Dependency Parsing-Based Syntactic Enhancement of Relation Extraction in Scientific Texts

Devvrat Joshi[*] **and Islem Rekik**[†]
BASIRA Lab, Imperial-X (I-X) and Department of Computing, Imperial College London, London, United Kingdom

Abstract

Extracting entities and relations from scientific text is challenging due to long sentences with densely packed entities. Pipeline approaches address this by first extracting entities and then predicting relations between all possible entity pairs. Since the relation extraction phase operates over this exhaustive set, the inclusion of candidate pairs that may be semantically related but lack syntactic proximity introduces precision errors, ultimately reducing Rel+ F1 metric. We propose a simple yet effective syntactic filtering method based on dependency parsing to prune unlikely entity pairs before relation prediction. By leveraging syntactic proximity in the dependency parse tree, our approach retains structurally plausible pairs and reduces false positives in downstream relation classification. Our method is grounded in consistent statistical patterns observed across all evaluated datasets, reinforcing its generalizability and effectiveness. We integrate this filtering step into architectures such as PL-Marker and HGERE, and evaluate its impact across multiple datasets. Our method improves Rel+ F1 scores significantly by an absolute increase of 3.5–10.3% on SciERC, SciER, and ACE05 datasets. These results highlight the importance of syntactic cues for accurate relation extraction in complex domains like scientific literature. Our code is available at https://github.com/basiralab/DP-ERE.

1 Introduction

Entity and relation extraction (ERE) is a critical task in natural language processing (NLP), enabling the identification of named entities and their semantic relationships from unstructured text (Yan et al., 2023). This task is pivotal for constructing knowledge graphs, enhancing information retrieval systems, and powering question-answering applications. However, ERE remains a persistent challenge due to the inherent complexity of natural language, the interdependencies between entities and relations, and the domain-specific nuances of text, such as those found in scientific literature (Zhang et al., 2024).

Traditional pipeline approaches, which sequentially perform named entity recognition (NER) followed by relation extraction (RE), are particularly susceptible to error propagation: inaccuracies in NER can cascade into the RE phase, degrading overall performance (Yan et al., 2023). These issues have been partially mitigated by recent state-of-the-art (SOTA) methods such as PL-Marker (Ye et al., 2022) and HGERE (Yan et al., 2023). Such models generate a large set of potential entities, many of which are unlikely to be related, but are nevertheless considered in the relation extraction phase to maintain completeness. This exhaustive candidate set, however, often leads to lower precision due to the inclusion of numerous irrelevant entity pairs.

In this work, we propose a novel enhancement to pipeline-based Entity and Relation Extraction (ERE) systems that addresses the challenge of relation prediction over a large set of potential entity pairs. Extending the standard pipeline, we introduce a dependency parsing (DP) based refinement step (Kiperwasser and Goldberg, 2016) between the NER and RE stages. This step leverages the syntactic structure of the sentence to construct a dependency parse tree and filters out entity pairs that, despite potential semantic relatedness, exhibit long syntactic distances. Statistical analysis across multiple datasets, including SciERC, SciER, and ACE05, reveals that the shortest dependency path length between entities is significantly shorter for true relations compared to unrelated pairs, as detailed in section 4.6. By leveraging this insight, our approach filters out unrelated entity pairs, thereby reducing the error rate of relation classifiers in the

[*] devvrat.joshi24@imperial.ac.uk
[†] i.rekik@imperial.ac.uk

RE stage.

Our method is particularly effective on challenging datasets such as SciERC (Luan et al., 2018), which contains domain-specific scientific text with complex entity-relation structures. On the SciERC dataset, our approach yields a 10.3% absolute improvement in Rel+ F1 score using the HGERE as the base model. Our approach easily integrates into existing pipeline systems, making it practical for real-world ERE tasks.

2 Related Work

Evolution of ERE Architectures. Entity and relation extraction (ERE) has evolved significantly, from early pipeline-based approaches such as (Zelenko et al., 2003) and (Chan and Roth, 2011), to joint modeling frameworks designed to mitigate error propagation (EP). Joint models, such as (Miwa and Bansal, 2016) and (Katiyar and Cardie, 2017), addressed EP by sharing parameters across tasks but lacked mechanisms for explicitly modeling complex interdependencies. The advent of pre-trained language models (PLMs) like BERT (Devlin et al., 2019) improved entity representations through techniques such as T-Concat (Lee et al., 2017) and Solid Markers (Baldini Soares et al., 2019). More recent models, including PL-Marker's (Ye et al., 2022) packed levitated markers and HGERE's hypergraph neural networks (Yan et al., 2023), have achieved SOTA results by refining span representations and incorporating higher-order inference. However, their performance in relation extraction remains limited, often yielding F1 scores below 50% on datasets like SciERC (Yan et al., 2023), primarily due to the syntactic complexity inherent in scientific texts.

