An Autoregressive Text-to-Graph Framework for Joint Entity and Relation Extraction

Urchade Zaratiana[1,2], **Nadi Tomeh**[2], **Pierre Holat**[1,2], **Thierry Charnois**[2]

[1] FI Group, Puteaux, France
[2] LIPN - Université Sorbonne Paris Nord - CNRS UMR 7030, Villetaneuse, France
{zaratiana, tomeh, charnois}@lipn.fr, pierre.holat@fi-group.com

Abstract

In this paper, we propose a novel method for joint entity and relation extraction from unstructured text by framing it as a conditional sequence generation problem. In contrast to conventional generative information extraction models that are left-to-right token-level generators, our approach is *span-based*. It generates a linearized graph where nodes represent text spans and edges represent relation triplets. Our method employs a transformer encoder-decoder architecture with pointing mechanism on a dynamic vocabulary of spans and relation types. Our model can capture the structural characteristics and boundaries of entities and relations through span representations while simultaneously grounding the generated output in the original text thanks to the pointing mechanism. Evaluation on benchmark datasets validates the effectiveness of our approach, demonstrating competitive results. Code is available at https://github.com/urchade/ATG.

Introduction

Joint entity and relation extraction is a fundamental task in Natural Language Processing (NLP), serving as the basis for various high-level applications such as Knowledge Graph construction (Ye et al. 2022b) and question answering (Chen et al. 2017). Traditionally, this task was tackled via pipeline models that independently trained and implemented entity recognition and relation extraction, often leading to error propagation (Brin 1999). Deep learning has led to the creation of end-to-end models, allowing for the use of shared representations and joint optimization of loss functions for both tasks (Wadden et al. 2019; Wang and Lu 2020; Zhao et al. 2021; Zhong and Chen 2021; Yan et al. 2021). Despite this advancement, these models essentially remain pipeline-based, with entity and relation predictions executed by separate classification heads, thereby ignoring potential interactions between these tasks.

Recent advancements have seen a shift towards "real" end-to-end solutions, where the prediction of entities and relations is intertwined, accomplished through autoregressive models. These models treat the joint entity-relation task as a process of generating plain text, employing *augmented languages* to encode and decode structural information (Paolini et al. 2021; Lu et al. 2022; Liu et al. 2022; Fei et al. 2022).

While these models have achieved remarkable performance, we argue that they also expose room for improvement, especially in terms of grounding the output in the input text.

In this paper, we present an autoregressive transformer encoder-decoder model that generates a linearized graph instead of generating plain text. Our model makes use of a pointing mechanism (Vinyals, Fortunato, and Jaitly 2015) on a dynamic vocabulary of spans and relations, providing explicit grounding in the original text. In fact, without grounding, models can generate output that are semantically coherent but contextually detached from the input. Our pointing mechanism mitigates this issue by ensuring that the decoder's outputs, specifically the entity spans, are directly tied to the input text. Furthermore, by generating spans and relations directly from the text, rather than producing standalone plain text, our model encode the structural characteristics and boundaries of entities/spans more accurately, which can be missed by previous generative information extraction models. The cornerstone of our solution is the explicit enumeration of all spans[1] at the encoder's output, making them readily available to the decoder. Although the number of spans can be extensive, we note that when bounding the span size, our model's vocabulary is typically smaller than that of traditional language models (Devlin et al. 2019; Raffel et al. 2019), as discussed in subsequent sections.

Moreover, as previous generative IE models operate at the token level, they scatter the information regarding an entity's span and its boundaries over multiple decoding steps. In contrast, generating an entity and its type in our approach is accomplished in a single decoding step, resulting in shorter sequence (Figure 5). Additionally, our method naturally ensures the well-formedness of the output while some generative IE models that produce text, often require non-regular constraints. As an example, in TANL (Paolini et al. 2021), if a portion of the generated sentence has an invalid format, that segment is discarded. Such issues are readily addressed in our model since the vocabulary can be fully controlled.

We evaluated our model on three benchmark datasets for joint entity and relation extraction: CoNLL 2004, SciERC, and ACE 05. Our model demonstrated competitive performance on all datasets. Our contributions can be summarized as follows:
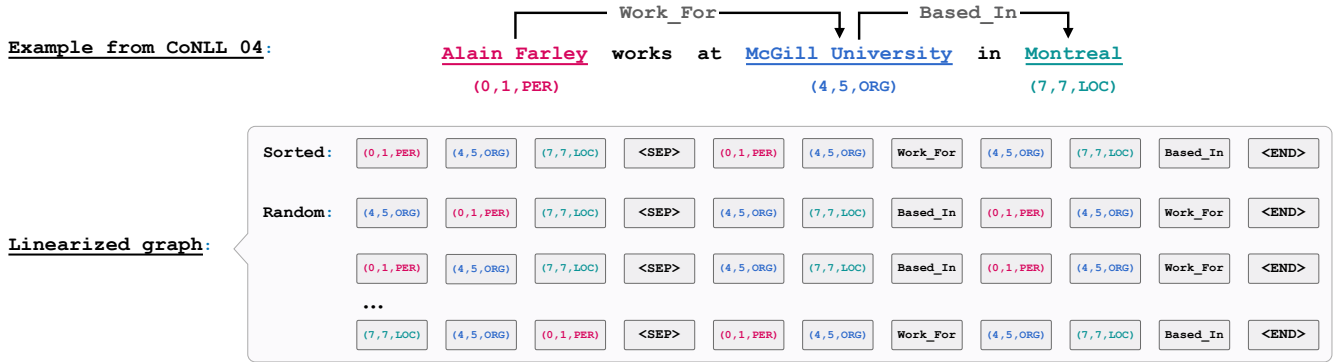
---

[1]Up to a certain length in practice.

Figure 1: Linearization for Information Graph Generation. The input text is mapped into an information extraction graph. The graph consists of entities and relation triplets, which are generated sequentially by first producing entity spans (represented by start word, end word, and entity type) followed by relation triplets (head entity, tail entity, and relation type).

- We propose a novel method for joint entity and relation extraction by framing it as a conditional sequence generation problem. Our approach generates a linearized graph representation, where nodes represent entity spans and edges represent relation triplets.

- Our model employs a transformer encoder-decoder architecture with a pointing mechanism on a dynamic vocabulary of spans and relation types. This allows to capture the structural characteristics and boundaries of entities and relations while grounding the generated output in the original text.

- We demonstrate the effectiveness of our approach through extensive evaluations on benchmark datasets, including CoNLL 2004, SciERC, and ACE 05. Our model achieves state-of-the-art results on CoNLL 2004 and SciERC, surpassing previous comparable models in terms of Entity F1 scores and Relation F1 scores.

