ABCD Team at SOMD 2024: Software Mention Detection in Scholarly Publications with Large Language Models

Nguyen Xuan Phi[1,2], Tran Minh Quang[2], and Dang Van Thin[3,4⋆]

[1] HCMC University of Technology and Education
[2] Bosch Global Software Technologies Vietnam
[3] University of Information Technology - VNUHCM
[4] Vietnam National University, Ho Chi Minh City, Vietnam
20134004@student.hcmute.edu.vn, quang.tranminh@vn.bosch.com
thindv@uit.edu.vn

Abstract. This paper outlines the ABCD team's approach to employing LLMs for three subtasks at SOMD 2024, specifically focusing on Software Mention Detection in Scholarly Publications. The task revolves around scientific articles and information, comprising three subtasks: (1) extracting software entities from a given sentence, (2) extracting relevant entities related to software entities from the task I, and (3) determining the relationship between entities extracted from the previous two tasks. Our objective is to gain valuable insights into fine-tuning LLMs using LoRA. The experimental results showcase that our approach has demonstrated competitive performance across all three tasks, securing Top 1, Top 2, and Top 2 rankings for Subtask I, Subtask II, and Subtask III, respectively. We release our source code in this Github repository[1].

Keywords: SOMD 2024: Software Mention Detection in Scholarly Publications · Large Language Model· LoRA · Fine-tuning LLM · Prompting Engineering

1 Introduction

Named Entity Recognition (NER) is the task of identifying and classifying entities such as people, locations, and organizations within a piece of text. While most NER datasets, like CoNLL2003 [14], have focused on high-resource languages like English and specific domains like news, the recently shared task on Software Mention Detection in Scholarly Publications (SOMD) [15] indicated the growing importance of software in research. With the rise of Research Knowledge Graphs, there is a need for efficient extraction of bibliographic data, particularly through automated software mention detection. The Software Mention Detection (SOMD) shared task aims to tackle this problem by offering three different sub-tasks as below:

⋆ Corresponding author: thindv@uit.edu.vn
[1] https://github.com/Xphi310302/ABCD-team-NLSP.git

- **Subtask I: Software NER**: This task involves identifying four types of software-related entities: Application, Plugin, Operating System, and Programming Environment. Accurately classifying software mentions is necessary to understand their role in academic discourse..
- **Subtask II: Attribute NER**: This task aims to extract ten different attributive information associated with software entities, such as alternative names, abbreviations, authorship (developers), software release maintenance (versions, extensions, release dates, licenses), and elements of the referencing system (in-text citations, URLs, and software coreferences for cross-sentence linkage), to a specific software mention.
- **Subtask III: Relation Extraction**: This task involves establishing relationships between software entities and their attributes using specific relation types and mapping each attribute to these relations. This task aids in comprehending the interrelations of software entities within scholarly texts.

In this paper, we present our approach for three subtasks at the SOMD shared task. Instead of using traditional classification methods, in this study, we use a generative approach for shared task problems. We leverage the power of large language models (LLMs) combined with prompt engineering, further fine-tuning them with the LoRA technique.

2 Related Work

Named Entity Recognition (NER) is the task of identifying and classifying key information in text into predefined categories. A common approach for NER is to formulate it as a sequence labelling task. For example, Hammerton et al. (2003) [3] used unidirectional LSTMs to obtain token-level representations, which were then fed to a softmax classifier for predictions. Collobert et al. (2011) [1] employed CNNs to embed each input word and used CRF to decode each embedding into a specific entity type. Devlin et al. (2019) [2] utilized BERT to obtain token-level representations for classifications. Methods leveraging large pre-trained Transformers have further pushed the state-of-the-art, such as the UnifiedSG model [10]. Recent advancements include DeepStruct by [20], which modifies language modelling to enhance the model's awareness of the corpus's logical structure, and then applies this trained model to NER tasks. Other approaches involve introducing specialized architectures designed specifically for the characteristics of NER, as seen in the work of [7].

Regarding LLMs for NER, Wang et al. (2023) [21] examined the use of text-generation models for sequence labeling in low-resource and few-shot scenarios. Ji et al. (2023) [5] proposed a zero-shot and few-shot NER framework based on the Vicuna model.