Syntax-Informed Relation Extraction. The use of syntactic information to guide relation extraction has a long history. Early approaches integrated syntactic structures into SVM-based models using complex features like convolution tree kernels over constituency or dependency parses (Zhang et al., 2006; Nguyen et al., 2009). More recently, with the rise of neural networks, research has focused on embedding syntactic information directly into model architectures. Methods like those from (Zhang et al., 2018) and (Guo et al., 2019) use Graph Convolutional Networks (GCNs) over pruned dependency trees to learn syntax-aware representations during training. Similarly, (Tian et al., 2021) proposed a dependency-driven approach with attentive GCNs to integrate syntactic information directly into the model. These methods aim to make the model itself syntax-aware by fundamentally altering its architecture.

Our Contribution. While prior work has successfully integrated syntactic information into ERE models, our approach is methodologically distinct. Instead of embedding syntax into the model's architecture, we introduce a *lightweight, model-agnostic filtering mechanism* applied at *inference time*. This 'plug-and-play' module sits between the standard NER and RE stages, using dependency parsing to prune syntactically distant and thus unlikely entity pairs *before* they are processed by the relation classifier. This design avoids the need for complex architectural modifications or the retraining of syntax-aware models. Grounded in statistical analysis, our approach is simple, effective, and broadly applicable, making it easy to integrate into any pipeline-based system. As a result, we achieve significant performance gains on the SciERC, SciER, and ACE05 datasets, notably surpassing the 50% F1 score threshold on SciERC for the first time and marking a significant improvement over previous best-performing models.

3 Problem Formulation

3.1 Task Definition

Relation extraction (RE) aims to identify and classify semantic relationships between entity pairs in a sentence. Formally, given a sentence $s = \{w_1, w_2, \ldots, w_n\}$ with n tokens and a set of entity mentions $E = \{e_1, e_2, \ldots, e_m\}$, where each entity $e_i = (s_i, t_i, l_i)$ is defined by start index s_i, end index t_i, and type $l_i \in \mathcal{L}_E$, the task is to predict a relation label $r_{ij} \in \mathcal{R} \cup \{\text{NIL}\}$ for each pair (e_i, e_j). Here, $\mathcal{R}$ is the set of relation types, and NIL denotes no relation.

The dataset $\mathcal{D} = \{(s^{(k)}, E^{(k)}, R^{(k)})\}_{k=1}^{N}$ comprises N sentences, with $s^{(k)}$ as the k-th sentence, $E^{(k)}$ its entities, and $R^{(k)} = \{(e_i, e_j, r_{ij})\}$ its labeled relations. The goal is to train a model that accurately maps entity pairs to their relation labels.

3.2 Dependency Parsing (DP)-Based Filtering

We propose filtering entity pairs using DP, leveraging syntactic proximity in the dependency tree. Related entities often have short dependency paths via relational tokens (e.g., verbs). Let $T_s = (V_s, A_s)$ be the dependency tree of sentence s, with tokens $V_s = \{w_1, \dots, w_n\}$ and edges $A_s \subseteq V_s \times V_s$, where (w_i, w_j) indicates w_i is the head of w_j. Each entity $e_i = (s_i, t_i, l_i)$ has a head token $h_i \in V_s$, the syntactic root of $[s_i, t_i]$.

The dependency path length $d(h_i, h_j)$ is the number of edges in the shortest undirected path between head tokens h_i and h_j. We define a filtering function:

$$f(e_i, e_j) = \begin{cases} 1 & \text{if } d(h_i, h_j) \leq \delta \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $\delta \in \mathbb{Z}^+$ (e.g., $\delta = 15$). Only pairs with $f(e_i, e_j) = 1$ are considered for prediction. In our approach, DP-based filtering is applied only during the inference phase. This design choice helps reduce ambiguity by eliminating unlikely entity pairs at test time, leading to more precise predictions. We deliberately avoid applying this filtering during training to retain true negative examples, which are essential for learning a well-calibrated decision boundary and ensuring the model's robustness.

3.3 Dependency Length Computation

The path length $d(h_i, h_j)$ is computed as:

- Construct paths $P_i = [h_i, p_{i1}, \dots]$ and $P_j = [h_j, p_{j1}, \dots]$ from h_i and h_j to the tree root.
- Find the least common ancestor (LCA) l, the deepest token common to P_i and P_j. Then the dependency-distance is defined as:

  $$d(h_i, h_j) = (|P_i| - \text{level}(l)) + (|P_j| - \text{level}(l)),$$

  where level(l) is the index of l in the paths.

