allows to test for relationships between pairs of named entities, identified in steps one and two: $is\_link(\mathbf{x}, n_1, n_2) = (t_1 \dots t_\rho)^T = \mathbf{t} \in \mathbb{R}^\rho$ where $\rho = |\mathbf{T}_R|$. For all $\mathbf{t}_i$ where $\sigma(\mathbf{t}_i) > 0.5$, the named entity ending at n_1 (also referred to as head entity) stands in relationship $\mathbf{T}_{R,i}$ with the named entity (tail entity) ending at position n_2. Note that this relationship is not symmetric, i.e. the ordering of head and tail entity is important: $is\_link(\mathbf{x}, n_1, n_2) \neq is\_link(\mathbf{x}, n_2, n_1)$

When dealing with more complex inputs, is_link must incorporate information about the positions of the left brackets as well, as spans are no longer uniquely determined by their ending position. The updated signature is shown in Eq. 10 This final step is visualized in Figure 2.

$$is\_link : \mathcal{V}^N \times \mathbb{N}^2 \times \mathbb{N}^2 \to \mathbf{T}_R \tag{10}$$

3.3 Training

The model of choice for this paper is the T5 (Raffel et al., 2020) Transformer architecture, which can also be used as an encoder, albeit primarily trained for autoregressive applications (Raffel et al., 2020). In order to circumvent error propagation between the three stages of ITER, training will include all three task functions simultaneously: is_left, is_span and is_link. ITER receives as input a sequence of hidden representations (*hidden states*) $\mathbf{h} = \langle h_1, h_2, \dots, h_N \rangle$, produced by the base transformer encoder (T5 in our case), instead of the raw sequence of tokens $\mathbf{x} \in \mathcal{V}^N$. The sequence of representations is shared across all three tasks. During training, the model will learn to minimize the following loss function:

$$L_{\text{ITER}} = \left(\sum_{i=1}^{N} \sum_{j=1}^{3} \begin{bmatrix} L_{is\_left}(i) \\ L_{is\_span}(i) \\ L_{is\_link}(i) \end{bmatrix}_j \right),$$

a combination of the loss terms for the three individual tasks: L_{is_left}, L_{is_span} and L_{is_link} (Appendix, Eq. 12,13 and 14). In a nutshell, in order to minimize the training loss, the model learns to assign weights greater than zero to the respective correct decisions in all three cases.

4 Experimental Results

In this section, we give an overview over the datasets used in our experiments (Section 4.1), followed by details about the hyperparameter search we performed (Section 4.2) and we conclude with our interpretation of the results from said experiments (Section 4.3).

FUNCTION	HEAD ENTITY	TAIL ENTITY	OUTPUT
$is\_link(\mathbf{x}, (1,2), (6,8))$	Barack Obama $(1,2)$	Honolulu, Hawaii $(6,8)$	*live_in* *born_in* *work_in* $(\ 0.05\ ,\ 0.98\ ,\ 0.01\ )$
$is\_link(\mathbf{x}, (1,2), (8,8))$	Barack Obama $(1,2)$	Hawaii $(8,8)$	*live_in* *born_in* *work_in* $(\ 0.17\ ,\ 0.45\ ,\ 0.02\ )$
$\vdots$	$\vdots$		$\vdots$
$is\_link(\mathbf{x}, (6,8), (8,8))$	Honolulu, Hawaii $(6,8)$	Hawaii $(8,8)$	*live_in* *born_in* *work_in* $(\ 0.03\ ,\ 0.07\ ,\ 0.1\ )$

Figure 2: Visualization of the third stage of the model. The output is already sigmoid-normalized. For illustration purposes, there are three relation types: $\mathbf{T}_R = \langle live\_in, born\_in, work\_in \rangle$. The named entity "Barack Obama" stands in relationship "born_in" with "Honolulu, Hawaii", as $0.98 > 0.5$, which is the criterion defined for this model.

