Software Mention Recognition with a Three-Stage Framework Based on BERTology Models at SOMD 2024

Nguyen Thi Thuy[1,2], Nguyen Viet Anh[1,2], Dang Van Thin[1,2*], and Ngan Luu-Thuy Nguyen[1,2]

[1] University of Information Technology - VNUHCM
[2] Vietnam National University, Ho Chi Minh City, Vietnam
`21521514@gm.uit.edu.vn`, `19521204@gm.uit.edu.vn`,
{`thindv`,`ngannlt`}`@uit.edu.vn`

Abstract. This paper describes our systems for the sub-task I in the Software Mention Detection in Scholarly Publications shared-task. We propose three approaches leveraging different pre-trained language models (BERT, SciBERT, and XLM-R) to tackle this challenge. Our best-performing system addresses the named entity recognition (NER) problem through a three-stage framework. (1) Entity Sentence Classification - classifies sentences containing potential software mentions; (2) Entity Extraction - detects mentions within classified sentences; (3) Entity Type Classification - categorizes detected mentions into specific software types. Experiments on the official dataset demonstrate that our three-stage framework achieves competitive performance, surpassing both other participating teams and our alternative approaches. As a result, our framework based on the XLM-R-based model achieves a weighted F1-score of 67.80%, delivering our team the 3rd rank in Sub-task I for the Software Mention Recognition task. We release our source code at this repository[3].

Keywords: Software mention recognition · Named entity recognition · Transformer · Three-stage framework.

1 Introduction

Named Entity Recognition (NER) is an important task in NLP that involves identifying and classifying named entities in text. That will transform them into structured data, making it easier to categorize and perform search processing or carry out other NLP tasks [5] on that data such as text classification, sentiment analysis, and contextual analysis, ... particularly in the domain of Biomedical Named Entity Recognition (Bio-NER), which is challenged by a range of entities like genes, proteins, medications, and diseases [9].

* Corresponding author: thindv@uit.edu.vn

[3] https://github.com/thuynguyen2003/NER-Three-Stage-Framework-for-Software-Mention-Recognition

The SOMD 2024 shared-task, hosted within Natural Scientific Language Processing and Research Knowledge Graphs (NSLP 2024) workshop [8], is designed to extract mentioned software and metadata from documents. In this context, both the software and the metadata are identified as specific intervals in the original documents. Understand and identify the software mentioned in documents, which is especially important to support information extraction in scientific documents.

In this paper, we present three different approaches to address the challenge of sub-task I, including:

- **Approach 1**: Fine-tuning pre-trained language models as a token classification problem.
- **Approach 2**: Two-stage framework for entity extraction and classification.
- **Apporach 3**: Three-stage framework for entity sentence classification, entity extraction, and entity type classification.

2 Related Work

In recent years, pre-training language models (PLMs) have made significant advancements in Named Entity Recognition (NER) tasks [16]. Among these, the most popular model is BERT [7] and its variations like SciBERT [2], RoBERT [3], and BiLSTM [11]. These models are often paired with machine learning techniques, particularly Conditional Random Fields (CRF) [10]. Additionally, some approaches involve breaking down the NER task into two simpler tasks using question-answering methods [1], achieving notable results on various datasets like BioNLP13CG, CTIReports, OntoNotes5.0 [12], and WNUT17 [6] based on the F1 measure.

With the emergence of ChatGPT, researchers have been exploring the use of Large Language Models (LLMs) for NER tasks [15,17], with some studies demonstrating that ChatGPT can be distilled into smaller UniversalNER models for open NER [18]. These UniversalNER models have shown exceptional accuracy across 43 datasets spanning diverse fields such as biomedicine, programming, social media, law, and finance, without requiring direct supervision. UniversalNER surpasses traditional guideline-tuned models like Alpaca and Vicuna by an average of over 30 F1 points and achieves a high F1 score of 0.8 on SoMeSci. In this paper, BERT, SciBERT, and XML-R models are still utilized to address the first task of the shared SOMD 2024 challenge.