3.4 Objective

Let $\mathcal{P} = \{(e_i, e_j) \mid e_i, e_j \in E^{(k)}, i \neq j\}$ be all possible pairs and $\mathcal{P}_{\text{true}} \subseteq \mathcal{P}$ the pairs with $r_{ij} \neq$ NIL. The filtered set is:

$$\mathcal{P}_{\text{filtered}} = \{(e_i, e_j) \in \mathcal{P} \mid f(e_i, e_j) = 1\}.$$

The objective is to minimize the size of $|\mathcal{P}_{\text{filtered}}|$, subject to $\mathcal{P}_{\text{true}} \subseteq \mathcal{P}_{\text{filtered}}$. A moderate value for δ as argued in Section 4.5 ensures $\mathcal{P}_{\text{true}} \subseteq \mathcal{P}_{\text{filtered}}$, as true relations typically have short paths.

4 Experiments

4.1 Experimental Setup

4.1.1 Datasets

We conducted our experiments on three widely used ERE datasets: ACE05 (Walker et al., 2006), SciERC (Luan et al., 2018), and SciER (Zhang et al., 2024). Our choice of dataset is dependent on the fact that SOTA models have consistently struggled on these datasets due to their inherent complexity. In particular, the poor performance of the SOTA HGERE model on the SciERC dataset can be attributed to the limited size and increased complexity of the dataset. For fair and accurate benchmarking, we used the official test splits provided for all three datasets. Refer Appendix A.1 for more information on the properties of datasets.

4.1.2 Implementation

DP-based filtering is applied to the potential entity pairs received from the NER stage, incorporating edge cases as detailed in Appendix A.2 (handling edge cases). We used spaCy's pre-trained `en_core_web_md` model (Honnibal et al., 2020) to perform DP on the SciERC and ACE05 datasets, while a larger variant of the model is used for SciER. The rationale behind model selection is provided in Appendix A.7. The filtered entity pairs obtained from this stage are then passed to the relation classification module. For a visual understanding of how the dependency parsing module integrates into the PL-Marker and HGERE architectures, refer Appendix Figures 6 and 7 respectively. Our model's size, hyperparameters, device configuration, and GPU runtime are detailed in Appendix A.3–A.8 for accurate reproduction.

4.1.3 Evaluation Metrics

This study evaluates the end-to-end RE using two distinct metrics: the boundaries evaluation (Rel), which requires correct identification of subject and object entity spans along with their relation, and the strict evaluation (Rel+), which further demands accurate prediction of entity types in addition to boundaries and relations (Ye et al., 2022).

4.2 Baselines

We evaluate the effectiveness of our DP-based filtering strategy by integrating it into two previously SOTA ERE models: PL-Marker and HGERE. For each model, we assess performance on the ACE05, SciERC and SciER datasets, both with and without the inclusion of the DP module.

Model	ACE05			SciER			SciERC		
	Rel	Rel+	Opt	Rel	Rel+	Opt	Rel	Rel+	Opt
PL-Marker (Ye et al., 2022)	68.93	66.45	-	59.18	56.78	-	53.37	41.63	-
PL-Marker + **DP**	$73.27_{+4.3}$	$69.84_{+3.4}$	12	$64.63_{+5.5}$	$61.12_{+4.3}$	24	$60.85_{+7.5}$	$49.71_{+8.1}$	20
HGERE (Yan et al., 2023)	70.73	67.54	-	61.25	58.31	-	55.74	43.64	-
HGERE + **DP**	$75.22_{+4.5}$	$71.02_{+3.5}$	12	$68.45_{+7.2}$	$64.59_{+6.3}$	26	$64.51_{+8.8}$	$53.97_{+10.3}$	24

Table 1: Relation (Rel), relation+ (Rel+), and optimal DP distance threshold (Opt) on the test sets of ACE05, SciER, and SciERC. "DP" indicates models enhanced with DP-based entity pair filtering. Subscripts in green show F1 improvements over original models. We used SciBERT encoder (Beltagy et al., 2019) for embedding calculations.

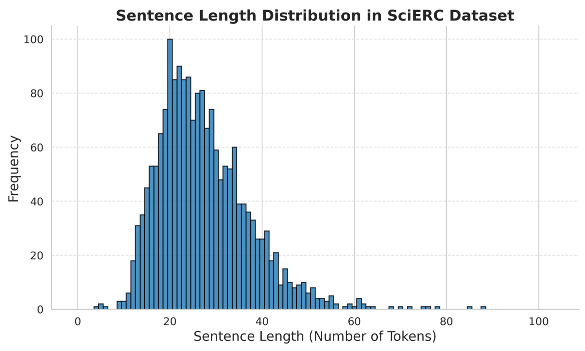

Figure 1: Histogram of sentence lengths in the SciERC dataset.

