SOMD@NSLP2024: Overview and Insights from the Software Mention Detection Shared Task

Frank Krüger[1][0000−0002−7925−3363], Saurav Karmakar[2][0009−0007−0124−5316], and Stefan Dietze[2][0009−0001−4364−9243]

[1] Wismar University of Applied Sciences, Wismar, Germany
frank.krueger@hs-wismar.de
[2] GESIS - Leibniz-Institut für Sozialwissenschaften, Köln, Germany
saurav.karmakar@gesis.org
stefan.dietze@gesis.org

Abstract. Software is a central part of the scientific process and involved in obtaining, analysing, visualising and processing research data. Understanding the provenance of research requires an understanding of the involved software. However, software citations in scientific publications often are informal, what creates challenges when aiming at understanding software adoption. This paper provides an overview of the Software Mention Detection (SOMD) shared task conducted as part of the 2024 Natural Scientific Language Processing Workshop, aiming at advancing the state-of-the-art with respect to NLP methods for detecting software mentions and additional information in scholarly publications. The SOMD shared task encompasses three subtasks, concerned with software mention recognition (subtask I), recognition of additional information (subtask II) and classification of involved relations (subtask III). We present an overview of the tasks, received submissions and used techniques. The best submissions achieved F1 scores of 0.74 (subtask I), 0.838 (subtask II) and 0.911 (subtask III) indicating both task feasibility but also potential for further performance gains.

Keywords: scholarly information processing · software mention extraction · software metadata identification · information extraction · relation classification

1 Introduction

Science across all disciplines has become increasingly data-driven, leading to additional needs with respect to software for collecting, processing and analysing data. Hence, transparency about software used as part of the scientific process is crucial to ensure reproducibility and to understand provenance of individual research data and insights. Knowledge about the particular version or software development state is a prerequisite for reproducibility of scientific results as even minor changes to the software might impact them significantly.

Furthermore, from a macro-perspective, understanding software usage, varying citation habits and their evolution over time within and across distinct disciplines can shape the understanding of the evolution of scientific disciplines,

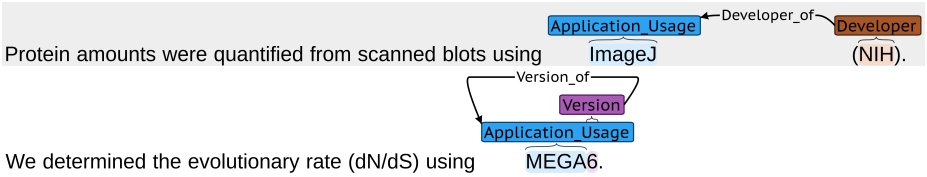

Fig. 1. Annotated sentences from SoMeSci. (Image taken from [18])

the varying influence of software on scientific impact and the emerging needs for computational support within particular disciplines and fields. Initial efforts are made to provide publicly accessible datasets that link open access articles to respective software that is used and cited, for instance, the OpenAIRE Knowledge Graph [10] or SoftwareKG [20]. Given the scale and heterogeneity of software citations, robust methods are required, able to detect and disambiguate mentions of software and related metadata. Despite the existence of software citation principles [21,6], software mentions in scientific articles are usually informal and often incomplete [19]—information about the developer or the version are often missing entirely, see Fig 1. Spelling variations and mistakes for software names, even common ones [20], increase the complexity of automatic detection and disambiguation. Training and evaluation of information extraction approaches require reliable ground truth data of sufficient size, raising the need for manually annotated gold standard corpora of software mentions.

With this shared task, we would like to advance the field of software mention detection, seeking novel methods that outperform the state-of-the-art on the provided three subtasks.

We use Codalab [14] as a platform to run all three competitions. Subtask I[3] received 22 registrations from participants, from which 10 results were submitted. In contrast, the more challenging subtasks II[4] and III[5] received 12 registrations and 11 registrations respectively, but for each, only 3 actual submissions were received.

The rest of the paper is structured as follows. Section 2 presents previous work related to SOMD in order to compare the presented systems to current research, Section 3 defines both subtasks along with the used evaluation metrics. In Section 4, we introduce the datasets and taxonomies used for both subtasks, delving into their construction methods. Section 5 showcases the results received from submissions of both subtasks, describing the system architectures when possible. Finally, Section 6 discusses those results along with their limitations, and provides concluding remarks.

