SOMD 2025: Fine-tuning ModernBERT for In- and Out-of-Distribution NER and Relation Extraction of Software Mentions in Scientific Texts

Projan Shakya*[1], Kristina Ghimire*[1], Kashish Bataju*[1], Ashwini Mandal*[1], Sadikshya Gyawali*[1], Manish Dahal*[1], Manish Awale*[1], Shital Adhikari [1,2], Sanjay Rijal[1,3], Vaghawan Ojha*[1]

[1]E.K. Solutions Pvt. Ltd., Nepal, [2]Steven Insititute of Technology, USA
[3]Institut de Física d'Altes Energies (IFAE), Spain

Correspondence: vaghawan.ojha@ekbana.net

Abstract

Software mentions are ubiquitous yet remains irregularly referenced among scientific texts. In this paper, we utilized the dataset and evaluation criteria defined by Software Mention Detection (SOMD 2025) competition to solve the problem of Named Entity Recognition (NER) and Relation Extraction (RE) in input sentences from scientific texts. During the competition, we achieved a leading F1 SOMD score of 0.89 in Phase I by first fine-tuning ModernBERT for NER, and then using the extracted entity pairs for RE. Additionally, we trained a model that jointly optimizes entity and relation losses, leading to an improvement in F1 SOMD score to 0.92. Retraining the same model on an augmented dataset, we achieved the second best F1 SOMD score of 0.55 in Phase II. In the Open Submission phase, we experimented with adapative fine-tuning, achieving an F1 SOMD score of 0.6, with the best macro average for NER being 0.69. Our work shows the efficiency of fine-tuning a niche task like software mention detection despite having limited data and the promise of adaptive fine-tuning on Out of Distribution (OOD) dataset.

Keywords: *BERT, ModernBERT, Named Entity Recognition, Relation Extraction, SciDeBERTa, SOMD*

1 Introduction

Software is an integral part of scientific research and scholarly documents (Allen et al., 2018; Du et al., 2021). Despite its significance, software is often referenced informally rather than being formally cited in academic publications (Schindler et al., 2021). Automatically identifying these inconsistencies in mentioning software and their associated attributes can enhance the transparency, accessibility, and reproducibility of scientific research (Schindler et al., 2021). The Software Mention Detection 2025 (Sharmila et al.) challenge aimed to advance the development of robust models that can jointly detect software entities and classify their relations in scholarly documents. The goal of this competition was to automatically extract structured information about software used in scientific research, improving understanding and reproducibility. Named Entity Recognition (NER) and Relation Extraction (RE) are two of the most important classical problems in Natural Language Processing (NLP) (Nasar et al., 2021). Since relation depends on the inherent named entity, these two problems had been posed as a joint tasks in the challenge (Sharmila et al.). The SOMD 2025 shared task aimed to tackle this problem by offering two phases as below:

- Phase I: This task involved identifying 14 types of software-related entities in BIO format [1] and 11 types of relations between those entities.
- Phase II: This task aimed to extract the entities and relation with OOD [2] test set. A new evaluation dataset was provided, introducing a data distribution shift between Phase I and Phase II. The objective of this phase was to adapt the model trained on Phase I to an OOD test set.

In this paper, we experiment with various architectures and identify the most effective ones for the joint extraction of named entities and their relations. We focus on approaches that not only perform well in the given tasks but could also generalize to similar joint extraction problems in the future.

*All authors contributed equally

[1]BIO format - B for the beginning of the entity, I for the inside of the entity, and O for non-entity tokens.

[2]OOD - Out of Distribution

Proceedings of the Fifth Workshop on Scholarly Document Processing (SDP 2025), pages 154–163

To this end, we fine-tuned encoder-only transformer models and evaluated their performance. In Phase I, our best-performing model was optimized by first performing NER through fine-tuning ModernBERT (Warner et al., 2024), followed by identifying relationships between the extracted entities. A different model trained after Phase I, referred to as the Modified Joint Model (see Section 4.1.4), later surpassed the performance of the previously best-performing model.

In Phase II, we augmented the training data using Large Language Model (LLM) generated examples and retrained the Modified Joint Model. During the Open Submission phase, we applied Adaptive fine-tuning (Ruder, 2021), which led to improvement in scores obtained during Phase II.

All relevant code and implementations have been made publicly available on GitHub[3] for reproducibility and further research.

2 Related Studies

NER and RE are key tasks in information extraction, with several models and datasets developed to improve their performance (Giorgi et al., 2019; Markus and Adrian, 2020; Nasar et al., 2021; Shang et al., 2022; Wang et al., 2022).