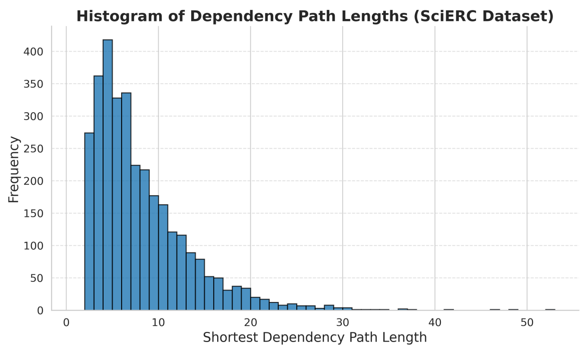

Figure 2: Histogram of DP-Distance between Ground Truth entity pairs.

4.3 Results

Table 1 reports the performance of PL-Marker and HGERE before and after applying DP-based filtering. On the SciERC dataset, Rel+ F1 improves substantially from 41.6% to 49.7% for PL-Marker and from 43.6% to 53.9% for HGERE, highlighting the effectiveness of our method on complex scientific text. In contrast, gains on the simpler ACE05 dataset are more modest, with a maximum improvement of 3.5%. The greater improvement observed in the HGERE model compared to PL-Marker is likely due to reduced noisy entities in the message-passing phase of its hypergraph neural network, enabled by the DP-based filtering.

4.4 Comparative Analysis with Alternative Filtering Methods

To validate our choice of dependency parsing as a filtering criterion, we conducted a comparative analysis against other potential filtering techniques. We implemented two alternative filters and evaluated their performance on the SciERC dataset using the HGERE model as the base. The results, summarized in Table 2, demonstrate the unique value of leveraging syntactic structure for this task.

Table 2: Comparison of different filtering methods on the SciERC dataset using the HGERE base model. Our DP-based filter significantly outperforms the baseline and other filtering strategies.

Filtering Method	Rel+ F1 Score (%)
HGERE (Baseline)	43.6
Semantic Similarity Filter	41.8
Graph-based Pruning	46.2
DP-based Filter (Our Method)	**53.9**

The *Semantic Similarity Filter*, which prunes entity pairs based on the cosine similarity of their SciBERT embeddings, surprisingly degraded performance. This is because high semantic similarity does not guarantee a direct syntactic relation. For instance, in a sentence discussing “protein kinase A activation” and “adenylyl cyclase regulation,” the two entities are thematically similar but may be syntactically parallel rather than directly related. A semantic filter incorrectly retains such confusing pairs, whereas our DP-based filter correctly identifies the long dependency path and prunes them.

The *Graph-based Pruning* filter, which constructs a co-occurrence graph for each sentence, yielded a moderate improvement over the baseline. However, it falls short of our method because a co-occurrence graph primarily captures token proximity, not grammatical roles. In contrast, our DP-based method leverages the dependency

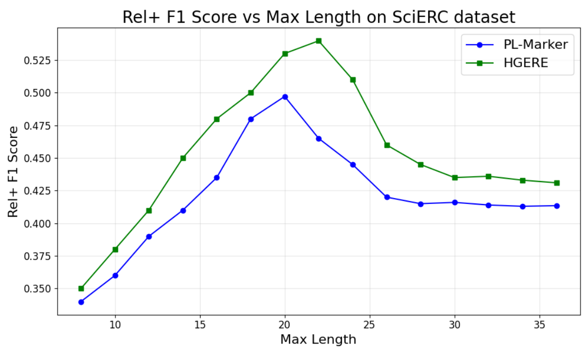

Figure 3: Ablation study: Relation of Rel+ F1 score while varying DP max length threshold on PL-Marker and HGERE models using SciERC dataset.

tree, which explicitly models the grammatical relationships (e.g., subject-object) that underpin most relational facts. This linguistic precision makes it a far more effective filter for reducing false positives while retaining true relations, ultimately leading to a substantial performance gain.

4.5 Ablation Study

Figure 3 illustrates the impact of varying the maximum dependency length on the Rel+ F1 score for PL-Marker and HGERE on the SciERC dataset. Performance improves with longer paths, peaking at length 20 for PL-Marker and 24 for HGERE, after which it declines and stabilizes near baseline levels. At longer lengths, filtering becomes ineffective as few entity pairs are excluded. HGERE shows greater resilience due to its hypergraph neural network, which models higher-order entity-relation interactions.

4.6 Case Study

This section analyzes the superiority of our DP-based distance over linear token distance for filtering entity pairs. We first evaluated a baseline filter using linear distance, measured as the number of tokens between entity spans. On the SciERC dataset, this approach yielded Rel+ F1 scores of 44.3% for PL-Marker and 44.9% for HGERE. While an improvement over the original baselines, this falls significantly short of our DP-based method. The limitation of linear distance is its inability to capture semantically meaningful but non-adjacent relationships, a challenge our syntax-aware method effectively overcomes.

