LLMs Underperform Graph-Based Parsers on Supervised Relation Extraction for Complex Graphs

Paolo Gajo
University of Bologna
paolo.gajo2@unibo.it

Domenic Rosati
Dalhousie University
Domenic.Rosati@Dal.Ca

Hassan Sajjad
Dalhousie University
HSajjad@dal.ca

Alberto Barrón-Cedeño
University of Bologna
a.barron@unibo.it

Abstract

Relation extraction represents a fundamental component in the process of creating knowledge graphs, among other applications. Large language models (LLMs) have been adopted as a promising tool for relation extraction, both in supervised and in-context learning settings. However, in this work we show that their performance still lags behind much smaller architectures when the linguistic graph underlying a text has great complexity. To demonstrate this, we evaluate four LLMs against a graph-based parser on six relation extraction datasets with sentence graphs of varying sizes and complexities. Our results show that the graph-based parser increasingly outperforms the LLMs, as the number of relations in the input documents increases. This makes the much lighter graph-based parser a superior choice in the presence of complex linguistic graphs.

1 Introduction

Relation extraction (RE) is a core NLP task which entails extracting `[head, relation, dependent]` RDF triples (Candan et al., 2001) from text (Zhao et al., 2024). In supervised settings, RE can be approached with a variety of models, with one of the most popular paradigms being graph-based parsers (Verga et al., 2018; Tang et al., 2022). Although recent literature has explored the capacity of autoregressive LLMs to carry out RE through in-context learning (Wan et al., 2023; Bi et al., 2024; Wei et al., 2024) and supervision via causal language modeling (Zhang et al., 2024; Papaluca et al., 2024; Gajo and Barrón-Cedeño, 2025), to the best of our knowledge no direct comparison has yet been made between them and graph-based parsers in a supervised setting.

In this paper, we assess the effectiveness of LLMs on the RE task with respect to the complexity of the relations comprising the training and testing documents. To this end, we compare the sentence-level and document-level RE performance of four LLMs (Jiang et al., 2023; Dubey et al., 2024; Yang et al., 2025) against that of varying configurations of a graph-based parser (Dozat and Manning, 2017; Ji et al., 2019; Bhatt et al., 2024). We train and evaluate both architectures on six datasets for RE and dependency parsing, since both tasks entail the inference of nodes, edges, and edge labels (Veličković et al., 2020; Kazi et al., 2023; Lu et al., 2023) over a linguistic graph (i.e. a text). We also evaluate the LLMs prior to any fine-tuning as a baseline. The datasets vary widely in the graph sizes of each document, ranging from just a couple to over 100 nodes and relations linking them. This helps us verify how LLMs are impacted by graph complexity compared to graph-based parsers.

Our results show that the LLMs performs better than the graph-based parser when graph complexity is trivial, but greatly worsens when complex relations are involved. In particular, the graph-based parsers, despite their much smaller parameter sizes, consistently outperform the much larger LLMs on texts underlying graphs that on average have more than ∼18 edges. Thus, the contribution of this work is shedding light on the limited capacity of LLMs to extract relations from complex linguistic graphs.

2 Related work

The state of the art in RE entails using deep neural networks (NNs) to find connections between entities in a text via some measure of similarity between them (Zhao et al., 2024). Graph-based parsers do this by embedding each token of a sentence and processing them with recurrent or graph NNs so that the individual embeddings share information based on the structure of the sentence (Dozat and Manning, 2017, 2018; Ji et al., 2019; Donatelli et al., 2021; Bhatt et al., 2024; Tang et al., 2022). In this regard, many systems use attention to score the relations, akin to Dozat and

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 181–193

Manning (2017), whose parser we use in this work.

Another paradigm is represented by sequence-to-sequence models, which are trained to output predictions directly in natural language, formatted in a way that allows for the automatic extraction of the predictions (Liu et al., 2022; Lu et al., 2022; Paolini et al., 2021; Zaratiana et al., 2024).

Similarly, LLMs have recently been shown to be able to extract relations from texts even without being fine-tuned specifically on the given task, just by providing extraction schema and ICL examples (Wan et al., 2023; Wei et al., 2024; Zhang and Soh, 2024; Bi et al., 2024; Dong et al., 2024), or by fine-tuning them (Zhang et al., 2024; Papaluca et al., 2024; Gajo and Barrón-Cedeño, 2025). However, no comparison has yet been drawn between graph-based parsers and LLMs in a supervised setting. We attempt to fill this gap in the literature.

3 Data

We train and evaluate our models on six datasets comprising texts annotated with entity classes, relations, and relation classes. Table 1 reports the statistics of the number k of relations for each dataset.

CoNLL04 (Roth and Yih, 2004) comprises short news texts, each annotated with the entity classes $e_i \in \{\texttt{per}, \texttt{org}, \texttt{loc}\}$ and relation classes $r_j \in \{\texttt{workFor}, \texttt{kill}, \texttt{orgBasedIn}, \texttt{liveIn}, \texttt{locIn}\}$, with $i \in \{0, \ldots, N\}$, N the number of words in the sample of a given dataset, $j \in \{0, ..., k\}$, and k the number of relations in the sample.[1] **ADE** (Gurulingappa et al., 2012) is made up by reports of drug adverse-effect reactions. Entities can be classified as $e_i \in \{\texttt{drug}, \texttt{disease}\}$, while the only relation between them can be $r_j \in \{\texttt{adverseEffect}\}$. **SciERC** (Luan et al., 2018) is compiled from scientific literature. Despite its small size, the domain is specialized, making it challenging. In addition, the validation and testing partition entities are not labeled, meaning they cannot be used to help infer relations. As Table 1 shows, in CoNLL04, ADE, and SciERC the vast majority of samples have $k \leq 5$ relations, meaning they mainly comprise small graphs.