(Devlin et al., 2019) introduced BERT, an encoder-only transformer model, which became a standard for extracting contextual embeddings in NLP tasks. Building on BERT, DYGIE++ proposed a dynamic span graph approach for joint NER, RE, and event extraction, achieving F1 scores of 67.5% for NER and 48.4% for RE on the SciERC dataset. (Markus and Adrian, 2020) presented SpERT, an attention-based model for span-based joint entity and relation extraction, which achieved F1 scores of 70.33% for entity recognition and 50.84% for relation extraction on SciERC using SciBERT. (Huguet Cabot and Navigli, 2021) introduced REBEL, a sequence-to-sequence model based on BART, which achieved an F1 score of 92.02% for RE tasks on the NYT dataset by linearizing triplets into text sequences.

(Shang et al., 2022) proposed OneRel, a joint entity and relation extraction model, which treats the task as a triple classification problem and achieved an F1 score of 92.9% on the NYT dataset. (Hennen et al., 2024) introduced ITER, an encoder-based model utilizing FLAN-T5, which performed NER and RE in three parallelizable steps, achieving F1 scores of 91.7% for NER and 71.9% for RE on ACE05.

ModernBERT was introduced as a state-of-the-art encoder-only model, optimizing BERT with modern enhancements (Warner et al., 2024). Trained on 2 trillion tokens, it showed significant improvements across various domains, though its application in NER and RE remained unexplored. (Jeong and Kim, 2022) explored SciDeBERTa, a model specialized for scientific text, in SOMD to extract entities and relations with domain-aware contextual embeddings.

In SOMD 2024, A three-stage framework based on XLM-RoBERTa achieved a macro-averaged F1 score of 67.8%, ranking third in Sub-task I (NER), with steps for entity classification, extraction, and type classification (Thi et al., 2024).

In this paper, we experiment with ModernBERT and SciDeBERTa for generating contextual embedding in a joint model with an added fine-tuning layers for NER and RE, aiming to enhance the state of art performance specifically in software mention detection tasks.

3 Dataset Description

Phase-I

The given train dataset (Schindler et al., 2021) consisted of 1149 sentences with their corresponding entity and relation labels, while the test set (in-distribution) had 203 sentences. The class distributions of entities and relations in the SOMD 2025 train dataset are illustrated in Figure 1 and Figure 2 respectively.

Phase II

The test dataset for Phase II consisted of 220 OOD sentences without any entity or relation labels. To enhance model performance on the Phase II OOD test set, we augmented the Phase I train dataset by using Large Language Models (LLMs). The augmentation process is explained below:

Dataset Augmentation

We began by identifying Out-of-Vocabulary (OOV) entities from the Phase II OOD test set. These included version patterns (e.g., spaced formats like `v 2 . 2 . 1`), citation formats (e.g., `[ 5 ]`, `[ 4 , 7 ]`), application names, and other uncommon expressions. These entities introduced unfamiliar vocabulary, contributing to the distribution shift in OOD test set.