A further data-driven motivation for DP-based filtering stems from the structural properties of scientific texts. As shown in Fig. 1, sentences in SciERC are, on average, significantly longer than those in general-domain datasets like ACE05. This leads to a quadratic increase in the number of possible entity pairs, escalating computational costs and injecting a high volume of irrelevant candidates that complicate relation extraction. However, a comparison of sentence lengths (Fig. 1) and the dependency path lengths of true relations (Fig. 2) reveals a crucial pattern: despite long sentences, the syntactic path between related entities is consistently short. Our DP-based filtering method capitalizes on this observation to prune a large number of syntactically implausible pairs early in the pipeline, thereby improving relation classification by significantly reducing false positives.

5 Conclusion

The integration of dependency parsing-based filtering into pipeline architectures marks a significant advancement in entity and relation extraction for scientific texts. By leveraging syntactic structures to prune irrelevant entity pairs, our approach enhances Rel+ F1 scores of models like PL-Marker and HGERE. This method's simplicity, robustness, and domain-generalizability make it a versatile solution for complex ERE tasks across diverse datasets. As scientific literature continues to grow in volume and complexity, our syntactic enhancement offers a scalable framework to improve knowledge extraction, fostering the development of more accurate knowledge graphs and advanced NLP applications.

6 Appendices

Limitations

While our dependency parsing-based filtering approach demonstrates significant improvements in relation extraction performance, particularly on complex scientific datasets like SciERC, several limitations should be acknowledged.

- The effectiveness of the filtering relies heavily on the quality of the dependency parser. We used spaCy's `en_core_web` based models, trained on general English text, but scientific texts may exhibit distinct syntactic structures, potentially reducing parsing accuracy.

- The dependency path length threshold (δ) which is used for inference, was optimized for each dataset; however, determining an optimal δ for new or diverse datasets may require additional tuning, limiting out-of-the-box applicability. For more information on hyperparameter tuning, refer Appendix A.4.

- Additionally, the approach's effectiveness across varied scientific domains or non-English languages remains untested, as it assumes syntactic proximity correlates with semantic relations, a premise that may not hold universally across languages and domains.

- The computational overhead of dependency parsing, though minimal, could also pose challenges for large-scale applications.

- Finally, our evaluation focused on pipeline architectures (PL-Marker and HGERE), leaving its integration with non-pipeline models unexplored.

Future work could address these by fine-tuning parsers on domain-specific data, incorporating syntactic information during training, and testing broader applicability.

Use of AI Assistants

We acknowledge the use of ChatGPT (OpenAI) and Grammarly to enhance the clarity, coherence, and grammatical correctness of our manuscript.

References

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905, Florence, Italy. Association for Computational Linguistics.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 551–560, Portland, Oregon, USA. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhijiang Guo, Yan Zhang, and Wei Lu. 2019. Attention guided graph convolutional networks for relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 241–251, Florence, Italy. Association for Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spacy: Industrial-strength natural language processing in python. *Zenodo*.

Arzoo Katiyar and Claire Cardie. 2017. Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 917–928, Vancouver, Canada. Association for Computational Linguistics.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on*

Empirical Methods in Natural Language Processing, pages 188–197, Copenhagen, Denmark. 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.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany. Association for Computational Linguistics.

Truc-Vien T. Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. 2009. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1378–1387, Singapore. Association for Computational Linguistics.

Yuanhe Tian, Guimin Chen, Yan Song, and Xiang Wan. 2021. Dependency-driven relation extraction with attentive graph convolutional networks. 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 4458–4471, Online. Association for Computational Linguistics.

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

Zhaohui Yan, Songlin Yang, Wei Liu, and Kewei Tu. 2023. Joint entity and relation extraction with span pruning and hypergraph neural networks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7512–7526, Singapore. Association for Computational Linguistics.

Deming Ye, Yankai Lin, Peng Li, and Maosong Sun. 2022. Packed levitated marker for entity and relation extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4904–4917, Dublin, Ireland. Association for Computational Linguistics.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106.

Min Zhang, Jie Zhang, and Jian Su. 2006. Exploring syntactic features for relation extraction using a convolution tree kernel. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 288–295, New York City, USA. Association for Computational Linguistics.

Qi Zhang, Zhijia Chen, Huitong Pan, Cornelia Caragea, Longin Jan Latecki, and Eduard Dragut. 2024. SciER: An entity and relation extraction dataset for datasets, methods, and tasks in scientific documents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13083–13100, Miami, Florida, USA. Association for Computational Linguistics.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215, Brussels, Belgium. Association for Computational Linguistics.

A Appendix

A.1 Dataset Information

	ACE2005	SciERC	SciER
#Entity Types	7	6	3
#Relation Types	6	7	9
#Entities	35706	8089	24518
#Relations	6684	4716	12083
#Docs	464	500	106
#Relations/Doc	15.2	9.4	114.0

Table 3: Properties of datasets used in experimentation

A.2 Handling Edge Cases

Several edge cases are addressed to ensure robustness:

1. **Invalid Spans**: If an entity span is empty or out of bounds (e.g., due to tokenization mismatches), spaCy may fail to assign a valid `span.root`. In such cases, the pair is retained to avoid discarding potentially valid relations, and a warning is logged (e.g., for subword index errors).

