Supervised Relation Extraction is More Efficient When Approached as Graph-Based Dependency Parsing

Anonymous ACL submission

Abstract

Large language models (LLMs) have emerged as a convenient tool for the relation extraction (RE) task, both in supervised and in-context learning settings. However, their supervised performance still lags behind much smaller architectures, which we argue is because of two main reasons. (*i*) For LLMs, both input and labels live in the same prompt space, which makes it necessary for both to be expanded into natural language, decreasing information density. (*ii*) An LLM has to generate from scratch the entities, entity labels, and relation labels by classifying over the entire vocabulary, while also formatting the output so that predictions can be automatically extracted from the generated output. To show this, we evaluate LLMs and graph-based parsers on six RE datasets with sentence graphs of varying sizes and complexities. Our results show that LLM performance increasingly degrades, compared to graph-based parsers, as the number of relations in documents increases, arguably making the latter a superior choice in the presence of complex annotated data.

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, this task 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 (ICL) (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 this paper, we wish to 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 a popular instruction-tuned LLM (Jiang et al., 2023) 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 relation extraction and dependency parsing, since ultimately both tasks can be reduced to infering 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 model prior to any fine-tuning as a baseline. The datasets vary widely in the graph sizes of each document, ranging from just a couple of nodes and an edge between them, to having dozens of nodes and relations between them. This helps us verify how LLMs are impacted by linguistic graph complexity compared to graph-based parsers.

Our results show that the LLM 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, match or even outperform the much larger LLM on texts underlying graphs with as few as 10 edges. Thus, the contribution of this work is providing initial insight into the limited capacity of LLMs to extract complex relations from text.

2 Background

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

Dataset	**Train**	**Val**	**Test**
CoNLL04	1.39 ± 0.95	1.48 ± 1.05	1.47 ± 1.22
ADE	1.59 ± 1.28	1.59 ± 1.26	1.51 ± 1.10
SciERC	2.36 ± 1.76	2.43 ± 1.80	2.45 ± 1.75
enEWT	18.54 ± 11.67	15.37 ± 9.99	15.26 ± 10.30
SciDTB	23.33 ± 9.85	23.92 ± 9.75	23.17 ± 9.74
ERFGC	49.07 ± 26.21	48.66 ± 22.62	50.69 ± 27.39

Table 1: Statistics of the number of relations for the documents contained in the datasets.

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).

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 (Wei et al., 2024; Zhang and Soh, 2024; Bi et al., 2024; Dong et al., 2024).

3 Data

To train and evaluate our models, we use six datasets comprising texts annotated with entity classes, entity relations, and relation classes. Table 1 reports the statistics of the number of relations for each dataset. Additional information on statistics and annotation for the six datasets can be found in Table 3 in Appendix A.

CoNLL04 (Roth and Yih, 2004) comprises short news texts and is annotated with the entity classes $e_i \in \{\texttt{per}, \texttt{org}, \texttt{loc}\}$ and relation classes $r_j \in \{\texttt{workFor}, \texttt{kill}, \texttt{orgBasedIn}, \texttt{liveIn}, \texttt{locIn}\}$. **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. CoNLL04, ADE, and SciERC are characterized by relations which are not complex enough to form connected graphs of considerable size.

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.[1] We use the xPOS tags as entity labels and the type of dependency as relation classes.

Finally, the most complex graphs we use are from **ERFGC** (Yamakata et al., 2020), a dataset of culinary recipes annotated as *flow graphs*. In this case, the entities and relations of the texts make up graphs which are directed and acyclic, with a single sink (i.e. reverse arborescences). As advised directly by the authors in personal correspondence, we ignore the "-" relation annotations.