[3] https://codalab.lisn.upsaclay.fr/competitions/16935
[4] https://codalab.lisn.upsaclay.fr/competitions/16936
[5] https://codalab.lisn.upsaclay.fr/competitions/16937

2 Related Work

Most works concerned with recognition of software mentions in scientific articles apply manual analysis on small corpora in order to answer specific questions [4,12] or are limited to specific software [8,9]. Automatic methods, enabling large scale analysis, have been implemented by iterative bootstrapping [13] as well as machine learning on manually engineered rules [2]. However, both achieve only moderate performance. Extraction through deep learning with a Bi-LSTM-CRF [20] shows promise, but requires sufficient and reliable ground truth data which only recently became available. More recently, Schindler et al. [17] provided robust information extraction models based on SciBERT and trained on the SoMeSci corpus [18] for NER and classification outperforms state-of-the-art methods for software extraction by 5 percentage points on average. A similar approach was taken by [5] to recognize software mentions across several million articles achieving Named Entity Recognition performances at a similar level. Given that performance of related works still widely varies and is far from robust, this shared task aims at advancing the field of software mention detection and disambiguation across various subtasks.

3 Tasks Description

Software is an important part of the scientific process and should therefore be recognized as first class citizen of research. Research Knowledge Graphs have recently been adopted to provide bibliographic data at scale that could be populated by automatic extraction of software mentions. Given the scale and heterogeneity of software citations, robust methods are required to detect and disambiguate mentions of software and related metadata. The SOftware Mention Detection in Scholarly Publications (SOMD) task utilises the SoMeSci - Software mentions in Science - corpus. Participants had the option to sign up for one or more subtasks. Automated evaluations of submitted systems are done through the Codalab platform.

- **Subtask I: Software Mention Recognition**: Software mentions are recognized from individual sentences. At the same time, software mentions had to be classified according to their mention type, e.g., mention, usage, or creation and their software type, e.g., application, programming environment, or package. Participants developed classifiers that take individual sentences from the different subsets of SoMeSci and output mentions of software further classified into their type of software and mention. Submissions were evaluated using the F1 score computed based on exact matches. Please note that subtask I deviates from the from the original publication in that it combines the identification of the software and the classification of mention and software type.
- **Subtask II: Additional Information**: For each software mention, additional information according to the SoMeSci schema shall be recognized

from the sentence. This includes information such as version, URL, and developer. Participants had to develop classifiers that take sentences with software mentions as input and identify all additional information within the sentence. As in Subtask I, submissions were evaluated through the F1 score based on exact matches.

– **Subtask III: Relation Classification**: For each software mention, relations to other recognized entities had to be classified. This includes versions and developers, but also URLs or host applications for plugins. The evaluation was based on exact matches rather than partial matches. F1 score had been used as an evaluation performance metric for all the subtasks.

4 Dataset

The shared tasks utilise SoMeSci-Software Mentions in Science - a gold standard knowledge graph of software mentions in scientific articles [18]. It contains high quality annotations (Inter-Rater Reliability, IRR: κ=.82) of 3756 software mentions in 1367 PubMed Central articles. Besides the plain mention of the software, it also provides relation labels for additional information, such as the version, the developer, a URL or citations and distinguishes between different types, such as application, plugin or programming environment, as well as different types of mentions, such as usage or creation. SoMeSci is the most comprehensive corpus about software mentions in scientific articles, providing training samples for Named Entity Recognition, Relation Extraction, Entity Disambiguation, and Entity Linking.

SoMeSci is created by manually annotating 3756 software mentions with additional information, resulting in 7237 labelled entities in 47,524 sentences from 1367 PubMed Central articles. Data is lifted into a knowledge graph (excerpt in Fig. 2) by using established vocabularies such as NLP Interchange Format (NIF) [3] and schema.org [16], disambiguated and linked to external resources, and shared as a publicly available 5-star Linked Open Data resource [1] that can be explored interactively.

In preparation of the three subtasks for the SOMD challenge, we released a new dataset implementing predefined splits for training and testing.

For each subtask, the same split was created, which also follows the original train-test split, as reported for SoMeSci[18], resulting in 39.768 sentences for train and 8180 sentences for testing for the first subtask.

