STAR: Improving Low-Resource Information Extraction by Structure-to-Text Data Generation with Large Language Models

Mingyu Derek Ma[†] **Xiaoxuan Wang**[†] **Po-Nien Kung**[†]
P. Jeffrey Brantingham[‡] **Nanyun Peng**[†] **Wei Wang**[†]
[†]Department of Computer Science [‡]Department of Anthropology
University of California, Los Angeles
{ma,xw27,ponienkung,violetpeng,weiwang}@cs.ucla.edu branting@ucla.edu

Abstract

Information extraction tasks such as event extraction require an in-depth understanding of the output structure and sub-task dependencies. They heavily rely on task-specific training data in the form of (passage, target structure) pairs to obtain reasonable performance. However, obtaining such data through human annotation is costly, leading to a pressing need for low-resource information extraction approaches that require minimal human labeling for real-world applications. Fine-tuning supervised models with synthesized training data would be a generalizable method, but the existing data generation methods either still rely on large-scale ground-truth data or cannot be applied to complicated IE tasks due to their poor performance. To address these challenges, we propose STAR, a data generation method that leverages Large Language Models (LLMs) to synthesize data instances given limited seed demonstrations, thereby boosting low-resource information extraction performance. Our approach involves generating target structures (Y) followed by generating passages (X), we further reduce errors and improve data quality through self-reflection error identification and self-refinement with iterative revision. Our experiments show that the data generated by STAR significantly improves the performance of low-resource event extraction and relation extraction tasks, even surpassing the effectiveness of human-curated data. Human assessment of the data quality shows STAR-generated data exhibits higher passage quality and better align with the task definitions compared with the human-curated data.

1 Introduction

Information extraction (IE) aims to extract knowledge of certain perspectives from natural language and consolidate it into an output structure [21]. To induce the target structure, the IE models need to understand fine-grained task requirements and constraints. Taking event extraction (EE), which is a component for IE systems to identify event triggers, event types, and their related details as arguments, as an example, task-specific rules include the predicted spans should be subsequences of the input passage, and arguments should be participants or attributes of the event. EE models are also expected to be aware of the dynamic skeleton of the event structure because the different predicted event types result in their respective sets of argument roles being filled in. Supervised models learn the implicit requirement and ontology knowledge from training data in the form of (passage, target structure) pairs. Prompt-based inference-only approaches with Large Language Models (LLMs) are shown to be unable to solve these complicated IE tasks [14, 4, 7]. In real-world applications, text from various sources and domains contains a broad range of output spaces and label definitions. It is costly and rigid to annotate sufficient training data, thus, performing IE given minimal seed data instances is of particular interest for realistic IE applications.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

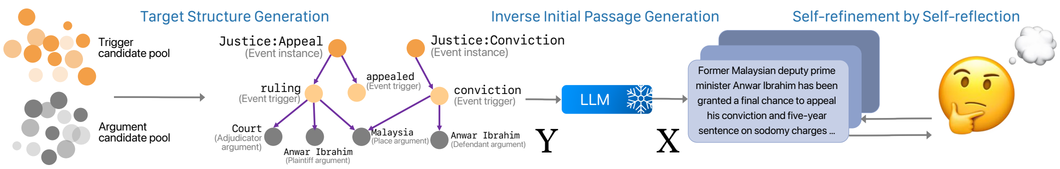

Figure 1: The STAR inverse data generation strategy using event extraction task as an example.

Synthesizing additional training data to fine-tune supervised models would be a generalizable method and has demonstrated its success for tasks like sentiment analysis [32] and relation extraction [10]. Some works annotate unlabeled examples with existing models as weak annotators ($X_{gold} \rightarrow Y$) [8, 2], produce analogous input ($X_{gold} \rightarrow X'$) [11, 12], or generate input assuming the labels are available ($Y_{gold} \rightarrow X$) [24, 5, 10]. However, these works require ground-truth X_{gold} or Y_{gold}, limiting their generalizability and scalability. What's more, they are designed for classification or straightforward IE tasks, and the performance drops significantly as the task complexity increases. Applying them to complicated IE tasks with sub-task dependencies and dynamic output structures such as EE produces noisy data that may impair task performance.