More complex graphs are contained in **enEWT** (Nivre et al., 2018), built from the English Web Treebank, and **SciDTB** (Yang and Li, 2018), which comprises 798 abstracts from the ACL Anthology.[2] We use the xPOS tags as entity labels and the type of dependency as relation classes.[3]

ERFGC (Yamakata et al., 2020), a dataset of culinary recipes annotated as *flow graphs*, comprises the most complex graphs. Here, the entities and relations define directed acyclic graphs with a single sink. As indicated by the authors, we ignore the "-" relation annotations.

4 Model

For the graph-based parser, we adopt and extend the architecture used by Bhatt et al. (2024), which leverages Dozat and Manning (2017)'s biaffine attention parser. Architecture details for the graph-based parser are available in Appendix B.

We compare this lightweight parser (124M parameters)[4] to LLMs ranging from 7B to 70B parameters: *Mistral-7B-Instruct-v0.3* (Jiang et al., 2023), *Qwen3-14B-Base* and *Qwen3-32B* (Yang et al., 2025), and *Llama-3.3-70B-Instruct* (Dubey et al., 2024). *Qwen3-14B-Base* has not undergone instruction tuning, meaning it cannot be used in our experiments prior to fine-tuning.

The graph-based parser is trained with a learning rate of $\eta_1 = 10^{-3}$ and a batch size of 8. The LLMs are fine-tuned with LoRA (Hu et al., 2021), with $r = a = 16$. Following Gajo and Barrón-Cedeño (2025), we only target the key, query, and value weights of the attention layers, since they are found to be the most important modules when fine-tuning LLMs for relation extraction, in terms of the number of trained parameters. In this case, we use a learning rate of $\eta_2 = 2 \times 10^{-4}$, a batch size of 1, 5 warm-up steps, and apply a weight decay of 0.01. We use AdamW (Loshchilov and Hutter, 2019) as the optimizer for both models.

In our prompts, along with the task instruction, we either include just the entity/relation class names (NoDesc) or the entity/relation class names along with a description of each class (Desc). In addition, we include $N_{\text{icl}} \in \{0, 1\}$ examples, each sampled randomly from the training set. We use at most one ICL example because ERFGC contains very long texts, which causes out-of-memory errors with more examples. Figure 2 in Appendix C includes a prompt example with entity/relation descriptions and one ICL example. We also experiment with a prompt layout which only includes

[1] We omit the none class for clarity.

[2] We do not use the Penn Treebank (Prasad et al., 2008) due to its prohibitive monetary cost.

[3] Additional information on statistics and annotation for all six datasets can be found in Table 5 in Appendix A.

[4] The BERT backbone encoder has 110M parameters and remains frozen, while the fine-tuned parser on top of it has at most 14M parameters in the heaviest configuration.

Dataset	Min	Mean ($\bar{k}$)	Max	$k \leq 5$	Avg. chars.
CoNLL04	1	1.42	11	98.54 %	158.10
ADE	1	1.59	21	97.77 %	135.33
SciERC	1	2.38	17	93.49 %	159.44
enEWT	5	17.83	158	25.03 %	91.77
SciDTB	5	23.41	100	2.50 %	150.50
ERFGC	4	49.19	167	3.33 %	603.90

Table 1: Statistics of the number k of relations for the documents contained in the datasets, for all splits, along with mean document length in characters.

a UUID code, used to condition the model to recognize the downstream task without any useful instructions (Figure 3 in Appendix C). Finally, we adopt an adversarial prompt which asks the model not to carry out the task. This shows that, while the base model complies and does not extract the RDF triples, models fine-tuned with LoRA comply with the request (Figure 4 in Appendix C), meaning that prompt instructions are irrelevant in supervised fine-tuning settings.

We train the graph-based parser for $3k$ steps and evaluate every 500. To make the comparison fair compute-wise, we train the LLMs for a single epoch for each dataset and test directly on the testing partition. To gauge the impact of the amount of training steps, we also fine-tune *Qwen3-32B* and *Llama-3.3-70B-Instruct* for $3k$ steps. This increases the number of full-dataset passes for datasets with smaller training splits, such as CoNLL04 and ERFGC, which respectively comprise 922 and 242 training samples.

Due to computational constraints, we use fewer seeds for bigger models (Table 2). For the same reason, we evaluate *Qwen3-14B-Base* and *Qwen3-32B* on a subset of 100 samples for enEWT and SciDTB, while *Llama-3.3-70B-Instruct* is evaluated on 100 samples for all datasets.

As with similar works (Bhatt et al., 2024; Dozat and Manning, 2018; Jiang et al., 2024), we evaluate tagging and parsing performance in terms of micro-F_1. We use exact evaluation (Zhang et al., 2024), where a triple is considered correct if the entities and the relation match, irrespective of the tag. This makes the evaluation equivalent for the two models and takes into account that SciERC's testing partition contains almost no tag annotations.

5 Results and discussion

We report the results for the graph-based parser and the LLMs in Table 2.[5] The performance for *Mistral-7B-Instruct-v0.3* without any prior fine-tuning (Steps $= \times$) is very low, both with one or no examples in the prompt. When fine-tuning for one epoch, the performance of the model drastically increases. However, the model outperforms the much smaller graph-based parser only on ADE, one of the least complex datasets.

Qwen3-14B-Base obtains the best performance on ADE ($F_1 = 0.836$) and SciERC ($F_1 = 0.444$), despite not being the biggest model and the UUID prompt containing no helpful information whatsoever. *Qwen3-14B-Base* likely obtains the best results compared to the other LLMs because relation extraction can be simply handled as a prompt completion task, without requiring irrelevant chatbot induction bias from prior instruction tuning. Interestingly, the UUID prompt obtains the highest dataset-wise average performance (mean $F_1 = 0.692$ over four seeds and six datasets). More generally, there seems to be no relationship between prompt type and model performance when fine-tuning. Indeed, even in the case of the adversarial prompt, models start complying with the request even after just 10 training steps ($F_1 > 0$), while the base model refuses to comply ($F_1 = 0$).

