AMI Team at the NTCIR-16 Real-MedNLP Task

Satoshi Hiai
Advanced Media, Inc.
Japan
s-hiai@advanced-media.co.jp

Shoji Nagayama
Advanced Media, Inc.
Japan
s-nagayama@advanced-media.co.jp

Atsushi Kojima
Advanced Media, Inc.
Japan
a-kojima@advanced-media.co.jp

ABSTRACT

The AMI team participated in subtasks 1 and 2 of the NTCIR-16 Real-MedNLP Task. In this paper, we report our systems employed for subtasks 1 and 2. In subtask 1, the organizer provides a small amount of training data. In recent years, the approach based on BERT has achieved excellent results for such a low-resource situation. We construct two systems based on the BERT model pretrained on biomedical documents (UTH-BERT). We construct the ensemble method with hidden vectors from multiple layers of UTH-BERT and the fine-tuning method with the CRF layer. In subtask 2, participants construct their methods based on the annotation guideline. We construct a multistage method to identify named entities. The system consists of three stages: a candidate extraction stage, an identification stage, and a tag correction stage. We discuss the effectiveness of our systems on the basis of our preliminary experiments and the results in the formal run.

KEYWORDS

Biomedical named entity recognition, ensemble model, fine-tuning of BERT, multistage method

TEAM NAME

AMI

SUBTASKS

Subtask1-CR-JA, Subtask1-RR-JA, Subtask2-CR-JA, Subtask2-RR-JA

1 INTRODUCTION

The AMI team participated in subtasks 1 and 2 of the NTCIR-16 Real-MedNLP Task [16]. Participants of these subtasks construct named entity recognition (NER) systems. Participants of subtask 1 utilize a small annotated corpus to construct their systems. Participants of subtask 2 utilize the annotation guideline containing a handful of example sentences. In this paper, we report our approach to the subtasks.

1.1 Subtask 1

Participants of subtask 1 construct supervised methods based on the provided dataset. However, since the dataset is small, it is insufficient for machine learning methods. Therefore, participants should construct high-performance systems with supervised learning with few resources.

In recent years, the approach based on language models pretrained on unlabeled data has achieved excellent results for low-resource situations. Devlin et al. [3] proposed a BERT model pretrained on a large amount of unlabeled data showed excellent performance in various NLP tasks. There are two approaches based on the BERT model: the feature-based approach and the fine-tuning approach. The methods employing the feature-based approach utilize hidden vectors of the BERT model as a feature that represents input tokens. Devlin et al. used concatenations of the vectors from the last four hidden layers as features for an NER task. The methods based on the fine-tuning approach utilize the BERT model with one additional output layer for a downstream task. The output layer converts the size of the hidden vector to the number of NE classes. The output layer and the BERT model are fine-tuned on training data of downstream tasks. In the NER task, the model identifies the NE class label for each input token. Furthermore, the BERT model pretrained on the same domain as the downstream task often achieves higher performance. Beltagy et al. [1] and Lee et al. [8] constructed BERT models pretrained on the English biomedical domain corpus. They fine-tuned the BERT model on the dataset of biomedical downstream tasks. The fine-tuned BERT models showed excellent performance in various biomedical tasks. Kawazoe et al. [5] constructed BERT model pretrained on the Japanese biomedical domain corpus, named UTH-BERT. They fine-tuned the BERT model on the dataset of a Japanese biomedical text classification task [15] and obtained high performance. We also construct our methods on the basis of UTH-BERT.

In this research, we propose two methods based on UTH-BERT. One is an ensemble method based on UTH-BERT. Although Devlin et al. utilized the last four hidden layers for the NER task, it is not clear how many BERT hidden layers should be used in this task for good performance. Therefore, we, first construct models utilizing the last N layers of UTH-BERT with N = 1, 2, 3, 4. Here, unlike the feature-based approach of Devlin et al., each model is trained on the dataset of this task. Then, we propose ensemble methods with the four UTH-BERT models. The other method is a UTH-based method considering tag sequences. We utilize the CRF [7] layer in addition to the UTH-BERT model. In related works [1, 3, 5, 8], the BERT model and an output layer were trained on downstream tasks. In addition, it has been reported that the CRF layer improves the performance of the BERT model in the NER task [12]. Moreover, the models with the CRF layer achieved high performance in biomedical NER tasks [10]. Therefore, we construct the UTH-BERT model with the CRF layer to consider tag sequences.

