BiTeM at WNUT 2020 Shared Task-1: Named Entity Recognition over Wet Lab Protocols using an Ensemble of Contextual Language Models

Julien Knafou[1,2,3], Nona Naderi[1,2], Jenny Copara[1,2,3], Douglas Teodoro[1,2], and Patrick Ruch[1,2]

Emails : {firstname.lastname}@hesge.ch
[1]University of Applied Sciences and Arts of Western Switzerland
[2]Swiss Institute of Bioinformatics, Geneva, Switzerland
[3]University of Geneva, Switzerland

Abstract

Recent improvements in machine-reading technologies attracted much attention to automation problems and their possibilities. In this context, WNUT 2020 introduces a Name Entity Recognition (NER) task based on wet laboratory procedures. In this paper, we present a 3-step method based on deep neural language models that reported the best overall exact match F_1-score (77.99%) of the competition. By fine-tuning 10 times, 10 different pretrained language models, this work shows the advantage of having more models in an ensemble based on a majority of votes strategy. On top of that, having 100 different models allowed us to analyse the combinations of ensemble that demonstrated the impact of having multiple pretrained models versus fine-tuning a pretrained model multiple times.

1 Introduction

The last decades have seen both the amount and the complexity of biological experiments grow. Coupling this phenomenon with the improvement in machine-reading technologies seem to have led researchers to look for ways to automate wet laboratory procedures. Such technologies should allow reproducibility while reducing human errors in the process. However, as current protocols are usually written in a natural language, a collection of wet laboratory protocols annotated with entities and relations would help assess current machine-reading performances in this specific setting (Kulkarni et al., 2018).

In this context, WNUT (Workshop on Noisy User-generated Text[1]) 2020 (Tabassum et al., 2020) proposes two tasks, a Named Entity Recognition (NER) task and a Relation Extraction (RE) task. In this paper, we present a 3-step method we used for the NER task. Our approach is essentially based on a deep neural language models supported by transformer-like architectures (Vaswani et al., 2017). First, we fine-tuned 10 different pretrained language models on the downstream task. Then, we generated 10 instances of those pretrained models, each time with a new random initialization of the last layer, namely the classifier. Finally, we used an ensemble strategy based on a majority of votes. Our approach achieves the exact-match F_1-score of 77.99% that ranks first in the shared task.

[1] http://noisy-text.github.io/2020/wlp-task.html

2 Related work

Deep learning approaches trained on large unstructured data have shown considerable success in NLP problems, including NER (Devlin et al., 2019; Liu et al., 2019; Lample et al., 2016; Beltagy et al., 2019; Jin et al., 2019). These models use the learned representations over the large data and reuse them in a supervised setting for a downstream task. For domain-specific tasks, the models that are trained on large general text can be further trained on domain specific large data and then adapted for a downstream task (Lee et al., 2019; Gururangan et al., 2020; Alsentzer et al., 2019) or the models can be trained only on domain-specific data and then adapted for a specific task (Beltagy et al., 2019).

3 Data

The data provided for this task is a subset of Kulkarni *et al.*'s corpus (Kulkarni et al., 2018). The dataset consists of 615 unique protocols annotated with 17 types of entities and *action* (an example is shown in Figure 1).

The organizers provided a set of protocols for training, development, and test. They further released a final set of unlabelled protocols for test during the competition (called test 2020). Table 1 shows the way the dataset has been split into a

Proceedings of the 2020 EMNLP Workshop W-NUT: The Sixth Workshop on Noisy User-generated Text, pages 305–313
Online, Nov 19, 2020.

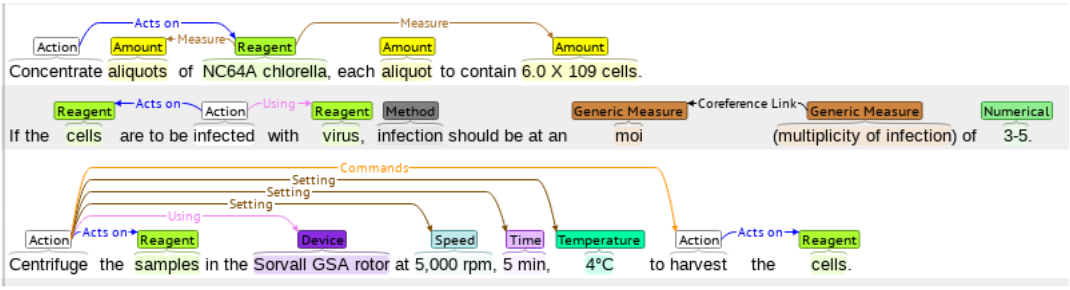

Figure 1: An example of the data.