In this paper, we present STAR, a **S**tructure-to-**T**ext Dat**A** Gene**R**ation pipeline to produce dependable data instances for low-resource IE. Instead of using existing models to produce silver target structures Y derived from the input passages X (*i.e.* $X_{gold} \rightarrow Y$) to enrich the training data, we propose to generate data instances inversely by producing target structures from scratch first and then prompting the LLM to generate a passage (*i.e.* $Y \rightarrow X$) containing the target structure information. This inverse design reformulates the synthetic data generation task from structure induction, where models often struggle, to conditional text generation, where LLMs excel. STAR contains three components. First, STAR generates diverse target structures Y from scratch, requiring minimal human efforts to initiate the data generation. In addition, with the ability to customize target structures and control their distribution, we can mitigate data imbalance and improve data diversity by producing target structures encompassing a broader range of triggers, event types and arguments, as well as their various combinations. Second, STAR performs instruction-guided data generation to prompt the LLM about the fine-grained task definition and constraints to produce passage X. Third, STAR detects errors in the generated data instances via self-reflection and provides hindsight natural language intervention to self-refine the generated data without additional human efforts.

Experimental results on event extraction (EE) and relation extraction (RE) tasks show that STAR is capable of generating human-level IE data instances given a couple of exemplar instances as demonstrations without the need for additional ground-truth passage or target structures. For EE, training the supervised model on STAR-generated data improves the argument classification sub-task by up to 12.91 points in F1 score on the ACE05 dataset, 2.9 points higher than using the same amount of human-curated data. For RE, we observe 5.41 points F1 score improvement on the TACRED dataset, which is comparable to using the human-curated data. The improvements brought by STAR-generated data to multiple supervised models across multiple IE tasks demonstrate the generalizability and compatibility of STAR. Our manual data examination indicates STAR-produced data exhibits higher passage quality and better aligns with the task definitions compared with human-curated data.

We further conduct a detailed analysis of different methods for using LLMs to improve EE performance, and we show that training the supervised models on STAR-generated data yields at least 27.51 points higher F1 score for argument classification over the best inference-only LLM formulation.

2 STAR: Structure-to-Text Data Generation

We introduce the method design of STAR as illustrated in Figure 1. We use event extraction (EE) as the exemplar IE task in this section as it covers more method details. The goal is to create N new data instances (X, Y) based on k demonstration instances to be used as additional training resources for supervised IE models. Each data instance is composed of a natural language **passage** X containing event information and a **target structure** Y containing 0 to any number of *events*, each contains an event trigger, its event type and 0 to any number of (argument mention, argument role) pairs.

2.1 Target Structure Generation

The output distribution of the dataset largely determines the generalizability and robustness of the model fine-tuned on it. We first generate a pool of valid seed words for triggers of each event type, and for arguments of each (event type, argument role) combination. We then create target structures.

Trigger candidate generation. We prompt the LLM with 1) a definition of the selected event type; and 2) a few passages that contain event triggers of the selected event type as demonstrations. We use special tags to wrap the trigger word in the demonstration passages and prompt the LLM to continuously generate more passages with trigger words wrapped within tag pairs. Then we parse the response and extract the trigger candidate words.

Argument candidate generation. We find reasonable arguments for a certain pair of event type and argument role by prompting LLM with: 1) definition of the argument role under this specific event type; 2) the entity type we are looking for (*e.g.* a vehicle). The prompt would be like "*Given the definition of Instrument argument as 'The device used to inflict the harm', what are some possible vehicle names that can be used as Instrument?*" for INSTRUMENT argument of LIFE:INJURE event type. The allowed entity types of arguments are provided in an event ontology, and we merge generated word pools returned from separate queries if there are multiple allowed entity types for an argument role. For example, the ORIGIN argument of the MOVEMENT:TRANSPORT event type could be of entity types `GPE` (geopolitical entity), `LOC` (location), or `FAC` (facility name). We further parse the numbered/bullet lists generated by LLM to get argument candidate words.