4.1 Data

To experimentally verify that our model can achieve performances on par or even higher than the baseline from ASP, we used 5 datasets from two different domains and tasks: CoNLL03 (Sang and Meulder, 2003, NER) (NER), CoNLL04 (Roth and Yih, 2004, RE) and NYT (Riedel et al., 2010, RE) were all annotated from news articles while ADE (Gurulingappa et al., 2012, RE) and GENIA (Kim et al., 2003, NER) contain training examples with biomedical context. Of those five datasets, three contain nested entities (NYT, ADE and GENIA), something that ASP cannot properly model, as shown in Section 2.3, which was another factor for our selection. This portfolio of datasets allows us to verify our claims across a wide range of applications and different levels of data complexity. An overview regarding the datasets can be obtained in Table 8. Following Li et al.; Eberts and Ulges and Cabot and Navigli, we evaluate our model in a *strict* setting: A predicted relation between two entities is only considered correct, if both the span and type of the entity match the gold standard. We report *micro* F1 scores, unless stated otherwise.

4.2 Hyperparameter search

Before training all of our models, we perform a hyperparameter search for all datasets using SMAC3 (Lindauer et al., 2022). For all datasets, we search for 8 hours, optimizing for high RE+ or NER F1, depending on the task. The search space consists of learning rates $lr \in [1\mathrm{e}{-3}, 2\mathrm{e}{-5}]$, learning rate schedules (*constant* or *linear*), warmup ratio $r \in \{0.0, 0.05, 0.1, 0.2\}$ and weight decay rate $wd \in [0, 0.1]$ for both the parameters of the base model (T5 in our case) and the parameters on top that are responsible for modeling the functions *is_left*, *is_span* and *is_link*, combined with batch size $bs \in \{8, 16, 32, 64\}$ and choice of activation function $act \in \{\mathrm{GELU}, \mathrm{ReLU}, \tanh\}$. The results of the hyperparameter search can be obtained in Table 7 in the Appendix.

4.3 Results

With the encoder of the FLAN-T5-large model as a base, ITER achieves state-of-the-art results on ADE with on average 0.5 F1 points of improvement (Table 3). Furthermore, it reaches competitive results to most generative approaches while the number of parameters is significantly smaller (Table 2, 4, 6). Specifically, its performance closely aligns with that of ASP+FLAN T5 base and ASP+FLAN T5 large, both of which possess a similar parameter count, with the latter having twice the parameters and only being slightly better. Table 5 answers another research question, which was to demonstrate that a higher inference speed can be obtained with a smaller model while reaching comparable results. Especially compared with DeepStruct our model performs well, considering its size and training time. DIFFUSIONNER performs exceptionally well, and we are not able to match its performance on the NER task, only coming close on CoNLL03. Again, supporting our hypothesis that encoder-only models —like DIFFUSIONNER—can outperform generative models like DeepStruct on structure prediction tasks.

To further improve our understanding of ITER and its shortcomings, we analyse ADE using confusion matrices. Figure 3 shows that our model does not struggle with the span-prediction task. The model also learned to predict the actions [and] at the same step where appropriate. The main challenge seems to be the named entity type *Adverse-*

Effekt, which is falsely predicted and missed several times.

Architecture	**Size**	**NER F1** (strict)	**RE F1** (strict)
ITER + FLAN T5 $_{large}$	374 M	89.770 ± 0.51	75.175 ± 0.39
ASP + FLAN T5 $_{base}$	247 M	89.4	73.8
ASP + FLAN T5 $_{large}$	783 M	90.5	76.2
TANL	222 M	89.8	72.6
REBEL *(pretrained)*	406 M	-	75.4
DeepStruct	10 B	88.4	72.8
DeepStruct *(finetuned)*	10 B	90.70	78.3
DiffusionNER	381M	92.78	

Table 2: Final training results for CoNLL04, averaged across five runs for each configuration.

Architecture	**NER F1** (strict)	**RE F1** (strict)
ITER + FLAN T5 $_{large}$	91.907 ± 0.72	84.300 ± 1.52
TANL *(multi-task)*	91.2	83.8
REBEL *(pretrained)*	-	82.2
DeepStruct *(finetuned)*	91.1	83.8

Table 3: Final training results for ADE with 10-fold cross-validation. F1 metrics are *macro*-averaged.