Split	# of protocols
Train	370
Dev	123
Test	123
Test 2020	111
Total	727

Table 1: Number of protocols in WNUT-NER dataset.

Entity	Train		Dev		Test		Test 2020	
	Count	%	Count	%	Count	%	Count	%
Action	12,355	25.91	4,011	25.49	4,138	25.32	5,346	23.04
Amount	3,432	7.20	1,090	6.93	1,190	7.28	1,223	5.27
Concentration	1,330	2.79	422	2.68	535	3.27	701	3.02
Device	1,752	3.67	616	3.92	468	2.86	888	3.83
Generic-Measure	484	1.02	132	0.84	143	0.87	173	0.75
Location	3,921	8.23	1,396	8.87	1,326	8.11	1,657	7.14
Measure-Type	857	1.79	324	2.06	272	1.66	720	3.10
Mention	257	0.54	83	0.53	56	0.34	142	0.61
Method	1,597	3.36	538	3.43	581	3.56	1,059	4.56
Modifier	4,588	9.62	1,547	9.83	1,601	9.79	3,416	14.72
Numerical	832	1.75	259	1.65	231	1.41	513	2.21
Reagent	11,121	23.33	3,594	22.93	3,995	24.44	5,012	21.60
Seal	210	0.44	92	0.58	64	0.39	119	0.51
Size	262	0.55	123	0.78	113	0.69	232	1.00
Speed	626	1.31	239	1.52	167	1.02	238	1.03
Temperature	1,592	3.34	486	3.09	532	3.25	744	3.21
Time	2,396	5.02	745	4.74	870	5.32	951	4.10
pH	67	0.14	37	0.23	62	0.38	66	0.28
Total	47,679		15,734		16,344		23,200	

Table 2: Entity distribution across the dataset (based on the Standoff format).

training set, a development set,[2] a test set, and the competition test (test 2020).

In Table 2, we see the distribution of all the entities by each subset. As we can see, we have 18 entities and only two of them (*Action* and *Reagent*) represent about 50% of annotations. This table also shows us that entities' proportions are fairly similar across all the subsets.

4 Method

Our models essentially focused on transformers-like (Vaswani et al., 2017) language models that we fine-tuned on the NER task by adding a fully connected layer on top of the token representations. The models include BERT (cased) (Devlin et al., 2019), BioBERT (BERT trained on PubMed abstracts and PMC full-text articles) (Lee et al., 2019), Bio+ClinicalBERT (BioBERT trained on notes in the MIMIC-III v1.4 database) (Alsentzer et al., 2019), PubMedBERT (Gu et al., 2020), RoBERTa (Liu et al., 2019), BioMed RoBERTa (Gururangan et al., 2020), and XLNet (Yang et al., 2019).

Our method has been driven in 3 steps. First, we chose 10 different pretrained models and fine-tuned them on the downstream task. Then, using a voting strategy, we created ensemble models. Finally, we fine-tuned 9 more times each model, each time with a new random initialization of the fully connected layer, to see if sampling ensemble models from this set of models would improve the results even more.

4.1 Transformers with a fully connected layer on top of the token representations

In order to use transformers as a NER model, the only preprocessing we had to do was to break each protocol into sentences. Those sentences will then be the sequences that are fed into our model. As there were no overlapping entities in the text, we used a $softmax$ function which allowed us to classify each token to only one entity.

As transformers usually use tokenizers that work on word bits (or sub-tokens), we had to deal with it by assigning a dummy entity to each sub-token that was part of a word. In such cases, at training time, we only assign the true entity to the first sub-token. This allowed us to build back the original text quite easily. Indeed, during prediction, a word will get the highest probable entity label among all the sub-tokens' predictions of that word. In other words, the highest probable entity label will be assigned to all the sub-tokens of the word and the sub-tokens will be merged to build back the original word with the respective assigned label. Finally, in a given

[2]Protocol 621 (in the development set) is a duplicate of protocol 570 (in the train set), but their labels do not totally match.