Creating target structure and distribution control. We randomly sample trigger and argument candidates to create target structure Y. However, unbalanced label distribution and the prevalence of dominant labels pose challenges in many existing human-curated datasets [36, 1, 35]. In EE, a single dominant trigger word eclipses other relevant terms, leading to an unbalanced representation [27]. We address the issues of imbalanced event type and trigger distribution by evenly generating data instances and significantly expanding the pool size for trigger candidates to 100, which is 1.4× to 50× larger for various event types compared to the human-curated ACE05 dataset. Additionally, we balance the **argument hallucination ratio** in the generated data. This involves ensuring that the generated dataset contains events with both many arguments and few arguments by uniformly replacing argument value with `None` across different argument hallucination ratios. Furthermore, we balance the **event density** in passage X by providing target structure Y with 0 to 5 events.

2.2 Instruction-Guided Passage Generation

We use task instruction from multiple task granularities to provide recipes for the LLM to generate passages containing structured event information. The instruction is appended to k in-context learning examples verbalized by our instance verbalizer. Finally, we provide the verbalized target structure information based on the target structure Y to prompt the LLM to generate the initial passage X_0.

Task-level instruction. We provide task-related instruction following the annotation guideline curated by experts to guide the human annotation process of the ACE05 dataset [3]. Specifically, we provide: 1) a definition of "event", "trigger", "participant arguments" and "attribute arguments"; 2) an overall task requirement that the goal is to generate a sentence containing the event trigger words and arguments; 3) hallucination clarification that instructs the model *not* to generate arguments of certain roles if we explicitly provide that "the argument is None"; 4) multiple event clarification that information from multiple events should be contained in a *single* passage.

Event type-level instruction. In this segment, we introduce meta-information provided by pre-defined event ontology for a specific event type, including the name and definition of the event type and *each* possible argument roles. We provide all possible argument roles instead of the ones with existing values to ensure the generated passage X does not contain hallucinated arguments that should not appear according to the output structure Y.

Instance-level verbalizer. We verbalize exemplar data instances and target structure Y into natural language sequences with three segments: 1) the number of events in the passage; 2) the content of the event target structure; 3) the passage X with tags wrapping triggers and arguments to explicit hint the LLM about the roles and positions of the keywords, *e.g.* "*<Plaintiff>He</Plaintiff> threatened to <Trigger>sue</Trigger> the company.*" could provide an explicit indicator to the LLM that "he" is served as a PLAINTIFF argument for the event triggered by "sue".

2.3 Self-refinement by Self-reflection

After the initial passage X_0 is generated, the self-refinement mechanism evaluates the quality and identifies potential errors and further improves X_0 through iterative updates [23, 29, 16]. In the t-th refinement iteration, we first identify the potential quality issues of X_{t-1} from a diverse set of quality dimensions (*e.g.* the passage contains CRIME argument information, but it should be "None" according to Y), then the issues are **feedback** to the LLM by providing a template-based natural language intervention (*e.g.* "*The passage contains a hallucinated argument* CRIME *incorrectly, remove* CRIME *information for event triggered by 'jailed'*.") along with the generated passage of the previous iteration X_{t-1}, so that the LLM could **refine** the passage and produce X_t.

We define a set of quality dimensions and their intervention template manually. For EE, they include 1) whether the trigger/argument mention is a subsequence of the passage; 2) whether a trigger is used to initiate an occurrence; 3) whether an argument is used as an event participant or attribute of the specific event; 4) whether the argument is serving the required argument role; 5) whether the passage contains information that could be served as an argument that should not appear; 6) whether POS tags of the argument mentions in the passage context match the provided ones.

For each quality dimension, we query LLM with questions like "*Is 'Syria' a* DESTINATION *argument describing the event triggered by 'flee'?*". We then standardize LLM's response to a binary error identification flag by checking whether the response entails a confirmative phrase "Yes, it is." with a Natural Language Inference model fine-tuned on MultiNLI [30] based on BART-large [13]. If a quality issue is flagged, we use the intervention template corresponding to the selected quality dimension as part of the feedback to the LLM for iterative revision. Such a self-reflection design makes the self-refinement process generalizable and robust since the entire error identification process through self-reflection and the revision process are done by the LLM itself without external add-on components.