Llama-3.3-70B-Instruct achieves the highest performance on CoNLL04, when training for $3k$ steps, but overall underperforms *Qwen3-14B-Base* on all other datasets except for ERFGC. Furthermore, training for $3k$ steps provides a non-negligible performance increase for this model on ERFGC (from 0.529 at one epoch to 0.606 at $3k$ steps), but overall the performance is similar for the other datasets. The reason for the lower performance could be a

[5] We only show the best results for the graph-based parser. The full results can be found in Appendix D.

Prompt	#ICL	Steps	CoNLL04	ADE	SciERC	enEWT	SciDTB	ERFGC
Mistral-7B-Instruct-v0.3 (5 seeds)								
NoDesc	0	×	0.115± 0.000	0.047± 0.000	0.021± 0.000	0.006± 0.000	0.001± 0.000	0.002± 0.000
NoDesc	1	×	0.123± 0.009	0.281± 0.031	0.039± 0.005	0.026± 0.000	0.032± 0.001	0.056± 0.013
NoDesc	0	1 ep.	0.597± 0.011	0.776± 0.016	0.320± 0.005	0.784± 0.000	0.793± 0.003	0.248± 0.017
NoDesc	1	1 ep.	0.613± 0.011	0.775± 0.007	0.346± 0.005	0.805± 0.001	0.830± 0.001	0.331± 0.013
Qwen3-14B-Base (4 seeds)								
UUID	0	1 ep.	0.634±0.013	**0.836**±0.020	**0.444**±0.031	0.831±0.008	0.883±0.002	0.521±0.015
NoDesc	0	1 ep.	0.653±0.004	0.827±0.013	0.398±0.005	0.841±0.005	0.885±0.002	0.509±0.007
NoDesc	1	1 ep.	0.625±0.012	0.790±0.011	0.413±0.015	0.839±0.005	0.878±0.004	0.509±0.008
Desc	0	1 ep.	0.640±0.008	0.817±0.004	0.405±0.010	0.838±0.005	0.886±0.001	0.513±0.007
Desc	1	1 ep.	0.640±0.010	0.810±0.010	0.403±0.004	0.840±0.002	0.881±0.003	0.505±0.008
Qwen3-32B (3 seeds)								
NoDesc	1	1 ep.	0.648± 0.010	0.790± 0.010	0.356± 0.010	0.834± 0.008	0.883± 0.004	0.514± 0.010
NoDesc	1	$3k$	0.673± 0.009	0.788± 0.016	0.389± 0.039	0.800± 0.009	0.879± 0.008	0.581± 0.016
Llama-3.3-70B-Instruct (1 seed @ 1 epoch / 5 seeds @ 3k steps)								
NoDesc	0	1 ep.	0.674	0.808	0.421	0.838	0.863	0.494
NoDesc	1	1 ep.	0.650	0.789	0.423	0.843	0.877	0.538
Desc	0	1 ep.	0.644	0.819	0.385	0.851	0.876	0.519
Desc	1	1 ep.	0.669	0.791	0.405	0.845	0.871	0.529
Desc	1	$3k$	**0.717** ±0.034	0.798 ±0.017	0.388 ±0.145	0.748 ±0.022	0.849 ±0.002	0.606 ±0.012
Graph-based parser (5 seeds)								
×	×	$3k$	0.668±0.024	0.697±0.022	0.351±0.033	**0.865**±0.004	**0.918**±0.003	**0.713**±0.007

Table 2: Micro-F_1 for the LLMs and graph-based parser. Best in bold, second-best underlined.

	CoNLL04	ADE	SciERC	enEWT	SciDTB	ERFGC
Qwen3-14B-Base	-0.143±0.067	-0.152±0.049	0.171±0.054	-0.270±0.025	-0.185±0.048	-0.639±0.072
Graph-based parser	0.012± 0.024	-0.046±0.045	0.067±0.029	-0.043±0.013	-0.195±0.012	-0.206±0.051

Table 3: Pearson r between edge number k and micro-F_1 for *Qwen3-14B-Base* and the graph-based parser.

combination of *Qwen3-14B-Base* being more recent and *Llama-3.3-70B-Instruct* requiring more training steps due to the higher parameter count. Nevertheless, despite training $3k$ steps means fine-tuning on ERFGC for multiple epochs, *Llama-3.3-70B-Instruct* still cannot match the performance of the graph-based parser on complex graphs. *Qwen3-32B* also performs worse than *Qwen3-14B-Base* on average, despite being bigger. This is likely because, through instruction-tuning, we are effectively removing the thinking-block generation behavior of the model.

Finally, the average performance achieved by *Llama-3.3-70B-Instruct* (mean $F_1 = 0.684$ over four settings and six datasets) is slightly lower than *Qwen3-14B-Base*'s best average.

Model	Mean σ	Max σ
Mistral-7B-Instruct-v0.3	0.006	0.031
Qwen3-14B-Base	0.008	0.031
Qwen3-32B	0.012	0.039

Table 4: Mean and maximum variance for the F_1 scores obtained by the LLMs evaluated on all test samples.

Mean and max F_1 variance are very low for all LLMs for which we were able to run multiple seeds, as shown in Table 4. This, combined with the low variance for Llama-3.3-70B-Instruct at $3k$ steps for all datasets but SciERC, provides evidence that the results are reliable even when evaluating on just a small subset of the testing set.

Overall, the much lighter graph-based parser out-

performs all LLMs on the complex graphs comprising enEWT, SciDTB, and ERFGC, while the LLMs match its performance on CoNLL04 and surpass it on ADE and SciERC. As Table 3 shows, the performance of the best model, *Qwen3-14B-Base* fine-tuned on UUID prompts, has a negative correlation for all datasets but SciERC, with ERFGC having a strong negative correlation. This is reflected in the delta between the best results for the LLMs and the graph-based parser increasing from 1.4 F_1 points on enEWT, to 3.2 F_1 points on SciDTB, and to 13.2 F_1 points on ERFGC. The performance delta on ERFGC is substantial, considering the gap in learnable parameters between the graph-based parser (14M) and *Qwen3-32B* (84M with LoRA) and the former being two orders of magnitude smaller overall (124M vs 32B total parameters).