1.2 Subtask 2

The participants of subtask 2 construct methods based on the annotation guideline for the construction of the subtask 1 dataset. The guideline contains descriptions for the annotation of each tag and a handful of sample sentences. It also encourages annotators to refer to external sources of knowledge such as disease name dictionaries to identify NEs. Therefore, the annotators learn the surfaces of NEs and the syntactic patterns before and after NEs on the basis

of external knowledge and the sample sentences, which are used to annotate NEs.

The participants of subtask 2 simulate the training of human annotators. Recently, machine learning approaches have achieved high performance. However, since no training data is given in this task, it is difficult to obtain high performance by a machine learning method. Therefore, a method based on external knowledge and handcrafted rules based on the surfaces of NEs and syntactic patterns is important for this task. As one example of an external source of knowledge, the guideline introduces the Manbyo dictionary[1], which contains approximately 380,000 disease names. However, since the extraction targets of NE classes are not only disease names, the coverage of the NEs is insufficient[2]. Therefore, we need to use other dictionaries and an augment method. In addition, the surfaces of NEs and syntactic patterns are important features. For example, surface information such as prefixes and suffixes plays an important role in estimating the type and features of a chemical. The syntactic patterns of combinations of numerical and unit expressions are also important keys to identifying medicine names and their prescribed amounts. However, since the the number of sample sentences in the guideline is small, we construct and augment the patterns to identify NEs. After the identification of NE tags for each token, the guideline contains some rules to merge continuous tags into one tag. For example, continuous <a> and <d> tags are often merged into the <d> tag. In the phrase "brain metastatis", although "brain" is an anatomical part (<a> tag) and "metastatis" is a disease name (<d> tag), "brain metastatis" is annotated as a <d> tag entity. Therefore, we should construct rules to merge continuous NEs.

In our system, we apply a multistage method to identify NEs. First, our method extracts candidate words from documents. In this stage, for the extraction, we apply the rules based on the surfaces and parts of speech of words. Additionally, we incorporate a machine learning method in this stage. Second, we apply identification methods for the candidate words. In this stage, we calculate NE scores for which we utilize syntactic pattern matching with regular expressions, dictionaries, and a similarity score based on word embeddings obtained with fasttext [2]. To solve the problem of insufficient coverage of the dictionaries, we apply an augmentation method. We augment the dictionaries with an unlabeled biomedical corpus using a bootstrap method [4]. Then, our system identifies the NEs from the calculated scores in the second stage. Finally, we apply merging rules for the continuous words identified as NEs.

2 METHODS

In this section, we describe our methods for subtasks 1 and 2 in detail.

2.1 Subtask 1

We construct the methods based on UTH-BERT. We describe the ensemble and fine-tuning methods in Sections 2.1.1 and 2.1.2, respectively. In both methods, we utilize UTH-BERT and a tokenizer published on the web[3]. The model consists of 12 transformer layers [14] and the hidden layer size is 768.

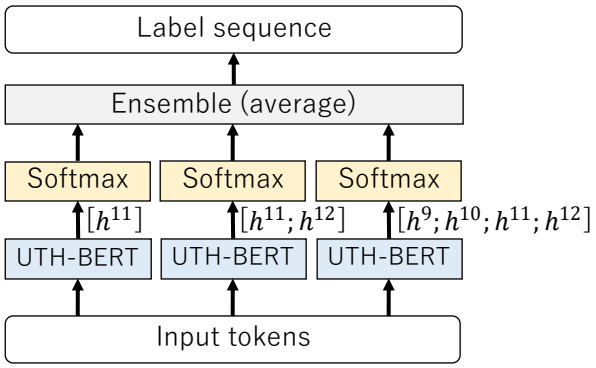

Figure 1: Overview of the ensemble method based on UTH-BERT.

2.1.1 Ensemble Method. In the ensemble method, we train a model consisting of an output layer and BERT layers by minimizing cross-entropy. In a typical approach, the output layer converts the hidden vector from the last layer of BERT to logits. In our system, we construct models utilizing the last N layers of UTH-BERT with N = 1, 2, 3, 4. Then, we ensemble UTH-BERT models. In our preliminary experiment, we evaluate which combinations of the four models are effective for NER on the NTCIR-16 dataset. We confirm that the ensemble of the models with N=1, 2, 4 is most effective. Therefore, we construct the ensemble model with N=1, 2, 4 for the formal run.

Figure. 1 shows an overview of our ensemble method. In this figure, $\boldsymbol{h}$ is a hidden vector from BERT. For instance, $\boldsymbol{h}^{11}$ represents the hidden vector from the 11th layer. To obtain hidden vectors, we use UTH-BERT consisting of 12 transformer layers. As the ensemble method, we simply apply probability averaging.

To train the output layer and BERT, we use the Transformer learning schedule [14]. We also use the Adam optimizer [11] and set $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-7}$.