2.4 Adaptation to Relation Extraction

RE's relation type would be the equivalent concept of "event type" in EE. For target structure generation, we generate entity candidates using seed data instances' entities as in-context examples. We then randomly pair entity candidates and assign a relation between the two entities. For initial passage generation, we use relation type definition instead. For self-refinement, we use the quality dimensions: 1) whether the given entities are contained in the generated passage, 2) whether there is a relation between them, and 3) whether they hold the certain relation provided in Y.

3 Experiments on Event Extraction

3.1 Baselines

We use two types of models as our baselines: the *inference-only methods*, and the *supervised models* fine-tuned on data created by various *data creation strategies*.

Inference-only EE methods. We use LLM GPT-3.5 [25] and GPT-4 [26] to perform inference. We adopt different EE input-target formulations to prompt LLMs, including formulations inspired by generative supervised models (1-3) and LLM prompting methods specifically designed for EE proposed by recent works (4-6). The formulations include: 1) **Examples & IO (Text2Event)** [20] uses a concise but *unnatural* template to represent event structure. 2) **Examples & IO (DEGREE)** [9] generates a filled-in *natural* language template. 3) **Examples & IO (DICE)** [22] is similar to DEGREE but uses separate queries for different argument roles. 4) **Task Instruction** [14] provides task description and pre-defined event type *names*. 5) **Instruction+Examples** [4] provides event type *definitions* and positive and negative examples, in addition to the task description. 6) **Code4Struct** [28] formulates task definition, event type definition and examples in Python code. For baselines 1-3, we follow the original input and target formulations and additionally provide k demonstration input-target pairs contained in the input prompt for in-context learning. Baselines 4 and 5 only support Tri-I and Tri-C, and baseline 6 only supports Arg-I and Arg-C.

Supervised EE models. We use two representative EE models as the testbed to evaluate the quality of the generated data. **OneIE** [15] is a multi-task sequence-tagging model trained with global features based on RoBERTa-large [17]. **DEGREE** [9] is a prompt-based model that fills in event type-specific human written templates based on a BART-large pre-trained model [13].

Data creation strategies. We introduce two other approaches to obtain training data. **Weakly Supervision**: we use the best inference-only model for EE to predict event structure Y' from passage X, and the (X, Y') pairs are used as training data. **Human**: data instances randomly sampled from the ACE05 dataset, which requires much more human efforts, as an ideal but unrealistic setting.

3.2 Experimental Setup

We denote k as the number of demonstration examples of each event type used as in-context demonstrations for the inference-only methods and data creation strategies, and N as the number of data instances per event type created by data creation strategies. Supervised models are trained on $k + N$ data instances per event type. We use the full test set for evaluation. We use the event ontology and data instances from the widely used sentence-level English event extraction dataset ACE05 [3].

We follow previous EE works [15] and report F1 scores for four tasks. 1) Trigger Identification: identified trigger span is correct. 2) Trigger Classification: its predicted *event type* is also correct. 3) Argument Identification: identified argument span is correct. 4) Argument Classification: its

Table 1: Event extraction performance (F1, %). *Inference-only methods* and *data creation methods* use the same set of k examples for each event type to prompt LLM to perform EE and generate data instances respectively. *Supervised EE models* are trained on $k + N$ data instances per event type. **Boldface** indicates the best performance among each group without additional human efforts. Gray background indicate using human-curated data, thus it is not comparable with other lines. Green background indicate STAR-generated data improve EE performance more than human-curated data.