Task Definition

We address the task of joint entity and relation extraction from text as a graph generation approach. Our proposed model generates nodes and edges as a single sequence, effectively integrating both entity and relation extraction into a unified framework. Formally, the task can be defined as follows: Given an input text sequence $\boldsymbol{x} = \{x_1, x_2, ..., x_L\}$, where x_i represents the i-th token in the sequence, our objective is to generate a linearized graph representation $\mathbf{y} = \{y_1, y_2, ..., y_M\}$, where y_j represents a token in the generated sequence. As shown in Figure 1, in Each token y_j can take one of three forms:

- **Entity span**: The token y_j represents an entity span, defined as $y_j = (s_j, e_j, t_j)$, where s_j and e_j denote the starting and ending positions of the entity span, and t_j denotes the type of the entity.

- **Relation type**: The token y_j represents a relation type between two entities, such as `Work_For` relation.

- **Special token**: The token y_j represents special tokens used in the generation process, such as `<SEP>` to sep-

arate entities and relations or the `<END>` to stop the generation.

The template employed in our model, we refer to **ATG** (**A**utoregrestive **T**ext-to-**G**raph), is depicted in Figure 1. It starts by generating the entities, followed by a `<SEP>` token, and ends with the relation triplets, each consisting of head node, tail node and edge/relation type. During training, we try two distinct orderings for the graph linearization: *sorted* and *random*. As shown in the Figure 1, the *sorted* linearization organizes entities and relations based on their positions in the original text, while the *random* linearization randomly shuffles entities and relations order.

Model Architecture

Our model, ATG, employed an encoder-decoder architecture, which processes the input text sequence and produces a linearized graph as illustrated in the Figure 2.

Encoder

The encoder in ATG utilizes a transformer layer that takes an input text sequence $\mathbf{x}$ and outputs token representations $\mathbf{H} \in \mathbb{R}^{L \times D}$, where D is the model dimension.

Vocabulary Construction

Dynamic vocabulary To enable the pointing mechanism in our decoder, we construct a dynamic vocabulary matrix $\mathbf{E}$ that includes embeddings for spans, special tokens, and relation types. While special tokens and relation type embeddings are randomly initialized and updated during training, the span embeddings are dynamically computed (Zaratiana et al. 2023), *i.e* their representations depend on the input sequence. More specifically, the embedding of a span (start, end, type) is computed as follows:

$$\mathbf{S}_{[\text{start,end,type}]} = \mathbf{W}_{\text{type}}^T [\mathbf{h}_{\text{start}} \odot \mathbf{h}_{\text{end}}] \qquad (1)$$

In this equation, $[\odot]$ represents a concatenation operation; $\mathbf{h}_{\text{start}}$ and $\mathbf{h}_{\text{end}}$ denote the representations of tokens at the start and end positions, respectively. $\mathbf{W}_{\text{type}} \in \mathbb{R}^{2D \times D}$ is a weight matrix associated with the entity type (*i.e*, there is a
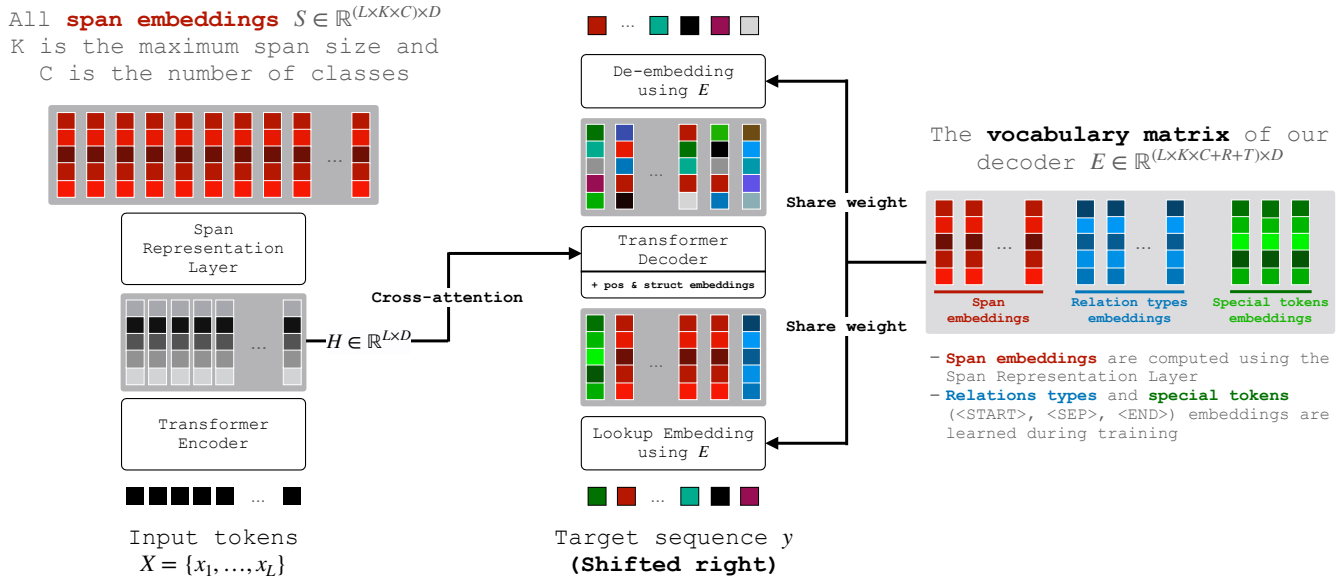
Figure 2: Illustration of the architecture of our model, ATG. (*left*) The Encoder takes in the input sequence X and generates representations of the tokens $\boldsymbol{H}$ and spans $\boldsymbol{S}$. (*middle*) The Decoder then generates the next token conditioned on the previous tokens and the input representation $\boldsymbol{H}$. (*right*) The vocabulary matrix used for decoding consists of the concatenation of span embeddings $\boldsymbol{S}$, learned relation type embeddings, and special token embeddings.

$\mathbf{W}_{\text{type}}$ for each entity types in a datasets). Finally, the vocabulary embedding matrix $\mathbf{E}$ is formed by stacking all the span embeddings $\mathbf{S}$, special token embeddings $\mathbf{T}$, and relation type embeddings $\mathbf{R}$.

Vocabulary size The size of the vocabulary matrix $\mathbf{E} \in \mathbb{R}^{V \times D}$ is $V = L \times K \times C + R + T$, where L represents the sequence length, K the maximum span size, C the number of entity types, R the number of relations, and T the number of special tokens (<START>, <END>, and <SEP>). Let's take the CoNLL 2004 dataset as an example to illustrate this. This dataset has the following characteristics: $K = 12$, $C = 4$, $R = 5$, and $T = 3$. Considering a sentence of length 114 (which is the maximum length in the training set), the resulting vocabulary size would be 5480. This size is considerably smaller when compared with the vocabulary size of a typical language model, which usually hovers around 30,000 distinct tokens.