The new dataset was released via Zenodo [7] and consists of three parts, one for each subtask. Each of the individual parts contains a list of sentences and labels for training and a list of sentences for testing. As subtask II and III require additional information for the test set, such as the already identified software mentions and their respective meta data, for both tasks, we provided an additional file with this information for train and test set respectively. As subtask II and III require already recognized software mentions, the number of sentences provided for these tasks reduced to 2353 (1091) for the train set and 374 (131) for the test set for subtask I (subtask II).

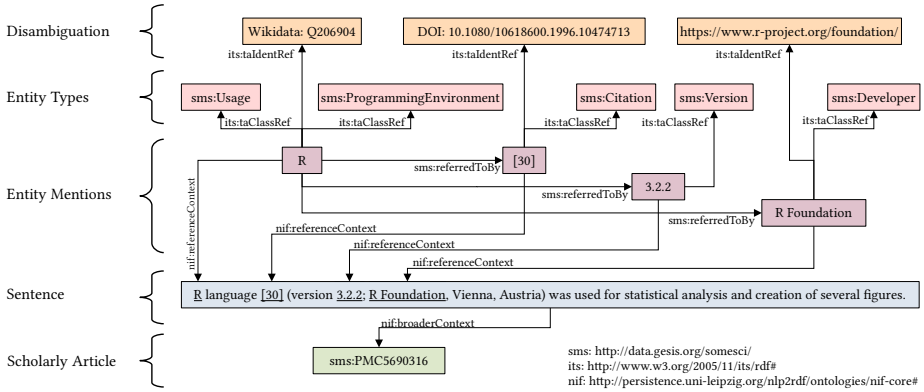

Fig. 2. Excerpt of the SoMeSci knowledge graph illustrating the textual references of software mentions and their version, developer and citation. The different levels of representation separate the main concerns of interest, natural language sentences, mentions of entities, their types and disambiguation to knowledge entities. For clarity, some information is omitted. (Image taken from [18])

Participants of the SOMD shared task were required to retrieve the dataset and use all available training data to establish a classifier to be tested on the provided test set. The prediction as created from the test set of the respective tasks was subject to submission at the Codalab platform. Finally, evaluation scripts implemented in Python by using the packages scikit-learn [15] and seqeval [11] were used to determined weighted Precision, Recall and F1 score, where the F1 score was used to rank the submissions.

5 Results

In this section we describe results reported as well the techniques and strategies adopted by different participants in the subtasks.

5.1 Subtask I

Overall, 23 participants registered for subtask I, from which we received nine submissions in total. Two of the submissions obtained F1 scores close to 0, resulting in seven valid submissions. An overview of the valid submissions including the achieved scores is provided in Table 1. As not all teams submitted system descriptions, in to following we focus on submissions that provided such descriptions. `phinx` achieved the highest F1 score, followed by `david-s477`, `ThuyNT03` and `ottogwg`.

Team `phinx` experimented with different LLMs namely BloomZ, Mistral, Llama-2 and Jaskier-7b-dpo, where Jaskier-7b-dpo provided the best performance (F1 score of 74%). They further finetuned pre-trained models using the LoRA (Low-Rank Adaptation) [22] technique.

Table 1. Evaluation results of submissions for subtask I

User	F1 (weighted)	precision (weighted)	recall (weighted)
phinx	**0.740**	**0.761**	**0.750**
david-s477	0.692	0.739	0.711
ThuyNT03	0.678	0.729	0.649
ottowg	0.652	0.679	0.664
vampire	0.648	0.682	0.637
fddaFIT	0.504	0.548	0.499
dainb	0.483	0.709	0.392

Team ThuyNT03 experimented with BERT based models specifically XLM-Roberta, BERT and SciBERT. Utilizing each model, they experimented with three approaches: in their first approach they chose direct classification, whereas in the second approach, classification was split into two stages, where the first stage produced BIO tags and the second stage produced actual entity labels. The third approach conducted a three stage classification that included a preliminary step to detect if a sentence contains any entities before continuing with the two stage classification.

The team fddaFIT investigated the effectiveness of the decoder-only Falcon-7b model, which is known for its performance across a wide range of NLP tasks. They experimented with different sampling schemes like selective sampling and adaptive sampling to compose finetuning data. They also experimented with different strategies but that did not yield enhancements in outcome. To address the class imbalance they used a weighted loss mechanism (where class weights are inversely proportional to class frequencies) and adaptive sampling, i.e. over-sampling the underrepresented data by a factor of 2 and undersampling the over-represented data to sizes equal to multiples (1, 1.5, 3) of the oversampled data volume.