#			$k=0$	5	10	$k=0$	5	10	$k=0$	5	10	$k=0$	5	10
			Trigger Iden.			Trigger Clas.			Argument Iden.			Argument Clas.		
			Inference-only Methods											
	LLM	*Formulation*												
1		E&IO (Text2Event)	0.00	9.23	11.30	0.00	2.12	3.47	0.00	0.87	1.03	0.00	0.31	0.44
2		E&IO (DEGREE)	0.00	14.39	17.52	0.00	3.17	6.21	0.00	1.02	2.47	0.00	0.92	1.98
3	GPT-3.5	E&IO (DICE)	0.00	15.13	16.94	0.00	4.11	7.09	0.00	0.71	1.65	0.00	0.33	0.97
4		Task Inst.[§]	18.31	18.31	18.31	8.37	8.37	8.37		—			—	
5		Inst.+Examples	29.44	47.24	59.71	21.56	40.57	53.29		—			—	
6		Code4Struct		—			—		12.33	18.34	23.74	9.72	14.85	19.10
7	GPT-4	Inst.+Examples	**34.31**	**52.55**	**62.12**	**27.35**	**46.57**	**56.46**		—			—	
8		Code4Struct		—			—		**17.51**	**24.50**	**27.62**	**11.89**	**24.28**	**25.48**
			Supervised Models (N = 50 except line 9 & 14)											
	EE Model	*Data Creation*												
9		None (N = 0)	0.00	57.24	60.55	0.00	52.38	54.84	0.00	29.06	36.45	0.00	25.85	33.56
10		Weak Sup.	29.48	49.23	51.66	23.61	45.02	45.23	16.19	24.35	26.84	10.47	19.14	22.94
11	OneIE	STAR (GPT-3.5)	42.61	63.08	64.12	36.65	56.61	57.29	30.32	39.76	43.40	24.36	36.17	40.93
12		STAR (GPT-4)	**45.42**	**64.63**	**66.77**	**39.15**	**58.84**	**60.76**	**32.23**	**42.76**	**46.22**	**27.47**	**39.53**	**43.25**
13		Human[†§]	65.62	65.62	65.62	60.10	60.10	60.10	44.76	44.76	44.76	41.60	41.60	41.60
14		None (N = 0)	0.00	55.62	57.65	0.00	50.69	52.49	0.00	31.77	42.29	0.00	30.19	40.08
15		Weak Sup.	27.51	46.48	49.70	22.23	41.65	43.55	18.14	32.53	33.33	13.45	27.38	30.01
16	DEGREE	STAR (GPT-3.5)	43.74	61.39	63.57	38.90	56.41	59.10	32.32	48.73	53.06	28.21	46.55	50.97
17		STAR (GPT-4)	**46.69**	**64.47**	**65.17**	**41.75**	**59.92**	**61.42**	**35.85**	**51.92**	**54.56**	**32.09**	**50.74**	**52.99**
18		Human[†§]	63.49	63.49	63.49	58.86	58.86	58.86	52.47	52.47	52.47	50.09	50.09	50.09

predicted *argument role* is also correct. Note that each task is dependent on the output of the previous task. We report the medium result for three runs of different random seeds.

3.3 Effectiveness of Data Generation

Table 1 shows the EE performance when using various numbers (k) of demonstrations and Figure 2 further shows the effects of various amounts (N) of augmented data instances. We show qualitative analysis, common error cases and more experimental results in Appendix **??**.

STAR boosts low-resource EE performance. Table 1 shows that data generated by STAR significantly improve the supervised models' performance (line 9 vs 11-12, and line 14 vs 16-17) across all tasks. The F1 scores of Arg-C are improved by 9.69 and 12.91 for OneIE and DEGREE when k = 10.

Data produced by STAR is more effective than human-curated ones given sufficient examples. Compared to human-curated data instances (line 13 and 18 in Table 1), training supervised models with STAR-generated data leads to better performance when 5 or more demonstrations are used for STAR and the supervised model is DEGREE except for Arg-I (indicated by underlined results in line 17). We also observe a similar trend for OneIE when 10 demonstrations are used for STAR (underlined results in line 12). This indicates we could boost low-resource EE performance as if we had additional ground-truth data without paying the human annotation efforts. Figure 2 further shows the superiority of STAR-generated data over human-curated ones regardless of the number of augmented data instances (N) using 10 demonstrations (k = 10) for all supervised models.

3.4 Ablation Studies

Structure generation method. The diversity and balanced distribution of the generated target structures produced by our target structure generation component (§2.1) result in an almost 4-point higher Arg-C F1 score compared to using human-annotated target structures sampled from the ACE05.