3 Approach

To address the Software Mention Recognition task, we utilize the power of different pre-trained transformer-based language model in different approaches. Figure 1 illustrates three approaches to participate in the competition. Because shared-task is related to each token in the sentence and whether words are in capital letters or not also greatly affects the recognition of entities. Therefore,

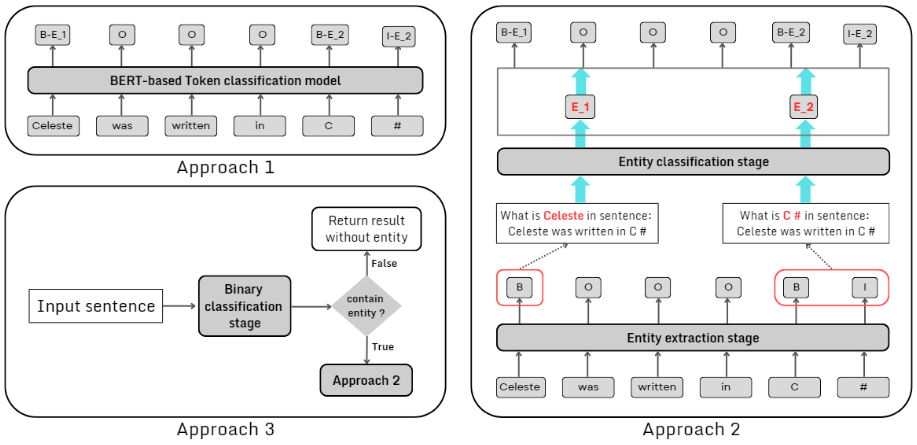
B-E_1
O
O
O
B-E_2
I-E_2
BERT-based Token classification model
Celeste
was
written
in
C
#
Approach 1
Input sentence
Binary classification stage
contain entity ?
False
True
Return result without entity
Approach 2
Approach 3
E_1
E_2
Entity classification stage
What is Celeste in sentence: Celeste was written in C #
What is C # in sentence: Celeste was written in C #
B
I
Entity extraction stage
Approach 2

Fig. 1. Overview system of three approaches: Sample input is "Celeste was written in C #" with two entities are E_1 and E_2. E_1 and E_2 play the role of two entity types in this example

we do not apply any preprocessing techniques but use data directly from the organizers. Also the tokenize method will depend on the default tokenier of the models. In our work, we employ various pre-trained language models, including the XLM-Roberta (XLM-R) [4], BERT [7], and SciBERT [2] as our main backbones. The detail of our three approaches are present as follow.

3.1 Approach 1: Token classification with BERTs

For the first approach, we address the task by fine-tuning different transformer BERT-base models for the token classification task. We adapted different pre-trained language models to the training dataset. After tokenizing the input, we feed the token sequence to backbones models to extract the fixed vector in the last layer as the final representation of the input sentence. Then, we apply a fully connected layer to process the vectors and predict labels for each input token using a softmax function. There are a total of 27 labels (in Table 1), where 26 correspond to 13 different entity types, and one label represents non-entities. Figure 1 illustrates the overview of our first approach.

3.2 Approach 2: Two-stage framework for Entity Extraction and Classification

Motivated by recent work by [1], we address Task 1 - Software Mention Recognition with a two-stage framework composed of entity extraction and entity classification components. However, our components are re-designed to improve the overall performance than original framework proposed by [1]. Figure 1 illustrates the overview of this approach, the detail of each component is presented below:

Table 1. List of labels for token classification task in Approach 1

Index	Label	Index	Label
1	B-Application_ Creation	15	B-PlugIn_Deposition
2	I-Application_ Creation	16	I-PlugIn_Deposition
3	B-Application_Deposition	17	B-PlugIn_Mention
4	I-Application_Deposition	18	I-PlugIn_Mention
5	B-Application_Mention	19	B-PlugIn_Usage
6	I-Application_Mention	20	I-PlugIn_Usage
7	B-Application_Usage	21	B-ProgrammingEnvironment_Mention
8	I-Application_Usage	22	I-ProgrammingEnvironment_Mention
9	B-OperatingSystem_Mention	23	B-ProgrammingEnvironment_Usage
10	I-OperatingSystem_Mention	24	I-ProgrammingEnvironment_Usage
11	B-OperatingSystem_Usage	25	B-SoftwareCoreference_Deposition
12	I-OperatingSystem_Usage	26	I-SoftwareCoreference_Deposition
13	B-PlugIn_Creation	27	O
14	I-PlugIn_Creation		

– **Stage 1 - Entity extraction**: This stage aims to identify whether each token in a given input sentence belongs to an entity or not. We achieve this through token classification, similar to Approach 1. However, instead of using 27 labels for different token types, we only use 3 labels as:
  - **O**: Non-entity token
  - **B-X**: Beginning token of an entity of type X (where X represents one of the 13 entity types)
  - **I-X**: Token within an entity of type X