Team ottogwg employed SciBERT pre-trained model and they also experiment with generative large language models. Various prompting strategies were used by the team for the subtask. Retrieval-Augmented Generation (RAG) with LLM has been applied using Generative Language Models (GLMs), specifically GPT 3.5 and GPT 4 for the task. They used a pipeline strategy that prioritizes selecting relevant text passages for GLM analysis, improving efficiency by filtering out unrelated content. Their performance optimization employs a hybrid method, combining a fine-tuned NER model for sentence selection with GLMs for information extraction. Their best configuration achieved an F1 score of 0.679 for subtask I using a generative LLM.

5.2 Subtask II

Out of 12 registrations for subtask II, we received submissions by two teams, namely phinx and ottogwg. Performance metrics for this subtask are reported

in Table 2. The team `ottowg` achieves the top performance with an F1 score of 0.838, whereas the `phinx` team achieves 0.743.

For this subtask, the `ottogwg` team adopted an approach similar to the approach of subtask I, but tuned for the extraction of associated software attributes. They utilised a retrieval mechanism to augment the task description in a few shot setup. For each sample, including those derived from few-shot learning, the process entailed presenting the sentence containing the software entity(ies) and then predicting a JSON list of identified entities.

The `phinx` team followed the same experimentation model as in subtask I with LLMs with modifications to the prompt engineering to accommodate the additional information for the software such as as version, URL, and developer etc. With multiple experimentation with various LLMs, they achieved 0.743 in F1 score as their best performing approach for this subtask using the Jaskier-7b-dpo model.

Table 2. Evaluation results of submissions for subtask II

User	F1 (weighted)	precision (weighted)	recall (weighted)
ottowg	**0.838**	**0.835**	**0.847**
phinx	0.743	0.745	0.748

5.3 Subtask III

Like the previous subtask, we received two submissions for subtask III by the same two same teams `phinx` and `ottogwg`. Table 3 depicts the results of their submission with `ottogwg` scored the best followed by `phinx` team.

For this subtask, the study for `ottogwg` proposed a novel approach by conceptualizing the task of relation extraction as a single-choice question-answering (QA) activity. This method entailed generating a comprehensive list of all possible entities within a sentence, drawing from the existing entities and their relationships as delineated in the training dataset. Each potential pair of entities was then evaluated to ascertain if a specific relation attribute types. These questions were then presented to a Large Language Model for answering.

The `phinx` team followed the same experimentation model as they applied for their earlier subtasks here with LLMs for modifications to the prompt engineering to accommodate the relations to other recognized entities which includes versions and developers, but also URLs or host applications for plugins. With multiple experimentation with various LLMs; again the Jaskier-7b-dpo model proved best for them by bringing 0.897 in F1 score as their best for this subtask.

Table 3. Evaluation results of submissions for subtask III

User	F1 (weighted)	precision (weighted)	recall (weighted)
ottogwg	**0.911**	**0.924**	**0.916**
phinx	0.897	0.904	0.897

6 Conclusion

In this paper, we presented an overview of the *Software Mention detection (SOMD)* shared task, that was run as part of the *2024 Natural Scientific Language Processing Workshop*, in conjunction with the Extended Semantic Web Conference 2024 (ESWC2024). The task is the first of its kind, proposing a set of three subtasks concerned with the detection of software mentions and related attributes in scholarly publications together with benchmark datasets and baselines. Given the important role of used software in the scientific process, understanding software citations is a crucial factor towards reproducibility of scientific works. This shared task provides the basis for advancing research into detecting and disambiguating software mentions. Unsurprisingly, the submissions to a large extent adopted various kinds of pre-trained language models as starting point for their pipelines. However, the diversity of submissions documented the range of techniques that can facilitate performance gains, starting from different base model choices, retrieval augmented approaches, sampling techniques or the use of prompt engineering as part of intermediate steps.