Architecture	**NER F1** (strict)	**RE F1** (strict)
ITER + FLAN T5 $_{large}$	94.726 ± 0.16	90.707 ± 0.34
TANL *(multi-task)*	94.7	90.7
REBEL	-	92.0
DeepStruct *(multi-task)*	95.4	93.7

Table 4: Final training results for NYT.

Transformer base model	**ASP** examples/s	**ITER** examples/s	**Speedup** over ASP
T5 (small)	44.459	1040.55	×23.41
T5 (large)	27.177	605.398	×22.28
T5 (3b)	29.427	334.843	×11.38

Table 5: Comparing the inference throughput (as *samples per second*) of ITER versus the autoregressive approach ASP (Liu et al., 2022) on the test set of CoNLL04. Computations were run on a single NVIDIA H100 GPU, a batch size of 64 combined with 10 epochs of training beforehand. Speedups of up to ×23.41 (×18.6 on avg.) are achieved when using ITER instead of ASP.

Architecture	**Dataset**	**F1** *(strict)*
ITER + FLAN T5 $_{large}$	CoNLL03	91.593 ± 0.39
ASP + T5 $_{base}$		91.8
DeepStruct *(multi-task)*		93.10
DiffusionNER		92.78
ITER + FLAN T5 $_{large}$	GENIA	80.153 ± 0.25
TANL *(multi-task)*		76.4
DeepStruct *(finetuned)*		80.8
DiffusionNER		81.53

Table 6: Final training results for CoNLL03 and GENIA. For CoNLL03, we trained ITER with a linear learning rate schedule, instead of the SMAC3 prediction to use a constant schedule, as the final performance did significantly degrade using a constant schedule.

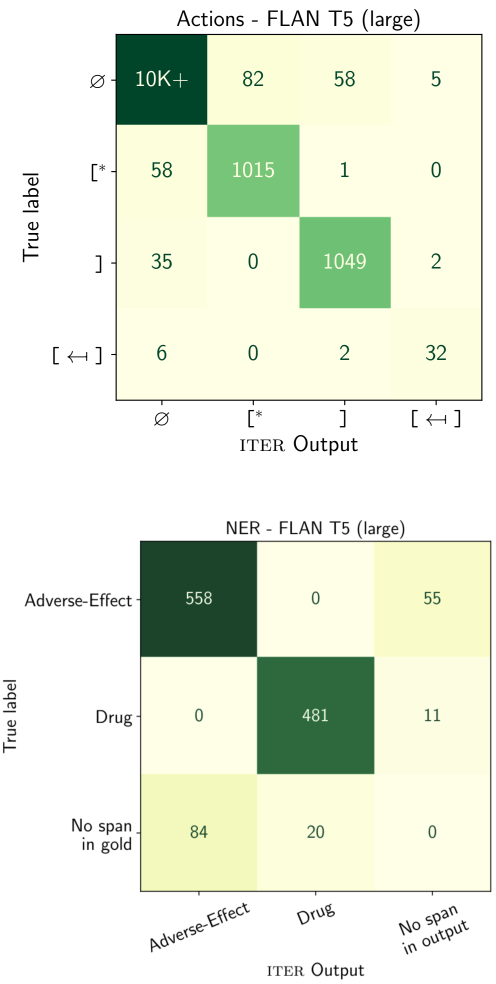

Figure 3: Confusion matrix for actions $\mathcal{A}$ and NER on ADE.

5 Limitations

One of ITER's limitations are named entities that are not directly contained in the input text. This issue can arise when combining the NER stage of ITER with tasks such as entity linking, where the task is then to not only identify the named entity and its type, but also to link said entity to a knowledge base entity, something particularly interesting when using a relation extraction pipeline to create knowledge graphs. The severity of this limitation strongly depends on the datasets used, we focussed on experiments with datasets where this issue cannot surface.