[3] https://github.com/ekbanasolutions/somd-2025

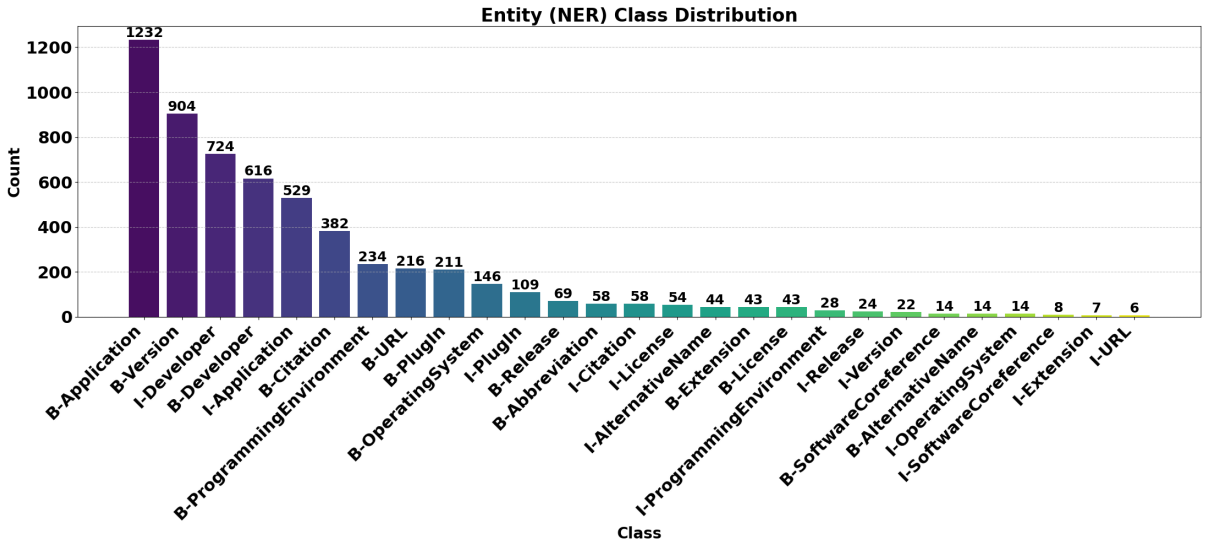

Figure 1: Class Distribution of Entities in the SOMD 2025 Training Dataset

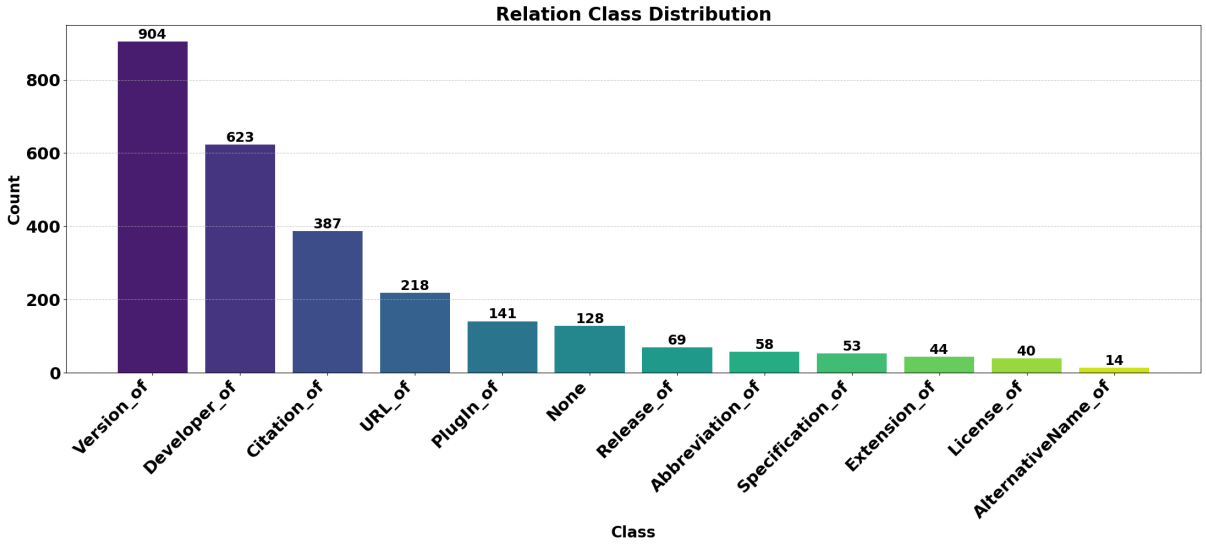

Figure 2: Class Distribution of Relations in the SOMD 2025 Training Dataset

Using the extracted keywords, we used `GPT-4o`, to generate natural language sentences. The sentence generation was guided by a carefully written prompt to ensure consistency and alignment with the task requirements. The prompt used to generate the sentences is shown in Appendix A.

The model was instructed to strictly maintain the mentioned formats. The generated sentences were then passed through our best-performing model after Phase I (Modified Joint Model, detailed in Section 4.1.4) to automatically assign entity and relation labels. These predictions were manually verified to correct missing or incorrect annotations and to filter out invalid samples. We refer to the curated dataset as LLM-Joint-Augmented dataset.

4 Methods

The methodologies employed in Phase I, Phase II and the Open Submission track are detailed in the following sections. For each phase, multiple methods are presented to reflect the step-by-step progression of our approach. These variations reflect iterative improvements made based on experimental results, showing the gradual development of our approach.

4.1 Phase I

In this phase, we evaluated two pre-trained language models: ModernBERT-base (Warner et al., 2024) and SciDeBERTa-cs (Jeong and Kim, 2022) to determine the most suitable base model for our

downstream tasks. Based on experimental results (see Table 1), ModernBERT-base performed better and was chosen for the further fine-tuning.

4.1.1 NER Fine-tuning

We fine-tuned ModernBERT-base on the SOMD 2025 training dataset (Schindler et al., 2021), as detailed in Section 3, which was randomly split into 80% for training and 20% for development. The official SOMD 2025 test set was reserved for final evaluation. To adapt ModernBERT-base for Named Entity Recognition (NER), we appended task-specific layers to pre-trained ModernBERT-base. These layers were initialized using Xavier initialization to ensure stable convergence. We employed a hidden dropout rate of 0.4 during training, alongside an attention dropout rate of 0.3 to regularize the self-attention mechanism. The ReLU activation function was used in the additional layers and also a dropout layer with a rate of 0.3 was included for further regularization. This setup is referred to as the EntityModel architecture, illustrated in Figure 3.

4.1.2 RE Fine-tuning

After extracting the entities using the EntityModel, we performed Relation Extraction (RE) based on the identified entity pairs. The RelationModel is detailed in Figure 3. Hidden state was passed from the final layer of the EntityModel. To stabilize and accelerate the training process, we also applied Layer Normalization and Multi-Head Attention (MHA) was incorporated to capture the contextualized relationships between the entities. Finally, the [CLS] token, which summarizes the entire sentence, was concatenated with the final output of the contextualized entity pair. GELU was used as the activation function. To prevent overfitting, a dropout rate of 0.5 was used in the Feed Forward layers, while an attention dropout of 0.1 was applied to the attention heads. Additionally, weight decay of 0.1 was added to make the model more robust and prevent overfitting since our dataset consisted of limited amount of data. Linear learning rate scheduler was used with the initial learning rate being $5e-5$.