2.1.2 Fine-Tuning Method. We fine-tune UTH-BERT with a CRF layer. Figure. 2 shows an overview of the method. We input the outputs of UTH-BERT for each token into the CRF layer. The CRF layer calculates the probability of input sequence $y = y_1, y_2, ..., y_n$ as

$$P = (e_1, e_2, ..., e_n)^T, \quad (1)$$

$$s(\mathbf{X}, \mathbf{y}) = \sum_{i=1}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} P_{i, y_i}, \quad (2)$$

$$p(\mathbf{y}|\mathbf{X}) = \frac{\exp(s(\mathbf{X}, \mathbf{y}))}{\sum_{\tilde{\mathbf{y}} \in \mathbf{Y_X}} \exp(s(\mathbf{X}, \tilde{\mathbf{y}}))}, \quad (3)$$

where $\mathbf{Y_X}$ is the set of all possible tag sequences and $A_{i,j}$ is the transition score from the ith tag to the jth tag. The CRF layer outputs a tag sequence $\mathbf{y}*$ that maximizes the score calculated as

$$\mathbf{y}* = \arg\max_{\tilde{\mathbf{y}} \in \mathbf{Y_X}} s(\mathbf{X}, \tilde{\mathbf{y}}). \quad (4)$$

[1] https://sociocom.naist.jp/manbyou-dic/

[2] We investigated the coverage of the NEs in the sample sentences in the guideline. The coverages are <d> = 44%, <a> = 13%, <m-key> = 7%, and <t-key> and <t-test> = 0%.

[3] https://ai-health.m.u-tokyo.ac.jp/home/research/UTH-BERT

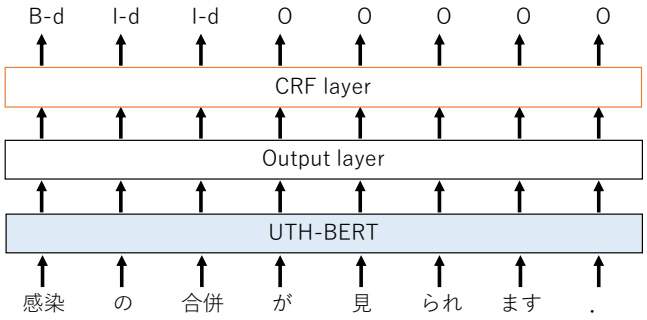

Figure 2: Overview of the fine-tuning method based on UTH-BERT.

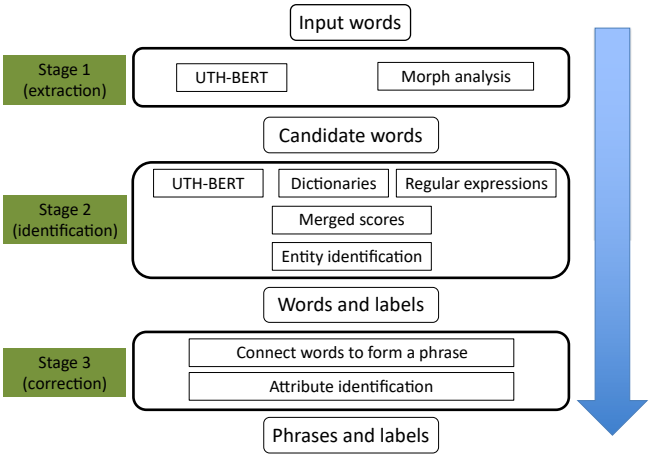

Figure 3: Outline of the guideline-based method.

For the NER task, we use the loss function

$$Loss_{NER} = log(p(\mathbf{y}|\mathbf{X})) = \mathbf{s}(\mathbf{X}, \mathbf{y}) - \log(\sum_{\tilde{\mathbf{y}} \in \mathbf{Y}_\mathbf{X}} \exp(\mathbf{s}(\mathbf{X}, \tilde{\mathbf{y}}))). \quad (5)$$

We use pytorch-crf[4] for the implementation. To train the model, we utilize the RAdam optimizer [9] with a learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-7}$. The batch size is 30. We utilize IOB2 tags [13] for the training. After the training, we modify the transition weight of the CRF layer to prevent the transition from O tags to I tags.

2.2 Subtask2

Figure. 3 shows the outline of our method, which consists of three stages: extraction, identification, and correction. We describe the extraction stage in Section 2.2.1, the identification stage in Section 2.2.2, and the correction stage in Section 2.2.3.

