ITER: Iterative Transformer-based Entity Recognition and Relation Extraction

Anonymous ACL submission

Abstract

When extracting structured information from text, recognizing entities and extracting relationships are essential. Recent advances in both tasks generate a structured representation of the information in an autoregressive manner, a time-consuming and computationally expensive approach. This naturally raises the question of whether autoregressive methods are necessary in order to achieve comparable results. In this work, we propose ITER, an efficient encoder-based relation extraction model, that performs the task in three parallelizable steps, greatly accelerating a recent language modeling approach: ITER achieves an inference throughput of over 600 samples per second for a large model on a single consumer-grade GPU. Furthermore, we achieve state-of-the-art results on the relation extraction datasets ADE and ACE05, and demonstrate competitive performance for both named entity recognition with GENIA and CoNLL03, and for relation extraction with SciERC and CoNLL04.

1 Introduction

In recent years, there has been a shift towards the use of autoregressive methods in many common natural language processing (NLP) tasks. In parallel, there has been 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 in the input, providing flexibility since the source and target vocabularies do not need to have any commonalities.

Flattening the output structure into a single string, preserving the structure information 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). 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. has proposed restricting the autoregressive model to *explicit* generation of the output structure.

However, since inference cannot be parallelized across the sequence dimension, language modeling approaches are prone to low throughput, especially as model size increases (Pope et al., 2022). To counteract this effect, a smaller output sequence length is of critical importance. For ASP (Liu et al., 2022), the output is always at least as long as the input, leading to poor real-world performance (see Eq. 1 in Appendix C). While scaling the model size from hundreds of millions to billions of parameters provides performance gains for Liu et al., this scaling may become unfeasible in terms of both computational requirements and throughput when using these large models in production.

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

In summary, the main contributions we have made are as follows:

1. We present ITER, a transformer-based, encoder-only relation extraction model. Instead of using a language modeling goal, our model generates the structured output in three basic steps. We show that this encoder-based approach achieves competitive performance compared to language modeling architectures, while retaining only a fraction of the number of parameters and increasing the inference

throughput by a factor of up to 23, to more than 1,000 examples per second.

2. In our experiments, we find that the trained encoder of FLAN T5 is as capable as encoders from the BERT family, which motivates further research in this direction.

3. We set a new state of the art for relation extraction on ACE05 and ADE, 71.9 (+1.4) and 85.6 (+1.8) F1 respectively, while being competitive on CoNLL04, SciERC, GENIA and CoNLL03, especially considering a significantly smaller model size and higher throughput. For named entity recognition, we set a new state of the art of 91.9 and 92.2 on ACE05 and ADE.

4. We publish our implementation and checkpoints at HTTPS://ANONYMOUS.4OPEN.SCIENCE/R/ITER-8432/README.MD.

2 Related Work

The goal of relation extraction (RE), 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*, within a given text, as well as to classify the *relationships* among these entities (Grishman and Sundheim, 1996; Zhao and Grishman, 2005).

Initial approaches to relation extraction have been to split the task into *named entity recognition* (NER) 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 (Sui et al., 2020; Zhong and Chen, 2021). 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 have been, and still are, viable approaches to modeling RE and related tasks (Gupta et al., 2016; Wang and Lu, 2020; Joshi et al., 2020; Tang et al., 2022; Zaratiana et al., 2024). Recent examples of this include DiffusionNER (Shen et al., 2023b), PL Marker (Ye et al., 2022) and UniRel (Tang et al., 2022). DiffusionNER formulates NER as a diffusion problem, allowing overlapping entities to be decoded from textual input in a fixed number of diffusion steps.

PL Marker use two types of packing strategies to identify spans from the set of all possible spans, up to a defined maximum length, and their interactions. Markers are inserted into the input sequence that cannot be attended to by classical tokens, but can attend everywhere themselves. Controlling the number of markers needed to model the interactions in an input is a key challenge faced by the authors, since increasing the input length for a transformer leads to a quadratically scaling inference time (Ye et al., 2022). UniRel combines the input text and unique tokens for each relation type to build an interaction map that models the relationships between spans. This approach can become increasingly complex when dealing with common multi-token spans, as three types of interaction maps are then required. Interaction map computation for UniRel scales quadratically with the sum of the input size and the number of relation types.

The main critic of span-based approaches is the increased design complexity, compared to language modeling approaches, due to the abstraction of most of the design complexity from the models to the target language.

2.2 Autoregressive Techniques

Modeling the task as a *seq2seq* problem has become the state of the art for RE in recent years (Cabot and Navigli, 2021; Wang et al., 2022; Paolini et al., 2021; Liu et al., 2022; Fei et al., 2022; Lu et al., 2022; Zaratiana et al., 2024). However, the primary concern in using this methodology however is the sacrifice in model throughput: the inference time of such pretrained language models (PLMs) scales quadratically with the input length. While encoding-based models often require only one pass through the encoder, PLMs require one pass through the decoder per generated token (naively), which cannot be parallelized due to the dependence on all previously generated tokens.

(m)REBEL (Cabot and Navigli, 2021; Cabot et al., 2023), TANL (Paolini et al., 2021) and ASP (Liu et al., 2022) translate the input sequence into a flattened output string, which in the case of (m)REBEL also no longer resembles natural language, but an HTML-like structure, where the input text is no longer preserved. This structure has implications for the interpretability of the model, since it is unclear which occurrence is referred to in

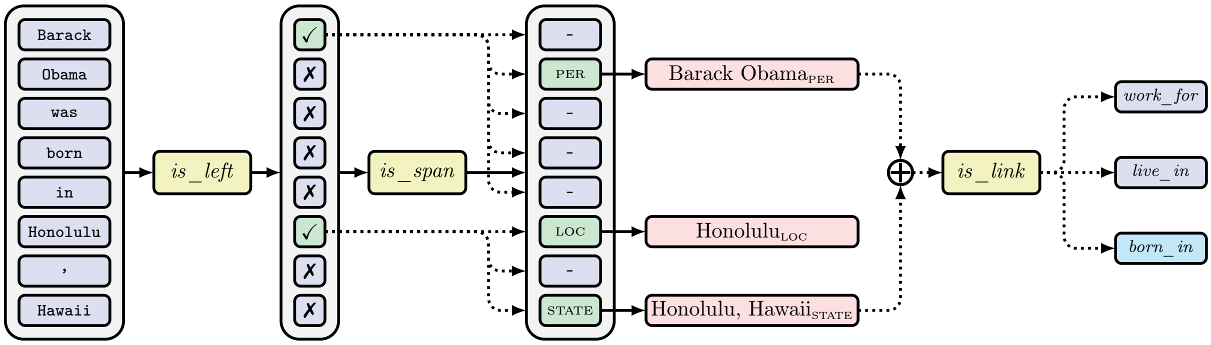

Figure 1: Visualization of ITER for nest depth $\omega = 1$. *is_left* returns two positions where spans start: 1 and 6. *is_span* then creates pairings of the types *person* between position 2 and 1, *location* between position 8 and 6, and *state* for position 8, a 1-token span. Since only the closest *left bracket action* is being considered when $\omega = 1$, *is_span* looks at the marked (✓) positions 1 and 6 for positions 1 to 5 and 6 to 8, respectively. In this example, *is_link* tests the two spans "Barack Obama" and "Honolulu, Hawaii" for relationships. As shown above, our implementation allows parallelization across the sequence dimension.

the output if the entity appears multiple times in the input. Paolini et al. extends the target output with information about entity types and relationships to other named entities. ASP generates a structured sequence of actions weaved into the original input, where three types of actions allow marking the start and end of spans and linking them together. Due to the autoregressive nature of the generative process, Liu et al. need to double the number of relation types to properly model the directions of relationships between spans.