Decoder

The decoder is a causal transformer trained to predict the next token in the sequence, akin to traditional language modeling. However, it is important to note that the vocabulary of our decoder consists of entity spans, relation types, and special tokens, rather than plain text. The decoder conditions its predictions on the previously generated tokens $y_{<j}$ using self-attention and on the input token representations $\mathbf{H}$ using cross-attention. This enables the decoder to attend to relevant information from both the previously generated tokens and the input text. Through attention visualizations, as depicted in Figure 8 and 9, we observed that the model effectively harnesses both sources of information. Finally,

The training objective aims to maximize the following conditional probability:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{j=1}^{M} p(y_j|\boldsymbol{y}_{<j}, \boldsymbol{H}) \qquad (2)$$

This is achieved during the training by minimizing the negative log-likelihood of a reference sequence obtained by linearizing the reference IE graph. Details about the decoder input and output are given in the subsequent paragraphs.

Decoder input embedding The embedding step feeds the previous decoder outputs $y_1, \ldots, y_{i-1}$ into the model using the vocabulary matrix $\mathbf{E}$, along with *positional* and *structural* embeddings as shown in Figure 3. This process can be expressed as follows:

$$\begin{aligned} \boldsymbol{z}_1, \ldots, \boldsymbol{z}_{i-1} =& \boldsymbol{E}[y_1, \ldots, y_{i-1}] \\ &+ \boldsymbol{E}_{pos}[1, \ldots, i-1] \\ &+ \boldsymbol{E}_{struct}[y_1, \ldots, y_{i-1}] \end{aligned} \qquad (3)$$

Here, $\boldsymbol{E}[y_1, \ldots, y_{i-1}]$ corresponds to the token embeddings, which may corresponds to spans, relation types, or special tokens embeddings depending on the nature of $y_{<i}$. The matrix $\boldsymbol{E}_{pos}$ represents absolute positional embedding. It allows to capture the positional information of the decoder outputs from y_1 to y_{i-1}. Additionally, the structural embedding $\boldsymbol{E}_{struct}$ serve as indicators to guide the model in generating specific elements. In particular, they provide information about whether the model should generate a Node (before <SEP> tokens), Tail, Head nodes, or Relation types (as illustrated in Figure 3). Both the absolute position encoding and the structural embedding are randomly initialized and

Decoder Output	(0,1,PER)	(4,5,ORG)	(7,7,LOC)	\<SEP>	(0,1,PER)	(4,5,ORG)	Work_For	(4,5,ORG)	(7,7,LOC)	Based_In	\<END>
Decoder Input Embedding	$E_{<START>}$	$E_{(0,1,PER)}$	$E_{(4,5,ORG)}$	$E_{(7,7,LOC)}$	$E_{<SEP>}$	$E_{(0,1,PER)}$	$E_{(4,5,ORG)}$	E_{Work_For}	$E_{(4,5,ORG)}$	$E_{(7,7,LOC)}$	E_{Based_In}
	+	+	+	+	+	+	+	+	+	+	+
Positional Embedding	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}
	+	+	+	+	+	+	+	+	+	+	+
Structural Embedding	E_{Node}	E_{Node}	E_{Node}	E_{Node}	E_{Head}	E_{Tail}	$E_{Relation}$	E_{Head}	E_{Tail}	$E_{Relation}$	E_{Head}

Figure 3: Input/ouptut of the decoder. The process starts with the special token \<START> and continues until the \<END> token is generated. To separate the generation of nodes and edges, a special token \<SEP> is used. At each position, the decoder takes in the sum of the embedding of the current token, absolute position embedding, and structural embedding.

updated during training. In summary, by combining these embeddings, ATG can capture the semantic, positional and structural information about the linearized graph.

Decoder output   We define $\tilde{z}_i$ as the hidden state at the last position of the decoder output sequence obtained by feeding the previous output embedding and the encoder outputs H (for cross-attention) to the decoder, *i.e*,

$$\tilde{z}_i = \mathcal{D}ecoder(z_1, \ldots, z_{i-1}; H)[-1] \qquad (4)$$

Then, to compute the probability distribution over the dynamic vocabulary for generating the next token, y_i, our model employs the softmax function on the dot product between the dynamic vocabulary embedding matrix E and $\tilde{z}_i$:

$$p(y_i|y_{<i}, H) = \frac{\exp E^T \tilde{z}_i}{\sum_{k=1}^{V}(\exp E^T \tilde{z}_i)_k} \qquad (5)$$

The probabilities generated by this formulation allow the model to select the appropriate token from the vocabulary for generating a span, special token, or relation type.

Constrained decoding   During inference, we sample from the model by enforcing constraints that preserve the well-
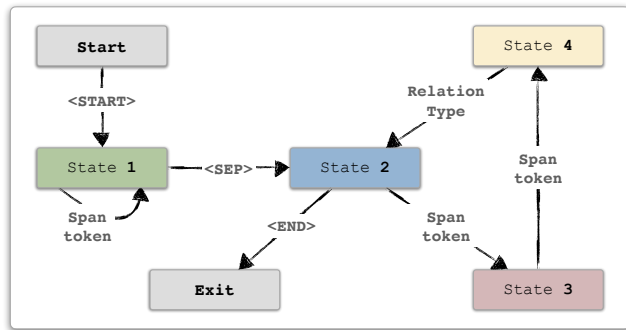

Figure 4: State-Transition diagram for constrained decoding. This diagram illustrates the state-based decision process used during the inference phase, which ensures the generation of a correct graph. Each state is represented by a node, and directed edges indicate valid actions. We use the same color code as the *structural embedding* in Figure 3.

inference, we feed our model with the \<START> token, and the generation process is guided by state-transition constraints as outlined in Figure 4. This structured approach ensures that each step in the generation aligns with the defined template, thereby maintaining the well-formedness of the output and allowing the production of valid IE graphs. All generations start with the start token **Start** state and continue until the \<END> token is sampled. In practice, we also add other constraints: in state **1**, we prevent the repetition of already generated spans, and in state **3**, we ensure the tail span is different from the head. Furthermore, it is also possible to incorporate domain knowledge into the prediction. For instance, if the type of a head entity is PER and the tail entity is ORG, the relation can be constrained to be Work_For (CoNLL 04 dataset).

Training with Sentence Augmentation

In our work, we observed oversmoothing (Kulikov, Eremeev, and Cho 2022), where the model prematurely generates the \<EOS>, *i.e* a bias towards short sequences (Murray and Chiang 2018; Xuewen et al. 2021). We found this bias to harm the recall of the tasks as the generation terminates before predicting all the entities/relations. To counteract this, we propose sentence augmentation, drawing inspiration from Pix2seq (Chen et al. 2022), who encounter a similar problem for their generative object detection model. This approach forms an augmented training sample s_{aug} by randomly concatenating sentences from the training set $\mathcal{D} = \{s_1, s_2, ..., s_N\}$:

$$s_{\text{aug}} = \bigoplus_{k=1}^{n} s_{i_k}, \quad i_k \sim \mathcal{U}(1, N), n \sim \mathcal{U}(1, B) \qquad (6)$$

Here, $\mathcal{U}$ denotes the uniform distribution, and B is a hyperparameter indicating the maximum number of sentences that can be concatenated. By applying this sentence augmentation technique during training, the model is exposed to diverse and longer output sequence lengths, reducing the risk of premature generation of the \<EOS> token and thus improving recall. We perform an ablation study of the effect of sentence augmentation in our experiments section, showing that it largely improves the overall performance of our model.