4.1.3 Few-Shot Prompting

We performed few-shot prompting for RE using GPT-4o, providing few carefully constructed examples. The model was instructed to follow a clear, step-by-step procedure to extract relations based solely on entities predicted by our fine-tuned NER

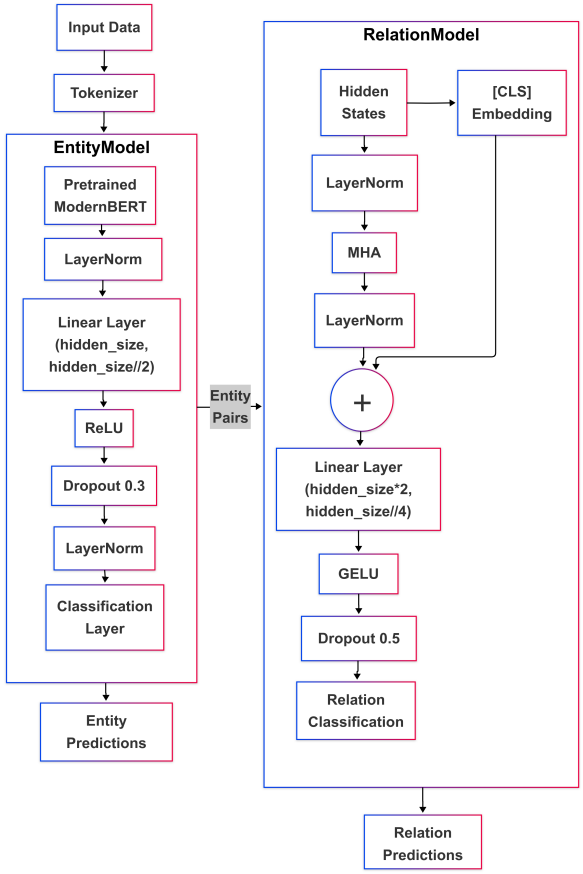

Figure 3: Model Architecture used in Phase I for NER and RE Fine-tuning

model detailed in Section 4.1.1. The model was explicitly instructed not to hallucinate any new relation classes.

4.1.4 Modified Joint Model (Post-Phase I)

After completing Phase I, we further experimented with a new architecture, which we refer to as the Modified Joint Model, designed to extract both entities and their relations.

For the entity extraction, we reused the same EntityModel, initializing it with the weights from our previously fine-tuned EntityModel in Section 4.1.1. To enable RE, we added additional layers on top of the EntityModel. Unlike the setup in Section 4.1.2, the RelationModel here followed a different architecture as shown in detail in Figure 4.

For relation extraction, we identified all non-"O" entity pairs predicted by EntityModel and concatenated their hidden states with the [CLS] token embedding to capture both local and global context. This combined representation was passed through two linear layers with ReLU activation and dropout of 0.4 for classification. Entity and relation losses were computed separately and summed to jointly

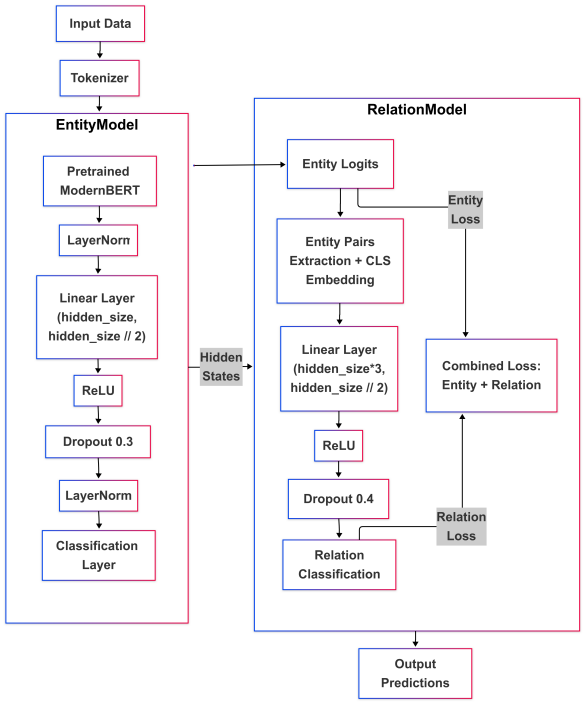

Figure 4: Model Architecture for Modified Joint Model

optimize both tasks in an end-to-end manner.