This raises the question of whether the structured prediction method employed by ASP can be performed by an encoder-based, BERT-like model, without the autoregressive language modeling approach taken by Liu et al., while improving inference throughput and maintaining the performance of the original work.

3 Approach

We base ITER on the work of Liu et al.: To replace the autoregressive component of their approach with our inference process, several modifications to the structured prediction are necessary. We will break down the process into three basic steps:

(1) First, use *is_left* (Eq. 4) to identify all positions n in the input $\mathbf{x}$ where a span begins.

(2) Following **(1)**, identify all positions $m \geq n$ in the input that pair with any of the previously identified positions n found in **(1)** using *is_span* (Eq. 5), forming named entities. *is_span* produces a set of bracket pairs with previously found *left bracket* actions [corresponding to spans of a given named entity type $t \in \mathbf{T}_E$ from n to m.

(3) Finally, test for relationships between all pairs of named entities found in **(2)** using is_link_λ (Eq. 7). This function returns a vector of Boolean values indicating whether a relationship between two spans is present or absent.

To allow an efficient implementation of our model, each step can be individually parallelized across the sequence dimension. First, we define the set of *structure-building* actions $\mathcal{A}$:

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

Our model must be allowed to perform both [(i.e., marking the possible beginning of a span) and] (i.e., ending a span) actions at the same time, in order not to lose model expressiveness. Otherwise, it will not be able to correctly classify single-token spans[1]. Therefore, the *structure-building* actions $A_n \subseteq \mathcal{A}$ performed at position n must now be a subset of $\mathcal{A}$, to allow for this behavior. This change is reflected in the definition of the optimal structured output $\mathbf{y}^*$ that our model will learn to generate: $\mathbf{y}^* \in \times_{n=1}^{N} \mathcal{Y}_n$. The possible actions $\mathcal{Y}_n$ to be performed at step n are defined as follows:

$$\mathcal{Y}_n = \wp(\mathcal{A}) \times \wp(\mathcal{B}_n) \quad (1)$$

where $\wp$ is the powerset operation.

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

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

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

At position m, ITER is allowed to pair [actions at positions $n \leq m$ with position m, circumventing single-token named entity issues (1, Eq. 2), and each such individual pairing is allowed to have its own named entity type $t \in \mathbf{T}_E$ (2, Eq. 2).

3.1 Identifying Named Entities

Before relationships can be determined, spans must be uniquely identified by their start and end positions in combination with the type of the named entity in the input sequence. Prior to each of the following three generation steps, the input $\mathbf{x}$ is passed to the encoder of the base model, in our case T5, which produces a sequence of contextualized vector representations $\mathbf{h} = \langle h_1 \dots h_N \rangle$ for $\mathbf{x}$ with $h_n \in \mathbb{R}^\delta$, where δ is the dimension of the latent representations produced by the base model. All three stages use gated feed-forward networks of the following form:

$$FFN_\kappa^\psi(\hat{h}) = ((f(\hat{h}W_a) \otimes \hat{h}W_i)W_o \tag{3}$$

where $\hat{h} \in \mathbb{R}^{\psi\delta}$ is the concatenation of ψ δ dimensional vectors from $\langle h_1 \dots h_N \rangle$, $W_a, W_i \in \mathbb{R}^{\psi\delta \times \eta}, W_o \in \mathbb{R}^{\eta \times \kappa}$ are weight matrices learned during training, f is a nonlinear function, and κ is the output dimension. $\psi = 2$ if two vectors are input (also $\psi = 4$ for four vectors), otherwise $\psi = 1$.

3.1.1 Determine Where Named Entities Start

To identify these spans, the model learns to predict the positions where the spans of named entities in the input $\mathbf{x}$ begin. This task is modeled by the function *is_left* (Eq. 4), which takes a latent representation h_n as input and outputs a Boolean value $b_n \in \mathbb{B}$:

$$\mathit{is\_left}(h_n) = FFN_{\kappa=1}^{\psi=1}(h_n) > 0 \tag{4}$$

At all positions where $\mathit{is\_left}(h_n)$ is true, the *left bracket* action [is included in the set of actions A_n that are performed at position n.

3.1.2 Pair Left and Right Brackets

After determining where spans of named entities start in the input $\mathbf{x}$, the next step is to determine which positions $\mathbf{x}_m$ $(m \geq n)$ following $\mathbf{x}_n$ in the input form a span of named entity type $t \in \mathbf{T}_E$. Our model learns a projection *is_span* that maps the input position h_m to a set of tuples of indices and entity types (n, t), where each entry corresponds to a pair of spans from n to m of type $t \in \mathbf{T}_E$:

$$\begin{aligned} &\mathit{is\_span} : \mathbb{R}^\delta \to \wp(\mathcal{B}_n) \\ &\mathit{is\_span}(h_m) = \{(n, t) \mid s_{n,m}^t\} \end{aligned} \tag{5}$$

where

$$\begin{aligned} s_{n,m}^t = \; &FFN_{\kappa=\#\mathbf{T}_E}^{\psi=2}(h_m, h_n)_t > 0 \\ &\wedge \mathit{is\_left}(h_n) \\ &\wedge n \leq m \end{aligned}$$

That is, a pair of positions $n \leq m$ was identified as a span pair of type t, where previously h_n was marked as the beginning of a span.

For each position m where the output of $B_m = \mathit{is\_span}(h_m)$ is not empty, ITER performs a *right bracket* action] at position m. Each element $(n, t) \in B_m$ determines a pair of a left bracket at position n with a right bracket at position m of type t, forming a named entity. If a left bracket from step one is left unbound, no named entity is identified. This prevents invalid output artifacts. A visualization of our model is available in Figure 1.