2. **Parser Errors**: If spaCy fails to parse a sentence or assigns an incorrect head token, the pair is retained to prevent false negatives. This is particularly relevant for scientific texts, which may deviate from the general English syntax of the `en_core_web_md` training data.

A.3 Model Parameters

We retain the original configurations of both PL-Marker and HGERE when integrating our dependency parsing module. All dataset-specific parameters, defined in the `shells` directory by the authors of HGERE[1] and the `scripts` directory in original PL-Marker[2], remain unchanged throughout our

[1]`https://github.com/yanzhh/HGERE`
[2]`https://github.com/thunlp/PL-Marker`

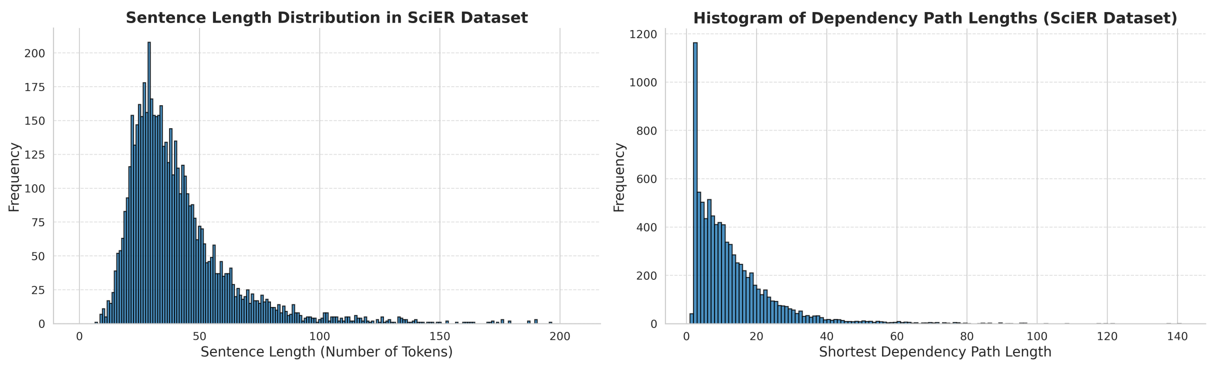

Figure 4: Left: Histogram of sentence lengths in the SciER dataset. Right: Histogram of DP distances between entities related in the ground truth of the SciER dataset.

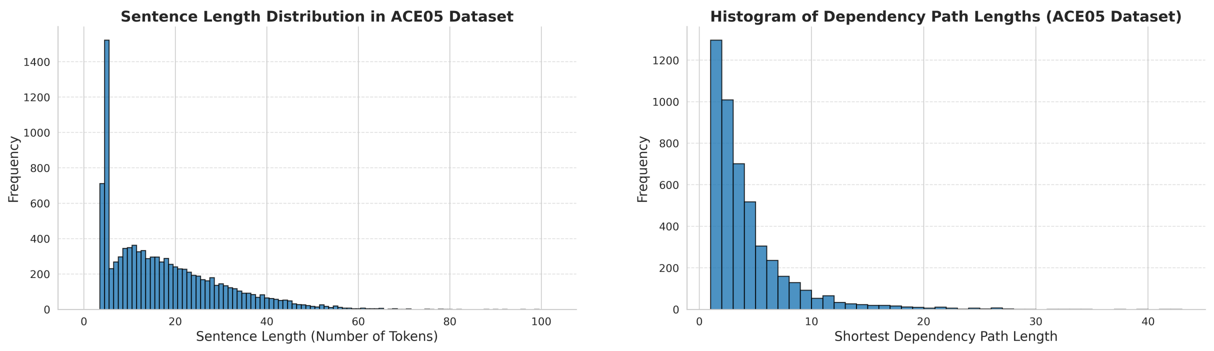

Figure 5: Left: Histogram of sentence lengths in the ACE05 dataset. Right: Histogram of DP distances between entities related in the ground truth of the ACE05 dataset.

experiments. The pre-trained models used in our experiments such as DP models - `en_core_web_md` and `en_core_web_lg` as well as the SciBERT encoder uses default parameters provided by their original python library.

A.3.1 Model Size

The core architecture and size of the PL-Marker and HGERE models are unmodified. The only addition is the integration of the spaCy dependency parser model `en_core_web_md`, which consists of approximately 20M parameters and `en_core_web_lg` consisting of 210M parameters, are in the same magnitude as that of the original PL-Marker and HGERE models.