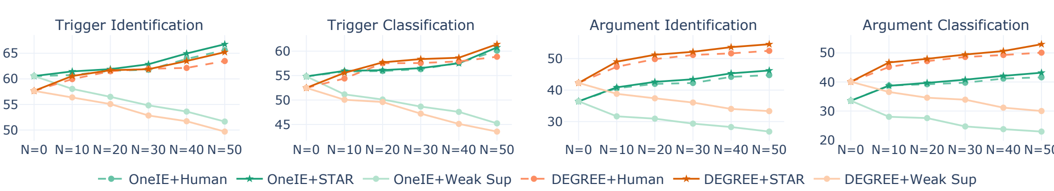

Figure 2: Event extraction performance (F1, %) when the EE models are trained on N augmented training data on top of 10 data points (k = 10) for each event type.

Error identification strategy. We also investigate the error identification capabilities of our self-reflection module with LLM as the backbone (§2.3). We compare it with two alternative methods and we utilize the same template to provide feedback on the identified errors. **Rule-based checking** uses heuristics to check whether a trigger/argument is a subsequence of the generated passage and uses an external NER module [31] to check whether a trigger/argument functions as the desired entity type. **Self-reflection (NLI)** uses the generated passage as the premier and a statement of a quality dimension as the hypothesis. We use entailment prediction of the NLI module used in §2.3 to identify whether a certain quality issue exists. The results are in lines 4-6 of Table 2. Our observations demonstrate both alternative methods help (line 3 vs 4-5), and self-reflection with LLM exhibits the highest effectiveness in error identification (line 6). Notably, the results underscore the effectiveness of the self-reflection design, resulting in a substantial 6-point increase in the F1 score for Arg-C without the need for additional annotation efforts.

Table 2: Ablation study on DEGREE's EE results while k = 10 and N = 10.

#	Method Variant	Tri-I	Tri-C	Arg-I	Arg-C
	Target Structure Y Generation Methods				
1	Ground-truth $Y^{\dagger}$	59.21	54.03	44.13	41.77
2	LLM generation	60.52	54.80	48.00	45.73
	Error Identification Strategies				
3	None	58.33	52.94	42.18	39.85
4	Rule-based checking	59.42	53.37	43.77	41.26
5	Self-reflection (NLI)	59.89	54.02	46.11	43.54
6	Self-reflection (LLM)	60.52	54.80	48.00	45.73

4 Experiments on Relation Extraction

To assess the generalizability of our proposed method, we conduct experiments on the sentence-level relation extraction (RE) task. We use relation definitions and seed examples in the widely-used TACRED dataset [34]. In this task, we generate (subject, relation, object) tuples from scratch, providing additional training data. The RE task aims to identify the relation between the given subject and object entities within a context passage. We train two representative RE models on the generated data instances. **SURE** [18] converts the task into a summarization formulation to leverage the indirect supervision with PEGASUS-large as the pre-trained encoder [33]. **GenPT** [6] transforms RE into an infilling problem with a RoBERTa-large model as backbone [17]. We use the same set of data creation baselines as in the EE experiments. We report micro F1 score across all relations (except for the "no relation" class) following prior works [18, 19].Table 3 presents the RE performance. The STAR-generated data significantly enhance the performance across the board compared to N = 0, with improvements of 5.4 and 3.6 F1 points when using SURE and GenPT respectively.

Table 3: Relation extraction performance (%) given 10 seed data instances k = 10.

#	RE Model	Data Gen	N = 0	10	40
1	GPT-3.5	—	27.91	27.91	27.91
2	SURE	Weak Sup.	27.61	28.02	28.32
3		STAR (GPT-3.5)		**30.50**	**33.02**
4		Human†		30.11	35.62
5	GenPT	Weak Sup.	33.38	30.93	30.29
6		STAR (GPT-3.5)		**34.55**	**37.01**
7		Human†		36.74	37.61

5 Quality Verification

Two annotators who are familiar with the EE task manually assess the quality of the EE data generated by STAR and curated by humans sampled from the ACE05 dataset for data instances in 100 sentences. Table 4 shows both sets of data demonstrate high satisfactory levels. STAR-generated data exhibits higher passage quality and better follows the task definition for most metrics, suggesting STAR produces EE annotations with comparable or even better quality than human annotators.

Table 4: Human assessment satisfactory rate (%).