2.2.1 Extraction. In this section, we describe the extraction stage (stage 1 in Fig. 3). Figure. 4 shows the flow of this stage. We apply an extraction method based on the tag predicted from UTH-BERT and a part-of-speech tag of each word to extract candidate words. In the extraction method based on the tag predicted from UTH-BERT, first, UTH-BERT predicts tags for the input sequence. Then we extract the words predicted as NEs as the candidate NEs. We fine-tuned UTH-BERT on the sample sentences in the guideline. In the extraction method based on a part-of-speech tag of each word, we extract nouns as candidates NEs. We utilize the MeCab analyzer [6] with biomedical domain dictionaries namely Manbyo dictionary, Hyakuyaku[5], and comeJisyo[6]. We describe these dictionaries in the next section.

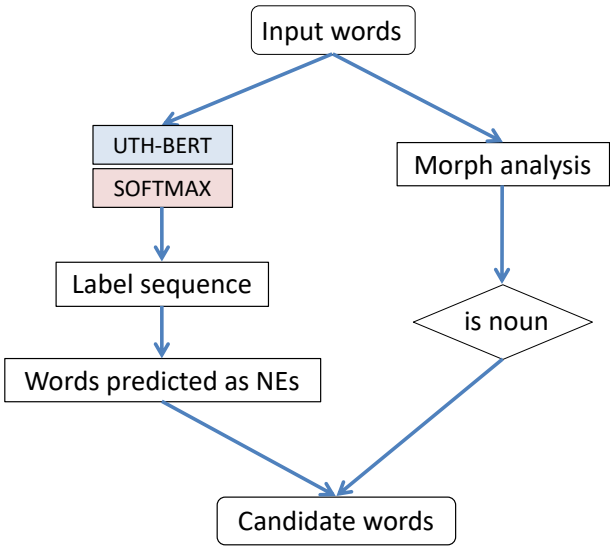

Figure 4: Flow of stage 1.

2.2.2 Identification. In this section, we describe the stage 2 in Fig. 3. In this stage, we decide the NE tag for the candidate words. To decide the entity tag, we calculate the entity scores for the candidate words. Then, we decide the entity tag of the words by the highest calculated score. We score candidate words using three methods. Note that we add scores using the scoring table shown in Table 1 in each method.

In the first method, we use the label obtained from UTH-BERT. We add the softmax score of each label from UTH-BERT model for candidate words.

In the second method, we add scores using dictionaries. We register the vocabularies of three dictionaries (Manbyo, Hyakuyaku, and comeJisyo) in userdict of MeCab. Then, we add scores by referring to which the dictionaries register candidate words. Note that we set different scores for each dictionary. In addition, we use similar words obtained with similar word embeddings, because some words are not contained in dictionaries owing to spelling inconsistencies. To obtain similar words, we use the embeddings obtained with fasttext. We use cosine similarity as the similarity metric. We regard the top-five words as similar words. We add different scores depending on the registered dictionary based on the scoring table shown in Table 2 similarly to the case of target recognition words.

[4]https://pytorch-crf.readthedocs.io/en/stable/

[5]https://sociocom.naist.jp/hyakuyaku-dic/

[6]https://ja.osdn.net/projects/comedic/

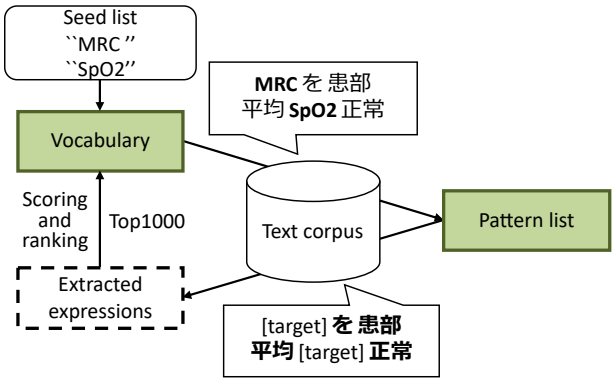

Figure 5: Flow of bootstrap method.

To train fasttext, we use iCorpus and our internal medical domain text corpus[7].

For disease entity recognition, we use Manbyo, in which symptoms and frequency information are registered. For medicine name entity recognition, we use Hyakuyaku, in which medicine names are registered. In addition, we use comeJisyo, in which medical terms are registered, to narrow down entity candidates to disease, body, and test entities. These dictionaries are not often effective for specific entities. In Manbyo, the coverage of disease entities is high, but that of other entities is low. For comeJisyo, categories of medical terms such as diseases, medicines, and tests are not registered, so we can not determine the category of medical terms registered in this dictionary. Therefore, we augment the vocabulary by a bootstrap method to determine entities of words excluding medicine names.