Input text: **Alain Farley** works at **McGill University** in **Montreal**

TANL	[Alain Farley | person | work for = McGill University] works at [McGill University | organization | based in = Montreal] in [Montreal | location]	**31 tokens**
UIE	((person: Alain Farley (work for: McGill University)) (organization: McGill University (based in: Montreal)) (location: Montreal))	**32 tokens**
LasUIE	{((Alain Farley, person [work for](McGill University, organization)) (McGill University, organization [based in] (Montreal, location)))}	**33 tokens**
ATG	(0,1,PER) (4,5,LOC) (7,7,LOC) <SEP> (0,1,PER) (4,5,LOC) Work_For (4,5,LOC) (7,7,LOC) Based_In	**10 tokens**

Figure 5: Linearization for different models. In contrast to existing approaches (*TANL* (Paolini et al. 2021), *UIE* (Lu et al. 2022), *LasUIE* (Fei et al. 2022)), our proposed model, *ATG*, generates spans (along with relation/special tokens) instead of text tokens, which allows for a shorter output sequence, richer (span-level) representation and fully controlled decoding.

Experimental Setup

Datasets

We evaluated our model on three benchmark English datasets for joint entity-relation extraction, namely SciERC (Luan et al. 2018), CoNLL 2004 (Carreras and Màrquez 2004), and ACE 05 (Walker et al. 2006). The statistics of the dataset is reported on Table 1.

ACE 05 is collected from a variety of domains, such as newswire, online forums and broadcast news. It provides a diverse set of entity types such as Persons (PER), Locations (LOC), Geopolitical Entities (GPE), and Organizations (ORG), along with intricate relation types that include ART (Artifact relationships), GEN-AFF (General affiliations), and PER-SOC (Personal social relationships). This dataset is particularly notable for its complexity and wide coverage of entity and relation types, making it a robust benchmark for evaluating the performance of IE models.

CoNLL 2004 is an annotated corpus collected from newswires and focuses on general entities such as People, Organizations, and Locations, and relations like Work_For and Live_in.

SciERC is a dataset that comes with entity, coreference, and relation annotations for a collection of documents from 500 AI paper abstracts. The dataset defines scientific term

Dataset	$|\mathcal{E}|$	$|\mathcal{R}|$	# Train	# Dev	# Test
ACE05	7	6	10,051	2,424	2,050
CoNLL 04	4	5	922	231	288
SciERC	6	7	1,861	275	551

Table 1: The statistics of the datasets. We use ACE04, ACE05, SciERC, and CoNLL 04 for evaluating end-to-end relation extraction.

types and relation types specifically designed for AI domain knowledge graph construction.

Evaluation Metrics

For the NER task, we adopt a span-level evaluation requiring precise entity boundaries and type predictions. To evaluate relations, we use two metrics: (1) Boundaries evaluation (**REL**) necessitates the correct prediction of entity boundaries and relation types; (2) Strict evaluation (**REL+**) additionally require accurate entity type prediction. We report the micro-averaged F1 score.

Baselines

We succinctly and briefly describe here the baseline that we compared with our model, which we separate into two categories: Span-based/table-filling and generative IE.

Span-based and table-filling *DyGIE++* (Wadden et al. 2019) is a model that uses a pretrained transformer to compute contextualized representations and employs graph propagation to update the representations of spans for prediction. *Tab-Seq* (Wang and Lu 2020) tackles the task of joint information extraction by treating it as a table filling problem. *PURE* (Zhong and Chen 2021) is a pipeline model for the information extraction task that learns distinct contextual representations for entities and relations. *PFN* (Yan et al. 2021) introduces methods that model two-way interactions between tasks by partitioning and filtering features. *UniRE* (Wang et al. 2021) proposes a joint entity and relation extraction model that eliminates the separation of label spaces for entity detection and relation classification. Their model uses a unified classifier to predict labels for each cell in a table of word pairs. In *TablERT* (Ma, Hiraoka, and Okazaki 2022), entities and relations are treated as tables, and the model utilizes two-dimensional CNNs to effectively capture and model local dependencies.

Models	SciERC			ACE 05			CoNLL 2004		
	ENT	REL	REL+	ENT	REL	REL+	ENT	REL	REL+
DYGIE++ (Wadden et al. 2019)	67.5	<u>48.4</u>	–	88.6	63.4	–	–	–	–
Tab-Seq (Wang and Lu 2020)	–	–	–	89.5	–	64.3	90.1	73.8	73.6
PURE (Zhong and Chen 2021)	66.6	48.2	35.6	88.7	66.7	63.9	–	–	–
PFN (Yan et al. 2021)	66.8	–	<u>38.4</u>	89.0	–	**66.8**	–	–	–
UniRE (Wang et al. 2021)	<u>68.4</u>	–	36.9	89.9	–	66.0	–	–	–
TablERT (Ma, Hiraoka, and Okazaki 2022)	–	–	–	87.8	65.0	61.8	**90.5**	73.2	72.2
GENERATIVE									
HySPA (Ren et al. 2021)	–	–	–	88.9	68.2	–	–	–	–
TANL (Paolini et al. 2021)	–	–	–	89.0	–	63.7	90.3	–	70.0
ASP (Liu et al. 2022) [†]	–	–	–	**91.3**	72.7	*70.5[†]*	<u>90.3</u>	–	*76.3*
UIE (Lu et al. 2022)	–	–	36.5	–	–	<u>66.6</u>	–	75.0	–
LasUIE (Fei et al. 2022)	–	–	–	–	–	66.4	–	75.3	–
ATG (*Our model*)	**69.7**	**51.1**	**38.6**	<u>90.1</u>	**68.7**	66.2	**90.5**	**78.5**	**78.5**

Table 2: Comparison of our proposed model with state-of-the-art methods. Results are reported in terms of Entity (ENT) F1, Relation (REL) F1, and Strict Relation (REL+) F1 scores. The best scores are shown in bold, and the second-best scores are underlined. [†] *Italic scores use undirected evaluation for relation extraction and thus are not strictly comparable to our results.*

Generative IE *HySPA* (Ren et al. 2021) is a model for text-to-graph extraction that has linear space and time complexity using a Hybrid span generator. *TANL* (Paolini et al. 2021) treat the joint IE task as translation from plain text to augmented natural languages by fine-tuning a T5 model (Raffel et al. 2019). This model has been further extended by *UIE* (Lu et al. 2022) and *LasUIE* (Fei et al. 2022), which both proposed better linearization and additional pretraining to enhance results. Finally, *ASP* (Liu et al. 2022) handles entity and relation extraction by encoding the target structure as a series of structure-building actions, using a conditional language model to predict these actions.