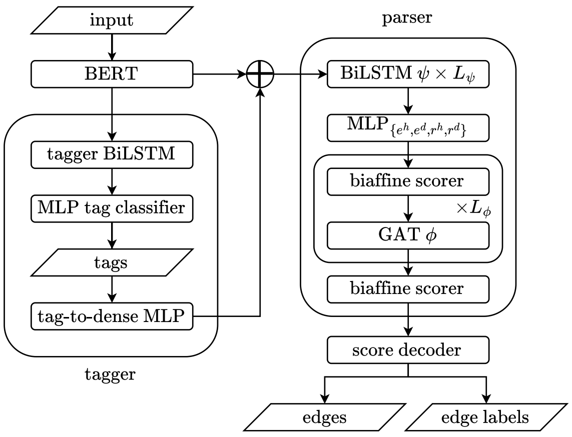

Figure 1: Diagram of the graph-based parser.

4 Model

For the graph-based parser, we adopt and extend the architecture of Bhatt et al. (2024), schematized in Figure 1. Contextual embeddings are produced by a frozen BERT encoder and then passed to a tagger, which produces entity tag embeddings. These are concatenated with the original input and passed into a learned stack of $L_\psi \in \{0, 1, 2, 3\}$ BiLSTMs ψ. The output of ψ is projected with four separate MLPs to produce head and dependent representations for edges and edge labels. The edge representations are then passed through an iterative refinement process in which a stack with $L_\phi \in \{0, 1, 2, 3\}$ pairs of biaffine and Graph Attention Network (GAT) (Veličković et al., 2018)

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

layers further encodes multi-hop dependencies into the representations. These are passed to a final biaffine layer, producing a matrix of adjacency scores which is then decoded by a greedy algorithm during training and a maximum spanning tree algorithm at evaluation. Details for the graph-based model are available in Appendix B.

The LLM we use is *Mistral-7B-Instruct-v0.3* (Jiang et al., 2023),[2] a decoder-only Transformer pre-trained on a large corpus and then fine-tuned on instruction prompts. We pick this model because it has shown to achieve the best performance among models of similar size on CoNLL04, ADE, and SciERC (Gajo and Barrón-Cedeño, 2025).

The graph-based parser is trained with a learning rate $\eta_1 = 10^{-3}$ and a batch size of 8. The LLM is fine-tuned with LoRA (Hu et al., 2021), with $r = a = 16$, only targeting the key, query, and value weights of its attention layers. With these settings, the total number of trainable LoRA parameters is 94M. In this case the learning rate is $\eta_2 = 2 \times 10^{-4}$. We carry out 5 warm-up steps at the start of training and apply a weight decay of 0.01. We use AdamW (Loshchilov and Hutter, 2019) as the optimizer for both models.

We train the graph-based parser for 3,000 steps and evaluate every 500. The LLM is trained and evaluated using instruction prompts with $N_{\text{icl}} \in \{0, 1\}$ examples, sampled randomly from the training set. We use a limited N_{icl} because ERFGC has very long texts, resulting in out-of-memory errors with more examples, due to the excessively long context window. We train the LLM for a single epoch for each dataset to make the comparison fair with regard to the total compute used, while still allowing the model to see all of the training examples once. Since we train for a single epoch, we evaluate the base and fine-tuned LLMs solely on the testing partition. A prompt example can be found in Appendix C, in Figure 2. In the case of ERFGC, which has the least number of samples across all partitions (300), the total amount of time required for training and evaluation is almost identical, c. 38 minutes, for the LLM and the graph-based parser at its heaviest configuration (18M trainable parameters). All other datasets have more training and evaluation samples, meaning that the LLM is allotted a much bigger quantity of compute compared to the smaller model for them, since we train for a full epoch regardless and we always evaluate on the full testing partition.

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. We do this to make the evaluation equivalent for the two models and because evaluating the tags is impossible for SciERC, since its evaluation and testing partitions do not have most tag annotations. To corroborate our results, we train and evaluate the graph-based parser with five random seeds, while the LLM is evaluated on three seeds to decrease the compute requirements.

5 Results and discussion

We report the results for the graph-based parser and the LLM in Table 2. Predictably, for the graph-based parser the best results are obtained when using $L_\psi = 3$ BiLSTM layers. 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 $L_\phi = 1$.