4.2 Phase II

In Phase II, we first used the trained Modified Joint Model detailed in Section 4.1.4, to infer on the provided OOD test data. In order to improve the score, we fine-tuned the ModernBERT-base model using a Masked Language Modeling (MLM) approach. The training process used the LLM-Joint-Augmented dataset as described in Section 3, by splitting it randomly into 80% training and 20% evaluation sets. Sentences were tokenized with a maximum length of 128 tokens and a masking probability of 15%. The model was trained for up to 60 epochs with early stopping (patience of 3) to prevent overfitting. The training configuration used a learning rate of $4e-4$ with weight decay of 0.01, batch sizes of 16, and FP16 precision for performance optimization. Then, for better generalization of OOD data provided in Phase II, we applied multiple approaches. The results from these approaches are presented in Table 2. Below we discuss some of the major approaches which improved the F1 scores.

4.2.1 Fine-tuning with Augmented Data

In this approach, we retrained our best-performing model (the Modified Joint Model, shown in Figure 4) using the MLM-pretrained ModernBERT-base on the LLM-Joint-Augmented dataset.

4.2.2 Separated Relation Classification

In this approach, we split relation classification into binary and multi-class tasks to reduce false positives. Two linear layers were added atop the Modified Joint Model: one used BinaryCrossEntropy to detect relation presence, and the other used CrossEntropy for relation type classification.

4.3 SOMD 2025 Open Submission

4.3.1 Adaptive fine-tuning

To perform better on the OOD evaluation set, we used Adaptive fine-tuning (Ruder, 2021) on the LLM-Joint-Augmented dataset, by adding three linear layers and a residual connection to ModernBERT-base in order to learn new features while preserving prior knowledge.

4.3.2 Fine-tuning ModernBERT

Given the limited improvements achieved with the previous approach, we fine-tuned ModernBERT-base from scratch (up to 20 layers) on the LLM-Joint-Augmented dataset. This resulted in a significant boost in NER score, increasing the F1 score by approximately 0.07, even without the use of adaptive layers. This yielded the highest NER F1 score among all participants and also led to improvements in the RE F1 score.

4.3.3 Post fine-tune Adaptation

Since fine-tuning ModernBERT up to layer 20 (Section 4.3.2) resulted in a significant improvement in NER performance, we incorporated the adaptive layer on top of the fine-tuned model. The adaptive layer followed the architecture described in Section 4.3.1. We experimented with various learning rates, batch sizes, and other hyperparameters to optimize performance. This overall approach is illustrated in Figure 5.

5 Results

After fine-tuning ModernBERT-base (Warner et al., 2024) on the SOMD 2025 dataset, we achieved an NER F1 score of 0.93. Another approach was fine-tuning SciDeBERTa-cs (Jeong and Kim, 2022), specifically at the 10^{th} layer, which resulted an NER F1 score of 0.89. This comparative experiment made us more inclined towards ModernBERT-base for further experiments. Building on the fine-tuned ModernBERT-base model, we experimented with various architectures for RE, achieving an RE F1 score of 0.84, that resulted in an F1 SOMD score of 0.89 in Phase I. Notably, our second model

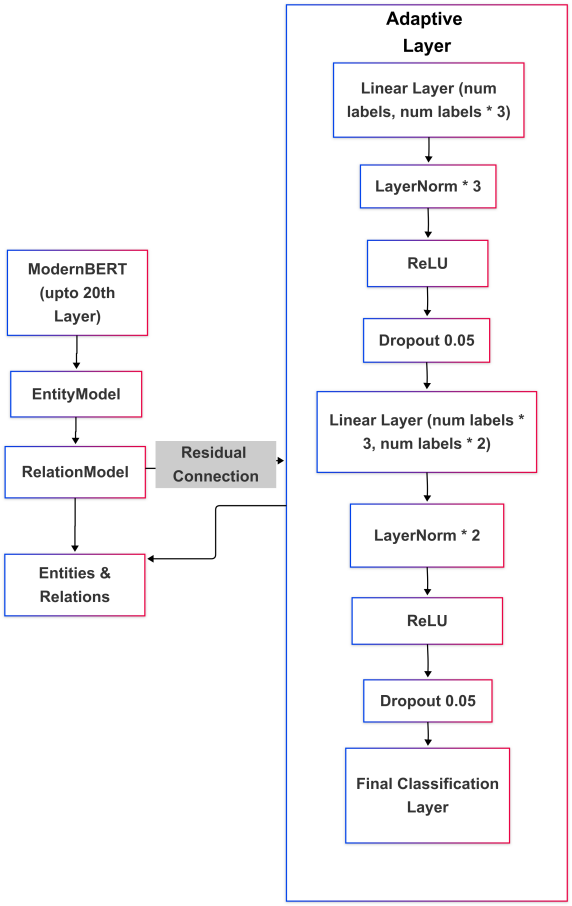

Figure 5: Model Architecture used in Section 4.3.3

(Modified Joint Model as shown in Figure 4), surpassed the previous approaches, achieving a higher overall F1 score of 0.92, as shown in Table 1. This result, however, was achieved after Phase I ended.