Hyperparameter Settings

Our model, ATG, employs a transformer encoder-decoder (Vaswani et al. 2017) architecture. We train it for a maximum of 70k steps using AdamW (Loshchilov and Hutter 2017) optimizer. We use learning rate warmup for the first 10% of training and then decay to 0. The base learning rates are 3e-5 for the encoder, 7e-5 for the decoder, and 1e-4 for other projection layers. Unlike other generative IE models that utilize pretrained encoder-decoder architectures, often relying on large models such as T5 (Raffel et al. 2019), we initialize ATG's encoder with pre-trained transformer encoders, while the decoder is randomly initialized. In our preliminary experiments, we observed that initializing ATG with a pretrained encoder-decoder led to suboptimal performance. We hypothesize that this is due to our decoder's utilization of a dynamic vocabulary, whereas existing pretrained encoder-decoder models have a fixed token vocabulary, which creates a large discrepancy. We use DeBERTa (He, Gao, and Chen 2021) for Conll-04 and ACE 05 and SciBERT (Beltagy, Lo, and Cohan 2019) for SciERC dataset. Across all configurations, the number of decoder layers is set to 6, though we noted that even a single layer

can be enough in certain cases. The sentence augmentation hyperparameter B is set to 5.

Main Results

Table 2 presents the main results of our experiments, along with comparable approaches from the literature. ATG demonstrates strong performance across all datasets. On the SciERC dataset, ATG achieves the highest scores across all metrics. It outperforms the second-best result by 0.2 in **REL+** and surpasses the best generative approach, UIE (Lu et al. 2022), by 2.1 points. On ACE 05, ATG provide a competitive performance, securing the second-highest scores. The reported top-performing model, ASP (Liu et al. 2022), operates under a relaxed, undirected relation evaluation, thereby limiting a fair comparison of results (Taillé et al. 2021). On the CoNLL 2004 dataset, ATG exhibits its superiority by outperforming the second-best result by 2.2 in terms of **REL+**. Overall, across all three datasets, our proposed model either holds the top position or showcases strong competitive performance.

Ablation Studies

Number of decoder layers The number of decoder layers impact is illustrate on the Figure 6. It has a varying impact on performance across the datasets. In SciERC, increasing the number of decoder layers leads to a gradual improvement in the performance, reaching a peak of 38.6 at 6 layer. For ACE 05, the score shows a slight improvement from 65.3 to 66.2 as the number of decoder layers increases from 1 to 6. For the CoNLL 2004 dataset, the score fluctuates with different numbers of decoder layers, achieving already strong performance with only a single layer. Overall, the choice of the number of decoder layers can have a noticeable impact on **REL+** performance but the effect may vary across datasets.
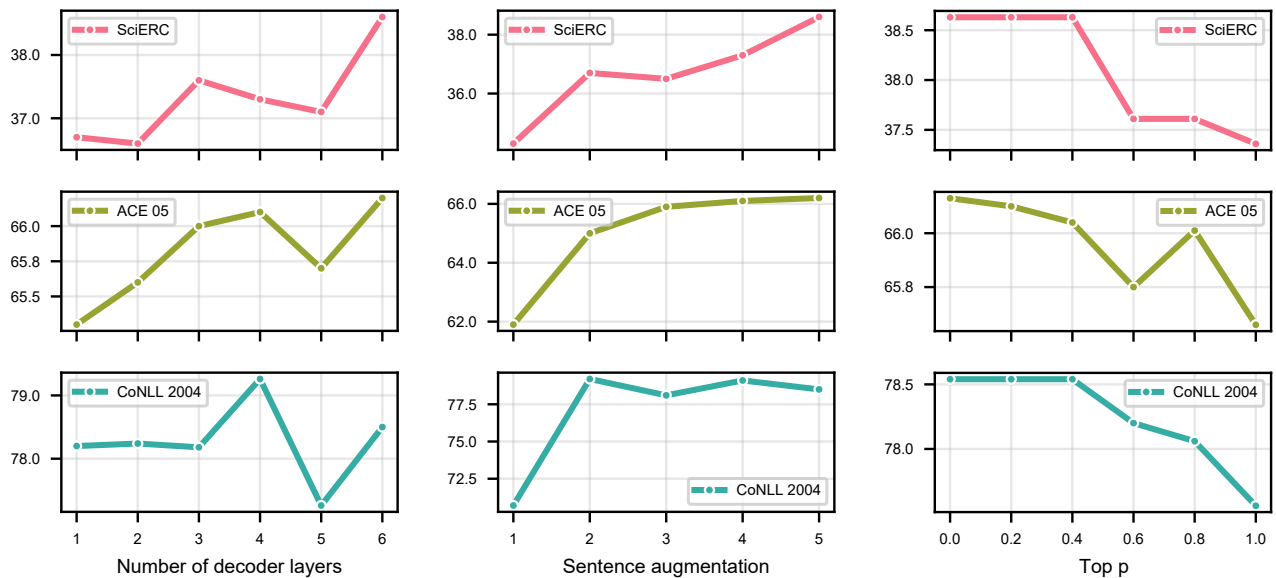
Figure 6: Investigation of the effect of different choices on model performance (REL+). (*Left*) Effect of the number of decoder layers, (*Center*) Impact of Sentence Augmentation, (*Right*) Study of different values of top-p for Nucleus Sampling.

Sentence augmentation The effect of sentence augmentation size on **REL+** performance is illustrated in Figure 6. The results reveal that increasing the number of sentence augmentations always improves performance across all datasets, except for CoNLL, where achieving state-of-the-art (SOTA) results is possible with just a size of 2. However, the absence of sentence augmentation leads to a significant decrease in **REL+**, proving its importance.

Nucleus sampling The impact of different top p values in nucleus sampling (Holtzman et al. 2019) on the performance (REL+) is shown in Figure 6. The scores across all datasets demonstrate a relatively stable trend, with minor variations observed as the top p value changes. This can be attributed to the application of constrained decoding, which ensures that the output remains well-formed. However, the lowest values of top p, corresponding to greedy decoding, consistently deliver the best performance.