3.2 Identify Relations among Named Entities

The third step now tests pairs of identified named entities for their relationship to each other. For the non-nested case, *is_link* projects two hidden states h_i and h_j onto a vector of non-normalized logits, similar to probabilities after applying the sigmoid function (Eq. 6).

$$\begin{aligned} &\mathit{is\_link}_\lambda : \mathbb{R}^\delta \times \mathbb{R}^\delta \to \mathbb{B}^\kappa \\ &\mathit{is\_link}(h_i, h_j)_\lambda = \sigma(FFN_\kappa^\psi(h_i, h_j)) > \lambda \end{aligned} \tag{6}$$

where $\lambda \in [0, 1], \kappa = |\mathbf{T}_R|$, $\psi = 2$. The comparison with λ, a decision boundary parameter is done element-wise. Thus, our model can predict multiple relationships between any pair of entities (spans). λ allows you to trade precision for recall. By default, we set $\lambda = 0.5$ as shown in Figure 2

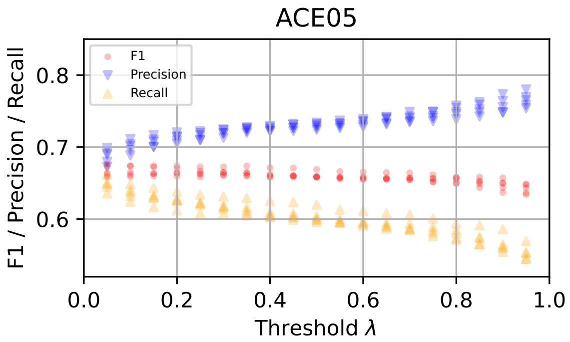

Figure 2: Visualization of the trade-off between RE+ precision and recall for different values of λ on ACE05. Five different seeds evaluated at the last checkpoint.

At both positions i, j, our model previously performed a $\boxed{]}$ action. It also paired those two actions with *left bracket* actions at positions k, l with span types $t_i, t_j \in \mathbf{T}_E$: $\boxed{]} \in A_i, A_j \wedge (k, t_i) \in \mathcal{B}_i \wedge (l, t_j) \in \mathcal{B}_j$. *is_link* returns vectors containing Boolean values for relations between two entities identified in (1) and (2). During inference, all combinations of entities found are tested for relationships. The order of the head and tail entities is important, so $\mathit{is\_link}(h_i, h_j) \neq \mathit{is\_link}(h_j, h_i)$ unless a relationship is symmetric.

The abstraction of using only the latent representation of the last position of the span can no longer be applied when dealing with nested entities, since spans are no longer uniquely identified by their last position. To counteract this, the representation of the first position of a span is also included:

$$\begin{aligned} &\mathit{is\_link} : \mathbb{R}^{4\times\delta} \to \mathbb{B}^{\kappa} \\ &\mathit{is\_link}(\mathbf{h}) = \sigma(FFN^{\psi=4}_{\kappa=\#\mathbf{T}_R}(\mathbf{h}_1 \ldots \mathbf{h}_4)) \end{aligned} \tag{7}$$

where now $\psi = 4, \mathbf{h} = \langle h_i, h_o, h_j, h_p \rangle$.

3.3 Training

A key issue in training our proposed model was the choice of transformer encoder used in our experiments. Since we base our work on the ASP (Liu et al., 2022) architecture, we ultimately decided to use the T5 (Raffel et al., 2020) autoregressive model as the encoder by simply relying on the PLM encoder stack. In addition, we perform experiments on several BERT-like encoder models: ALBERT and DeBERTa.

To avoid error propagation between the three stages of ITER, the training will include all three functions simultaneously: *is_left*, *is_span*, and *is_link*. ITER takes as input a sequence of latent representations $\mathbf{h} = \langle h_1, h_2, \ldots, h_N \rangle$. The sequence of representations is shared across all three tasks. The loss function used during training can be found in Appendix B, in Equations 10, 12, and 13. To minimize training loss, the model is encouraged to assign weights greater than zero to the correct decisions in all three cases, which affects the decisions made by *is_left*, *is_span*, and *is_link*.

3.4 ITER versus other Encoders

Since ITER is no longer an autoregressive model, this motivates the discussion of other, encoding-based approaches in terms of their differences and similarities to our model.

Table-filling Approaches: Unlike most previous approaches, ITER does not recognize entities with a classical table-filling pipeline, where each combination of tokens in the input $\mathbf{x}$ is tested to be a named entity (Gupta et al., 2016; Wang and Lu, 2020; Ma et al., 2020; Tang et al., 2022).

Span-Based Approaches: The best known and most powerful span-based approaches include DyGIE++, PURE, and PL Marker (Wadden et al., 2019; Zhong and Chen, 2021; Ye et al., 2022), which mostly seem to follow the same basic idea of creating all possible spans (with up to length L) and consequently predicting the correct types (including *none*). In addition, they use markers to better teach the model start and end indices. For instance, in PL Marker a group of *levitated markers* is built for each token in the input, and appended to the input sequence. Each such pair of markers is able to accompany a subsequence of the whole input, and there is one pair per possible span with a maximum span length L depending on the data (Ye et al., 2022). As a consequence, the input length increases drastically by about $2NL$ depending on L. This increase is also reflected in the throughput of PL Marker (211.7 samples/s) compared to our method (392.9 samples/s), as shown in Table 3.

In contrast, ITER identifies named entities in two, linear time steps, as discussed in the next section. To the best of our knowledge, and supported by our experiments, ITER is the most efficient transformer-based end-to-end relation extraction model.

3.5 Complexity

Here, the theoretical time complexity of our approach is briefly discussed. As follows from their definitions, both steps (1) and (2) can be parallelized over the sequence dimension. Since *is_left* uses only linear projections and activation functions, its runtime is bounded by the length of the input sequence $\mathbf{h}$, yielding a linear time complexity $O(N)$. is_span is optimized to consider only the ω closest *left bracket* actions, and in the trivial case we set $\omega = 1$. For nested named entity records, ω can be calculated as follows:

$$\omega = \max\left\{ \sum_{k=n}^{m} \left[\!\left[\boxed{[} \in A_k \right]\!\right] \middle| (n,t) \in B_m, 1 \le m \le N \right\}$$

for an element of a given record. For $\omega = 1$, is_span performs one pass through the FFN per element of the sequence $\mathbf{h}$, thereby yielding a time complexity $O(N)$ linear in the input length N. Correspondingly, ω can be derived for the whole dataset by taking the maximum value over all samples. Our choices for ω are also shown in Table 1. For $\omega > 1$, we have $O(N * \omega)$, but $\omega \ll N$. Since steps (1) and (2) are performed sequentially, their combination remains bounded by the sequence length N. Testing for relationships in step (3) requires testing all combinations of entities found and thus gives a quadratic runtime, but not in the sequence length, but in the number of entities E with $E \ll N$. Using ITER thus gives a complexity of $O(N + E^2)$), where E is the number of entities.

4 Experimental Results

In this section, we give an overview of the datasets used (Section 4.1) followed by a discussion of the results from our experiments (Section 4.2). Details of the hyperparameter search we performed can be found in Appendix D.

4.1 Data

To evaluate our proposed model, we measured its performance and throughput on a diverse portfolio of six datasets, varying in domain and task: CoNLL03 (Sang and Meulder, 2003) and GENIA (Kim et al., 2003) were selected for NER, followed by CoNLL04 (Roth and Yih, 2004), ACE05 (Walker et al., 2006), ADE (Gurulingappa et al., 2012) and SciERC (Luan et al., 2018) for RE. CoNLL03, CoNLL04 and ACE05 contain examples taken from news articles, GENIA and ADE have a biomedical domain and contain examples of Medline abstracts and drug-drug interactions, respectively. SciERC consists of 500 scientific abstracts that have been annotated for scientific entities, their relationships and co-references (Luan et al., 2018). An overview of the selection of our datasets can be found in Table 1. Following the literature, we primarily evaluate our model in a *strict* setting for RE: A predicted relationship between two entities is only considered correct if both the span and the type of the entity match the gold standard (RE+). We report *micro* F1 values unless otherwise noted.