  Using separate labels for the beginning (B) and inside (I) positions of tokens within an entity allows us to efficiently extract all words belonging to the same entity in stage 2.
– **Stage 2 - Entity classification:** In this stage, we classify the detected entities from stage 1. We use a classifier with 13 labels corresponding to the 13 entity types, discarding the B-I prefix distinction used for token position. This classifier is built by fine-tuning a transformer-based model like BERT. [14] During fine-tuning for classification tasks, it's common practice to use the hidden state associated with the [CLS] token as input for a classifier. However, in this approach, we fine-tune the entire transformer model end-to-end. This means the hidden states are not treated as fixed features, but are trained alongside the classification head (a component added on top of the pre-trained model) for optimal performance. Additionally, to leverage the knowledge of transformer models, we format this classifier as a question-and-answering model by constructing the input as the following prompt:
  - **Input**: What is <entity> in the sentence: <input sentence>
  - **Output**: Type of entity

Table 2. General statistics in the training set and private test set

Information	Training set	Private test set
#Sentence	39768	8180
#Sentence with entity	2353	374
Total entity	3241	515
Max length	568	347
Avg length	28.32	28.82

3.3 Approach 3: Three-stage framework

Our analysis in Table 2 revealed a limited number of sentences containing entities within the training set. This disparity raised concerns about potential biases in the label information during the training process for the previously mentioned approaches. To address this, we introduce a new three-stage framework, which integrate a binary classification with Approach 2. We simply built a binary classification model to detect the sentences which contain the entity. As shown in Figure 1, if a sentence is classified as class 0, assign all tokens in the sentence as O, otherwise, this sentence will be passed to Approach 2 to extract the entity and its type.

4 Experimental Setup

4.1 Data and Evaluation Metrics

This shared-task uses the SoMeSci dataset [13] which included 39768 sentences and 3756 software mentions divided into a training set and a private test set. We train our systems only on the training set and evaluate the performance of our model on the private test set using weighted precision, recall, and F1-score. In Table 2, we summarize some general information about the two data sets. Where #Sentence denotes the number of sentences, #Sentence with entity denotes the number of sentences containing the entity, and Total entity is the total of entities in all sentences. Max length and Avg length are the maximum length and average length of the sentences in each set, respectively. This dataset contains six groups of entity Application, OperatingSystem, PlugIn, ProgrammingEnvironment, and SoftwareConference. Each group can have the entity belong to four types [Creation, Deposition, Mention, Usage]. In Table 3 we indicate the distribution of each entity in the dataset

4.2 System Settings

We conduct all experiments on three approaches, using three base-version backbones: XLM-R[4], BERT[5], and SciBERT[6]. We loaded the weights of the backbones from the HuggingFace library and carried out training on an NVIDIA

[4] https://huggingface.co/FacebookAI/xlm-roberta-base
[5] https://huggingface.co/google-bert/bert-base-uncased
[6] https://huggingface.co/allenai/scibert-scivocab-uncased

Table 3. Statistics the number of entities in each entity type entity in each entity group in the training set and private test set

Entity group	**Entity**	**Training set**		**Testing set**	
		Quantity	**Total**	**Quantity**	**Total**
Application	Application_Creation	150	2353	47	348
	Application_Deposition	80		22	
	Application_Mention	162		31	
	Application_Usage	1958		248	
OperatingSystem	OperatingSystem_Mention	13	140	17	33
	OperatingSystem_Usage	127		16	
PlugIn	PlugIn_Creation	53	344	17	81
	PlugIn_Deposition	21		8	
	PlugIn_Mention	40		11	
	PlugIn_Usage	230		45	
ProgrammingEnvironment	ProgrammingEnvironment_Mention	41	372	6	49
	ProgrammingEnvironment_Usage	331		43	
SoftwareCoreference	SoftwareCoreference_Deposition	35	35	4	4

T4(x2) GPU provided by Kaggle. The corresponding hyper-parameters for each approach are presented below:

- **Approach 1:** batch size = 32, learning rate = 5e-05, and the number of epoch = 25 with XLM-R model and the number of epoch = 20 both remain backbones.
- **Approach 2:**
  - Stage 1: batch size = 32, learning rate = 5e-05 and the number of epoch = 20 for all three backbones.
  - Stage 2: batch size = 16, learning rate = 2e-05 and the number of epoch = 25 with XLM-R model and epoch = 20 two remainder models.
- **Approach 3:**
  - Stage 1: batch size = 32, learning rate = 2e-5 and the number of epoch = 10 for all three backbones.
  - Stage 2 and Stage 3: Using the configuration and architecture as the Approach 2.

5 Main results

According to the organizing committee, this sub-task will be evaluated by F1-Score based on exact matches. As shown in Table 4, we provide a tabulated summary of 9 experiments, each representing one of the 9 final systems generated from three different approaches and using three distinct backbones.