Pretrained Models		Corpus type	# Parameters
BERT (Devlin et al., 2019)	base	General	110M
	large		340M
BioBERT (Lee et al., 2019)		Bio	110M
Bio+ClinicalBERT (Alsentzer et al., 2019)		Bio	110M
PubMedBERT (Gu et al., 2020)		Bio	110M
RoBERTa (Liu et al., 2019)	base	General	110M
	large		340M
BioMed RoBERTa (Gururangan et al., 2020)		Bio	110M
XLNet (Yang et al., 2019)	base	General	110M
	large		340M

Table 3: Pretrained models features

sequence, if two adjacent words were given the same entity prediction, we would consider the two words as a passage related to that entity.

Using the above setup, we fine-tuned 10 pretrained transformers for 10 epochs using an Adam optimizer (Kingma and Ba, 2014), a learning rate of $3e^{-5}$, a batch size of 24 and a maximum sequence length of 256 tokens. We used 1x T4 GPU for all base models and 2x T4 GPUs for the large ones. For a given model, it took in average roughly 16 minutes per epoch to train, thus about 2.67 hours for the 10 epochs. After each epoch, we predicted the development set, computed the F_1-score and saved the model if it improved the previous epoch score. Table 3 shows more information about all the pretrained models that we fine-tuned on the NER task. Indeed, 4 models out of 10 were trained on Biomedical corpus, such as PubMed and/or BioMed whereas the others were trained on general corpora, such as Wikipedia. Another key difference is the model type which defines the way a given model has been trained. This includes the training task (e.g., MLM, next sentence prediction, ...), the tokenizer algorithm, the optimizer and more. We used 5 different kinds of BERT-based, 3 of RoBERTa-based and 2 of XLNet-based models. For more details regarding the specifics of the architectures, please refer directly to their respective papers.

4.2 An ensemble based on a voting strategy

As implemented in (Copara et al., 2020b,a), our ensemble model strategy is based on a majority of votes. This means that for a given ensemble model composition, each composing model has the right to vote. In other words, for a given protocol and a given sequence, each model will return its predictions which can be interpreted as passage/entity combinations. Once we collected all models' predictions, we then counted all the passage/entity combinations and validated only those that had cast a majority of votes.

4.3 Sampling

Once we had all the models trained and ready, we were wondering if we could improve efficiency by adding more voters. The idea is to repeat the first step where each time we have a new random initialization on the fully connected layer. We ended up with 100 different models, corresponding to 10 different pretrained models fine-tuned 10 times.

With only a few models to choose from, we would have been able to predict all the possible model compositions; however, as using 50 models out of 100 would have resulted in about 10^{29} possible ensembles, we had to sample randomly ensemble model compositions. For each number of models taken into account in a given ensemble, we took a sample size of 1000 combinations. This will later allow us to show the results distribution of our ensemble models and examine how it will behave in certain circumstances.

The ensemble model we chose to use for the submission was the one that gave us the best F_1-score on the test set. It is a composition of 14 models that were fine-tuned on the task. It contained the following pretrained models: 2× BioBERT (BioBERT models with two different random initializations or seeds), 2× BioClinicalBERT (2 random seeds), 3× PubMedBERT (3 random seeds), 2× RoBERTa$_{base}$ (2 random seeds), 1× RoBERTa$_{large}$ 1× BioMed RoBERTa and 3× XLNet$_{large}$ (3 random seeds).

5 Results and Discussions

In Table 4, we see the F_1-score for all the 10 models we fine-tuned across all the 18 entities. The reported baseline is the CRF baseline[3] that was provided for the shared task. First, we can see that the ensemble model outperforms the baseline by far. When comparing all the models (ensemble apart), we also notice that PubMedBERT is quite consistent as it often outperforms all the other models, including the ensemble for a few entities, namely *Mention*, *Seal*, *Temperature* and *pH*. Additionally, when compared to its peers, it clearly shows the best micro and macro F_1-scores.