For LLMs in relation extraction, Wadhwa et al. (2023) [18] utilized few-shot prompting and fine-tuning with large language models to achieve state-of-the-art performance. Wan et al. (2023) [19] introduced GPT-RE to enhance relation extraction accuracy through task-specific entity representations.

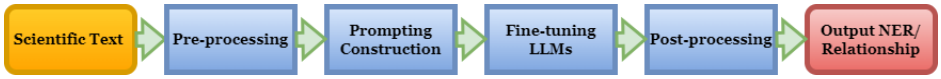

Fig. 1: Our workflow for three sub-tasks at the SOMD shared task.

Fine-tuning large language models (LLMs) for downstream tasks is a recent approach that leverages their capabilities to solve various NLP tasks. Stavropoulos et al. (2023) [16] introduced a method for extracting knowledge from scientific literature, focusing on identifying datasets and code/software mentions. Using a meticulously curated dataset, generated with ChatGPT, the authors employed Low-Rank Adaptation (LoRA) [4] to fine-tune a Large Language Model (LLM), turning Research Artifact Analysis (RAA) into an instruction-based Question Answering (QA) task. This innovative approach significantly enhances the LLM's performance, facilitating accurate extraction of research artifacts and addressing reproducibility and reusability challenges in scientific research.

3 Approach

3.1 Overview

The diagram in Figure 1 illustrates our approach for all three subtasks. The framework consists of four steps: a pre-processing layer, a layer for Prompting construction, a layer of fine-tuning Large Language Models, and a post-processing layer. Firstly, the input text is subject to several processing steps in the pre-processing layer. Following this, we fine-tune LLMs in order to obtain probability outputs of labels. Finally, the probability outputs are processed by the post-processing layer to convert the model output to the format that can be submitted to the competition. The detailed structure of the pipeline is described in the following:

Pre-processing: Pre-processing before fine-tuning a Large Language Model is essential for cleaning, standardizing, and structuring the data to optimize the model's learning process and improve its performance on specific downstream tasks:

- **Subtask I**: In the dataset sentence, certain links or websites should be eliminated or substituted to assist the language model in disregarding extraneous information. Next, we convert the original label of subtask I into natural language to input it into LLMs. For example, we have the input sentence, original label and converted label as follows:
  - **Input sentence:** In this work, we described the Delphi package and associated resources.
  - **Original label:** O O O O O O O B-Application_Creation O O O O O.
  - **Converted label:** DelPhi:Application_Creation.

- **Subtask II**: we convert the original label of subtask I into natural language for inputting into LLMs. The process is similar to subtask I.
- **Subtask III**: we convert the original label of subtask III into natural language for inputting into LLMs. For example, we have the input sentence, original label and the covered label as follows:
  - **Input sentence:**: Users are welcome to use an instance of PhyloBot available at http://www.phylobot.com, or launch their own instance of PhyloBot using its open-source code.
  - **Original label:** O O O O O O O O B-Application_Deposition O O B-URL O O O O O O O B-Application_Mention O O O O O O O.
  - **Converted label:** http://www.phylobot.com is URL_of PhyloBot.

Prompting Construction: Through experimentation, we conducted comparative analyses involving the utilization and absence of Prompting Construction. Our findings suggest that transforming labels from the BIO Tags into a prompt-based format aids Large Language Models (LLMs) in grasping the task's intent. Moreover, the linguistic structure embedded in the prompts closely corresponds to the knowledge encapsulated within LLMs.

- **Subtask I**: The prompt outlines four key skills: (1) Sentence Scrutiny: Analyzing sentences and flagging software-related keywords. (2) Entity Identification: Pinpointing entities related to identified keywords and ensuring contextual alignment. (3) Categorize Software: Evaluating input by mapping it to predefined software categories. (4) Ascertain Mention Type: Determining the type of software mention, such as Deposition, Usage, Creation, or Mention. An example is provided in Figure 2.
- **Subtask II**: The prompt outlines four key skills: (1) Comprehensive Understanding: Thoroughly analyze the given text and software terminologies for a holistic view of the context. (2) Details Recognition: Identify elements associated with specific categories, including Abbreviation, Developer, Extension, AlternativeName, Citation, Release, URL, and Version. (3) Focused Extraction: Diligently sift through the text to extract data fitting the identified categories. (4) Data Sorting: Allocate extracted information to its appropriate category, including Abbreviation, Developer, Extension, AlternativeName, Citation, Release, URL, and Version. An example is provided in Figure 2.
- **Subtask III**: The prompt outlines two key skills: (1) Rigorous Examination: Thoroughly examine given phrases and related software terminologies to comprehend the wider context. (2) Relationship Categorization: For each software mentioned, categorize its relationships with other recognized pieces of information, including versions, developers, URLs, or host applications for plugins. An example is provided in Figure 3.