Quality Dimension	STAR	Human
Grammaticality of X	**96**	90
Informativeness of X	**79**	78
Commonsense of X	**95**	93
Trigger span describes event occurrence	**99**	**99**
Event follows event type definition	**99**	97
Argument span describes an event	**100**	99
Argument associated with correct trigger	**98**	95
Argument follows role definition	98	**99**

6 Conclusion

We present STAR, an inverse data generation pipeline designed for low-resource IE that generates complicated output structure first and then curates input passage containing structure content, all with LLMs. STAR also contains self-refinement capabilities to fix self-identified error cases. Experimental results on EE and RE show that the generated data instances could significantly improve the performance and they are even more effective than human-curated data.

References

[1] Y. Cao, Y. Pruksachatkun, K.-W. Chang, R. Gupta, V. Kumar, J. Dhamala, and A. Galstyan. On the intrinsic and extrinsic fairness evaluation metrics for contextualized language representations. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 561–570, Dublin, Ireland, May 2022. Association for Computational Linguistics. URL `https://aclanthology.org/2022.acl-short.62`.

[2] Y. K. Chia, L. Bing, S. Poria, and L. Si. RelationPrompt: Leveraging Prompts to Generate Synthetic Data for Zero-Shot Relation Triplet Extraction. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 45–57, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.5. URL `https://aclanthology.org/2022.findings-acl.5`.

[3] G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, May 2004. European Language Resources Association (ELRA). URL `http://www.lrec-conf.org/proceedings/lrec2004/pdf/5.pdf`.

[4] J. Gao, H. Zhao, C. Yu, and R. Xu. Exploring the Feasibility of ChatGPT for Event Extraction, Mar. 2023. URL `http://arxiv.org/abs/2303.03836`.

[5] T. Gao, A. Fisch, and D. Chen. Making Pre-trained Language Models Better Few-shot Learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.295. URL `https://aclanthology.org/2021.acl-long.295`.

[6] J. Han, S. Zhao, B. Cheng, S. Ma, and W. Lu. Generative prompt tuning for relation classification. *arXiv preprint arXiv:2210.12435*, 2022.

[7] R. Han, T. Peng, C. Yang, B. Wang, L. Liu, and X. Wan. Is Information Extraction Solved by ChatGPT? An Analysis of Performance, Evaluation Criteria, Robustness and Errors, May 2023. URL `http://arxiv.org/abs/2305.14450`.

[8] X. He, I. Nassar, J. R. Kiros, G. Haffari, and M. Norouzi. Generate, Annotate, and Learn: Generative Models Advance Self-Training and Knowledge Distillation. Oct. 2021. URL `https://openreview.net/forum?id=oC12z8lkbrU`.

[9] I.-H. Hsu, K.-H. Huang, E. Boschee, S. Miller, P. Natarajan, K.-W. Chang, and N. Peng. DEGREE: A Data-Efficient Generation-Based Event Extraction Model. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1890–1908, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.138. URL `https://aclanthology.org/2022.naacl-main.138`.

[10] M. Josifoski, M. Sakota, M. Peyrard, and R. West. Exploiting Asymmetry for Synthetic Training Data Generation: SynthIE and the Case of Information Extraction, Mar. 2023. URL `http://arxiv.org/abs/2303.04132`.

[11] V. Kumar, A. Choudhary, and E. Cho. Data Augmentation using Pre-trained Transformer Models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26, Suzhou, China, Dec. 2020. Association for Computational Linguistics. URL `https://aclanthology.org/2020.lifelongnlp-1.3`.

[12] K. Lee, K. Guu, L. He, T. Dozat, and H. W. Chung. Neural data augmentation via example extrapolation. *arXiv preprint 2102.01335*, 2021.

[13] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. 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, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL `https://aclanthology.org/2020.acl-main.703`.

[14] B. Li, G. Fang, Y. Yang, Q. Wang, W. Ye, W. Zhao, and S. Zhang. Evaluating ChatGPT's Information Extraction Capabilities: An Assessment of Performance, Explainability, Calibration, and Faithfulness, Apr. 2023. URL `http://arxiv.org/abs/2304.11633`.