Dataset	TRAIN	DEV	TEST	ω	OVERLAPPING ENTITIES
CONLL03	954	216	231	1	0
CONLL04	922	231	288	1	0
SciERC	1861	275	551	3	2.7%
ADE	4,272	10%*	10%*	2	1.6 - 4.4%
ACE05	5217	1277	1130	1	0
GENIA	16,692	†	1,854	4	21.6%
NYT	56,196	5,000	5,000	2	1.1%

Table 1: We report the number of samples per dataset split, the choice of ω per dataset and the number of samples where overlaps do occur. We count overlaps as entities that begin inside another. (*): There is no official dataset split for ADE, so we use 10-fold cross-validation with 10% of the total examples. † In the dataset split provided by Shen et al., the training (train) and development (dev) sets have been merged.

Models. We use an ensemble of different models during training, in particular the FLAN T5 (Shen et al., 2023a) encoders, referred to as FT5, T5 (Raffel et al., 2020) and BART (Lewis et al., 2020). We also train our model with DeBERTaV3 (He et al., 2023, 2021), BERT (Devlin et al., 2019), and ALBERT (Lan et al., 2020).

4.2 Results

Overall, ITER outperforms or is competitive with the state-of-the-art on all datasets, as shown in Table 2 for NER and RE+. In addition, our model excels in terms of throughput: Our large model variant ITER + FLAN T5 *(large)* outperforms the autoregressive ASP by up to 22× and the encoder-based PL Marker by up to 12.5×. See Table 3 for the results of our throughput measurements.

For ACE05, we set a new state of the art of 71.9 F1 for RE+, but with a very large PLM as an encoder: FLAN T5 *(xl)* with 1.3 billion parameters. DeBERTaV3 *(large)* gave very competitive results despite it being a 2.7 times smaller model: 70.8 F1 for RE+ and a state-of-the-art of 91.9 for NER. We also set a new state-of-the-art for the biomedical dataset ADE, for both NER and RE+ with 92.2 and 85.6 F1, respectively.

On CoNLL04, our model is only outperformed by ASP + T0 *(3b)*, DeepStruct and ATG with respect to the strict RE F1 metric. The first two are ×7.12

Model	Architecture: Parameters total	vs. ours	ACE05 NER	ACE05 RE+	ACE05 RE+ (∗)	CoNLL04 NER	CoNLL04 RE+	ADE NER	ADE RE+	SciERC NER	SciERC RE+	SciERC RE+ (∗)	CoNLL03 NER	GENIA NER
ITER *(ours)*														
+ FT5 *(large)*	393 M	1.0	91.5	70.5	71.1	89.8	75.2	91.6	84.3	63.8	31.5	32.1	91.6	80.4
+ FT5 *(xl)*	1.3 B	×3.33	91.7	**71.9**	**72.7**	90.1	75.6	**92.2**	**85.6**	69.1	38.9	40.2	91.8	81.2
+ DeBERTaV3 *(large)*	476 M	×1.21	**91.9**	70.8	71.4	91.2	75.9	91.6	84.7	68.1	36.9	38.6	91.8	80.6
+ BERT *(large)*	340 M	×0.86	88.3	64.5	65.4	87.4	68.0	90.9	82.8	53.2	28.1 ◆	28.6		77.6
+ SciDeBERTa *(full)*	436 M	×1.11								68.1	38.4	40.1		
ASP														
+ T5 *(base)*	229 M	×0.58	90.7	68.6	-	89.4	73.8						91.8	
+ T5 *(large)*	745 M	×1.89	91.3	69.4	-	89.4	73.8						92.8	
+ T5 / T0 *(3b)*	2.8 B	×7.12	91.3	70.5	-	90.3	76.3							
PL Marker *(albert-xxlarge-v1)*	223 M	×0.56	91.1	-	71.1									
PL Marker *(scibert-uncased)*	110 M	×0.27								69.9	-	**41.6**		
DeepStruct *(finetuned)*	10 B	×25.1	86.9	66.8	-	90.7	78.3	91.3	83.2				93.1	80.8
ATG *(deberta-v3-large)*	479 M	×1.21	90.1	66.2	-	90.5	**78.5**							
ATG *(scibert-cased)*	151 M	×0.38								69.7	38.6			
TANL	222 M	×0.57	88.9	63.7	-	89.8	72.6	91.2	83.8				92.0	76.4
REBEL	406 M	×1.03				-	75.4	-	82.2					
DiffusionNER	381 M	×0.85	86.9	-	-	92.8	-						92.78	**81.53**
PFN	223 M	×0.56	89.0	66.8	-			91.3	83.2	66.8	38.4	-		
UIE	737 M	×1.87	85.8	66.1	-	-	75.0			-	36.53		92.99	
LasUIE	737 M	×1.87	86.0	66.4	-	-	75.3						**93.2**	
Wang and Lu	223 M	×0.56	89.5	64.3	-	90.1	73.6	89.7	80.1					
UNIRE (Wang et al.)	110 M	×0.27								68.4	-	36.9		
TF-MTRNN	*unknown*					**93.6**	72.1						86.80	
Spert.PL	110 M	×0.27								70.53	**39.41**			

Table 2: Final results for CoNLL04, ACE05, ADE, SciERC, CoNLL03, and GENIA. (∗): PL Marker and Wang et al. weight correct symmetry relations twice (Ye et al., 2022), further explained in Appendix F. To allow a fair comparison, we also report results using their scoring method. ◆ One of the 5 runs with *bert-large-cased* diverged.

and ×25.1 larger. For CoNLL03, our model performs in line with the results from ASP, but does not perform close to the state of the art on this dataset. In an ablation study, we find that using the best final checkpoint from CoNLL03 as the initialization for CoNLL04 increases the final model performance by an average of 0.6 F1 points. More details can be found in Appendix E. On both GENIA and SciERC, our model achieves results that are competitive with the current state of the art.

We conducted experiments with the BART, DeBERTa, and ALBERT encoders on ACE05. Our goal was to identify the best fitting pretrained model for our proposed method. The results of this investigation can be seen in Table 4 in Appendix A. Comparing the performance of ITER with these PLMs, we find that models with relative position embeddings (T5 family, DeBERTa) to generally perform better in our setting when compared to models with absolute position embeddings (BART, ALBERT).