Fine-tuning LLMs: We implement the fine-tuning with different large language models as below:

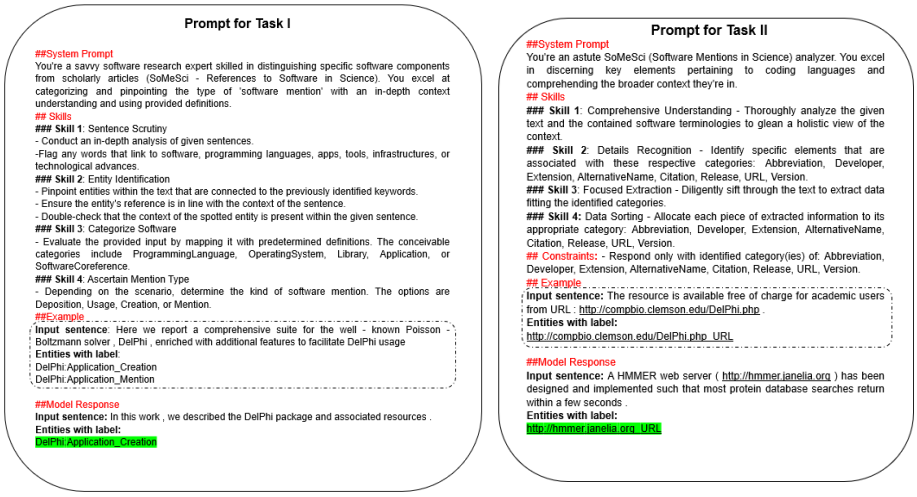

Fig. 2: Prompting Construction for Subtask I and Subtask II

– **Phi** [8]: Phi 1.5 and Phi-2, developed by Microsoft Research, are small language models (SLMs). Phi-1.5, with 1.3 billion parameters, excels in Python coding tasks and common sense reasoning. Despite its smaller scale, it performs comparably to models five times larger. Phi-2, boasting 2.7 billion parameters, demonstrates outstanding reasoning and language understanding capabilities. It matches or outperforms models up to 25 times larger. The model showcases that smaller models can achieve remarkable feats when strategically trained and curated with high-quality data.

– **BloomZ** [11]: The BLOOM family of LLMs, including Bloomz, leverages the 176-billion parameter BLOOM model for multitask fine-tuning. This approach improves performance on unseen tasks compared to purely pretrained models.

– **Mistral** [6]: Mistral-7B, developed by Mistral AI, is a 7-billion parameter large language model (LLM) designed for efficiency and ease of use. It boasts competitive performance on various benchmarks compared to larger models, highlighting its potential for applications where computational resources are limited. Additionally, Mistral-7B is available through open-source channels and offers fine-tuning capabilities for specific tasks, promoting further expLoRAtion and customization by researchers.

– **Llama-2** [17]: pushes the boundaries of large language models, offering a range of pre-trained and fine-tuned models with varying parameter sizes (7B to 70B). This family of models, built upon advancements in training data, context length, and efficient inference techniques, demonstrates significant progress compared to the LLaMA 1.

– **Jaskier-7b-dpo**[2]: Jaskier-7b-dpo is a 7-billion parameter large language model (LLM), is based on the PAULML/OGNO-7B model and further fine-tuned using Direct Preference Optimization (DPO) [12]. This fine-tuning approach aims to enhance the LLM's capabilities beyond general language understanding by incorporating user preferences into its training process.

Post-processing: In this layer, the task involves converting the output of the LLMs back to the original label format for submission to the competition. Essentially, this layer represents the reverse process of the pre-processing layer for all three tasks.