A.4 Hyperparameter Tuning

Our approach introduces only a single hyperparameter, which defines the maximum allowable dependency path length between entity pairs. To determine its optimal value, we perform a grid search over the set {8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36} across all three datasets. Since dependency parsing is performed once and filtering is applied only at inference time, we cache the parse trees and recompute only the candidate entity pairs for each path length threshold. This results in a highly efficient search process with minimal computational overhead. Despite its simplicity, this tuning procedure is sufficient to achieve peak Rel+ F1 scores, underscoring the effectiveness and practicality of our method.

A.5 GPU Training Time

Total training time (hours) of GPU required for each experiment is provided in Table 5.

A.6 Theoretical Analysis

Our filtering method's effectiveness is grounded in linguistic theory, specifically in how dependency parsing is uniquely suited to handle the complex syntactic structures common in scientific texts. Scientific writing often obscures direct relationships through *long-distance dependencies* (where related entities are separated by many words), *nominalizations* (where actions are expressed as nouns, like "the *inhibition* of X"), and frequent use of the *passive voice*. A simple linear distance metric fails in these cases, but dependency grammar excels because it models grammatical relationships rather

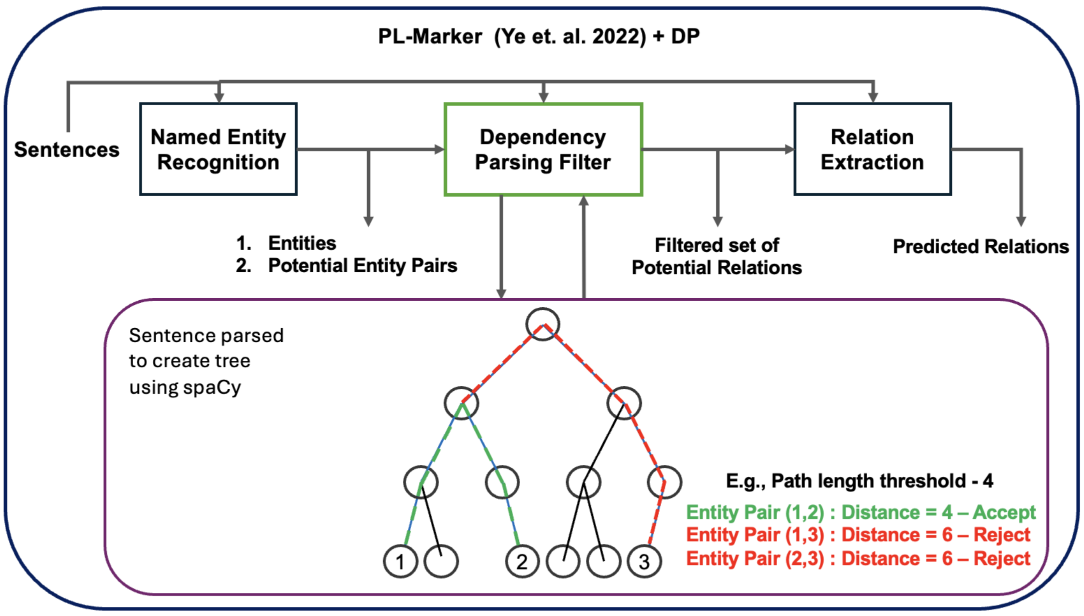
PL-Marker (Ye et. al. 2022) + DP
Sentences
Named Entity Recognition
Dependency Parsing Filter
Relation Extraction
1. Entities
2. Potential Entity Pairs
Filtered set of Potential Relations
Predicted Relations
Sentence parsed to create tree using spaCy
E.g., Path length threshold - 4
Entity Pair (1,2) : Distance = 4 – Accept
Entity Pair (1,3) : Distance = 6 – Reject
Entity Pair (2,3) : Distance = 6 – Reject
1
2
3

Figure 6: Visual pipeline for understanding PL-Marker+DP model

than token position. It correctly identifies the syntactic heads of entities, creating a direct path between them that bypasses intervening modifiers and correctly interprets grammatical roles even when word order is inverted, such as through relations like `nsubj:pass`.

Ultimately, this means a short dependency path is a robust and theoretically sound indicator of a true relation, a principle that does not apply to linear distance. Our DP-based filter is therefore not just a heuristic; it is a direct application of linguistic theory. It imposes a strong *inductive bias* tailored to the predicate-argument structure inherent in language. This is precisely what allows it to effectively prune the noisy, syntactically implausible candidate pairs that confuse modern neural models, which often lack this explicit syntactic awareness.

A.7 Selection of DP Model

We evaluated all variants of spaCy's `en_core_web` models at inference, such as `small`, `medium`, and `large`, which progressively increase in the number of parameters. For the SciERC and ACE05 datasets, the medium-sized model yielded the best performance with the HGERE baseline. In contrast, the large model performed best on the SciER dataset, likely due to the greater complexity and longer sentence lengths present in the dataset (Refer Table 4 and Figure 4).