Positional and structural smbeddings Table 3 illustrates the importance of positional and structural encoding on the Relation F1 score. When employing both encoding, **ATG** achieves the best performance across all datasets (38.6 for

	SciERC	ACE 05	CoNLL 2004
Full	**38.6**	**66.2**	**78.5**
- *Pos*	<u>36.4</u>	<u>66.0</u>	78.3
- *Struct*	36.1	65.8	<u>78.4</u>
- *Both*	35.4	65.4	78.0

Table 3: Effect of positional (*Pos*) and structural embedding (*Struct*) on REL+.
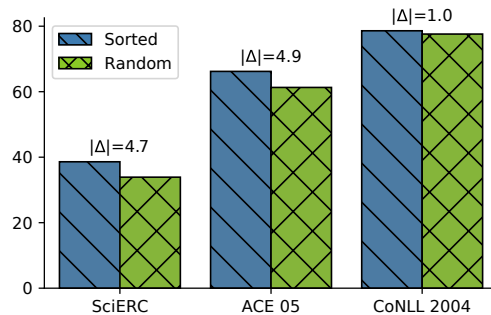

Figure 7: Impact of sequence ordering on REL+.

SciERC, 66.2 for ACE 05, and 78.5 for CoNLL 2004). Excluding positional encoding causes only slight performance drops, since the span representations may contain some positional information. Omitting structural encoding leads to similar, but slightly larger drops. Finally, when both are removed, the scores decrease the most, indicating their importance for the task.

Sequence ordering Figure 7 compares the effects of sorted and random sequence ordering across different datasets. The results clearly show that the sorted ordering approach consistently outperforms the random one. The difference in performance is particularly significant on SciERC and ACE 05, with improvements of 4.7 and 4.9, respectively. On the CoNLL 04 dataset, although the sorted approach still leads, the difference narrows to 1. Interestingly, we initially hypothesized that random ordering would deliver better performance, given that any generation errors in a sorted order could be difficult to rectify.

Figure 8: Decoder Self-Attention Visualization. This figure illustrates the attention patterns among elements in the generated sequence.
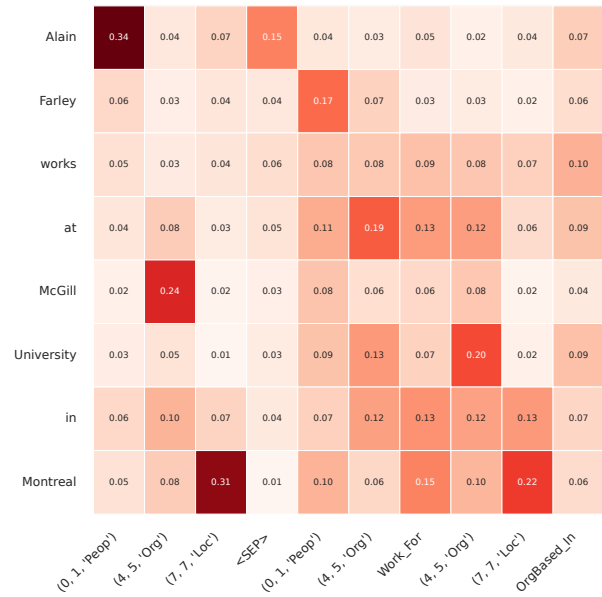

Figure 9: Decoder Cross-Attention Visualization. This map shows how each target element in the decoder interacts with and utilizes the original input text.

Interpretability Analysis

Attention Maps

Here we analyze the attention of the model during the decoding step, which allow us to explain some of the model's decision. We investigate both the self-attention (Fig. 8) and cross-attention (Fig. 9).

Self-attention The self-attention map, shown in Figure 8, depicts the distribution of attention across preceding tokens during generation. One notable observation is the model's tendency to focus on the head and tail entities that comprise the relation when predicting relation types. For example, when predicting the Work_For relation, the model allocates most of its attention weight to the tokens (0,1,Peop) and (4,5,Org).

Cross-attention The cross-attention map in Figure 9 indicates the specific areas in the input sequence that the decoded tokens attend to during generation. For entity labels in the output sequence such as (0,1,Peop), (4,5,Org), and (7,7,Loc), we can observe higher attention scores for the words Alain, McGill, and Montreal, respectively, in the input sequence. This indicates that the model tends to focuses on the beginning of each entity span when generating these entities in the output sequence. Furthermore, when predicting tail entities for relations, significant attention is directed toward the prepositions 'at' and 'in' in the input sequence. This suggests that the model has learned to associate these prepositions with specific relations between entities.

Learned Structure Embedding

To investigate the impact of the learned structure embedding on model performance, we analyze the similarity between the structure embeddings learned during training, depicted in Figure 10. Notably, we observe a consistent pattern across datasets: the embeddings for the Head and Tail exhibit a high negative correlation. This finding may suggest that the model learns to differentiate between the Head and Tail entities, capturing their distinct characteristics.
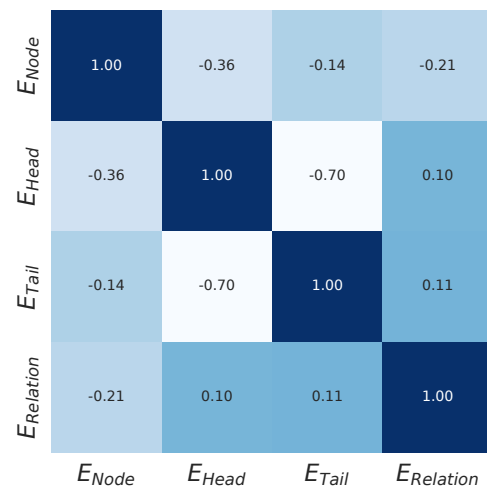

Figure 10: Structure embedding similarity. This map shows the cosine similarity between pairs of structure embedding.
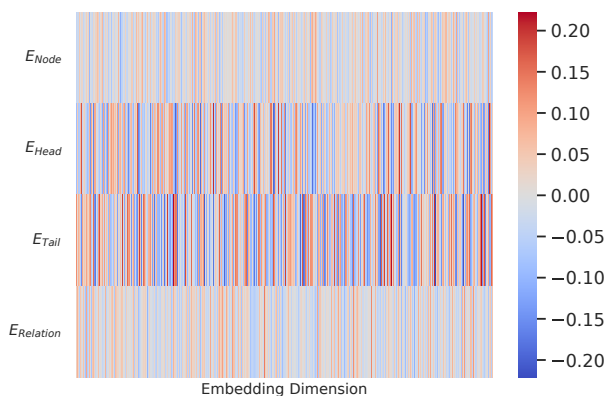
Figure 11: Structure embedding values. This shows the values taken by the learned structure embeddings.

However, we do not have a clear interpretation of this phenomenon. Additionally, Figure 11 illustrates structure embedding values over 512 dimensions. Noticeably, E_{Head} and E_{Tail} show higher values, indicating that predicting head and tail entities may be the most challenging for the model.