S.N.	Approach	NER	RE	F1 SOMD
1. NER fine-tuning (Section 4.1.1)				
	ModernBERT	0.93	-	-
	SciDeBERTa	0.89	-	-
2. RE fine-tuning (Section 4.1.2)				
	Baseline	0.93	0.80	0.87
	+ LayerNorm	0.93	0.80	0.87
	+ MultiHeadAttention	0.93	0.83	0.88
	+ GELU	0.93	**0.84**	**0.89**
3. Few-Shot Prompting (Section 4.1.3)				
	GPT-4o	-	0.38	-
4. Modified Joint Model (Section 4.1.4)				
	ModernBERT	**0.95**	**0.89**	**0.92**

Table 1: SOMD score for each approach of Phase I

In Phase II, we achieved SOMD score of 0.55 by fine-tuning the Modified Joint Model using the LLM-Joint-Augmented dataset, which was ranked the second-best in SOMD 2025. We obtained macro average F1 score of 0.64 for NER and 0.46 for RE. The precision, recall and F1 score for each entity and relation class in the test set of Phase I and Phase II are shown in Table 4 and Table 5.

After the phase was completed, we experimented with multiple approaches which resulted in improvement of the F1 score as mentioned in the Table 2. In open submission, we experimented with adaptive learning and further fine-tuning the model which resulted in an F1 SOMD score of 0.6 as explained in Section 4.3.3.

Approach	F1 SOMD
Phase II Submission	
Using Phase I Modified Joint Model (Section 4.1.4)	0.51
Fine-tuning with Augmented Data (Section 4.2.1)	**0.55**
Separated Relation Classification (Section 4.2.2)	0.54
Open Submission	
Adaptive Learning (Section 4.3.1)	0.57
Fine-tuning ModernBERT (Section 4.3.2)	0.58
Post fine-tune Adaptation (Section 4.3.3)	**0.60**

Table 2: SOMD F1 scores for different approaches in Phase II and Open Submission

All the macro F1 score for NER and RE are computed using exact match via seqeval library (Nakayama, 2018).

Limitations

The model demonstrated poor generalization to OOD dataset, as seen when transitioning from Phase I to Phase II of the SOMD 2025 dataset. A key challenge inherent to the task was heavy reliance of RE on accurate NER; thus, errors in NER propagated to RE, resulting in incorrect relation pairings. Another limitation was the class imbalance in the dataset, as illustrated in Figures 1 and 2. Furthermore, the entity hidden states along with the [CLS] token representation was passed through several layers for RE, which made the early errors have even bigger impact later. A notable limitation was also the unavailability of large corpus of sentences to fine-tune the model.

Discussion and Conclusion

Compared to the results reported in SOMD 2025[4], our model achieved the highest F1 score of 0.89 in Phase I, outperforming the *TU Graz Data Team*, which scored 0.88, and *gabrielrsilva*, who scored 0.39. Thus, our model attained the top performance among all participants in this phase. In Phase II, our model achieved an F1 score of 0.55, ranking second overall. The highest score in this phase was

[4]SOMD 2025 Results: https://www.codabench.org/competitions/5840/#/results-tab

Phase	F1 SOMD	NER (macro avg)			RE (macro avg)		
		F1	Precision	Recall	F1	Precision	Recall
Phase I	0.89	0.93	0.93	0.95	0.84	0.85	0.86
Post-Phase I (Modified Joint Model)	0.92	0.95	0.95	0.96	0.89	0.95	0.85
Phase II	0.55	0.64	0.67	0.65	0.46	0.69	0.39
Open Submission	0.60	0.69	0.74	0.69	0.51	0.71	0.42

Table 3: SOMD results for NER and RE in each phase

Entity	Phase I				Phase II			
	Precision	Recall	F1 score	Support	Precision	Recall	F1 score	Support
Abbreviation	1.0000	0.7500	0.8571	4	0.9000	0.7500	0.8182	12
AlternativeName	1.0000	1.0000	1.0000	2	1.0000	0.4118	0.5833	17
Application	0.9283	0.9539	0.9409	217	0.7120	0.7218	0.7168	363
Citation	1.0000	0.9811	0.9905	53	0.8744	0.9679	0.9188	187
Developer	0.9762	0.9840	0.9801	125	0.5882	0.5000	0.5405	20
Extension	0.8571	0.8571	0.8571	7	0.0000	0.0000	0.0000	6
License	1.0000	1.0000	1.0000	7	-	-	-	-
OperatingSystem	1.0000	1.0000	1.0000	22	1.0000	1.0000	1.0000	2
PlugIn	0.8750	0.8235	0.8485	34	0.3784	0.7000	0.4912	20
ProgrammingEnvironment	0.9474	0.9730	0.9600	37	0.9000	0.7500	0.8182	24
Release	0.9286	1.0000	0.9630	13	0.7000	0.7000	0.7000	10
SoftwareCoreference	0.5000	1.0000	0.6667	1	0.0000	0.0000	0.0000	3
URL	1.0000	1.0000	1.0000	32	1.0000	1.0000	1.0000	70
Version	0.9879	0.9702	0.9790	168	0.6911	0.8854	0.7763	96
Micro Avg	0.9586	0.9626	0.9606	722	0.7626	0.8012	0.7814	830
Macro Avg	0.9286	0.9495	0.9316	722	0.6726	0.6451	0.6433	830
Weighted Avg	0.9595	0.9626	0.9607	722	0.7663	0.8012	0.7778	830