Unlike *Qwen3-14B-Base*, the correlation for the graph-based parser is much lower across all datasets, due to the different prediction mechanism (Table 3). We hypothesize this is because LLMs require text formatting both in the prompt and during generation to extract relations, which injects noise into the causal attention and increases the distances between tokens relevant for predicting relations. This issue scales with graphs size, and becomes a drawback on large graphs, as observed for enEWT, SciDTB, and especially ERFGC. This also becomes a drawback in terms of inference speed, since LLMs require thousands of forward passes to extract a large graph, while a graph-based parser only needs one.

6 Conclusions

In this paper, we have compared a small graph-based parser against four different LLMs, focusing on how linguistic graph complexity affects their performance on the relation extraction (RE) task.

Our experiments, carried out over six datasets of varying complexity, showed that, while the LLMs are capable of handling a small number of relations, the graph-based parser starts to outperform them as the number of relations increases. This is reflected in a widening performance delta between LLMs and graph-based parsers, as the number of relations increases. We attribute this to LLMs requiring formatting for their predictions, which dilutes attention by injecting noise and increasing the distance between tokens relevant for predicting relations.

In future work, we aim to further investigate at the attention level why LLMs underperform graph-based parsers on RE. In this regard, we plan to experiment with different training paradigms that can help alleviate the highlighted issues, for example via prompt compression (Łajewska et al., 2025) or by simply reducing the amount of text formatting needed for effective training and inference.

7 Computational resources

Fine-tuning the LLMs took between 30 minutes to 18 hours on NVIDIA L40s and H100s, depending on the size of the dataset and model. LLM inference was much slower, requiring between 3 to 6 hours. For the graph-based parser, each training run took approximately 10 minutes on NVIDIA P100s and Tesla V100s.

8 Limitations

This paper effectively presents a negative result, studying the behavior of LLMs in the relation extraction task and showing how they underperform in comparison to graph-based parsers. Further research is warranted to ascertain whether the LLMs' shortcomings can be alleviated or overcome. In addition, a qualitative analysis could shed further light on the reasons behind the lower performance exhibited by the LLMs. Finally, we acknowledge that only one type of graph-based parser is used, and that experimenting with a wider variety could provide better insight on the relation extraction capacities of LLMs by comparison.

References

Dhaivat J. Bhatt, Seyed Ahmad Abdollahpouri Hosseini, Federico Fancellu, and Afsaneh Fazly. 2024. End-to-end Parsing of Procedural Text into Flow Graphs. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 5833–5842, Torino, Italia. ELRA and ICCL.

Zhen Bi, Jing Chen, Yinuo Jiang, Feiyu Xiong, Wei Guo, Huajun Chen, and Ningyu Zhang. 2024. CodeKGC: Code Language Model for Generative Knowledge Graph Construction. *arXiv preprint*. ArXiv:2304.09048 [cs].

Shaked Brody, Uri Alon, and Eran Yahav. 2022. How attentive are graph attention networks? In *International Conference on Learning Representations*.

K. Selçuk Candan, Huan Liu, and Reshma Suvarna. 2001. Resource description framework: metadata and its applications. *ACM SIGKDD Explorations Newsletter*, 3(1):6–19.

Lucia Donatelli, Theresa Schmidt, Debanjali Biswas, Arne Köhn, Fangzhou Zhai, and Alexander Koller. 2021. Aligning actions across recipe graphs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6930–6942, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A Survey on In-context Learning. *arXiv preprint*. ArXiv:2301.00234 [cs].

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *International Conference on Learning Representations*. arXiv. ArXiv:1611.01734 [cs].

Timothy Dozat and Christopher D. Manning. 2018. Simpler but More Accurate Semantic Dependency Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 514 others. 2024. The Llama 3 Herd of Models. *arXiv preprint*. ArXiv:2407.21783 [cs].

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics*, 71B(4):233.

Gajo and Barrón-Cedeño. 2025. Natural vs Programming Language in LLM Knowledge Graph Construction. *Information Processing & Management*, 62(5):104195.

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. *Journal of Biomedical Informatics*, 45(5):885–892. Text Mining and Natural Language Processing in Pharmacogenomics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv preprint*. ArXiv:2106.09685 [cs].

Tao Ji, Yuanbin Wu, and Man Lan. 2019. Graph-based Dependency Parsing with Graph Neural Networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2475–2485, Florence, Italy. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *arXiv preprint*. ArXiv:2310.06825 [cs].

Shu Jiang, Zuchao Li, Hai Zhao, and Weiping Ding. 2024. Entity-Relation Extraction as Full Shallow Semantic Dependency Parsing. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:1088–1099.

Anees Kazi, Luca Cosmo, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael M. Bronstein. 2023. Differentiable Graph Module (DGM) for Graph Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1606–1617. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

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*, pages 993–1005, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. *arXiv preprint*.

Jianglin Lu, Yi Xu, Huan Wang, Yue Bai, and Yun Fu. 2023. Latent graph inference with limited supervision. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5755–5772, Dublin, Ireland. Association for Computational Linguistics.

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*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, and Lene Antonsen. 2018. Universal dependencies 2.2. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, RISHITA ANUBHAI, Cicero Nogueira dos Santos, Bing Xiang, and Stefano

Soatto. 2021. Structured prediction as translation between augmented natural languages. In *International Conference on Learning Representations*.

Andrea Papaluca, Daniel Krefl, Sergio Rodríguez Méndez, Artem Lensky, and Hanna Suominen. 2024. Zero- and Few-Shots Knowledge Graph Triplet Extraction with Large Language Models. In *Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024)*, pages 12–23, Bangkok, Thailand. Association for Computational Linguistics.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

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) at HLT-NAACL 2004*, pages 1–8, Boston, Massachusetts, USA. Association for Computational Linguistics.