The experimental results in Table 4 indicate that Approach 3, a three-stage system, demonstrates the best performance across all backbones, with the XLM-RoBERTa backbone exhibiting the highest efficacy among all approaches. However, this result is for reference only and is only true in all of my experiments. It's important to acknowledge that different contexts, set up or datasets might yield different outcomes, and we are not sure this is the best result that each

Table 4. Comparative performance of our three Approaches with different pre-trained language models on the test set.

Models	**Approach 1**			**Approach 2**			**Approach 3**		
	Precision	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
BERT	0.675	0.594	0.625	0.682	0.643	0.653	0.690	0.629	0.650
SciBERT	0.658	0.621	0.623	0.719	0.645	0.670	0.736	0.631	0.670
XLM-R	0.716	0.614	**0.649**	0.707	0.654	**0.671**	0.729	0.649	**0.678**

Table 5. Official scoreboard[7]for the sub-task I: Software mention recognition.

Participant	**Ranking**	**Evaluation metrics**		
		Precision	**Recall**	**F1-score**
phinx	Top 1	**0.761**	**0.750**	**0.740**
david-s477	Top 2	0.739	0.711	0.692
ottowg	Top 4	0.679	0.664	0.652
vampire	Top 5	0.682	0.637	0.648
Our best system	Top 3	0.729	0.649	0.678

backbone could give in other cases. Finally, the best system was built according to approach 3 with XLM-R backbone and our best submission was ranked 3rd. Table 5 show the final score of the top 5 participants.

With the test dataset labels provided by the organizing committee, we evaluated the performance of our best system for each entity class in Table 6. We observed that the SoftwareCoreference_Deposition entity achieved the highest Precision score, while the ProgrammingEnvironment_Usage entity attained the highest Recall and F1 score, top 5 F1-score classes are ProgrammingEnvironment_Usage, SoftwareCoreference_Deposition, and OperatingSystem_Mention. It is evident that entities belonging to the PlugIn group typically scored lower than those in other groups shows that it has difficulty in the regconization process. Although, the number of PlugIn_Usage entities in the training set is pretty large the result on the test set is not positive. Besides that, PlugIn_Creation and PlugIn_Deposition entities have the sample in the training set are pretty low and their score moves forward to zero. The number of OperatingSystem_Mention entities in the training set is low and the score on the test set is high so we predict the mention entity type in this group is featured and easier to recognize than other groups.

Additionally, in Table 7, we evaluated each individual stage in our final three-stage system by assuming that the accuracy of the stages before it is 100%. The first stage works well with an F1-score of 0.992 in classifying whether a sentence contains an entity or not. Moving to stage 2, tasked with detecting entities in sentences, achieved an F1 score at a relatively good level, but a significant difference between Precision and Recall (12.6% difference) is evident, which also affects the overall system performance. In the final stage, the scores between the three metrics are relatively balanced, but it appears that the task of classifying

[7] https://codalab.lisn.upsaclay.fr/competitions/16935#results

Table 6. Performance of the final system on the test dataset across entity classes evaluated by Precision, Recall, and F1-score.

Entity class	**Precision**	**Recall**	**F1-score**
Application_Creation	0.692	0.766	0.727
Application_Deposition	0.615	0.727	0.667
Application_Mention	0.560	0.452	0.500
Application_Usage	0.812	0.730	0.769
OperatingSystem_Mention	0.867	0.765	0.812
OperatingSystem_Usage	0.579	0.688	0.629
PlugIn_Creation	0.200	0.059	0.091
PlugIn_Deposition	0.000	0.000	0.000
PlugIn_Mention	0.667	0.364	0.471
PlugIn_Usage	0.682	0.333	0.448
ProgrammingEnvironment_Mention	0.500	0.167	0.250
ProgrammingEnvironment_Usage	0.886	**0.907**	**0.897**
SoftwareCoreference_Deposition	**1.000**	0.750	0.857

Table 7. Performance of components in our final three-stage framework.

Stage	**Precision**	**Recall**	**F1-score**
Stage 1	0.992	0.992	0.992
Stage 2	0.912	0.786	0.845
Stage 3	0.786	0.806	0.784

13 entity classes had some impact on this stage with relatively lower overall performance. The propagation of errors between the three stages has a significant impact on the entire system, with the final F1-score of the entire system being 0.678.