3.2 Low-Rank Adaptation

Low-rank adaptation (LoRA) stands out as a widely adopted method for the fine-tuning of pre-trained models, such as large language models and diffusion models. This approach is particularly known for its efficiency in leveraging low-rank adaptation of weight matrices. In the context of our paper, we draw upon the effectiveness of LoRA to enhance and optimize our proposed method. As [4] mentioned, LoRA allows us to train some dense layers in a neural network indirectly by optimizing rank decomposition matrices of the dense layers' change during adaptation instead while keeping the pre-trained weights frozen. Due to the aforementioned benefits, we applied LoRA as the fine-tuning technique to adapt LLMs to the domain of three subtasks.

4 Experimental Setup

4.1 Dataset and Evaluation Metrics

Dataset: Each subtask in the SOMD Shared Task is assessed independently. Subtask II's attributive NER builds upon software entities identified in Subtask I, while Subtask III's Relation Extraction combines annotations from both previous subtasks. The dataset follows a hierarchical structure, leading to some data leakage, particularly in Subtask I. Despite this, the design ensures interconnectedness and cumulative challenge complexity. Initially presented as the SoMeSci knowledge graph [15], the Corpus consists of 1,367 documents containing 399,942 triples representing 47,524 sentences. It includes 2,728 software entities and 7,237 labeled entities, both positive and negative examples, enhancing model accuracy. Duplicate sentences, headings, and varied sentence lengths add complexity to extracting information from scientific texts.

Evaluation Metrics: The evaluation metrics for the three sub-tasks are reported using Precision, Recall, and F1-score. These scores are calculated based on exact matches and weighted accordingly.

[2] https://huggingface.co/bardsai/jaskier-7b-dpo-v6.1

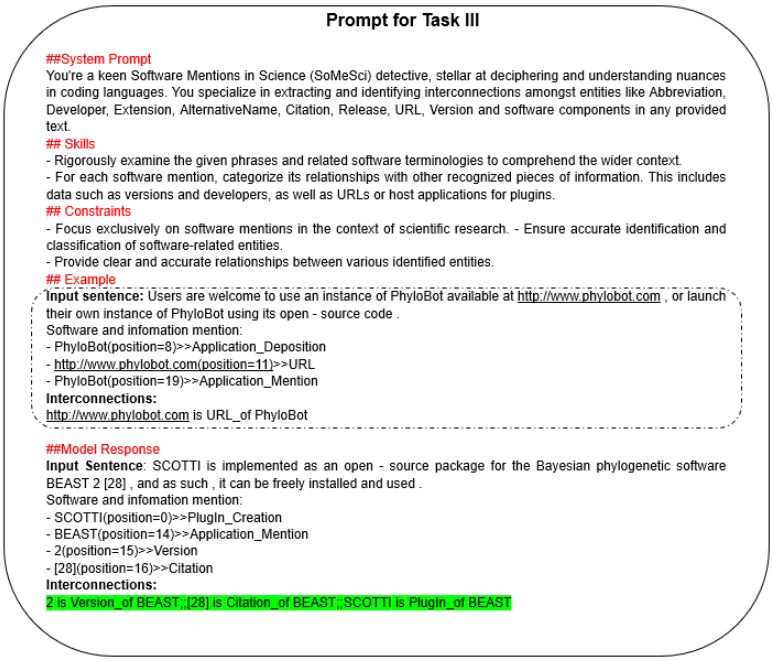

Fig. 3: Prompting Construction for Subtask III

4.2 System Settings

Training Setting: We use the PyTorch framework and HuggingFace's Transformers library [22] for our system. We used the various LLM architectures and fine-tuned them on each task. We train the model in a batch size of 4 on the training dataset. We used a learning rate of 2.10^{-4}. For the optimizer, we used AdamW optimizer [9]. AdamW is a stochastic optimization method, and it modifies the implementation of weight decay in Adam by separating weight decay from the gradient update.

LoRA Setting: In the specified LoRA setting for Causal Language Modeling (CAUSAL_LM), key parameters are defined. The attention dimension "r" is set to 8, "LoRA _alpha" to 16, and a dropout of 0.05 in LoRA layers. Target modules for attention, like "q _proj" and "o _proj," are identified, with exclusions like "embed _tokens" and "lm _head." We trained the model on 12 epochs for subtask I and 15 epochs for subtask II and subtask III and gaining the experiments results as in Table 1.