[15] Y. Lin, H. Ji, F. Huang, and L. Wu. A Joint Neural Model for Information Extraction with Global Features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.713. URL `https://aclanthology.org/2020.acl-main.713`.

[16] H. Liu, C. Sferrazza, and P. Abbeel. Chain of Hindsight Aligns Language Models with Feedback, Mar. 2023. URL `http://arxiv.org/abs/2302.02676`.

[17] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach, July 2019. URL `http://arxiv.org/abs/1907.11692`.

[18] K. Lu, I.-H. Hsu, W. Zhou, M. D. Ma, M. Chen, et al. Summarization as indirect supervision for relation extraction. In *EMNLP - Findings*, 2022.

[19] K. Lu, I.-H. Hsu, W. Zhou, M. D. Ma, and M. Chen. Multi-hop evidence retrieval for cross-document relation extraction. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10336–10351, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.657. URL `https://aclanthology.org/2023.findings-acl.657`.

[20] Y. Lu, H. Lin, J. Xu, X. Han, J. Tang, A. Li, L. Sun, M. Liao, and S. Chen. Text2Event: Controllable Sequence-to-Structure Generation for End-to-end Event Extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2795–2806, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.217. URL `https://aclanthology.org/2021.acl-long.217`.

[21] M. D. Ma, J. Sun, M. Yang, K.-H. Huang, N. Wen, S. Singh, R. Han, and N. Peng. EventPlus: A Temporal Event Understanding Pipeline. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages 56–65, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-demos.7. URL `https://aclanthology.org/2021.naacl-demos.7`.

[22] M. D. Ma, A. K. Taylor, W. Wang, and N. Peng. DICE: Data-Efficient Clinical Event Extraction with Generative Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. arXiv, Aug. 2022. URL `http://arxiv.org/abs/2208.07989`.

[23] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Welleck, B. P. Majumder, S. Gupta, A. Yazdanbakhsh, and P. Clark. Self-Refine: Iterative Refinement with Self-Feedback, Mar. 2023. URL `http://arxiv.org/abs/2303.17651`.

[24] Y. Meng, J. Huang, Y. Zhang, and J. Han. Generating Training Data with Language Models: Towards Zero-Shot Language Understanding, Oct. 2022. URL `http://arxiv.org/abs/2202.04538`.

[25] OpenAI. Chatgpt: Optimizing language models for dialogue. `https://openai.com/blog/chatgpt/.`, 2022.

[26] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[27] M. Tong, B. Xu, S. Wang, M. Han, Y. Cao, J. Zhu, S. Chen, L. Hou, and J. Li. DocEE: A Large-Scale and Fine-grained Benchmark for Document-level Event Extraction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3970–3982, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.291. URL https://aclanthology.org/2022.naacl-main.291.

[28] X. Wang, S. Li, and H. Ji. Code4Struct: Code Generation for Few-Shot Event Structure Prediction, May 2023. URL http://arxiv.org/abs/2210.12810.

[29] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi. Self-Instruct: Aligning Language Model with Self Generated Instructions, Dec. 2022. URL http://arxiv.org/abs/2212.10560.

[30] A. Williams, N. Nangia, and S. Bowman. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL https://aclanthology.org/N18-1101.

[31] I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Matsumoto. LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.523. URL https://aclanthology.org/2020.emnlp-main.523.

[32] J. Ye, J. Gao, Q. Li, H. Xu, J. Feng, Z. Wu, T. Yu, and L. Kong. ZeroGen: Efficient Zero-shot Learning via Dataset Generation, Oct. 2022. URL http://arxiv.org/abs/2202.07922.

[33] J. Zhang, Y. Zhao, M. Saleh, and P. Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR, 2020.

[34] Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning. Position-aware Attention and Supervised Data Improve Slot Filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1004. URL https://aclanthology.org/D17-1004.

[35] J. Zhao, T. Wang, M. Yatskar, V. Ordonez, and K.-W. Chang. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2003. URL https://aclanthology.org/N18-2003.

[36] Y. Zhou, M. Kaneko, and D. Bollegala. Sense embeddings are also biased – evaluating social biases in static and contextualised sense embeddings. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1924–1935, Dublin, Ireland, May 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.acl-long.135.