However, when looking at *Speed*, it seems that the transformers-based models we used are not able to do a better job than the baseline. A closer look at the errors should be done in order to see what caused such a difference with the baseline (see Section 5.4).

[3] https://github.com/jeniyat/WNUT_2020_NER/tree/master/code/baseline_CRF

Entity	BERT (cased)		BioClinical BERT	BioBERT	RoBERTa		BioMed RoBERTa	PubMed BERT	XLNet		Ensemble	Baseline
	base	*large*			*base*	*large*			*base*	*large*		
Action	$88.98_{0.2}$	$88.51_{0.2}$	$88.87_{0.2}$	$89.06_{0.2}$	$88.78_{0.3}$	$88.75_{0.2}$	$88.70_{0.2}$	$89.29_{0.3}$	$89.32_{0.1}$	$89.11_{0.3}$	**90.00** (0.68↑)	84.40
Amount	$85.73_{0.2}$	$85.14_{0.3}$	$85.21_{0.4}$	$85.96_{0.5}$	$85.59_{0.3}$	$85.53_{0.7}$	$85.99_{0.4}$	$85.66_{0.2}$	$85.00_{0.3}$	$84.47_{0.6}$	**86.41** (0.42↑)	84.19
Concentration	$82.61_{0.7}$	$81.83_{0.7}$	$82.31_{0.6}$	$83.14_{0.4}$	$82.68_{0.5}$	$82.14_{1.0}$	$83.34_{0.5}$	$83.81_{0.4}$	$82.25_{0.7}$	$82.40_{1.1}$	**84.64** (0.83↑)	78.49
Device	$62.96_{1.1}$	$62.18_{0.8}$	$63.57_{0.6}$	$64.24_{1.2}$	$63.39_{0.8}$	$64.21_{1.0}$	$63.44_{0.9}$	$64.17_{1.0}$	$63.27_{0.9}$	$63.65_{1.2}$	**67.11** (2.87↑)	58.67
Generic-Measure	$30.44_{1.6}$	$29.73_{1.9}$	$26.37_{1.8}$	$30.40_{1.7}$	$32.12_{1.1}$	$32.23_{2.0}$	$30.54_{1.7}$	$31.57_{1.8}$	$30.33_{1.3}$	$31.10_{2.1}$	**33.17** (0.94↑)	29.88
Location	$75.53_{0.4}$	$75.05_{0.4}$	$75.54_{0.3}$	$76.06_{0.7}$	$75.85_{0.5}$	$76.10_{0.6}$	$75.32_{0.6}$	$76.24_{0.3}$	$75.62_{0.3}$	$75.53_{1.0}$	**77.50** (1.26↑)	69.17
Measure-Type	$52.75_{1.4}$	$53.32_{1.4}$	$52.42_{1.7}$	$54.23_{1.1}$	$53.89_{1.6}$	$52.62_{1.1}$	$53.61_{0.9}$	$52.47_{0.6}$	$54.61_{1.3}$	$55.35_{2.0}$	**55.36** (0.01↑)	48.93
Mention	$71.00_{3.0}$	$67.15_{2.5}$	$68.40_{1.9}$	$68.18_{1.7}$	$68.70_{1.0}$	$70.35_{2.2}$	$68.44_{2.1}$	**72.35**$_{0.9}$	$68.66_{2.8}$	$63.02_{11.1}$	71.79 (0.56↓)	60.18
Method	$48.70_{1.3}$	$48.40_{1.0}$	$47.22_{0.8}$	$47.01_{1.3}$	$49.64_{1.6}$	$48.86_{1.1}$	$47.85_{1.5}$	$47.58_{1.9}$	$49.89_{1.4}$	$49.89_{1.0}$	**53.19** (3.30↑)	43.63
Modifier	$58.83_{0.8}$	$57.93_{0.7}$	$58.80_{0.6}$	$59.07_{0.6}$	$58.68_{0.8}$	$59.56_{0.8}$	$58.45_{0.6}$	$59.72_{0.6}$	$59.47_{0.6}$	$58.44_{1.2}$	**60.27** (0.55↑)	53.94
Numerical	$62.56_{3.6}$	$64.73_{2.2}$	$63.10_{2.7}$	$61.40_{4.5}$	$59.88_{2.6}$	$62.51_{3.2}$	$61.53_{2.7}$	$64.83_{4.0}$	$63.41_{3.8}$	$63.05_{3.7}$	**66.98** (2.15↑)	56.96