If it is the case that the input has not been preprocessed, our model also requires a very tedious preprocessing-step that requires the programmer to correctly align the input string with the tokens that the model will be trained on. This is a limitation of the *sentencepiece* (Kudo and Richardson, 2018) tokenizer used in our experiments, as the tokenization process does not guarantee entity-level boundaries being respected during tokenization, meaning that a token spanning the characters i to j might contain the beginning of a span k ($i < k < j$). While generative approaches can circumvent this problem by introducing additional tokens into the target language text, encoder-based approaches such as our work are limited to dealing with this issue pre-tokenization.

Another limitation of ITER would be the strong task-dependent design of the functions is_left, is_span and is_link. This prevents a few-shot task transfer without finetuning for new relations or entity types.

6 Conclusion

In this paper, we identified several key drawbacks in a fairly recent state-of-the-art method for relation extraction, ASP, and proposed several improvements to counteract those issues. We investigated whether it is possible to translate the autoregressive process into a constant-in-time-complexity approach, whilst maintaining an equal level of performance.

We unify the aforementioned proposed improvements together with a new three-stage process in ITER, an encoder-based *non-autoregressive* relation extraction model. Our model achieves performances on par with state-of-the-art methods on all datasets and sets a new state-of-the-art on ADE of 84.3 F1, whilst being functionally more expressive and reducing inference time significantly, when compared to ASP. In our experiments, we highlight the time saving benefits of encoder-based models over autoregressive *seq2seq* approaches, suggesting that they are just as viable in a structure prediction task.

7 Future Work

While our model is currently built on top of a T5 encoder stack, it might be insightful to explore the performance of this architecture with other pretrained (encoder-only) language models such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) or the Nyströmformer (Xiong et al., 2021)

One area of future work might be exploring an even larger set of datasets. Unfortunately, there exist no benchmark suites for relation extraction, which itself might be an area of future work. While most datasets are open-source, there exist proprietary datasets, preventing the democratization of research in NLP and in machine learning in general. Evaluating architectures on a diverse portfolio of datasets instead of a limited amount of selected or hand-picked datasets should also allow to gain more significant insights into the performance, capabilities and limitations of relation extraction systems in general.

References

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. REBEL: relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 2370–2381. Association for Computational Linguistics.

Pere-Lluís Huguet Cabot, Simone Tedeschi, Axel-Cyrille Ngonga Ngomo, and Roberto Navigli. 2023. Red$^{\mathrm{fm}}$: a filtered and multilingual relation extraction dataset. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 4326–4343. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Markus Eberts and Adrian Ulges. 2020. Span-based joint entity and relation extraction with transformer pre-training. In *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, pages 2006–2013.

Jenny Rose Finkel and Christopher D. Manning. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, 6-7 August 2009, Singapore, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 141–150. ACL.

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference- 6: A brief history. In *16th International Conference on Computational Linguistics, Proceedings of the Conference, COLING 1996, Center for Sprogteknologi, Copenhagen, Denmark, August 5-9, 1996*, pages 466–471.

Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 2537–2547.

Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. 2012. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *J. Biomed. Informatics*, 45(5):885–892.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Trans. Assoc. Comput. Linguistics*, 8:64–77.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. In *Proceedings of the Eleventh International Conference on Intelligent Systems for Molecular Biology, June 29 - July 3, 2003, Brisbane, Australia*, pages 180–182.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, pages 66–71.

Fei Li, Meishan Zhang, Guohong Fu, and Donghong Ji. 2017. A neural joint model for entity and relation extraction from biomedical text. *BMC Bioinform.*, 18(1):198:1–198:11.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. 2022. SMAC3: A versatile bayesian optimization package for hyperparameter optimization. *J. Mach. Learn. Res.*, 23:54:1–54:9.

Tianyu Liu, Yuchen Eleanor Jiang, Nicholas Monath, Ryan Cotterell, and Mrinmaya Sachan. 2022. Autoregressive structured prediction with language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 993–1005. Association for Computational Linguistics.

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

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cícero 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, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2022. Efficiently scaling transformer inference. *CoRR*, abs/2211.05102.

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

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part III*, volume 6323 of *Lecture Notes in Computer Science*, pages 148–163. Springer.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning, CoNLL 2004, Held in cooperation with HLT-NAACL 2004, Boston, Massachusetts, USA, May 6-7, 2004*, pages 1–8. ACL.