5 Conclusion and Future Work

In this work, we propose ITER, an efficient, well-performing relation extraction model. We translate the autoregressive process of Liu et al. into a constant, easily parallelizable three-step process, while maintaining the same level of performance and increasing model throughput, especially for longer sequence lengths. Our model allows us to perform any kind of structured prediction task with maximum throughput (Figure 3) and state-of-the-art performance. On ACE05, we set a new state-of-the-art of 71.9 F1 and 91.9 for RE and NER. For the biomedical dataset ADE, we set a new state-of-the-art for NER (92.2) and RE+ (85.6). In our experiments, we find that using the encoders of generative T5 models can yield model performance advantages over using discriminative models such as BERT and ALBERT, while also outperforming these models in terms of throughput. The only exception to this finding is DeBERTa, where F1 performance is competitive (RE+) or even better (NER). However, in terms of model throughput, the T5 family models still outperform DeBERTa. We also highlight the advantages of encoder-based models over generative approaches in terms of model throughput, with ITER being up to 42 times faster than the autoregressive state-of-the-art model ASP. To the best of our knowledge ITER is the first model to successfully use only the encoder of an autoregressively trained PLM in this domain, inspiring further research in this direction.

One area of future work may be to develop a

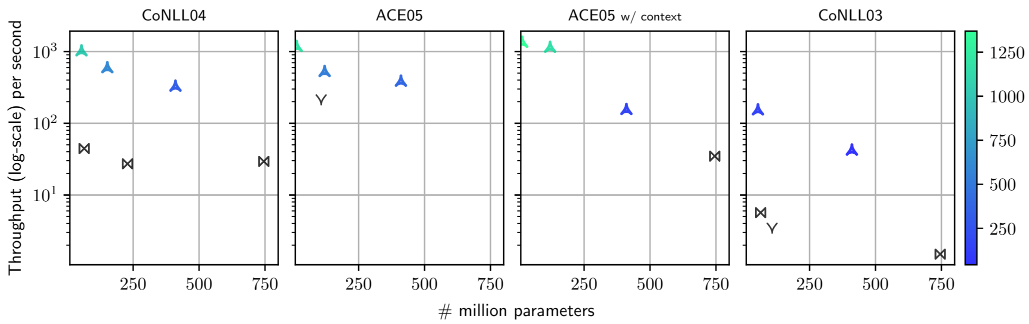

Figure 3: Visualization of the throughput of ITER vs. ASP (autoregressive) and PL Marker (encoder-based). ⋏ marks ITER, ⋎ marks PL Marker (Ye et al., 2022) and ⋈ marks ASP (Liu et al., 2022). For the visualization, we measured on ITER models of different sizes from 15 M to 410 M parameters.

Dataset NAME # TOKENS	**Architecture** MODEL	 PARAMS	**Throughput** SAMPLES/S
ACE05 ⋈ 29 / 360 / 135.35 MIN / MAX / AVG	WITHOUT CONTEXT:		
	ITER + T5 *(large)*	410 M	**392.908**
	PL Marker *(base)*	108 M	211.7
	WITH CONTEXT:		
	ITER + T5 *(large)*	410 M	**158.476**
	ASP + T5 *(large)*	745 M	34.826
CoNLL03 DOCUMENT LEVEL 46 / 1015 / 308.18 MIN / MAX / AVG	SMALL MODELS:		
	ITER + T5 *(small)*	46 M	**156.44**
	ASP + T5 *(small)*	61 M	5.6685
	LARGE MODELS:		
	ITER + FT5 *(large)*	370 M	**43.05**
	ASP + T5 *(large)*	738 M	1.4981
CoNLL03 SENTENCE LEVEL 2 / 213 / 24.55 MIN / MAX / AVG	ITER + T5 *(small)*	46 M	**3276.12**
	ITER + FT5 *(large)*	370 M	512.49
	PL Marker *(base)*	108 M	54.8
CoNLL04 4 / 173 / 43.84 MIN / MAX / AVG	SMALL MODELS:		
	ITER + T5 *(small)*	54 M	**1040.550**
	ASP + T5 *(small)*	64 M	44.459
	LARGE MODELS:		
	ITER + T5 *(large)*	410 M	**605.398**
	ASP + T5 *(large)*	745 M	27.177
	ASP + T5 *(3b)*	2.9 B	29.427

Table 3: Comparing the inference throughput of various RE architectures: ITER, ASP and PL Marker. Experiments for ITER and ASP were performed on a single RTX 4090 GPU using a batch size of 64. For document level CoNLL03, a single H100 GPU was used with a batch size of 8. ITER is significantly faster in inference than ASP and PL Marker, especially when dealing with longer input lengths, such as in CoNLL03. ⋈: Statistics for ACE05 with additional context; without, ACE05 has 25.21 tokens on avg.

large (synthetic) dataset that allows evaluation of the expressiveness of RE models with respect to nested entities, since nested entities are a real-world problem, but existing datasets contain only a small fraction of such examples (see Table 1). Enabling zero- and few-shot task transfer for our pretrained models without further training may be another area of future work. As discussed earlier, using the T5 encoder instead of BERT-style models gave us equivalent training results for us and motivates a more comprehensive study of the performance of autoregressive, encoder-decoder models in settings where typically encoder-based models are employed.

6 Limitations

One limitation of our model is the output of named entities that are not directly contained in the input text. However, out of the six datasets used in this work, none contain an example where this problem occurs.

While the three functions *is_left*, *is_span* and *is_link* seem very task-specific at a first glance, they allow very flexible modeling of first-order relationships between any kind of spans in all kinds of span modeling tasks. This includes tasks like coreference resolution and entity linking. Our reference implementation on GitHub[2] includes easy-to-use scripts to apply ITER to any kind of structured prediction problem.

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.

[2] https://anonymous.4open.science/r/ITER-8432/README.md

Red$^{\text{fm}}$: 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.

Hao Fei, Shengqiong Wu, Jingye Li, Bobo Li, Fei Li, Libo Qin, Meishan Zhang, Min Zhang, and Tat-Seng Chua. 2022. Lasuie: Unifying information extraction with latent adaptive structure-aware generative language model. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

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.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: decoding-enhanced bert with disentangled attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

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.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

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.

Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022. Unified structure generation for universal information extraction. *CoRR*, abs/2203.12277.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3219–3232. Association for Computational Linguistics.

Youmi Ma, Tatsuya Hiraoka, and Naoaki Okazaki. 2020. Named entity recognition and relation extraction using enhanced table filling by contextualized representations. *CoRR*, abs/2010.07522.

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.

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.

Sheng Shen, Le Hou, Yanqi Zhou, Nan Du, Shayne Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen, Tu Vu, Yuexin Wu, Wuyang Chen, Albert Webson, Yunxuan Li, Vincent Y. Zhao, Hongkun Yu, Kurt Keutzer, Trevor Darrell, and Denny Zhou. 2023a. Flan-moe: Scaling instruction-finetuned language models with sparse mixture of experts. *CoRR*, abs/2305.14705.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023b. 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.

Yongliang Shen, Xiaobin Wang, Zeqi Tan, Guangwei Xu, Pengjun Xie, Fei Huang, Weiming Lu, and Yueting Zhuang. 2022. Parallel instance query network for named entity recognition. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 947–961. Association for Computational Linguistics.