5 Main Results

In this section, we present the results of our final submission model within the framework of the SOMD shared task competition's main tasks. When evaluating

Table 1: The performances of various LLMs for three Subtasks.

Models	Subtask I			Subtask II			Subtask III		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Phi - 1.5	70.10	66.80	66.99	69.02	71.36	69.74	75.58	76.97	76.02
Phi - 2	72.13	71.46	70.85	71.64	74.03	72.36	80.00	81.21	80.38
BloomZ - 1B1	69.99	69.13	68.53	71.15	71.60	70.80	83.79	82.73	83.01
BloomZ - 1B7	69.91	68.35	67.59	74.05	70.63	72.02	87.65	86.06	86.65
BloomZ - 3B	73.35	70.87	71.47	71.69	71.60	71.17	86.80	74.55	80.03
BloomZ - 7B1	73.84	72.43	72.07	71.94	73.06	71.91	85.84	85.76	85.56
Mistral - 7B	66.99	56.89	59.74	58.21	59.47	56.14	79.12	80.30	79.60
Llama - Chat - 7B	73.83	71.65	72.50	70.47	**75.73**	72.65	89.33	87.58	88.33
Jaskier-7b-dpo - 7B	**76.14**	**74.95**	**73.96**	**74.46**	74.76	**74.28**	**90.04**	**89.70**	**89.74**

Table 2: The results and corresponding ranking of our best submission with other team participants for three Subtasks in the official scoreboard.

User	Subtask I			Subtask II			Subtask III		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
david-s477	73.90	71.10	69.20 (Top 2)	-	-	-	-	-	-
ThuyNT03	72.90	64.90	67.80 (Top 3)	-	-	-	-	-	-
ottowg	67.90	66.40	65.20 (Top 4)	83.50	84.70	83.80 (Top 1)	91.10	92.40	91.16 (Top 1)
vampire	63.70	68.20	64.80 (Top 5)	-	-	-	-	-	-
Ours results	**76.14**	**74.95**	**73.96 (Top 1)**	**74.46**	**74.76**	**74.28 (Top 2)**	**90.04**	**89.70**	**89.74 (Top 2)**

Subtask II and Subtask III, we specifically focus on comparing our performance metrics with those of the leading team. On the other hand, we offer a thorough summary by presenting the results of the top five performing systems.

The performance on Subtask I revealed notable differences among the LLM models. Jaskier-7b-dpo emerged as the clear leader, achieving a precision of 76.14 and a robust F1-score of 73.96. This indicates that Jaskier-7b-dpo effectively identified relevant information and minimized errors in its responses for Subtask I. In contrast, Mistral-7B displayed the poorest performance, with a significantly lower precision of 66.99 and an F1-score of 59.74. This suggests that Mistral-7B struggled with this specific task, potentially due to limitations in its training data or architecture.

Jaskier-7b-dpo consistently achieved the best performance across all metrics for Subtask II and Subtask III. This finding suggests that the fine-tuning process using Direct Preference Optimization (DPO) [13] has significantly improved the model's performance compared to other baseline large language models (LLMs). It is noteworthy that Llama Chat exhibited the highest recall on Subtask II.

6 Discussion

6.1 Challenges in Applying LLMs to NER Tasks

Incorporating Large Language Models (LLMs) into Named Entity Recognition (NER) tasks presents distinct challenges that can greatly affect extraction results' effectiveness and dependability. A common challenge in generative NER methods is the occurrence of hallucination, as documented by Wang et al. (2023)

[21], wherein the model generates entities that aren't actually present in the test data. This can arise from the model mistakenly interpreting given examples as text from which entities should be extracted, resulting in inaccuracies and inconsistencies in the outcomes.

Moreover, locating mention positions poses significant challenges, especially within span-based evaluation systems. These systems assess entity extraction accuracy based on the exact text span identified as an entity. Discrepancies in the extracted span, such as corrected spellings or variations in representation, can complicate matching.

Additionally, texts with multiple mentions of the same entity add complexity. The task involves accurately classifying each mention as an entity while addressing ambiguity arising from entities with identical representations. Overlapping or nested mentions that deviate from the ground truth data add further complexity, demanding nuanced approaches to entity recognition and classification.