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, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 142–147. ACL.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Diffusionner: Boundary diffusion for named entity recognition. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 3875–3890.

Dianbo Sui, Yubo Chen, Kang Liu, Jun Zhao, Xiangrong Zeng, and Shengping Liu. 2020. Joint entity and relation extraction with set prediction networks. *CoRR*, abs/2011.01675.

Wei Tang, Benfeng Xu, Yuyue Zhao, Zhendong Mao, Yifeng Liu, Yong Liao, and Haiyong Xie. 2022. Unirel: Unified representation and interaction for joint relational triple extraction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 7087–7099.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Chenguang Wang, Xiao Liu, Zui Chen, Haoyun Hong, Jie Tang, and Dawn Song. 2022. Deepstruct: Pretraining of language models for structure prediction. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 803–823.

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 2020, Online, November 16-20, 2020*, pages 1706–1721.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. 2021. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 14138–14148. AAAI Press.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pages 419–426. The Association for Computer Linguistics.

Zexuan Zhong and Danqi Chen. 2021. A frustratingly easy approach for entity and relation extraction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 50–61.

A Appendix

Definitions. Let $\mathbf{x} \in \mathbb{R}^N$ be a vector of reals. Then the LSE operation is defined the following way:

$$\mathrm{LSE}_{n=1}^{N}(\mathbf{x}) = \log \sum_{n=1}^{N} \exp(\mathbf{x}_n) \tag{11}$$

The loss function L_{is_left} is defined as:

$$L_{\text{left}}(n) = \mathrm{LSE}_{j=1}^{2} \gamma - \mathrm{LSE}_{j=1}^{2} \Gamma \tag{12}$$

where $\mathbf{h}_n = \mathit{is\_left}(\mathbf{x})_n \in \mathbb{R}$ is the real-valued output of $\mathit{is\_left}$,

$$\gamma = \begin{bmatrix} \mathbf{h}_n \\ 0 \end{bmatrix}, \Gamma = \begin{bmatrix} \mathbf{h}_n + (1-\alpha) * (-\infty) \\ \alpha * (-\infty) \end{bmatrix}$$

and

$$\alpha = \begin{cases} 1 & \text{iff. } [\in \mathcal{A}_n \\ 0 & \text{otherwise} \end{cases}$$

equals to one if the model should perform a [action at time-step n. Accordingly, we define L_{is_left}:

$$L_{\text{lr}}(n) = \mathrm{LSE}_{j=1}^{2} \pi - \mathrm{LSE}_{j=1}^{2} \Pi \tag{13}$$

where

$$\pi = \begin{bmatrix} (\mathrm{LSE}_{i=1}^{\eta}\ \mathbf{h}_{n,m,i}) \\ 0 \end{bmatrix}$$

$$\Pi = \begin{bmatrix} (\mathrm{LSE}_{i=1}^{\eta}\ \mathbf{h}_{n,m,i} + \Delta_{n,m,i}) + (1-\beta) * (-\infty) \\ 0 + \beta * (-\infty) \end{bmatrix}$$

$\eta = |\mathbf{T}_E|$ is the number of entity types and $\mathbf{h}_{n,m} = \mathit{is\_span}(\mathbf{x}, n, m) \in \mathbb{R}^\eta$ is a vector containing one logit per such entity type.

$$\beta = \begin{cases} 1 & \text{iff. }] \in \mathcal{A}_n \\ 0 & \text{otherwise} \end{cases}$$

equals one iff. the performing] is a correct action at time-step n. We also define $m = \max\{m \mid m \leq n \wedge [\in A_m\}, m \leq n$, the largest index of the preceeding positions where [$\in \mathcal{A}_m$. Finally, we define

$$\Delta_{n,m,i} = \begin{cases} 0 & \text{iff. } (m, t_i) \in \mathcal{B}_n, t_i \in \mathbf{T}_E \\ -\infty & \text{otherwise} \end{cases}$$