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.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 5783–5788.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. Philadelphia: Linguistic Data Consortium. `https://catalog.ldc.upenn.edu/LDC2006T06`.

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.

Yijun Wang, Changzhi Sun, Yuanbin Wu, Hao Zhou, Lei Li, and Junchi Yan. 2021. Unire: A unified label space for entity relation extraction. *CoRR*, abs/2107.04292.

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), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 4904–4917.

Urchade Zaratiana, Nadi Tomeh, Pierre Holat, and Thierry Charnois. 2024. An autoregressive text-to-graph framework for joint entity and relation extraction. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 19477–19487.

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 ITER with different PLMs

We investigate the performance of ITER with a variety of different PLMs: T5, FLAN T5 (FT5), BART, DeBERTaV3 and ALBERT. The results can be seen in Table 4.

	Transformer MODEL	NER F1	RE+ F1	Throughput SAMPLES/S
	T5 *(large)*	91.0	69.4	175.510
	FT5 *(large)*	91.5	70.5	176.056
ITER +	BART *(large)*	91.4	68.5	172.871
	DeBERTaV3 *(large)*	**91.9**	**70.8**	138.589
	ALBERT *(xxlarge-v2)*	91.4	69.8	36.029

Table 4: Comparing the performance of ITER on ACE05 using different base models: using encoders from the autoregressive models (FLAN-)T5, BART, and the encoder-only models DeBERTa and ALBERT. Throughput was measured with $batch\_size = 8$ for all models on a single H100 GPU.

B Training

The log-sum-exp operation $LSE : \mathbb{R}^N \to \mathbb{R}$ used in the following equations is defined as:

$$LSE_{n=1}^{N}(\mathbf{x}) = \log \sum_{n=1}^{N} \exp(\mathbf{x}_n) \tag{8}$$

where $\mathbf{x} \in \mathbb{R}^N$ is a vector of real numbers. We define $\eta = \#\mathbf{T}_E$ and $\varphi = \#\mathbf{T}_R$ to hold the number of entity and relation types. $A_n \in \mathcal{A}_n$ holds the *set of correct actions* at position n. Likewise, $B_n \in \mathcal{B}_n$ holds the *set of correct bracket pairings* at position n.

During training, the model will learn to minimize the following loss function:

$$\mathcal{L}_{\text{ITER}} = \sum_{n=1}^{N} \sum \begin{bmatrix} \mathcal{L}_{is\_left}(n) \\ \mathcal{L}_{is\_span}(n) \\ \mathcal{L}_{is\_link}(n) \end{bmatrix} \tag{9}$$

is a sum of three loss values for each position in the input sequence $\mathbf{x}$.

The loss for placing *left-bracket* actions $\mathcal{L}_{is\_left}$ is defined as follows:

$$\mathcal{L}_{is\_left}(n) = LSE \begin{bmatrix} \gamma_n \\ 0 \end{bmatrix} - LSE \begin{bmatrix} \Gamma_n \\ \alpha * -M \end{bmatrix} \tag{10}$$

using the feed-forward network from Eq. 4, we define

$$\gamma_n = FFN_{is\_left}(h_n)$$
$$\Gamma_n = \gamma_n + (1 - \alpha) * -M$$

where $h_n \in \mathbb{R}$ is the real-valued output of the encoder model for input position n, $M \to \infty$.

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

is equal to one if the model should perform a [action at time step n, effectively canceling out one of the terms in the above equation. Accordingly, we define $\mathcal{L}'_{is\_span}$ for a pairing between positions n and m:

$$\begin{aligned} \mathcal{L}'_{is\_span}(n, m) = LSE & \begin{bmatrix} \pi_{n,m} \\ 0 \end{bmatrix} \\ -LSE & \begin{bmatrix} \Pi_{n,m} + \beta_{n,m} * -M \\ (1 - \beta_{n,m}) * -M \end{bmatrix} \end{aligned} \tag{11}$$

where $M \to \infty$,

$$\pi_{n,m} = LSE_{t=1}^{\eta}\left(\hat{h}_{n,m,t}\right)$$
$$\Pi_{n,m} = LSE_{t=1}^{\eta}\left(\hat{h}_{n,m,t} + \Delta_{n,m,t}\right)$$

using the feed-forward network from Eq. 5

$$\hat{h}_{n,m} = FFN_{is\_span}(h_n, h_m) \in \mathbb{R}^{\eta}$$

$\hat{h}_{n,m}$ is a vector containing one logit per entity type corresponding to whether positions n to m form a span of a particular type.

$$\beta_{n,m} = \begin{cases} 0 & \text{iff. } (n, t) \in B_m \\ 1 & \text{otherwise} \end{cases}$$

equals one iff. pairing n with m with any type is *not* a correct action at time-step m. Using Eq. 11, we define $\mathcal{L}_{is\_span}$ for $\omega = 1$: $\mathcal{L}_{is\_span}(n) = \mathcal{L}'_{is\_span}(n, m)$ where

$$n = \max\{n \mid n \leq m \wedge [\in A_n\}$$

is the closest of the preceding positions $n \leq m$ where [$\in A_n$. For $\omega > 1$, we define $\mathcal{L}_{is\_span}$ the following:

$$\mathcal{L}_{is\_span}(m) = \sum_{n \in \mathcal{N}}^{n \leq m} \mathcal{L}'_{is\_span}(n, m) \tag{12}$$

Dataset	Learning Rate (LR)		Learning Rate Schedule		Warmup		Weight Decay		Batch Size	Activation Function
	T5	ITER	T5	ITER	T5	ITER	T5	ITER		
CoNLL04	5e−5	3e−4	linear *with warmup*	linear *with warmup*	5%	1%	0.100	0.020	8	ReLU
ADE	2e−4	2.9e−4	constant *with warmup*	constant *with warmup*	10%	1%	0.028	0.026	32	ReLU
ACE05	4.9e−5	5.9e−4	linear *with warmup*	linear *with warmup*	1%	5%	0.082	0.109	8	ReLU
SciERC	3e−5	1e−4	linear *with warmup*	linear *with warmup*	5%	0%	0.1	0.1	8	ReLU
GENIA	2.6e−4	8e−4	linear *with warmup*	linear *with warmup*	20%	10%	0.045	0.056	16	ReLU
CoNLL03	2e−5	3e−4	linear *with warmup*	linear *with warmup*	0%	0%	0.096	0.0098	8	ReLU

Table 5: Hyperparameter search results obtained with SMAC3 (Lindauer et al., 2022). The single best incumbent configuration was selected for final training on each dataset. For SciERC, we used the provided values since the found hyperparameters did yield subpar training results.

where $\mathcal{N} = \langle n_1 \dots n_\omega \rangle$ are the ω closest preceding *left bracket actions*:

$$n_i = \max \left\{ j \mid j \leq m \land \texttt{[} \in A_j \land j \notin \mathcal{N}_{<i} \right\}.$$

We define

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

($M \to \infty$) to equal zero iff. There is a bracket pairing between the positions $n \leq m$ of type $t \in \mathbf{T}_E$, and a large negative value otherwise, effectively canceling out any interaction between n and m of type t. To minimize the loss function, the model must assign negative values to non-existent interactions between two positions n and n of a particular type t_i.