References

1. Berners-Lee, T.: Is your linked open data 5 star? (2010), http://www.w3.org/DesignIssues/LinkedData#fivestar
2. Duck, G., Nenadic, G., Filannino, M., Brass, A., Robertson, D.L., Stevens, R.: A survey of bioinformatics database and software usage through mining the literature. PLOS ONE **11**(6), 1–25 (06 2016). https://doi.org/10.1371/journal.pone.0157989
3. Hellmann, S., Lehmann, J., Auer, S., Brümmer, M.: Integrating nlp using linked data. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) The Semantic Web – ISWC 2013. pp. 98–113. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
4. Howison, J., Bullard, J.: Software in the scientific literature: Problems with seeing, finding, and using software mentioned in the biology literature. Journal of the Association for Information Science and Technology **67**(9), 2137–2155 (2016)
5. Istrate, A.M., Li, D., Taraborelli, D., Torkar, M., Veytsman, B., Williams, I.: A large dataset of software mentions in the biomedical literature (2022). https://doi.org/10.48550/ARXIV.2209.00693
6. Katz, D., Hong, N.C., Clark, T., Muench, A., Stall, S., Bouquin, D., Cannon, M., et al.: Recognizing the value of software: a software citation guide. F1000Research **9**, 1257 (Jan 2021). https://doi.org/10.12688/f1000research.26932.2
7. Krüger, F.: SOMD - SOftware Mention Detection (Jan 2024). https://doi.org/10.5281/zenodo.10472161

8. Li, K., Lin, X., Greenberg, J.: Software citation, reuse and metadata considerations: An exploratory study examining lammps. Proceedings of the Association for Information Science and Technology **53**(1), 1–10 (2016)
9. Li, K., Yan, E., Feng, Y.: How is r cited in research outputs? structure, impacts, and citation standard. Journal of Informetrics **11**(4), 989–1002 (2017)
10. Manghi, P., Bardi, A., Atzori, C., Baglioni, M., Manola, N., Schirrwagen, J., Principe, P.: The OpenAIRE research graph data model (2019). https://doi.org/10.5281/ZENODO.2643199
11. Nakayama, H.: seqeval: A python framework for sequence labeling evaluation (2018), https://github.com/chakki-works/seqeval, software available from https://github.com/chakki-works/seqeval
12. Nangia, U., Katz, D.S.: Understanding software in research: Initial results from examining nature and a call for collaboration. In: 2017 IEEE 13th International Conference on e-Science (e-Science). pp. 486–487. IEEE (2017)
13. Pan, X., Yan, E., Wang, Q., Hua, W.: Assessing the impact of software on science: A bootstrapped learning of software entities in full-text papers. Journal of Informetrics **9**(4), 860–871 (2015)
14. Pavao, A., Guyon, I., Letournel, A.C., Tran, D.T., Baro, X., Escalante, H.J., Escalera, S., Thomas, T., Xu, Z.: Codalab competitions: An open source platform to organize scientific challenges. Journal of Machine Learning Research **24**(198), 1–6 (2023), http://jmlr.org/papers/v24/21-1436.html
15. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
16. Ronallo, J.: Html5 microdata and schema. org. Code4Lib Journal (16) (2012)
17. Schindler, D., Bensmann, F., Dietze, S., Krüger, F.: The role of software in science: a knowledge graph-based analysis of software mentions in pubmed central. PeerJ Computer Science **8** (2022)
18. 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 and Knowledge Management (CIKM '21). Association for Computing Machinery, Virtual Event, QLD, Australia (Nov 2021). https://doi.org/10.1145/3459637.3482017
19. Schindler, D., Hossain, T., Spors, S., Krüger, F.: A multi-level analysis of data quality for formal software citation. Quantitative Science Studies (2024), https://arxiv.org/abs/2306.17535
20. Schindler, D., Zapilko, B., Krüger, F.: Investigating software usage in the social sciences: A knowledge graph approach. In: The Semantic Web, pp. 271–286. No. 12123 in Lecture Notes in Computer Science, Springer International Publishing, Heraklion, Greece (May 2020). https://doi.org/10.1007/978-3-030-49461-2_16
21. Smith, A.M., Katz, D.S., Niemeyer, K.E.: Software citation principles. PeerJ Computer Science **2**, e86 (2016). https://doi.org/10.7717/peerj-cs.86
22. Yu, Y., Yang, C.H.H., Kolehmainen, J., Shivakumar, P.G., Gu, Y., Ren, S.R.R., Luo, Q., Gourav, A., Chen, I.F., Liu, Y.C., et al.: Low-rank adaptation of large language model rescoring for parameter-efficient speech recognition. In: 2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). pp. 1–8. IEEE (2023)