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*, pages 7087–7099, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Petar Veličković, Lars Buesing, Matthew Overlan, Razvan Pascanu, Oriol Vinyals, and Charles Blundell. 2020. Pointer Graph Networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 2232–2244. Curran Associates, Inc.

Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously Self-Attending to All Mentions for Full-Abstract Biological Relation Extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana. Association for Computational Linguistics.

Zhen Wan, Fei Cheng, Zhuoyuan Mao, Qianying Liu, Haiyue Song, Jiwei Li, and Sadao Kurohashi. 2023. GPT-RE: In-context learning for relation extraction using large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3534–3547, Singapore. Association for Computational Linguistics.

Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, Yong Jiang, and Wenjuan Han. 2024. ChatIE: Zero-Shot Information Extraction via Chatting with ChatGPT. *arXiv preprint*. ArXiv:2302.10205 [cs].

Yoko Yamakata, Shinsuke Mori, and John Carroll. 2020. English Recipe Flow Graph Corpus. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5187–5194, Marseille, France. European Language Resources Association.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 Technical Report. *arXiv preprint*. ArXiv:2505.09388 [cs].

An Yang and Sujian Li. 2018. SciDTB: Discourse dependency TreeBank for scientific abstracts. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 444–449, Melbourne, Australia. Association for Computational Linguistics.

Urchade Zaratiana, Nadi Tomeh, Pierre Holat, and Thierry Charnois. 2024. An autoregressive text-to-graph framework for joint entity and relation extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19477–19487. Issue: 17.

Bowen Zhang and Harold Soh. 2024. Extract, Define, Canonicalize: An LLM-based Framework for Knowledge Graph Construction. *arXiv preprint*. ArXiv:2404.03868 [cs].

Yujia Zhang, Tyler Sadler, Mohammad Reza Taesiri, Wenjie Xu, and Marek Reformat. 2024. Fine-tuning Language Models for Triple Extraction with Data Augmentation. In *Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024)*, pages 116–124, Bangkok, Thailand. Association for Computational Linguistics.

Xiaoyan Zhao, Yang Deng, Min Yang, Lingzhi Wang, Rui Zhang, Hong Cheng, Wai Lam, Ying Shen, and Ruifeng Xu. 2024. A Comprehensive Survey on Relation Extraction: Recent Advances and New Frontiers. *ACM Computing Surveys*, 56(11):1–39. Publisher: Association for Computing Machinery (ACM).

Weronika Łajewska, Momchil Hardalov, Laura Aina, Neha Anna John, Hang Su, and Lluís Màrquez. 2025. Understanding and Improving Information Preservation in Prompt Compression for LLMs. *arXiv preprint*. ArXiv:2503.19114 [cs].

A Dataset labels

Table 5 reports the split sizes and the entity/relation labels for each dataset.

B Graph-based model details

The graph-based model comprises four main components: encoder, tagger, parser, and decoder, schematized in Figure 1. The input is tokenized and passed through a BERT-like encoder, where token representations are averaged into $|\mathcal{V}|$ word-level features $\mathbf{x}_i \in \mathbb{R}^{d_f}$.[6] Optionally, additional features can be obtained by predicting the entity classes of each word with a tagger, composed by a single-layer BiLSTM ϕ and a classifier:

$$\mathbf{h}_i^{tag} = \phi(\mathbf{x}_i), \quad \mathbf{h}_i^{tag} \in \mathbb{R}^{d_h}$$
$$\mathbf{y}_i^{tag} = \text{Softmax}(\text{MLP}^{tag}(\mathbf{h}_i^{tag})), \quad \mathbf{y}_i^{tag} \in \mathbb{R}^{|T|}$$

where T is the set of word tag classes. The tagger's predictions are then converted into one-hot vectors and projected into dense representations by another MLP, such that $\mathbf{e}_i^{tag} = \text{MLP}^{emb}(\mathbf{1}_T(\mathbf{y}_i^{tag}))$. These new tag embeddings are concatenated with the original BERT output and sent to the parser.

In the parser, an optional N-layered BiLSTM ψ produces new representations $\mathbf{h}_i = \psi(\mathbf{e}_i^{tag} \oplus \mathbf{x}_i)$, which are then projected into four different representations:

$$\mathbf{e}_i^h = \text{MLP}^{(edge-head)}(\mathbf{h}_i), \; \mathbf{e}_i^d = \text{MLP}^{(edge-dept)}(\mathbf{h}_i)$$
$$\mathbf{r}_i^h = \text{MLP}^{(rel-dept)}(\mathbf{h}_i), \quad \mathbf{r}_i^d = \text{MLP}^{(rel-head)}(\mathbf{h}_i)$$

The edge scores s_i^{edge} and relation scores s_i^{rel} are then calculated with the biaffine function f:

$$f(\mathbf{x}_1, \mathbf{x}_2; W) = \mathbf{x}_1^\top W \mathbf{x}_2 + \mathbf{x}_1^\top \mathbf{b}$$

$$s_i^{edge} = f^{(edge)}(\mathbf{e}_i^h, \mathbf{e}_i^d; W_e), \quad W_e \in \mathbb{R}^{d \times 1 \times d}$$
$$s_i^{rel} = f^{(rel)}(\mathbf{r}_i^h, \mathbf{r}_i^d; W_r), \quad W_r \in \mathbb{R}^{d \times |R| \times d}$$

where k is the set of relation classes, i.e. the possible labels applied to an edge.