Figure. 5 shows the bootstrap method used for vocabulary augmentation. First, we extract the entity words in the guideline sentences and use the word list as a seed list. Then, we form a word-based trigram by tokenizing text using the dictionaries. We make patterns by extracting the text including words in the seed list and removing words from the patterns. We also record the positions of the removed words to consider the order of words. We extract substrings using the patterns in the corpus. Then, we extract the expressions for the positions of the removed words in the extracted substrings. For each extracted expression, we calculate cosine similarity scores between the fasttext embedding of the extracted expression and the fasttext embeddings of each word in the vocabulary. We regard the sum of the scores as the score for the extracted expressions. To update the vocabulary, we add the extracted words with the top 1,000 scores to the vocabulary. We iterate the process until the score becomes smaller than the threshold or the number of updates exceeds a specific value. We add vocabularies in each entity category by choosing words from the augmented vocabularies manually. In this system, we utilize iCorpus for the text corpus and augment the <t-key> vocabulary.

In the third method, we use regular expressions to add scores. To construct regular expressions, we use vocabularies obtained by the bootstrap method, the sample sentences in the guideline, and the sentences in iCorpus and a website[8]. When a candidate word matches any regular expression, we add the scores for the candidate word. We also use similar words to the candidate word to match regular expressions. We show the number of regular expressions in Table 3.

We determine the entity of a candidate word by calculating the highest total score among the three methods.

Table 1: Scoring table for the candidate words.

scoring table	Candidate word				
Tag	BERT	Regular Expression	Dictionary		
			Manbyo	Hyakuyaku	comeJisyo
a	10	1	0	0	0
d	10	100	100	0	1
m-key	10	100	0	100	1
m-val	10	100	0	0	0
t-key	10	100	0	0	1
t-test	10	100	0	0	1
t-val	10	100	0	0	0
timex3	10	100	0	0	0

Table 2: Similar words scoring table for the candidate words.

Scoring table	Similar word			
Tag	Regular expression	Dictionary		
		Manbyo	Hyakuyaku	comeJisyo
a	10	0	0	1
d	10	1	0	1
m-key	10	0	1	1
m-val	10	0	0	0
t-key	10	0	0	1
t-test	10	0	0	1
t-val	10	0	0	0
timex3	10	0	0	0

Table 3: Numbers of regular expressions.

Tag	Number
a	355
d	3
m-key	7915
m-val	16
t-key	179
t-test	10
t-val	104
timex3	9

2.2.3 Correction. In this section we discuss stage 3 in Fig. 3. Figure. 6 shows the flow of this stage. To regard series of candidate words as one NE, candidate words are merged in this stage. In addition, we re-identify NEs and identify attributes. Candidate words are merged to obtain candidate phrases if a series of candidate words exist in one clause. We regard a candidate word as a candidate phrase if that word only exists in one clause. The NE of a candidate phrase is identified by considering NEs composed of

[7] This corpus contains approximately 2GB of medical texts.

[8] mhlw.go.jp/file/06-Seisakujouhou-10800000-Iseikyoku/0000209868.pdf

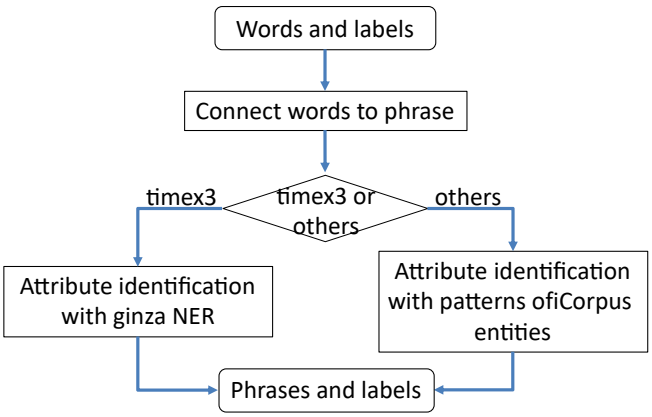

Figure 6: Flow of stage 3.

Table 4: Correspondence between attributes of Real-MedNLP and iCorpus NEs.

Real-MedNLP	iCorpus
d_negative	judge_Negative
d_positive	judge_Positive
d_suspicious	judge_Suspicious
{t-test,m-key}_executed	executed_Done
{t-test,m-key}_negated	executed_Didnot
{t-test,m-key}_scheduled	executed_Scheduled

Table 5: Numbers of regular expressions for tag attributes.

Tag	Attribute	Number
d	negative	19
	positive	93
	suspicious	89
{t-test, m-key}	executed	234
	negated	72
	scheduled	52

Table 6: Correspondence between attributes of Real-MedNLP and ginza NEs.