Table 4: Performance of HGERE with different sizes of the general-domain `en_core_web` parser. Best scores for each dataset are in bold.

Dataset	Small	Medium	Large
SciERC	62.11	**64.59**	64.28
SciER	51.57	53.62	**54.97**
ACE05	69.93	**71.02**	70.86

A.8 Device Configuration

The computational experiments in this study were conducted on a high-performance computing system equipped with an NVIDIA Tesla A30 GPU, with Driver Version 556.123 and CUDA Driver Version 12.4. The system features a GPU with a memory capacity of 24.00 GiB.

A.9 Reproducibility and Fairness

All the experiments were conducted using three different random seeds similar to the codebase of PL-Marker and HGERE. The results provided in Table 1 and Figures 3 are the mean performance over these random seeds to ensure fairness. The results can be reproduced with a minimal change to the initialization and evaluate functions present in the original code repositories of PL-Marker[3] and HGERE[4], while adding another function for the distance calculation mechanism.

[3]`https://github.com/thunlp/PL-Marker`
[4]`https://github.com/yanzhh/HGERE`

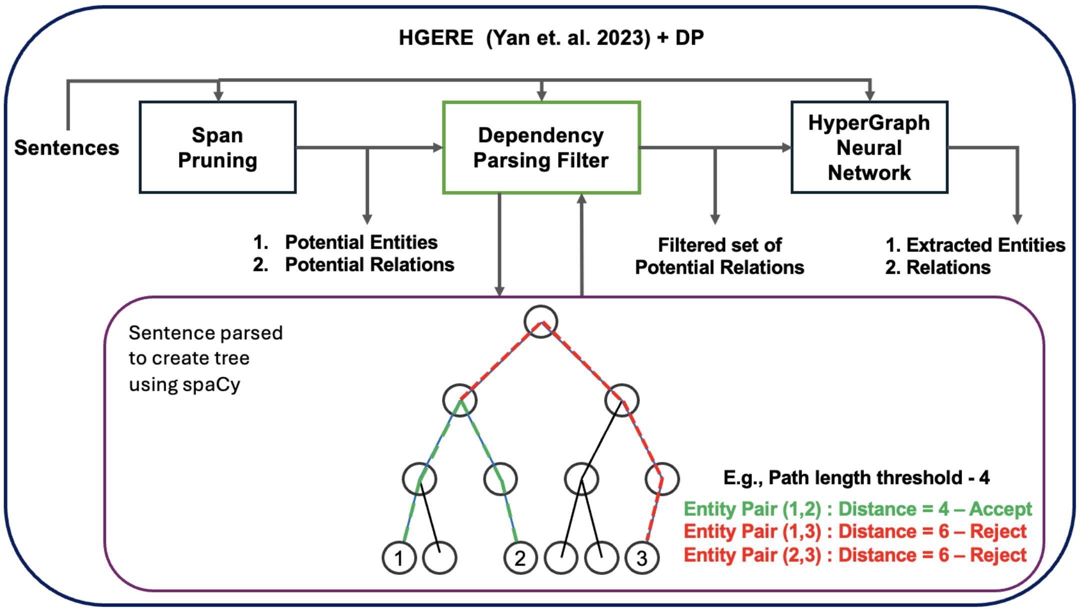
HGERE (Yan et. al. 2023) + DP
Sentences
Span Pruning
Dependency Parsing Filter
HyperGraph Neural Network
1. Potential Entities
2. Potential Relations
Filtered set of Potential Relations
1. Extracted Entities
2. Relations
Sentence parsed to create tree using spaCy
E.g., Path length threshold - 4
Entity Pair (1,2) : Distance = 4 – Accept
Entity Pair (1,3) : Distance = 6 – Reject
Entity Pair (2,3) : Distance = 6 – Reject
1
2
3

Figure 7: Visual pipeline for understanding HGERE+DP model

Base Model	Dataset	NER Stage		RE Stage		Seeds	DP Infer Time (hrs)	Total (hrs)
		Epochs	Time (hrs)	Epochs	Time (hrs)			
PL-Marker	SciERC	20	5.4	10	0.5	3	<0.01	17.7
	SciER	20	8.1	10	1.9	3	0.03	30
	ACE05	10	3.1	10	0.8	3	<0.01	11.7
HGERE	SciERC	8	2.7	30	5.1	3	<0.01	23.4
	SciER	8	4.9	30	9.5	3	0.03	43.2
	ACE05	5	2.5	15	4.8	3	<0.01	21.9

Table 5: Total GPU time used for experiments. "Epochs" refer to the training iterations per stage. "NER Time" and "RE Time" are total training durations for Named Entity Recognition and Relation Extraction stages respectively. "DP Infer Time" is the preprocessing time at inference for dependency parsing. "Total Time" is calculated as `(NER Time + RE Time + DP Infer Time) × Seeds`.