6 Conclusion and Future Work

In this paper, we present and evaluate three approaches for tackling sub-task I in the Software Mention Detection in Scholarly Publications shared task. While we explored the use of suitable transformer models like BERT, our three-stage system leveraging the XLM-R model achieved the highest performance in the competition. As a result, our best system achieved the Top 3 in the private test. In future work, our intention is to analyze the error propagation between the three stages to enhance the performance of the entire three-stage system. Additionally, with access to more substantial computational resources, we aim to experiment with fine-tuning sub-tasks using larger batch sizes and epochs for each backbone in order to investigate the effects of these hyper-parameters on the model's performance.

Acknowledgements

This research was supported by The VNUHCM-University of Information Technology's Scientific Research Support Fund. We also thank the anonymous reviewers for their valuable comments on our manuscript.

References

1. Arora, J., Park, Y.: Split-NER: Named entity recognition via two question-answering-based classifications. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 416–426. Association for Computational Linguistics, Toronto, Canada (Jul 2023). https://doi.org/10.18653/v1/2023.acl-short.36, https://aclanthology.org/2023.acl-short.36
2. Beltagy, I., Lo, K., Cohan, A.: SciBERT: A pretrained language model for scientific text. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 3615–3620. Association for Computational Linguistics, Hong Kong, China (Nov 2019). https://doi.org/10.18653/v1/D19-1371, https://aclanthology.org/D19-1371
3. Chen, T., Qian, Y., Wang, Y., Chen, X., Ouyang, D., Dong, S., Li, X., Zhao, J., Huang, L.: Robert-agr: An entity relationship extraction model of massive agricultural text based on the roberta and crf algorithm. In: 2023 IEEE 8th International Conference on Big Data Analytics (ICBDA). pp. 113–120. IEEE (2023)
4. Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., Stoyanov, V.: Unsupervised cross-lingual representation learning at scale. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J. (eds.) Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 8440–8451. Association for Computational Linguistics, Online (Jul 2020). https://doi.org/10.18653/v1/2020.acl-main.747, https://aclanthology.org/2020.acl-main.747
5. Dash, A., Darshana, S., Yadav, D.K., Gupta, V.: A clinical named entity recognition model using pretrained word embedding and deep neural networks. Decision Analytics Journal p. 100426 (2024)
6. Derczynski, L., Nichols, E., van Erp, M., Limsopatham, N.: Results of the WNUT2017 shared task on novel and emerging entity recognition. In: Proceedings of the 3rd Workshop on Noisy User-generated Text. pp. 140–147. Association for Computational Linguistics, Copenhagen, Denmark (Sep 2017). https://doi.org/10.18653/v1/W17-4418, https://www.aclweb.org/anthology/W17-4418
7. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) 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). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-1423, https://aclanthology.org/N19-1423

8. Elena Simperl, Peter Clark, K.K.: Natural scientific language processing and research knowledge graphs. In: Lecture Notes in Artificial Intelligence (2024)
9. Li, L., Zhou, R., Huang, D.: Two-phase biomedical named entity recognition using crfs. Computational biology and chemistry **33**(4), 334–338 (2009)
10. Lopez, P., Du, C., Cohoon, J., Ram, K., Howison, J.: Mining software entities in scientific literature: document-level ner for an extremely imbalance and large-scale task. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management. pp. 3986–3995 (2021)
11. Luo, L., Yang, Z., Yang, P., Zhang, Y., Wang, L., Lin, H., Wang, J.: An attention-based bilstm-crf approach to document-level chemical named entity recognition. Bioinformatics **34**(8), 1381–1388 (2018)
12. Pradhan, S., Moschitti, A., Xue, N., Ng, H.T., Björkelund, A., Uryupina, O., Zhang, Y., Zhong, Z.: Towards robust linguistic analysis using OntoNotes. In: Proceedings of the Seventeenth Conference on Computational Natural Language Learning. pp. 143–152. Association for Computational Linguistics, Sofia, Bulgaria (Aug 2013), https://aclanthology.org/W13-3516
13. 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)
14. Tunstall, L., Werra, L.v., Wolf, T.: Natural language processing with transformers: Building language applications with hugging face. O'Reilly (2022)
15. 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)
16. Zhang, H., Li, X., Gu, R., Qu, X., Meng, X., Hu, S., Liu, S.: Samsung research china-beijing at semeval-2023 task 2: An al-r model for multilingual complex named entity recognition. In: Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023). pp. 114–120 (2023)
17. Zhang, Z., Zhao, Y., Gao, H., Hu, M.: Linkner: Linking local named entity recognition models to large language models using uncertainty. arXiv preprint arXiv:2402.10573 (2024)
18. Zhou, W., Zhang, S., Gu, Y., Chen, M., Poon, H.: Universalner: Targeted distillation from large language models for open named entity recognition. In: The Twelfth International Conference on Learning Representations (2023)