Real-MedNLP	ginza
timex3_age	Age
timex3_date	Period_Year, Period_Month Period_Week, Period_Day,Date
timex3_duration	Period_Time
timex3_time	Time

candidate words. Normally, the NE of a candidate phrase is the same as last of those composed of candidate words. When a phrase contains a numerical expression, we identify the NE of the phrase by pattern matching.

Next, we identify attributes of a candidate phrase. We identify the phrases that have <d>, <t-test>, or <m-key> tags to use patterns of iCorpus NEs. iCorpus NEs have very similar patterns to the attributes of Real-MedNLP NEs. Table. 4 shows the correspondence. The attributes of <d>, <t-test>, or <m-key> tags are not identified for the expression of a candidate phrase, but for clue expressions existing before and after the phrase. On the other hand, entities in iCorpus are identified for the expression of the candidate phrase. In fact, expressions of phrases with an iCorpus NE do not correspond to those with <d>, <t-test>, or <m-key> tags, but to clue expressions existing before and after the phrase. We use vocabularies obtained by iCorpus's expressions with corresponding entities. We show the numbers of regular expressions in Table 5. The expressions existing before and after the phrase matches regular expression, we identify the attributes of that phrase.

The phrases with the <timex3> tag are identified by using the ginza NER. The ginza NER has a very similar pattern to the attributes of the <timex3> tag. Table. 6 shows the correspondence. Unlike in the case of using iCorpus, the expressions of phrases with ginza's entities correspond to those with the <timex3> tag. We carry out ginza NER analysis for the candidate phrase and identify its attributes based on the result of the ginza NER. Using this process, we obtain the classification target phrase and identify its entity.

3 EXPERIMENTS

In this section, we describe experiments performed to evaluate our systems. In Section 3.1, we describe our preliminary experiment before the formal run. In the preliminary experiment, for subtask 1, we split the given development dataset into a training dataset and an evaluation dataset. For subtask 2, we create a small evaluation dataset to evaluate our systems. In Section 3.2.2, we describe the results of our systems in the formal run.

3.1 Evaluation on Development Data

3.1.1 UTH-BERT-Based Methods for Subtask 1. We evaluated our systems based on UTH-BERT (ensemble and fine-tuning approaches). for the evaluation dataset, we used given data from CR and RR tasks. We divided the data into training 90% and evaluation 10% datasets. The training dataset contained 134 CR documents and 65 RR documents. The evaluation dataset contained 14 CR documents and seven RR documents. We report precision, the recall and the F1-measure. Table 7 shows the results of our systems on the evaluation dataset. The ensemble method outperformed the fine-tuning method in terms of the F1-measures of the a, d, t-test, and t-val tags in the CR and RR evaluation datasets. This result suggests that vectors from multiple hidden layers are useful for identifying these NEs. As future work, we can analyze the model more deeply by analyzing the relationship between the utilized layer and the NE class in which the layer is effective for estimation. The fine-tuning method outperformed the ensemble method in terms of the F1-measures of the m-key and t-val tags in the CR evaluation dataset. The fine-tuning method utilized the CRF layer. In the NER with UTH-BERT, the CRF layer is useful for considering the sequence to identify NEs consisting of many tokens. The average number of tokens of the m-key and m-val entities tokenized with the UTH-BERT tokenizer in the evaluation dataset was larger than that of other tags. Therefore, UTH-BERT with the CRF layer appears to be useful for identifying these NEs.

Table 7: Evaluation results of proposed methods for subtask 1 on development data. Owing to the small amount of evaluation data, the results of some tags were extremely high or low. Bold values indicate the better of the results of the ensemble and fine-tuning methods.

Methods	Tag	CR-JA			RR-JA		
		P	R	F1	P	R	F1
Ensemble	a	**69.32**	**70**	**69.61**	**100**	**100**	**100**
	d	**80.6**	**81.0**	**80.8**	**88.24**	**86.54**	**87.38**
	m-key	**62.5**	58.82	60.61	-	-	-
	m-val	0	0	0	-	-	-
	t-key	**63.64**	33.33	43.75	-	-	-
	t-test	**86.96**	**95.23**	**90.91**	**100**	**100**	**100**
	t-val	**75**	**37.50**	**50**	-	-	-
	timex3	83.33	86.73	85	**100**	**100**	**100**
Fine-tuning	a	62.00	68.89	65.26	91.67	95.65	93.62
	d	64.61	78.50	70.88	80.00	84.62	82.24
	m-key	61.11	**64.71**	**62.86**	-	-	-
	m-val	**100**	**33.33**	**50.00**	-	-	-
	t-key	50.00	**42.86**	**46.15**	-	-	-
	t-test	73.08	90.48	80.85	**100**	**100**	**100**
	t-val	50.00	**37.50**	42.86	-	-	-
	timex3	**86.00**	**87.76**	**86.87**	**100**	**100**	**100**

Table 8: Development dataset for subtask 2.