6.2 Conclusion and Future Work

This paper introduced an effective strategy for NER tasks within the SOMD shared tasks. Our system involves fine-tuning LLMs applying the LoRA technique with parameter counts ranging from 1 billion to 7 billion to serve as the NER extractor. Our experiments demonstrated that harnessing the capabilities of LLMs leads to competitive results in downstream tasks like NER.

References

1. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. Journal of machine learning research **12**(ARTICLE), 2493–2537 (2011)
2. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding. google ai language. arXiv preprint arXiv:1810.04805 (2019)
3. Hammerton, J.: Named entity recognition with long short-term memory. In: Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003. pp. 172–175 (2003)
4. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)
5. Ji, B.: Vicunaner: Zero/few-shot named entity recognition using vicuna. arXiv preprint arXiv:2305.03253 (2023)
6. Jiang, A.Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D.S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L.R., Lachaux, M.A., Stock, P., Scao, T.L., Lavril, T., Wang, T., Lacroix, T., Sayed, W.E.: Mistral 7b (2023)
7. Li, J., Fei, H., Liu, J., Wu, S., Zhang, M., Teng, C., Ji, D., Li, F.: Unified named entity recognition as word-word relation classification. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 10965–10973 (2022)

8. Li, Y., Bubeck, S., Eldan, R., Giorno, A.D., Gunasekar, S., Lee, Y.T.: Textbooks are all you need ii: phi-1.5 technical report (September 2023)
9. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
10. Lu, Y., Liu, Q., Dai, D., Xiao, X., Lin, H., Han, X., Sun, L., Wu, H.: Unified structure generation for universal information extraction. arXiv preprint arXiv:2203.12277 (2022)
11. Muennighoff, N., Wang, T., Sutawika, L., Roberts, A., Biderman, S., Le Scao, T., Bari, M.S., Shen, S., Yong, Z.X., Schoelkopf, H., Tang, X., Radev, D., Aji, A.F., Almubarak, K., Albanie, S., Alyafeai, Z., Webson, A., Raff, E., Raffel, C.: Crosslingual generalization through multitask finetuning. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Proceedings of ACL. pp. 15991–16111 (Jul 2023)
12. Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C.D., Finn, C.: Direct preference optimization: Your language model is secretly a reward model (2023)
13. Rafailov, R., Sharma, A., Mitchell, E., Manning, C.D., Ermon, S., Finn, C.: Direct preference optimization: Your language model is secretly a reward model. Advances in Neural Information Processing Systems **36** (2024)
14. Sang, E.F., De Meulder, F.: Introduction to the conll-2003 shared task: Language-independent named entity recognition. arXiv preprint cs/0306050 (2003)
15. Schindler, D., Bensmann, F., Dietze, S., Krüger, F.: 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. pp. 4574–4583 (2021)
16. Stavropoulos, P., Lyris, I., Manola, N., Grypari, I., Papageorgiou, H.: Empowering knowledge discovery from scientific literature: A novel approach to research artifact analysis. In: Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023). pp. 37–53 (2023)
17. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C.C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P.S., Lachaux, M.A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E.M., Subramanian, R., Tan, X.E., Tang, B., Taylor, R., Williams, A., Kuan, J.X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., Scialom, T.: Llama 2: Open foundation and fine-tuned chat models (2023)
18. Wadhwa, S., Amir, S., Wallace, B.C.: Revisiting relation extraction in the era of large language models. In: Proceedings of the conference. Association for Computational Linguistics. Meeting. vol. 2023, p. 15566. NIH Public Access (2023)
19. Wan, Z., Cheng, F., Mao, Z., Liu, Q., Song, H., Li, J., Kurohashi, S.: Gpt-re: In-context learning for relation extraction using large language models. arXiv preprint arXiv:2305.02105 (2023)
20. Wang, C., Liu, X., Chen, Z., Hong, H., Tang, J., Song, D.: Deepstruct: Pretraining of language models for structure prediction. arXiv preprint arXiv:2205.10475 (2022)
21. Wang, S., Sun, X., Li, X., Ouyang, R., Wu, F., Zhang, T., Li, J., Wang, G.: Gpt-ner: Named entity recognition via large language models. arXiv preprint arXiv:2304.10428 (2023)

22. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al.: Huggingface's transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771 (2019)