Related Works

Classification-based IE

In the field of information extraction (IE), traditional pipeline models have been used, consisting of separate stages for entity recognition and relation extraction (Roth and Yih 2004). Entity recognition is performed to identify mentioned entities (Chiu and Nichols 2015; Lample et al. 2016; Zaratiana et al. 2022a,b, 2023), followed by relation extraction to determine the relationships between these entities (Zelenko, Aone, and Richardella 2002; Bach and Badaskar 2007; Lin et al. 2016; Wu, Bamman, and Russell 2017). However, this approach suffers from error propagation, where mistakes in entity recognition can negatively impact the accuracy of relation extraction (Brin 1999; Roth and Yih 2004; Nadeau and Sekine 2007). To address these challenges, there has been a shift towards end-to-end models that jointly optimize both entity recognition and relation extraction. This joint optimization aims to harness the interplay between the two tasks, thereby enhancing overall performance (Sun et al. 2021; Zhao et al. 2021; Ye et al. 2022a). Noteworthy directions in this domain include table-filling methods (Wang and Lu 2020; Ma, Hiraoka, and Okazaki 2022), span pair classification (Eberts and Ulges 2019; Wadden et al. 2019), set prediction (Sui et al. 2020), augmented sequence tagging mechanisms (Ji et al. 2020), fine-grained triplet classification (Shang, Huang, and Mao 2022), and the use of unified labels for the task (Wang et al. 2021).

Generative IE

Recent advancements in generative Information Extraction (IE) emphasize the use of language models (LMs) to produce entities and relations, either as text or as a sequence of actions (Paolini et al. 2021; Lu et al. 2022; Nayak and Ng 2020; Liu et al. 2022; Fei et al. 2022; Wan et al. 2023). Typically, these models employ pretrained encoder-decoder architectures, such as T5 (Raffel et al. 2019) or BART (Lewis et al. 2020), to encode an input text and subsequently decode it into a structured output. Their primary advantage over non-generative methods is their ability to seamlessly integrate tasks by treating them as a unified generation process. A comprehensive review of this approach is available in (Ye et al. 2022b; Xu et al. 2023). Generative models have also found applications in other IE tasks, including entity linking (Cao et al. 2021), event extraction (Li, Ji, and Han 2021), and document-level IE (Giorgi, Bader, and Wang 2022).

Constrained Decoding

In the generative IE paradigm, the model can in principle generate any sequence over the LM's vocabulary if the decoder is not constrained in some way. One might resort to *controlled* to bias the model and guide the generation (Li et al. 2022; Kumar, Paria, and Tsvetkov 2022; Amini, Du, and Cotterell 2023). These approaches still generate a sequence over LM's vocabulary that needs to be mapped to an output graph using some kind of a *parser* that analyzes the output and extracts a well-formed structure (Paolini et al. 2021). Another approach incorporates constraints explicitly into the decoding algorithm to restrict the LM's vocabulary to allowed tokens. For instance, Cao et al. (2021) and Josifoski et al. (2022) use constrained beam search to force the output to a set of allowed entities and relations from a knowledge base schema. Another solution is to build a custom decoder that is constrained to a tailored vocabulary and decoding algorithm guaranteed to produce a linearized well-formed structure. Liu et al. (2022) and Lu et al. (2022) use a specialized decoder with some sort of explicit grammar over a specific vocabulary to restrict the output to valid sequences. Our work falls in this category which can be formalized as generating words from a formal language described by a grammar over a specialized alphabet (Willard and Louf 2023; Geng et al. 2023).

Conclusion

In conclusion, our autoregressive text-to-graph framework for joint entity and relation extraction has demonstrated its effectiveness in achieving state-of-the-art or competitive results on multiple benchmark datasets. By directly generating a linearized graph representation instead of plain text, ATG successfully captures the structural characteristics, boundaries, and interactions of entities and relations. Moreover, the pointing mechanism on dynamic vocabulary provides robust grounding in the original text, which also allows our model's decoding to be fully controllable.

Acknowledgments

References

Amini, A.; Du, L.; and Cotterell, R. 2023. Structured Voronoi Sampling. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Bach, N.; and Badaskar, S. 2007. A Review of Relation Extraction.

Beltagy, I.; Lo, K.; and Cohan, A. 2019. SciBERT: A Pretrained Language Model for Scientific Text. In *EMNLP 2019*.

Brin, S. 1999. Extracting Patterns and Relations from the World Wide Web. In *The World Wide Web and Databases*. ISBN 978-3-540-48909-2.

Cao, N. D.; Izacard, G.; Riedel, S.; and Petroni, F. 2021. Autoregressive Entity Retrieval. In *International Conference on Learning Representations*.

Carreras, X.; and Màrquez, L. 2004. Introduction to the CoNLL-2004 Shared Task: Semantic Role Labeling. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*.

Chen, D.; Fisch, A.; Weston, J.; and Bordes, A. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Chen, T.; Saxena, S.; Li, L.; Fleet, D. J.; and Hinton, G. 2022. Pix2seq: A Language Modeling Framework for Object Detection. In *International Conference on Learning Representations*.

Chiu, J. P. C.; and Nichols, E. 2015. Named Entity Recognition with Bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 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*.

Eberts, M.; and Ulges, A. 2019. Span-based Joint Entity and Relation Extraction with Transformer Pre-training. *ArXiv*, abs/1909.07755.

Fei, H.; Wu, S.; Li, J.; Li, B.; Li, F.; Qin, L.; Zhang, M.; Zhang, M.; and Chua, T.-S. 2022. LasUIE: Unifying Information Extraction with Latent Adaptive Structure-aware Generative Language Model. In *Advances in Neural Information Processing Systems*.

Geng, S.; Josifoski, M.; Peyrard, M.; and West, R. 2023. Grammar-Constrained Decoding for Structured NLP Tasks without Finetuning. In *EMNLP 2023*.

Giorgi, J.; Bader, G. D.; and Wang, B. 2022. A sequence-to-sequence approach for document-level relation extraction. In *Workshop on Biomedical Natural Language Processing*.

He, P.; Gao, J.; and Chen, W. 2021. DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing. *ArXiv*, abs/2111.09543.

Holtzman, A.; Buys, J.; Forbes, M.; and Choi, Y. 2019. The Curious Case of Neural Text Degeneration. *ArXiv*, abs/1904.09751.

Ji, B.; Yu, J.; Li, S.; Ma, J.; Wu, Q.; Tan, Y.; and Liu, H. 2020. Span-based Joint Entity and Relation Extraction with Attention-based Span-specific and Contextual Semantic Representations. In *Proceedings of the 28th International Conference on Computational Linguistics*. International Committee on Computational Linguistics.

Josifoski, M.; De Cao, N.; Peyrard, M.; Petroni, F.; and West, R. 2022. GenIE: Generative Information Extraction. In *NAACL 2022*.

Kulikov, I.; Eremeev, M.; and Cho, K. 2022. Characterizing and addressing the issue of oversmoothing in neural autoregressive sequence modeling. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Kumar, S.; Paria, B.; and Tsvetkov, Y. 2022. Gradient-based Constrained Sampling from Language Models. In *EMNLP 2022*.

Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural Architectures for Named Entity Recognition. In *NAACL 2016*.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 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*.