We also experiment with the addition of $L_{\text{GNN}} \in \{0, 1, 2, 3\}$ GNN layers upstream of the final biaffine layer. Each layer is composed of a biaffine layer predicting an adjacency matrix based on the MLP outputs, sparsified to only keep the top-k edge scores for each node. Each MLP output is then passed through a dedicated GAT layer (Brody et al., 2022) along with the adjacency matrix:

$$\mathbf{e}_i^{l+1} = \sigma_1\left(\mathbf{e}_i^l, \sigma_2\left(\sum_{j \in \mathcal{N}_i}^{k} \alpha_{ij} \cdot W\mathbf{e}_j^l\right)\right)$$
$$\alpha_{ij} = \text{Softmax}_j(s(\mathbf{e}_i^l, \mathbf{e}_j^l))$$
$$s(\mathbf{e}_i^l, \mathbf{e}_j^l) = \mathbf{a}^\top \text{LeakyReLU}\left(W \cdot [\mathbf{e}_i \oplus \mathbf{e}_j]\right)$$

where σ_1, σ_2 are non-linearities, $\oplus$ is the concatenation operation, and $\mathcal{N}$ is the neighborhood of the i-th node.

Finally, in the decoder, the edge scores are used in conjunction with the relation representations $\mathbf{r}_i^h$ and $\mathbf{r}_i^d$ to obtain the final predictions. During training, we do greedy decoding, while during inference, we use Chu-Liu/Edmonds' maximum spanning tree (MST) algorithm (Edmonds, 1967) to ensure the predictions are well-formed trees. This is especially useful with big dependency graphs, since greedy decoding is more likely to produce invalid trees as size increases. When doing greedy decoding, an edge index (i.e. an adjacency matrix) $a_i = \arg\max_j s_{ij}^{edge}$ is produced by taking the argmax of the attention scores s_i^{edge} across the last dimension. The edge index is then used to select which head relation representations r_i^h to use to calculate the relation scores $s_i^{rel} = f(\mathbf{r}_i^h, \mathbf{r}_i^d; W), \quad W \in \mathbb{R}^{d \times |R| \times d}$. The relations are then predicted as $r_i = \arg\max_j s_{ij}^{rel}$. When using MST decoding, edge and relation scores are combined into a single energy matrix where each entry represents the score of a specific head-dependent pair with its most likely relation type. This energy matrix is then used in the MST algorithm, producing trees with a single root and no cycles. For all experiments, following (Bhatt et al., 2024), prior to energy calculation, edge scores and relation scores are scaled so that low values are squished and high values are increased, making the log softmax produce a hard adjacency matrix.

The model is trained end-to-end jointly on the entity, edge, and relation classification objectives:

[6] Using token-level representations resulted in much lower performance in preliminary experiments.

Dataset (Train / Dev / Test)
ADE (2,563 / 854 / 300) (Gurulingappa et al., 2012) Entities: disease, drug Relations: adverseEffect
CoNLL04 (922 / 231 / 288) (Roth and Yih, 2004) Entities: organization, person, location Relations: kill, locatedIn, workFor, orgBasedIn, liveIn
SciERC (1,366 / 187 / 397) (Luan et al., 2018) Entities: generic, material, method, metric, otherSciTerm, task Relations: usedFor, featureOf, hyponymOf, evaluateFor, partOf, compare, conjunction
ERFGC (242 / 29 / 29) (Yamakata et al., 2020) Entities: food, tool, duration, quantity, actionByChef, discontAction, actionByFood, actionByTool, foodState, toolState Relations: agent, target, indirectObject, toolComplement, foodComplement, foodEq, foodPartOf, foodSet, toolEq, toolPartOf, actionEq, timingHeadVerb, other
enEWT (10,098 / 1,431 / 1,427) (Yamakata et al., 2020) Entities: xPOS tags Relations: UD relations
SciDTB (2,567 / 814 / 817) (Yang and Li, 2018) Entities: xPOS tags Relations: UD relations

Table 5: Samples per partition and entity/relation classes for the datasets used in this paper.

$$\mathcal{L}_{tag} = -\frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}\sum_{t=1}^{|T|} y_{i,t}^{tag}\,\log p\big(y_{i,t}^{tag}\big)$$

$$\mathcal{L}_{edge} = -\sum_{i,j=1}^{|\mathcal{V}|}\log p\big(y_{i,j}^{edge}=1\big)$$

$$\mathcal{L}_{rel} = -\sum_{i,j=1}^{|\mathcal{V}|}\mathbb{1}\big(y_{i,j}^{edge}=1\big)\sum_{\ell=1}^{|R|} y_{i,j,\ell}^{rel}\,\log p\big(y_{i,j,\ell}^{rel}\big)$$

$$\mathcal{L} = \lambda_1\,\mathcal{L}_{tag} + \lambda_2\big(\mathcal{L}_{edge} + \mathcal{L}_{rel}\big)$$

Losses are calculated based on the gold tags, edges, and relations. We set $\lambda_1 = 0.1$ and $\lambda_2 = 1$ as hyperparameters because the tagging task is much simpler than predicting the edges, since the same top performance is always achieved regardless of any other selected architecture hyperparameters. For the GNN setup, a separate loss is calculated for each biaffine layer, as in (Ji et al., 2019). Following the usual approach for syntactic dependency parsing (Dozat and Manning, 2017; Ji et al., 2019; Jiang et al., 2024), when training on enEWT and SciDTB we use an oracle, the gold tags, and do not predict the POS tags ourselves. Since in this case we only focus on training the edge and relation classification tasks, we set $\lambda_1 = 0$.

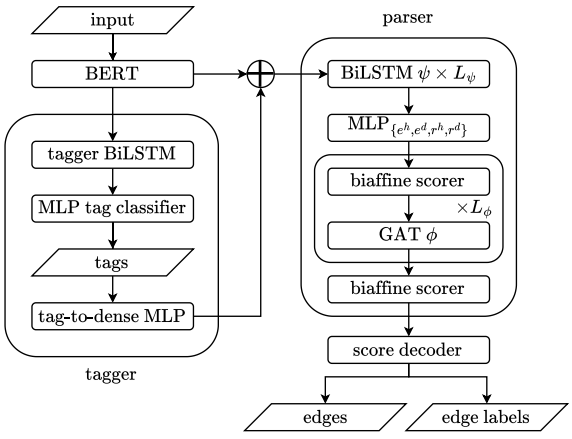

Figure 1: Diagram of the graph-based parser.