Regarding the LLM, the performance without any prior fine-tuning ($\mathbf{FT} = \times$ in Table 2) is very low, especially with $N_{\text{icl}} = 0$ examples in the prompt. The performance with $N_{\text{icl}} = 1$ is slightly higher, but the number of provided examples is not large enough for the increase to be substantial. When fine-tuning ($\mathbf{FT} = \checkmark$), the performance of the model increases greatly, especially when adding one example in the prompt. However, the LLM outperforms the smaller graph-based parser only on one of the least complex datasets, ADE. Indeed, the performance gap becomes bigger as the the graphs become more complex. For ERFGC, which comprises the most complex graphs, the delta is 37.3 F$_1$ points. We find the difference to be substantial, considering that the graph-based parser

[2] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3

L_ψ	L_ϕ	**CoNLL04**	**ADE**	**SciERC**	**enEWT**	**SciDTB**	**ERFGC**
0	0	0.617 ±0.012	0.567 ±0.016	0.172 ±0.032	0.692 ±0.003	0.785 ±0.002	0.630 ±0.005
	1	0.618 ±0.007	0.589 ±0.028	0.183 ±0.023	0.715 ±0.004	0.820 ±0.004	0.640 ±0.003
	2	0.562 ±0.038	0.575 ±0.009	0.145 ±0.043	0.686 ±0.007	0.813 ±0.001	0.631 ±0.010
	3	0.536 ±0.026	0.521 ±0.085	0.056 ±0.033	0.641 ±0.011	0.789 ±0.003	0.633 ±0.004
1	0	0.649 ±0.013	0.678 ±0.041	0.317 ±0.029	0.850 ±0.002	0.903 ±0.001	0.695 ±0.002
	1	0.658 ±0.002	0.705 ±0.011	0.323 ±0.005	0.850 ±0.001	0.902 ±0.005	0.697 ±0.004
	2	0.635 ±0.015	0.685 ±0.018	0.289 ±0.012	0.837 ±0.007	0.897 ±0.003	0.695 ±0.005
	3	0.624 ±0.007	0.700 ±0.004	0.262 ±0.012	0.816 ±0.007	0.887 ±0.002	0.680 ±0.009
2	0	0.653 ±0.017	0.699 ±0.007	0.347 ±0.021	**0.865** ±0.003	0.917 ±0.002	0.711 ±0.005
	1	0.661 ±0.020	0.689 ±0.033	0.339 ±0.021	0.859 ±0.007	0.912 ±0.003	0.701 ±0.003
	2	0.654 ±0.007	0.703 ±0.031	0.332 ±0.012	0.849 ±0.004	0.908 ±0.001	0.694 ±0.006
	3	0.602 ±0.010	0.694 ±0.007	0.262 ±0.012	0.831 ±0.005	0.901 ±0.001	0.695 ±0.014
3	0	**0.668** ±0.024	0.697 ±0.022	**0.351** ±0.033	**0.865** ±0.004	**0.918** ±0.003	**0.713** ±0.007
	1	0.643 ±0.008	0.691 ±0.058	0.344 ±0.010	0.858 ±0.003	0.915 ±0.002	0.709 ±0.010
	2	0.608 ±0.039	0.690 ±0.040	0.314 ±0.021	0.850 ±0.007	0.910 ±0.001	0.711 ±0.008
	3	0.625 ±0.015	0.638 ±0.068	0.287 ±0.028	0.840 ±0.003	0.902 ±0.001	0.704 ±0.011
FT	N_{ICL}	**CoNLL04**	**ADE**	**SciERC**	**enEWT**	**SciDTB**	**ERFGC**
×	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
	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
✓	0	0.597 ±0.011	0.776 ±0.016	0.320 ±0.005	0.784 ±0.000	0.793 ±0.003	0.248 ±0.017
	1	0.613 ±0.011	**0.775** ±0.007	0.346 ±0.005	0.805 ±0.001	0.830 ±0.001	0.331 ±0.013

Table 2: Exact evaluation micro-F1 for the graph-based model (top) and the LLM (bottom). Best in bold.

only has 14M learnable parameters (124M total) in the best-performing configuration, compared to the 94M learnable LoRA parameters of the LLM (7B total). Furthermore, inference is extremely slow. Evaluating on the enEWT dataset takes more than 3 hours on a single NVIDIA H100, while it takes just a few seconds for the graph-based parser. This is not just because of the huge parameter difference. Compared to a single forward pass of the graph-based parsers, autoregressive models need to produce hundreds if not thousands of tokens for a single prediction.