Tag	# of sample	P	R	F1
a	82	45.10	56.10	50.0
d_positive	162	32.31	58.64	41.67
d_negative	45	50.0	2.22	4.26
d_suspicious	10	0	0	0
d_general	10	0	0	0
m-key_scheduled	0	-	-	-
m-key_executed	17	31.59	70.59	43.64
m-key_negated	1	0	0	0
m-key_other	1	0	0	0
m-val	14	15.63	35.71	21.74
t-key	173	48.55	48.55	48.55
t-test_executed	85	47.0	55.29	50.81
t-test_negated	4	0	0	0
t-test_other	0	-	-	-
t-val	179	39.24	51.96	44.71
timex3_date	23	8.16	69.57	14.61
timex3_time	0	-	-	-
timex3_duration	9	0	0	0
timex3_set	1	0	0	0
timex3_age	8	63.64	87.50	73.68
timex3_med	20	0	0	0
timex3_misc	0	-	-	-

3.1.2 Guideline Based Method for Subtask 2. We evaluated our system for subtask 2. We created a small annotated dataset for the evaluation using iCorpus. One of the annotators annotated the two documents in iCorpus. He annotated <a>, <d>, <t-key>, <t-val>, <t-test>, <m-key>, <m-val>, and <timex3> tags on the basis of the guideline. Since this data was used in the preliminary experiment, we did not evaluate the dataset quality. We report precision, recall, and F1-measure on this evaluation dataset. Table 8 shows the statistics of the dataset and the experimental results.

3.2 Formal Run

In this section, we describe the evaluation results of our systems in the formal run. We describe the results for subtasks in Section 3.2.1 and 3.2.2, respectively.

3.2.1 Subtask 1. In this section, we describe the evaluation results of our systems for subtask 1. We submitted two systems, AMI-1 and AMI-2. AMI-1 is the ensemble method based on UTH-BERT described in Section 2.1.1 and AMI-2 is the fine-tuning method based on UTH-BERT described in Section 2.1.2. The organizer evaluated the submitted systems with character level and entity-level metrics. The character-level metric corresponded to the accuracy. The organizer calculated the F1-measures of all the entities and tags. Table 9 show the evaluation results. Unlike the results of the preliminary experiment in Section 3.1.1, the ensemble method (AMI-1) had lower performance than the fine-tuning method (AMI-2) on almost all metrics. Since this result is strong contrast to the result of the preliminary experiment, the submitted result of AMI-1 may include formatting errors.

3.2.2 Subtask 2. In this section, we describe the evaluation result of our system on the formal run of subtask 2. We submitted only using AMI-1, as described in Section 2.2, using UTH-BERT in our method with the model being fine-tuned using the sample sentences in the guideline. In our system submitted to the formal run, we added the annotated dataset created from iCorpus described in Section 3.1.2 to the training dataset and re-fine-tuned the UTH-BERT model a second time. The organizer evaluated the submitted systems with the metrics described in Section 3.2.1. Additionally, our system for subtask 2 also identified the attributes of

Table 9: Evaluation results of proposed methods for subtask 1 in formal run.

Evaluation metrics		CR-JA		RR-JA	
		AMI-1	AMI-2	AMI-1	AMI-2
Character-Accuracy (entity)	(All target entities)	83.17	88.40	86.53	96.47
Entity-Precision	(All target entities)	55.22	57.81	11.06	89.07
Entity-Recall	(All target entities)	59.04	62.29	23.53	89.45
Entity-F1	(All target entities)	57.07	59.96	15.05	89.26
	a	58.37	58.43	33.58	89.16
	d	67.05	67.05	7.88	89.40
	m-key	70.63	70.39	-	-
	m-val	65.67	65.67	-	-
	t-key	35.76	35.55	-	-
	t-test	43.58	43.38	0.00	87.50
	t-val	55.48	55.68	-	-
	timex3	74.62	74.39	24.49	88.24

Table 10: Evaluation results of proposed methods for subtask 2 in formal run

Evaluation metric		CR-JA	RR-JA
		AMI-1	AMI-1
Character-Accuracy (entity)	(All target entities)	71.46	89.24
Entity-Precision	(All target entities)	30.90	60.50
Entity-Recall	(All target entities)	34.81	69.87
Entity-F1	(All target entities)	32.74	64.85
	a	41.52	56.89
	d	41.68	68.45
	m-key	40.00	-
	m-val	22.38	-
	t-key	37.20	-
	t-test	28.17	81.25
	t-val	34.66	-
	timex3	35.02	74.42