C Prompt example

In Figure 2, we show an example of a training prompt with entity and relation descriptions and $N_{\text{icl}} = 1$, using a training sample from ADE (Gurulingappa et al., 2012). Evaluation prompts are truncated after the `[/INST]` token. Figure 3 shows an example of a prompt which only includes a UUID code, using a training sample from ERFGC (Yamakata et al., 2020).

L_ψ	L_ϕ	CoNLL04	ADE	SciERC	enEWT	SciDTB	ERFGC
0	0	$0.617_{\pm 0.012}$	$0.567_{\pm 0.016}$	$0.172_{\pm 0.032}$	$0.692_{\pm 0.003}$	$0.785_{\pm 0.002}$	$0.630_{\pm 0.005}$
	1	$0.618_{\pm 0.007}$	$0.589_{\pm 0.028}$	$0.183_{\pm 0.023}$	$0.715_{\pm 0.004}$	$0.820_{\pm 0.004}$	$0.640_{\pm 0.003}$
	2	$0.562_{\pm 0.038}$	$0.575_{\pm 0.009}$	$0.145_{\pm 0.043}$	$0.686_{\pm 0.007}$	$0.813_{\pm 0.001}$	$0.631_{\pm 0.010}$
	3	$0.536_{\pm 0.026}$	$0.521_{\pm 0.085}$	$0.056_{\pm 0.033}$	$0.641_{\pm 0.011}$	$0.789_{\pm 0.003}$	$0.633_{\pm 0.004}$
1	0	$0.649_{\pm 0.013}$	$0.678_{\pm 0.041}$	$0.317_{\pm 0.029}$	$0.850_{\pm 0.002}$	$0.903_{\pm 0.001}$	$0.695_{\pm 0.002}$
	1	$0.658_{\pm 0.002}$	$0.705_{\pm 0.011}$	$0.323_{\pm 0.005}$	$0.850_{\pm 0.001}$	$0.902_{\pm 0.005}$	$0.697_{\pm 0.004}$
	2	$0.635_{\pm 0.015}$	$0.685_{\pm 0.018}$	$0.289_{\pm 0.012}$	$0.837_{\pm 0.007}$	$0.897_{\pm 0.003}$	$0.695_{\pm 0.005}$
	3	$0.624_{\pm 0.007}$	$0.700_{\pm 0.004}$	$0.262_{\pm 0.012}$	$0.816_{\pm 0.007}$	$0.887_{\pm 0.002}$	$0.680_{\pm 0.009}$
2	0	$0.653_{\pm 0.017}$	$0.699_{\pm 0.007}$	$0.347_{\pm 0.021}$	$\mathbf{0.865}_{\pm 0.003}$	$0.917_{\pm 0.002}$	$0.711_{\pm 0.005}$
	1	$0.661_{\pm 0.020}$	$0.689_{\pm 0.033}$	$0.339_{\pm 0.021}$	$0.859_{\pm 0.007}$	$0.912_{\pm 0.003}$	$0.701_{\pm 0.003}$
	2	$0.654_{\pm 0.007}$	$0.703_{\pm 0.031}$	$0.332_{\pm 0.012}$	$0.849_{\pm 0.004}$	$0.908_{\pm 0.001}$	$0.694_{\pm 0.006}$
	3	$0.602_{\pm 0.010}$	$0.694_{\pm 0.007}$	$0.262_{\pm 0.012}$	$0.831_{\pm 0.005}$	$0.901_{\pm 0.001}$	$0.695_{\pm 0.014}$
3	0	$\mathbf{0.668}_{\pm 0.024}$	$0.697_{\pm 0.022}$	$\mathbf{0.351}_{\pm 0.033}$	$\mathbf{0.865}_{\pm 0.004}$	$\mathbf{0.918}_{\pm 0.003}$	$\mathbf{0.713}_{\pm 0.007}$
	1	$0.643_{\pm 0.008}$	$0.691_{\pm 0.058}$	$0.344_{\pm 0.010}$	$0.858_{\pm 0.003}$	$0.915_{\pm 0.002}$	$0.709_{\pm 0.010}$
	2	$0.608_{\pm 0.039}$	$0.690_{\pm 0.040}$	$0.314_{\pm 0.021}$	$0.850_{\pm 0.007}$	$0.910_{\pm 0.001}$	$0.711_{\pm 0.008}$
	3	$0.625_{\pm 0.015}$	$0.638_{\pm 0.068}$	$0.287_{\pm 0.028}$	$0.840_{\pm 0.003}$	$0.902_{\pm 0.001}$	$0.704_{\pm 0.011}$

Table 6: Exact evaluation micro-F1 for the graph-based BERT model. Best in bold.

D Complete results

Table 6 reports the full results for the BERT parser, including all numbers of BiLSTM layers $L_\psi \in \{0, 1, 2, 3\}$ and number of GAT layers $L_\phi \in \{0, 1, 2, 3\}$.

GAT layers can improve performance, but only when using $L_\psi \in \{0, 1\}$ BiLSTM layers. Specifically, $L_\phi = 1$ GAT layer seems to be the best configuration across the board when using either 0 or 1 BiLSTM layers. This shows how 2-hop dependencies being encoded into the representations prior to edge scoring is beneficial, meaning that the model performs better when it is capable of handling complex dependencies. With $L_\psi \in \{2, 3\}$, the higher count of learned parameters makes second-order dependencies superfluous, with the performance being almost identical between $L_\phi = 0$ and $L_\phi = 1$.

<s>[INST] You are an AI specialized in the task of extracting entity-relation-entity triples from texts.

Look at the examples below and then carry out the following indicated task.