Table 4: Precision, Recall, F1 score, and Support of Each Entity Class in Test Dataset of Phase I and Phase II

Relation	Phase I				Phase II			
	Precision	Recall	F1 score	Support	Precision	Recall	F1 score	Support
Developer_of	0.8722	0.9206	0.8958	126	0.8889	0.4000	0.5517	20
Citation_of	0.8276	0.9057	0.8649	53	0.6761	0.5134	0.5836	187
Version_of	0.8378	0.9226	0.8782	168	0.8600	0.4479	0.5890	96
PlugIn_of	0.9524	0.8000	0.8696	25	0.5882	0.7692	0.6667	13
URL_of	0.8286	0.9062	0.8657	32	0.4286	0.3000	0.3529	70
License_of	0.8571	0.8571	0.8571	7	0.0000	0.0000	0.0000	0
AlternativeName_of	1.0000	1.0000	1.0000	2	1.0000	0.1176	0.2105	17
Release_of	0.6667	0.9231	0.7742	13	1.0000	0.3000	0.4615	10
Abbreviation_of	1.0000	0.5000	0.6667	4	0.8000	0.6667	0.7273	12
Extension_of	0.7778	1.0000	0.8750	7	0.0000	0.0000	0.0000	6
Specification_of	0.6667	0.7143	0.6897	14	0.0000	0.0000	0.0000	0
Micro Avg	0.8392	0.9024	0.8697	451	0.6773	0.4432	0.5358	431
Macro Avg	0.8443	0.8591	0.8397	451	0.6935	0.3905	0.4604	431
Weighted Avg	0.8432	0.9024	0.8696	451	0.6984	0.4432	0.5267	431

Table 5: Precision, Recall, F1 score, and Support of Each Relation Class in Test Dataset of Phase I and Phase II

obtained by *TU Graz Data Team*, with scores of 0.63. In the Open Submission track, our model achieved an F1 SOMD score of 0.60, which, at the time of writing, ranked second overall, and attained the highest NER F1 score of 0.69 among all submissions.

Compared to SOMD 2024's results, our model achieved a higher F1 score in the NER task which was at the time divided into Subtask I and Subtask II. The top-performing team in Subtask I last year, *Team phinx*, had an F1 score of 0.74 (Xuan et al., 2024), while *Team ottowg* scored 0.838 (Otto et al., 2024) in Subtask II. Our model outperformed both of these benchmarks. However, in the RE task, our model showed a slight drop in performance. The top-performing team last year, *Team ottowg*, achieved an F1 score of 0.911. It is important to note that their approach (Otto et al., 2024) utilized a

question-answering framework, which constrained the number of candidate entities and explicitly highlighted possible relations, thereby simplifying the task and potentially boosting performance.

Better results could have been achieved by pretraining a Masked Language Model (MLM) on a large corpus. This would have enabled the model to learn richer contextual representations of language, improving its ability to understand sentence structure and semantics. Consequently, leading to more accurate predictions for both entity recognition and relation extraction tasks. Pretraining provides a strong foundation, especially when fine-tuned on task-specific data, as it helps the model generalize better–particularly in low-resource settings or when dealing with unseen vocabulary and complex sentence structures (Zhou et al., 2022; Sonkar et al., 2022).

Since limited research has explored the use of ModernBERT for NER and RE, we initiated our study using it as the foundation. Our model was built on ModernBERT, and further enhanced through prior training on the NER task and adaptive fine-tuning. This combination enabled it to achieve strong performance on both in-domain and OOD datasets. While relation extraction under OOD conditions remains challenging, the results highlight the effectiveness of combining robust pretraining, task-specific fine-tuning, and a joint optimization strategy for software mention understanding.

6 Acknowledgments

We thank E.K. Solutions Pvt. Ltd. (EKbana Nepal) for kindly providing the time and resources needed to participate in this competition. We would also like to extend our gratitude to the SOMD 2025 competition organizers.

References

Alice Allen, Peter J. Teuben, and P. Wesley Ryan. 2018. Schroedinger's code: A preliminary study on research source code availability and link persistence in astrophysics. *The Astrophysical Journal Supplement Series*, 236(1):10.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Caifan Du, Johanna Cohoon, Patrice Lopez, and James Howison. 2021. Softcite dataset: A dataset of software mentions in biomedical and economic research publications. *Journal of the Association for Information Science and Technology*, 72(7):870–884.