Dataset	Learning Rate		-Schedule		Warmup		Weight Decay		Batch	Activation
	T5	ITER	T5	ITER	T5	ITER	T5	ITER	-size	-function
CoNLL04	1e−4	4e−4	linear *with warmup*	constant *with warmup*	0.2	0.01	0.070	0.133	8	GELU
ADE	2e−4	2.8e−4	constant *with warmup*	linear *with warmup*	0.1	0.01	0.028	0.027	32	ReLU
NYT	2.5e−4	1e−4	linear *with warmup*	linear	0.2	0	0.016	0.07	8	ReLU
GENIA	2.6e−4	8e−4	linear *with warmup*	linear *with warmup*	0.2	0.1	0.045	0.056	16	ReLU
CoNLL03	2e−4	9.7e−5	constant	constant	0	0	0.096	0.0098	8	ReLU

Table 7: Hyperparameter search results obtained using SMAC3 (Lindauer et al., 2022). For all datasets, the search was performed for 8 GPU hours using a single NVIDIA H100 GPU per dataset. The single best incumbent configuration has been selected for final training on the respective datasets.

to equal zero iff. there is a bracket pairing between the positions m and n of type $t_i \in \mathbf{T}_E$, and negative infinity otherwise. Lastly, L_{is_link} is defined as the binary cross entropy loss function:

$$L_{is\_link}(n) = \sum_{m} \sum_{i=1}^{|\mathbf{T}_R|} \begin{cases} \mu & \text{iff. }] \in \mathcal{A}_{n,m} \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

where

$$\mu = \sum \begin{bmatrix} \theta_{n,m,i} * \log\left(\mathbf{h}_{n,m,i}\right) \\ (1 - \theta_{n,m,i}) * \log\left(1 - \mathbf{h}_{n,m,i}\right) \end{bmatrix}$$

with

$$\mathbf{h}_{n,m,i} = \mathit{is\_link}(\mathbf{x}, n, m)$$

and $\theta_{n,m,i} = 1$ iff. the spans ending at positions n and m are in relationship i, $\theta_{n,m,i} = 0$ otherwise.

B Dataset Statistics

Dataset	TRAIN	DEV	TEST	Nested Entities
ADE	4,272	10%*	10%*	✓
NYT	56,196	5,000	5,000	✓
CONLL03	954	216	231	✗
CONLL04	922	231	288	✗
GENIA	16,692	†	1,854	✓

Table 8: Number of samples per dataset split. * No official dataset split exists for ADE so we employ 10-fold cross-validation with 10% of the total examples following . † GENIA comes with only two files.

C Proofs

Theorem 1. *Let* $\mathbf{x} \in \mathcal{V}^N$ *be a sequence of tokens with* $x_N = \text{EOS}$*. If* $\mathbf{y} \in \mathcal{Y}_1 \times \ldots \mathcal{Y}_M$ *is the decoded sequence of actions, then* $M \geq N$ *holds for all* $\mathbf{x} \in \mathcal{V}^{\mathbb{N}}$.

Proof. Let a_m be the action chosen at step m, $\#\texttt{copy}(m) = \sum_{i=1}^{m} \mathbb{1}_{[a_i = \texttt{copy}]}$ be the number of tokens x_n that have been copied until generation step m. Recall: generation completes at step m when $x_{\#\texttt{copy}(m)} = \text{EOS} \wedge a_m = \texttt{copy}$ (1), i.e. the EOS token has been copied into the output.

Let $\#\mathcal{A}(m) = m$ be the number of actions performed up until a certain point m in the output sequence $\mathbf{y}$ of length M. It holds that
$\#\mathcal{A}(m) = \underbrace{\sum_{i=1}^{m} \mathbb{1}_{a_i = \texttt{copy}}}_{\geq 0} + \underbrace{\sum_{i=1}^{m} \mathbb{1}_{a_i \neq \texttt{copy}}}_{\geq 0}$.
With that, it follows that $\#\texttt{copy}(m) \leq \#\mathcal{A}(m)$ (2). Using (1) we get $\#\texttt{copy}(M) = N$ and with (2) we then get $N \leq \#\mathcal{A}(M) = M \implies N \leq M \Leftrightarrow M \geq N$ □