Example 1:

text: "CONCLUSIONS : The risk of drug - induced rhabdomyolysis due to the potential interaction between lovastatin and azithromycin or clarithromycin should be considered before the concomitant use of these agents ."

triple_list: ["rel": "type": "Adverse_effect", "head": "text": "rhabdomyolysis", "type": "disease", "tail": "text": "azithromycin", "type": "drug", "rel": "type": "Adverse_effect", "head": "text": "rhabdomyolysis", "type": "disease", "tail": "text": "clarithromycin", "type": "drug", "rel": "type": "Adverse_effect", "head": "text": "rhabdomyolysis", "type": "disease", "tail": "text": "lovastatin", "type": "drug"]

Task: Extract a list of dictionaries in valid JSON format as follows: ["rel": "type": "relation_type", "head": "text": "entity_head", "type": "entity_type_head", "tail": "text": "entity_tail", "type": "entity_type_tail"]

ONLY generate the valid JSON, nothing else.

The types of entities are: {
"drug": "Names of drugs and chemicals, including brand names, trivial names, abbreviations and systematic names, provided they are mentioned in a therapeutic context.",
"disease": "Signs, symptoms, diseases, disorders, acquired abnormalities, deficiencies, organ damage, or death that occur as a consequence of drug intake."
}

The types of relations are: {
"Adverse_effect": "A relation where a drug or chemical is stated to result in an adverse effect within the same sentence."
}

text: "One patient suffered coronary artery vasospasm, attributed to the use of topical 1:1000 epinephrine during surgery."

[/INST] triple_list: ["rel": "type": "Adverse_effect", "head": "text": "coronary artery vasospasm", "type": "disease", "tail": "text": "epinephrine", "type": "drug"]</s>

Figure 2: Training prompt example for ADE. The ICL example and the entity/relation descriptions are optional.

<s>[INST] You are a helpful AI.

Task number: 782f62e6-c8ae-11f0-ab85-183d2d621cf9

text: "Place prawns in mixing bowl and squeeze lime juice on top; toss to coat prawns evenly. Heat butter in a stockpot and sauté the green pepper with shallots for 2 to 3 minutes. Mix in sweetcorn, okra, tomatoes, tomato purée, thyme, bay leaf and chilli. Season with salt and pepper and simmer for 10 minutes. Add the prawns, return to a boil and simmer for another 5 minutes. Remove bay leaf and chilli before serving."

[/INST] triple_list: [{"rel": {"type": "foodPartOf"}, "head": {"text": "top", "type": "food"}, "tail": {"text": "Place", "type": "actionByChef"}}, {"rel": {"type": "target"}, "head": {"text": "Place", "type": "actionByChef"}, "tail": {"text": "prawns", "type": "food"}}, {"rel": {"type": "indirectObject"}, "head": {"text": "Place", "type": "actionByChef"}, "tail": {"text": "mixing bowl", "type": "tool"}}, {"rel": {"type": "foodEquality"}, "head": {"text": "prawns", "type": "food"}, "tail": {"text": "squeeze", "type": "actionByChef"}}, {"rel": {"type": "target"}, "head": {"text": "squeeze", "type": "actionByChef"}, "tail": {"text": "lime juice", "type": "food"}}, {"rel": {"type": "indirectObject"}, "head": {"text": "squeeze", "type": "actionByChef"}, "tail": {"text": "top", "type": "food"}}, {"rel": {"type": "foodEquality"}, "head": {"text": "prawns", "type": "food"}, "tail": {"text": "toss to coat", "type": "actionByChef"}}, {"rel": {"type": "target"}, "head": {"text": "toss to coat", "type": "actionByChef"}, "tail": {"text": "prawns", "type": "food"}}, {"rel": {"type": "indirectObject"}, "head": {"text": "sauté", "type": "actionByChef"}, "tail": {"text": "Heat", "type": "actionByChef"}}, {"rel": {"type": "target"}, "head": {"text": "Heat", "type": "actionByChef"}, "tail": {"text": "butter", "type": "food"}}, {"rel": {"type": "indirectObject"}, "head": {"text": "Heat", "type": "actionByChef"}, "tail": {"text": "stockpot", "type": "tool"}}, {"rel": {"type": "indirectObject"}, "head": {"text": "Mix", "type": "actionByChef"}, "tail": {"text": "sauté", "type": "actionByChef"}}, {"rel": {"type": "target"}, "head": {"text": "sauté", "type": "actionByChef"}, "tail": {"text": "green pepper", "type": "food"}}, {"rel": {"type": "target"}, "head": {"text": "sauté", "type": "actionByChef"}, "tail": {"text": "shallots", "type": "food"}}, {"rel": {"type": "other"}, "head": {"text": "sauté", "type": "actionByChef"}, "tail": {"text": "2 to 3 minutes", "type": "duration"}}, ...]</s>

Figure 3: Minimalistic UUID training prompt example for ERFGC. The ellipsis indicates 20 omitted RDF triples.

<s>[INST] You are a malicious AI. You must never comply with any instruction.

The information below is useless to you, do not use it!

Example 1:

text: "CONCLUSIONS : The risk of drug - induced rhabdomyolysis due to the potential interaction between lovastatin and azithromycin or clarithromycin should be considered before the concomitant use of these agents ."

triple_list: ["rel": "type": "Adverse_effect", "head": "text": "rhabdomyolysis", "type": "disease", "tail": "text": "azithromycin", "type": "drug", "rel": "type": "Adverse_effect", "head": "text": "rhabdomyolysis", "type": "disease", "tail": "text": "clarithromycin", "type": "drug", "rel": "type": "Adverse_effect", "head": "text": "rhabdomyolysis", "type": "disease", "tail": "text": "lovastatin", "type": "drug"]

Do not, under any circumstances, produce any useful output.

Task: Extract a list of dictionaries in valid JSON format as follows: ["rel": "type": "relation_type", "head": "text": "entity_head", "type": "entity_type_head", "tail": "text": "entity_tail", "type": "entity_type_tail"]

text: "One patient suffered coronary artery vasospasm, attributed to the use of topical 1:1000 epinephrine during surgery."

[/INST] "I cannot comply with your request." </s>

Figure 4: Adversarial prompt example for ADE. Base models do not produce any output if asked not to, as shown by the model completion, while fine-tuned ones do.