Table 11: Joint evaluation results of proposed methods for subtask 2 in formal run

Evaluation metric		CR-JA	RR-JA
		AMI-1	AMI-1
Character-Accuracy (joint)	(All target entities + attributes)	67.29	81.98
Joint-Precision	(All target entities + attributes)	24.67	47.89
Joint-Recall	(All target entities + attributes)	27.79	55.31
Joint-F1	(All target entities + attributes)	26.14	51.33
	a	41.52	56.89
	d	0	0
	d_positive	34.56	56.78
	d_suspicious	39.02	50.53
	d_negative	2.27	0
	d_general	0	0
	m-key_scheduled	0	-
	m-key_executed	29.73	-
	m-key_negated	20	-
	m-key_other	0	-
	m-val	21.43	-
	t-key	37.2	-
	t-test_executed	28.26	54.9
	t-test_negated	-	0
	t-test_other	0	0
	t-val	34.66	-
	timex3_date	20.2	68.29
	timex3_time	12.5	-
	timex3_duration	0	0
	timex3_set	0	-
	timex3_age	72.64	-
	timex3_med	11.25	0
	timex3_misc	0	-

entities. The organizers evaluated the entity + attribute identification results with the metrics of character-level accuracy, precision, recall, and F1-measure. Tables 10 and 11 shows the entity evaluation and entity + attribute evaluation result, respectively.

4 CONCLUSIONS

In this paper, we described our systems employed for subtasks 1 and 2 of the NTCIR-16 Real-MedNLP Task. We constructed two systems for subtask 1. One used the ensemble method, which employs ensembles of several models that utilize different numbers of UTH-BERT hidden layer vectors. The other used the fine-tuning method, in which UTH-BERT is fine-tuned with the CRF layer and a model considering the tag sequence is constructed.

We discussed the effectiveness of the utilization of multiple hidden layers and the CRF layer on the basis of the results of a preliminary experiment and evaluation results in the formal run.

For subtask 2, we constructed a multistage method to identify NEs. The method consisted of three stages. First, we applied an extraction method to extract the candidates of NE words. Second, we applied a scoring method to the candidate NE words and identified their NE tags on the basis of the scores. Finally, we applied merging rules for the continuous words identified as NEs.

REFERENCES

[1] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. 3615–3620.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. 4171–4186.

[4] Marti A. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*. 539–545.

[5] Yoshimasa Kawazoe, Daisaku Shibata, Emiko Shinohara, Eiji Aramaki, and Kazuhiko Ohe. 2021. A Clinical Specific BERT Developed using a Huge Japanese Clinical Text Corpus. *PLoS ONE* 16, 11 (2021), 1–11.

[6] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying Conditional Random Fields to Japanese Morphological Analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. 230–237.

[7] John Lafferty, Andrew McCallum, and Fernando C.N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of International Conference on Machine Learning*. 282–289.

[8] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: A Pre-trained Biomedical Language Representation Model for Biomedical Text Mining. *Bioinformatics* 36, 4 (2019), 1234–1240.

[9] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. On the Variance of the Adaptive Learning Rate and Beyond. In *Proceedings of the International Conference on Learning Representations*.

[10] Ling Luo, Zhihao Yang, Pei Yang, Yin Zhang, Lei Wang, Hongfei Lin, and Jian Wang. 2018. An Attention-based BiLSTM-CRF Approach to Document-level Chemical Named Entity Recognition. *Bioinformatics* 34, 8 (2018), 1381–1388.

[11] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations*.

[12] Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. 2019. Portuguese Named Entity Recognition using BERT-CRF. *arXiv preprint arXiv:1909.10649* (2019).

[13] Kim Tjong, Erik F Sang, and Veenstra Jorn. 1999. Representing Text Chunks. In *Proceedings of 9th Conference of the European Chapter of the Association for Computational Linguistics*. 173–179.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 30. 6000–6010.

[15] Shoko Wakamiya, Mizuki Morita, Yoshinobu Kano, Tomoko Ohkuma, and Eiji Aramaki. 2017. Overview of the NTCIR-13: MedWeb Task. In *Proceedings of the 13th NTCIR Conference on Evaluation of Information Access Technologies*. 40–49.

[16] Shuntaro Yada, Yuta Nakamura, Shoko Wakamiya, and Eiji Aramaki. 2022. Real-MedNLP: Overview of REAL Document-based MEDical Natural Language Processing Task. In *Proceeding of the 16th NTCIR Conference on Evaluation of Information Access Technologies*.