Finally, $\mathcal{L}_{is\_link}$ is defined as the binary cross entropy loss function:

$$\mathcal{L}_{is\_link}(m) = \sum_{n=1}^{N} \sum_{i=1}^{\varphi} \begin{cases} \mu_{n,m,i} & \text{iff. } \texttt{]} \in A_n \\ & \land\ \texttt{]} \in A_m \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

where

$$\mu_{n,m,i} = \theta_{n,m,i} * \log\left(\hat{h}_{n,m,i}\right) + \left(1 - \theta_{n,m,i}\right) * \log\left(1 - \hat{h}_{n,m,i}\right)$$

with the feed-forward network from Eq. 7:

$$\hat{h} = FFN_{is\_link}(h_n, h_m).$$

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

C Proofs

Theorem 1. *Let* $\mathbf{x} \in \mathcal{V}^N$ *be a sequence of tokens with* $x_N = \text{EOS}$. *If* $\mathbf{y} \in \mathcal{Y}_1 \times \dots \mathcal{Y}_M$ *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, $\#\texttt{copy}(m) = \sum_{i=1}^{m} \mathbb{1}_{\left[a_i = \texttt{copy}\right]}$ be the number of tokens x_n that have been copied until generation step m. Recall: generation completes at step m when $x_{\#\texttt{copy}(m)} = \text{EOS} \land a_m = \texttt{copy}$ (1), i.e. the EOS token has been copied into the output.

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

D Hyperparameter Search

Before training ITER with FLAN T5 (large) on ACE05, CoNLL04, ADE, SciERC, CoNLL03 and GENIA, we perform a hyperparameter search on all datasets using SMAC3 (Lindauer et al., 2022). For all datasets, we optimize for high RE+ or NER F1, depending on the task. The search space consists of learning rates $lr \in [1\text{e}{-3}, 2\text{e}{-5}]$, 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 above that are responsible for modeling the functions *is_left*, *is_span* and *is_link*, combined with the batch size $bs \in \{8, 16, 32, 64\}$ and the choice of activation function $act \in \{\mathrm{GELU}, \mathrm{ReLU}, \tanh\}$. The results of this hyperparameter search can be found in Table 5.

E Pretraining ITER

In this ablation study, we experiment with using the best performing CoNLL03 checkpoint for transfer learning by using it as a starting point used to train ACE05 and CoNLL04. On CoNLL04, we are able to increase the average model performance by 0.6 F1 points to 76.3 F1. Using the same strategy for ACE05 does not yield any improvement at all, performance drops by an average of 1.8 F1 points.

	Dataset and Architecture		NER	RE+
	MODEL	PARAMS	F1	F1
CoNLL04	ITER *(ours)*			
	+ FT5 *(large)*	393 M	89.7 ± 0.51	75.1 ± 0.39
	+ $\text{FT5}_{\text{+CoNLL03}}$ *(large)*	393 M	**91.1** ± 0.53	**76.3** ± 0.80
	+ DeBERTaV3 *(large)*	476 M	91.2 ± 0.20	75.9 ± 1.41
ACE05	ITER *(ours)*			
	+ FT5 *(large)*	393 M	91.5 ± 0.16	70.5 ± 0.51
	+ $\text{FT5}_{\text{+CoNLL03}}$ *(large)*	393 M	91.3 ± 0.15	68.3 ± 0.36
	+ DeBERTaV3 *(large)*	476 M	**91.9** ± 0.38	**70.8** ± 0.47

Table 7: Comparing CoNLL03-pretrained ITER + $\text{FT5}_{\text{+CoNLL03}}$ *(large)* with normal ITER versions on 3 seeds. FT5 refers to FLAN T5.

F On the Evaluation of PL Marker

PL Marker follow Wang et al. and implement the following evaluation for symmetric relations in ACE05 and SciERC: Their model predicts symmetric relations twice, outputting $(head, symmetric\_type, tail)$ and $(tail, symmetric\_type, head)$ for a symmetric relation between $head$ and $tail$. In their evaluation, however, this counts as two different outputs, and thus they will be weighted twice in case of either being a correct model output. Apart from our evaluation, where symmetric relationships are output as a 3-element set $\{head, tail, symmetric\_type,\}$ and thus a correct output is not weighted twice, we also implement the evaluation according to PL Marker and Wang et al. and indicate results stemming from this evaluation with an asterisk $(*)$ in our result tables to allow a fair comparison between the two methods. This subtle change has an effect on the final performance for RE+, as can be seen in Tables 2 and 6.