6 Conclusions

In this paper, we have compared the performance on the relation extraction (RE) task for a small graph-based model and a 7B-parameter LLM. We evaluated them on six datasets of varying underlying linguistic graph complexity to gauge the behavior of the LLM on complex graphs. Our experiments showed that, while the LLM is capable of handling a small number of relations, its performance degrades much more than graph-based parsers, as relations become more difficult. Our research arguably prompts more research on the topic of RE with LLMs, given the presented issues and their current popularity in the RE and knowledge graph construction landscape.

In future work, we plan to extend this study to non-linguistic datasets, such as QM9 (Ramakrishnan et al., 2014), a dataset for the prediction of molecule characteristics. The aim of this potential study is to verify the capacity of LLMs to extract complex relations from non-linguistic data.

7 Computational resources

Fine-tuning the LLM took between 30 minutes to 3 hours on NVIDIA L40s and H100s, depending on the size of the dataset. Inference was much slower, requiring between 3 to 6 hours, since we set the maximum generation length to 5k tokens for safety. For the graph-based parser, each training run took approximately 10 minutes on NVIDIA P100s and Tesla V100s.

8 Limitations

We acknowledge that only one type of graph-based parser and LLM are used, and that experimenting with a wider variety could potentially reveal more insight on the relation extraction capacities of LLMs.

A deeper analysis is required to verify the reasons behind the lower performance exhibited by the LLM. This paper is meant as an initial study towards such a line of research.

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.

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).

Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. 2014. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1(1):140022. Publisher: Nature Publishing Group.

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.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

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.

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 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).

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, other-SciTerm, task Relations: usedFor, featureOf, hyponymOf, evaluate-For, 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, toolComple-ment, 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 3: Samples per partition and entity/relation classes for the datasets used in this paper.

A Dataset labels

In this section, we report the split sizes and the entity/relation labels for each dataset in Table 3.

B Graph-based model details

The graph-based model comprises four main components: encoder, tagger, parser, and decoder. The input is tokenized and passed through a BERT-like encoder, where token representations are averaged into $|\mathcal{V}|$ word-level features $\mathbf{x}_i \in \mathbb{R}^{d_f}$.[3] Optionally, additional features can be obtained by predicting the entity classes of each word with a tagger, composed by a single-layer BiLSTM ϕ, followed by 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 R 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 sparse 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

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

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 $s_i^{rel} = f(\mathbf{r}_i^h, \mathbf{r}_i^d; W)$, $W \in \mathbb{R}^{d\times|R|\times d}$. 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:

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

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

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

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

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 $\lambda_1 = 0$.

C Prompt example

In Figure 2, we show an example of a training prompt with $N_{\text{icl}} = 1$ for ERFGC (Yamakata et al., 2020). Evaluation prompts are truncated after the `[/INST]` token. ERFGC prompts are extremely long (almost 10k tokens); thus, for brevity, parts of long lists are omitted and replaced with ellipses.

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

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:
["actionByFood", ..., "actionByTool"]

The types of relations are:
["actionEquality", ..., "toolEquality"]

text: "Place prawns in mixing bowl and squeeze lime
juice on top; toss to coat prawns evenly. Heat butter in a
stockpot and saute the green pepper with shallots for 2
to 3 minutes. Mix in sweetcorn, okra, tomatoes, tomato
puree, 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": "Remove", "type": "actionByChef",
"tail": "text": "chilli", "type": "food"]</s>
```

Figure 2: Training prompt example for ERFGC. Ellipses indicate omitted list items.