John Giorgi, Xindi Wang, Nicola Sahar, Won Young Shin, Gary D. Bader, and Bo Wang. 2019. End-to-end named entity recognition and relation extraction using pre-trained language models. *Preprint*, arXiv:1912.13415.

Moritz Hennen, Florian Babl, and Michaela Geierhos. 2024. ITER: Iterative transformer-based entity recognition and relation extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11209–11223, Miami, Florida, USA. Association for Computational Linguistics.

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. REBEL: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yuna Jeong and Eunhui Kim. 2022. Scideberta: Learning deberta for science technology documents and fine-tuning information extraction tasks. *IEEE Access*, 10:60805–60813.

Eberts Markus and Ulges Adrian. 2020. *Span-Based Joint Entity and Relation Extraction with Transformer Pre-Training*. IOS Press.

Hiroki Nakayama. 2018. seqeval: A python framework for sequence labeling evaluation. Software available from https://github.com/chakki-works/seqeval.

Zara Nasar, Syed Waqar Jaffry, and Muhammad Kamran Malik. 2021. Named entity recognition and relation extraction: State-of-the-art. *ACM Comput. Surv.*, 54(1).

Wolfgang Otto, Sharmila Upadhyaya, and Stefan Dietze. 2024. Enhancing software-related information extraction via single-choice question answering with large language models. In *Proceedings of the Workshop on Natural Scientific Language Processing and Research Knowledge Graphs*, Cologne, Germany. GESIS - Leibniz Institute for the Social Sciences.

Sebastian Ruder. 2021. Recent Advances in Language Model Fine-tuning. `http://ruder.io/recent-advances-lm-fine-tuning`.

David Schindler, Felix Bensmann, Stefan Dietze, and Frank Krüger. 2021. Somesci- a 5 star open data gold standard knowledge graph of software mentions in scientific articles. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 4574–4583, New York, NY, USA. Association for Computing Machinery.

Yu-Ming Shang, Heyan Huang, and Xian-Ling Mao. 2022. Onerel:joint entity and relation extraction with one module in one step. *Preprint*, arXiv:2203.05412.

Upadhyaya Sharmila, Otto Wolfgang, Krueger Frank, and Stefan Dietze. 5th workshop on scholarly document processing.

Shashank Sonkar, Zichao Wang, and Richard G. Baraniuk. 2022. Maner: Mask augmented named entity recognition for extreme low-resource languages. *Preprint*, arXiv:2212.09723.

Thuy Nguyen Thi, Anh Nguyen Viet, Thin Dang Van, and Ngan Nguyen Luu Thuy. 2024. Software mention recognition with a three-stage framework based on bertology models at somd 2024. *Preprint*, arXiv:2405.01575.

Xinyu Wang, Jiong Cai, Yong Jiang, Pengjun Xie, Kewei Tu, and Wei Lu. 2022. Named entity and relation extraction with multi-modal retrieval. *Preprint*, arXiv:2212.01612.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *Preprint*, arXiv:2412.13663.

Phi Nguyen Xuan, Quang Tran Minh, and Thin Dang Van. 2024. ABCD Team at SOMD 2024: Software Mention Detection in Scholarly Publications with Large Language Models. In *Proceedings of the Workshop on Natural Scientific Language Processing and Research Knowledge Graphs (NSLP-KG)*, Ho Chi Minh City, Vietnam. CEUR Workshop Proceedings.

Ran Zhou, Xin Li, Ruidan He, Lidong Bing, Erik Cambria, Luo Si, and Chunyan Miao. 2022. MELM: Data augmentation with masked entity language modeling for low-resource NER. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2251–2262, Dublin, Ireland. Association for Computational Linguistics.

Appendix

A. Prompt to LLM for Data Augmentation

The following prompt was used, where `{text}` was dynamically replaced with the extracted OOV keywords:

> **System:** `You are a helpful assistant that generates natural sentences. Return only the sentence without any additional text.`
>
> **User:** `Generate a natural sentence using these keywords: {text}. The sentence should include citations such as [1], [3 , 4], or formatted styles like (Battke et al . , 2010) or (Jellyfish , https://scicrunch.org/resolver/RRID:SCR_005491) [28]. The sentence should also follow version patterns like: v . 2 . 4 . 0, v 2 . 31 . 9, v 10, version 20170127, version 2 . 3 . 0, (v 0 . 36 . 5), etc. Some sentences should include the extensions like: SPSS / PASW , SAS / SPSS , R / RStudio, etc. Some sentences should include the applications like: R, Python, GraphPad Prism, etc. Some sentences should include the operating systems like: Windows, Linux, Mac OS, etc. Strictly maintain sentence extensions, versions, applications, operating systems, etc. as given examples.`