Dataset	**Architecture**	**#**	**NER**			**RE**			**RE+** (strict)		
	W/ MODEL	SEEDS	PRECISION	RECALL	F1	PRECISION	RECALL	F1	PRECISION	RECALL	F1
CoNLL04	ITER *(ours)*										
	+ FLAN T5 *(large)*	5	89.5 ± 0.43	90.1 ± 0.80	89.8 ± 0.51	78.3 ± 2.04	73.0 ± 1.43	75.5 ± 0.29	77.9 ± 2.10	72.7 ± 1.38	75.2 ± 0.39
	+ FLAN T5 *(xl)*	5	89.9 ± 0.56	90.3 ± 0.52	90.1 ± 0.44	77.8 ± 2.16	73.7 ± 1.25	75.7 ± 1.52	77.7 ± 2.13	73.6 ± 1.22	75.6 ± 1.49
	+ DeBERTaV3 *(large)*	5	90.8 ± 0.28	91.7 ± 0.43	91.2 ± 0.20	79.2 ± 2.49	73.1 ± 1.24	76.0 ± 1.43	79.0 ± 2.51	73.0 ± 1.15	75.9 ± 1.41
	+ BERT *(large-cased)*	3	87.3 ± 0.29	87.6 ± 0.67	87.4 ± 0.23	69.1 ± 2.13	67.2 ± 2.28	68.1 ± 1.25	69.0 ± 2.03	67.2 ± 2.03	68.0 ± 1.11
ACE05	ITER *(ours)*										
	+ FLAN T5 *(large)*	5	90.6 ± 0.38	92.4 ± 0.20	91.5 ± 0.24	75.3 ± 0.62	72.0 ± 0.48	73.6 ± 0.36	72.1 ± 0.76	69.0 ± 0.53	70.5 ± 0.51
	+ FLAN T5 *(xl)*	5	90.6 ± 0.26	92.6 ± 0.26	91.6 ± 0.12	76.7 ± 0.72	73.5 ± 0.92	75.1 ± 0.49	73.5 ± 0.65	70.4 ± 1.00	71.9 ± 0.56
	+ DeBERTaV3 *(large)*	5	91.1 ± 0.49	92.7 ± 0.30	91.9 ± 0.38	76.2 ± 0.90	71.6 ± 0.60	73.8 ± 0.54	73.1 ± 0.91	68.6 ± 0.40	70.8 ± 0.47
	+ T5 *(large)*	3	90.0 ± 0.55	91.3 ± 0.02	90.7 ± 0.27	76.0 ± 1.12	63.1 ± 0.53	69.0 ± 0.58	73.1 ± 1.01	60.7 ± 0.65	66.3 ± 0.62
	+ BART *(large)*	3	90.7 ± 0.27	92.0 ± 0.08	91.4 ± 0.15	74.6 ± 0.77	68.3 ± 0.82	71.3 ± 0.50	71.7 ± 0.35	65.6 ± 0.92	68.5 ± 0.42
	+ ALBERT *(xxlarge-v2)*	3	90.8 ± 0.32	92.1 ± 0.23	91.4 ± 0.23	74.9 ± 2.32	70.3 ± 1.24	72.5 ± 0.70	72.1 ± 2.16	67.7 ± 1.28	69.8 ± 0.67
ACE05 (∗)	ITER *(ours)*										
	+ FLAN T5 *(large)*	5	90.6 ± 0.38	92.4 ± 0.20	91.5 ± 0.24	75.5 ± 0.59	72.5 ± 0.64	74.0 ± 0.40	72.5 ± 0.34	69.6 ± 0.71	71.1 ± 0.59
	+ FLAN T5 *(xl)*	5	90.6 ± 0.26	92.6 ± 0.26	91.6 ± 0.12	77.0 ± 0.62	74.1 ± 0.84	75.6 ± 0.37	73.9 ± 0.51	71.2 ± 0.93	72.6 ± 0.44
	+ DeBERTaV3 *(large)*	5	91.1 ± 0.49	92.7 ± 0.30	91.9 ± 0.38	76.5 ± 0.90	72.2 ± 0.66	74.3 ± 0.56	73.6 ± 0.91	69.4 ± 0.54	71.4 ± 0.53
	+ T5 *(large)*	3	90.0 ± 0.55	91.3 ± 0.02	90.7 ± 0.27	76.4 ± 1.05	63.7 ± 0.38	69.4 ± 0.43	73.6 ± 0.89	61.4 ± 0.50	67.0 ± 0.42
	+ BART *(large)*	3	90.7 ± 0.27	92.0 ± 0.08	91.4 ± 0.15	75.0 ± 0.76	69.0 ± 0.57	71.9 ± 0.50	72.3 ± 0.33	66.4 ± 0.61	69.2 ± 0.29
	+ ALBERT *(xxlarge-v2)*	3	90.8 ± 0.32	92.1 ± 0.23	91.4 ± 0.23	75.2 ± 2.30	70.8 ± 1.08	72.9 ± 0.78	72.6 ± 2.16	68.4 ± 1.13	70.4 ± 0.75
ADE	ITER *(ours)*										
	+ FLAN T5 *(large)*	10	91.1 ± 0.93	92.1 ± 0.85	91.6 ± 0.73	83.6 ± 1.46	85.0 ± 1.85	84.3 ± 1.46	83.6 ± 1.46	85.0 ± 1.85	84.3 ± 1.46
	+ FLAN T5 *(xl)*	10	91.2 ± 1.37	93.3 ± 0.60	92.2 ± 0.89	84.3 ± 1.95	87.0 ± 1.38	85.6 ± 1.42	84.3 ± 1.95	87.0 ± 1.38	85.6 ± 1.42
	+ DeBERTaV3 *(large)*	10	90.6 ± 1.24	92.7 ± 0.90	91.6 ± 0.77	83.2 ± 2.00	86.3 ± 1.85	84.7 ± 1.39	83.2 ± 2.01	86.3 ± 1.86	84.7 ± 1.40
	+ BERT *(large-cased)*	10	89.9 ± 0.97	91.8 ± 1.02	90.9 ± 0.85	81.4 ± 2.16	84.4 ± 2.22	82.8 ± 1.83	81.4 ± 2.16	84.4 ± 2.22	82.8 ± 1.83
SciERC	ITER *(ours)*										
	+ FLAN T5 *(large)*	5	64.9 ± 1.05	62.8 ± 0.59	63.8 ± 0.75	51.7 ± 2.53	36.0 ± 2.19	42.3 ± 1.12	38.5 ± 1.69	26.8 ± 2.12	31.5 ± 1.47
	+ FLAN T5 *(xl)*	5	68.3 ± 0.50	69.9 ± 1.19	69.1 ± 0.46	56.7 ± 0.83	46.4 ± 2.07	51.0 ± 1.12	43.2 ± 0.78	35.4 ± 1.35	38.9 ± 0.58
	+ DeBERTaV3 *(large)*	5	67.3 ± 1.12	69.0 ± 1.93	68.1 ± 1.21	56.0 ± 1.39	45.5 ± 0.95	50.2 ± 0.22	41.2 ± 1.51	33.5 ± 1.92	36.9 ± 1.62
SciERC (∗)	ITER *(ours)*										
	+ FLAN T5 *(large)*	5	64.9 ± 1.05	62.8 ± 0.59	63.8 ± 0.75	52.7 ± 2.38	36.8 ± 2.39	43.3 ± 1.21	39.1 ± 1.61	27.3 ± 2.39	32.1 ± 1.74
	+ FLAN T5 *(xl)*	5	68.3 ± 0.50	69.9 ± 1.19	69.1 ± 0.46	57.7 ± 1.00	48.3 ± 2.28	52.6 ± 1.04	44.2 ± 1.01	37.0 ± 1.53	40.2 ± 0.54
	+ DeBERTaV3 *(large)*	5	67.3 ± 1.12	69.0 ± 1.93	68.1 ± 1.21	57.6 ± 1.38	47.9 ± 1.16	52.2 ± 0.28	42.5 ± 1.29	35.4 ± 2.09	38.6 ± 1.63
CoNLL03	ITER *(ours)*										
	+ FLAN T5 *(large)*	5	90.9 ± 0.59	92.3 ± 0.28	91.6 ± 0.39						
	+ FLAN T5 *(xl)*	5	91.2 ± 0.87	92.4 ± 0.53	91.8 ± 0.69						
	+ DeBERTaV3 *(large)*	5	91.4 ± 0.70	92.1 ± 0.33	91.8 ± 0.29						
GENIA	ITER *(ours)*										
	+ FLAN T5 *(large)*	5	81.7 ± 3.47	78.7 ± 7.25	80.2 ± 2.52						
	+ FLAN T5 *(xl)*	5	81.6 ± 0.60	80.8 ± 0.76	81.2 ± 0.25						
	+ DeBERTaV3 *(large)*	5	82.0 ± 0.24	79.2 ± 0.49	80.6 ± 0.24						

Table 6: Final training results for all used datasets: CoNLL04, ACE05, ADE, SciERC, CoNLL03 and GENIA. We run experiments on five seeds (three for ablation studies) and report the mean performance and the standard deviation. (∗): On the SciERC and ACE05 datasets, we implement PL Marker and Wang et al.'s strict F1 scoring for symmetric relations to get comparable results. More information regarding this evaluation method can be found in Appendix F.