Li, S.; Ji, H.; and Han, J. 2021. Document-Level Event Argument Extraction by Conditional Generation. In *NAACL 2021*.

Li, X.; Thickstun, J.; Gulrajani, I.; Liang, P. S.; and Hashimoto, T. B. 2022. Diffusion-LM Improves Controllable Text Generation. In *Advances in Neural Information Processing Systems*.

Lin, Y.; Shen, S.; Liu, Z.; Luan, H.; and Sun, M. 2016. Neural Relation Extraction with Selective Attention over Instances. In *Annual Meeting of the Association for Computational Linguistics*.

Liu, T.; Jiang, Y. E.; Monath, N.; Cotterell, R.; and Sachan, M. 2022. Autoregressive Structured Prediction with Language Models. In *Findings of EMNLP 2022*.

Loshchilov, I.; and Hutter, F. 2017. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

Lu, Y.; Liu, Q.; Dai, D.; Xiao, X.; Lin, H.; Han, X.; Sun, L.; and Wu, H. 2022. Unified Structure Generation for Universal Information Extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Luan, Y.; He, L.; Ostendorf, M.; and Hajishirzi, H. 2018. Multi-Task Identification of Entities, Relations, and Coreferencefor Scientific Knowledge Graph Construction. In *EMNLP 2018*.

Ma, Y.; Hiraoka, T.; and Okazaki, N. 2022. Joint Entity and Relation Extraction Based on Table Labeling Using Convolutional Neural Networks. In *Proceedings of the Sixth Workshop on Structured Prediction for NLP*.

Murray, K.; and Chiang, D. 2018. Correcting Length Bias in Neural Machine Translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*.

Nadeau, D.; and Sekine, S. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*.

Nayak, T.; and Ng, H. T. 2020. Effective Modeling of Encoder-Decoder Architecture for Joint Entity and Relation Extraction. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, 8528–8535. AAAI Press.

Paolini, G.; Athiwaratkun, B.; Krone, J.; Ma, J.; Achille, A.; ANUBHAI, R.; dos Santos, C. N.; Xiang, B.; and Soatto, S. 2021. Structured Prediction as Translation between Augmented Natural Languages. In *International Conference on Learning Representations*.

Raffel, C.; Shazeer, N. M.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *ArXiv*, abs/1910.10683.

Ren, L.; Sun, C.; Ji, H.; and Hockenmaier, J. 2021. HySPA: Hybrid Span Generation for Scalable Text-to-Graph Extraction. In *Findings of ACL-IJCNLP 2021*.

Roth, D.; and Yih, W.-t. 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) at HLT-NAACL 2004*.

Shang, Y.-M.; Huang, H.; and Mao, X.-L. 2022. OneRel: Joint Entity and Relation Extraction with One Module in One Step. In *AAAI Conference on Artificial Intelligence*.

Sui, D.; Chen, Y.; Liu, K.; Zhao, J.; Zeng, X.; and Liu, S. 2020. Joint Entity and Relation Extraction with Set Prediction Networks. *IEEE transactions on neural networks and learning systems*, PP.

Sun, K.; Zhang, R.; Mensah, S.; Mao, Y.; and Liu, X. 2021. Progressive Multi-task Learning with Controlled Information Flow for Joint Entity and Relation Extraction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15): 13851–13859.

Taillé, B.; Guigue, V.; Scoutheeten, G.; and Gallinari, P. 2021. Let's Stop Incorrect Comparisons in End-to-end Relation Extraction! arXiv:2009.10684.

Vaswani, A.; Shazeer, N. M.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *NIPS*.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer Networks. In *Advances in Neural Information Processing Systems*.

Wadden, D.; Wennberg, U.; Luan, Y.; and Hajishirzi, H. 2019. Entity, Relation, and Event Extraction with Contextualized Span Representations. In *EMNLP 2019*.

Walker, C.; Strassel, S.; Medero, J.; and Maeda, K. 2006. ACE 2005 Multilingual Training Corpus.

Wan, Z.; Cheng, F.; Mao, Z.; Liu, Q.; Song, H.; Li, J.; and Kurohashi, S. 2023. GPT-RE: In-context Learning for Relation Extraction using Large Language Models. In *EMNLP 2023*.

Wang, J.; and Lu, W. 2020. Two are Better than One: Joint Entity and Relation Extraction with Table-Sequence Encoders. In *EMNLP 2020*.

Wang, Y.; Sun, C.; Wu, Y.; Zhou, H.; Li, L.; and Yan, J. 2021. UniRE: A Unified Label Space for Entity Relation Extraction. In *ACL 2021*.

Willard, B. T.; and Louf, R. 2023. Efficient Guided Generation for Large Language Models. arXiv:2307.09702.

Wu, Y.; Bamman, D.; and Russell, S. J. 2017. Adversarial Training for Relation Extraction. In *EMNLP 2017*.

Xu, D.; Chen, W.; Peng, W.; Zhang, C.; Xu, T.; Zhao, X.; Wu, X.; Zheng, Y.; and Chen, E. 2023. Large Language Models for Generative Information Extraction: A Survey. arXiv:2312.17617.

Xuewen, S.; Heyan, H.; Ping, J.; and Yi-Kun, T. 2021. Reducing Length Bias in Scoring Neural Machine Translation via a Causal Inference Method. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics*.

Yan, Z.; Zhang, C.; Fu, J.; Zhang, Q.; and Wei, Z. 2021. A Partition Filter Network for Joint Entity and Relation Extraction. In *EMNLP 2021*.

Ye, D.; Lin, Y.; Li, P.; and Sun, M. 2022a. 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)*.

Ye, H.; Zhang, N.; Chen, H.; and Chen, H. 2022b. Generative Knowledge Graph Construction: A Review. In *EMNLP 2022*.

Zaratiana, U.; Tomeh, N.; El Khbir, N.; Holat, P.; and Charnois, T. 2023. Filtered Semi-Markov CRF. In *Findings of EMNLP 2023*.

Zaratiana, U.; Tomeh, N.; Holat, P.; and Charnois, T. 2022a. GNNer: Reducing Overlapping in Span-based NER Using Graph Neural Networks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*.

Zaratiana, U.; Tomeh, N.; Holat, P.; and Charnois, T. 2022b. Named Entity Recognition as Structured Span Prediction. In *Proceedings of the Workshop on Unimodal and Multimodal Induction of Linguistic Structures (UM-IoS)*.

Zelenko, D.; Aone, C.; and Richardella, A. 2002. Kernel Methods for Relation Extraction. In *Journal of machine learning research*.

Zhao, T.; Yan, Z.; Cao, Y.; and Li, Z. 2021. A Unified Multi-Task Learning Framework for Joint Extraction of Entities and Relations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16): 14524–14531.

Zhong, Z.; and Chen, D. 2021. A Frustratingly Easy Approach for Entity and Relation Extraction. In *NAACL 2021*.