Reagent	$80.38_{0.2}$	$80.34_{0.3}$	$80.60_{0.2}$	$80.94_{0.2}$	$80.51_{0.3}$	$80.84_{0.2}$	$80.35_{0.2}$	$81.24_{0.3}$	$81.19_{0.1}$	$80.68_{0.3}$	**82.59** (1.35↑)	74.98
Seal	$65.56_{5.1}$	$59.90_{1.9}$	$62.46_{2.7}$	$63.07_{2.6}$	$64.13_{2.8}$	$66.42_{3.4}$	$63.95_{2.7}$	**69.51**$_{3.9}$	$62.65_{3.3}$	$62.83_{3.7}$	69.42 (0.09↓)	67.72
Size	$58.36_{1.8}$	$60.16_{0.9}$	$60.07_{1.8}$	$61.65_{1.5}$	$59.79_{1.8}$	$56.47_{2.1}$	$59.30_{2.1}$	$62.48_{2.1}$	$61.12_{1.3}$	$59.63_{3.0}$	**62.56** (0.08↑)	57.14
Speed	$83.92_{0.8}$	$83.64_{1.0}$	$82.21_{0.8}$	$83.06_{0.8}$	$84.71_{0.7}$	$83.67_{1.0}$	$84.33_{0.6}$	$84.13_{1.0}$	$84.45_{0.7}$	$83.43_{2.3}$	84.24 (0.47↓)	**85.46**
Temperature	$91.47_{0.5}$	$90.20_{0.6}$	$91.36_{0.5}$	$90.84_{0.8}$	$91.32_{0.6}$	$90.70_{0.6}$	$91.39_{0.7}$	**92.20**$_{0.5}$	$91.49_{0.5}$	$90.20_{1.4}$	92.11 (0.09↓)	91.76
Time	$88.75_{0.3}$	$88.92_{0.7}$	$88.78_{0.4}$	$88.85_{0.5}$	$88.70_{0.6}$	$89.02_{0.7}$	$89.28_{0.4}$	$88.49_{0.6}$	$88.87_{0.4}$	$88.20_{1.5}$	**89.49** (0.21↑)	88.64
pH	$72.56_{2.3}$	$67.07_{3.0}$	$72.47_{1.4}$	$72.41_{2.3}$	$71.27_{2.8}$	$66.78_{2.2}$	$72.50_{4.1}$	**78.04**$_{1.7}$	$69.35_{2.8}$	$70.77_{2.9}$	77.59 (0.45↓)	65.49
micro F_1-score	$78.38_{0.2}$	$77.94_{0.2}$	$78.27_{0.1}$	$78.56_{0.2}$	$78.42_{0.2}$	$78.52_{0.2}$	$78.31_{0.1}$	$79.00_{0.2}$	$78.76_{0.1}$	$78.34_{0.3}$	**80.42** (1.42↑)	74.39
macro F_1-score	$70.06_{0.7}$	$69.12_{0.3}$	$69.43_{0.3}$	$69.98_{0.5}$	$69.98_{0.4}$	$69.82_{0.3}$	$69.91_{0.3}$	$71.32_{0.3}$	$70.05_{0.4}$	$69.54_{0.8}$	**72.47** (1.15↑)	66.65

Table 4: F_1-score by model on the test set. We reported averages across the 10 random seeds for all the pretrained model results, for those, subscripts represents the standard deviations. The improvements of the ensemble model over the best performing transformer model is shown in parentheses in the ensemble column.

When comparing the micro to the macro F_1-score standard deviations across all the models, we can see that the macro F_1-score standard deviations are systematically higher. This is probably due to the fact that some entities, namely *Generic-Measure*, *Mention*, *Seal*, *Size* and *pH*, which account for less than 1% of the test set each (see Table 2), seem to have a relatively high F_1-score standard deviations level. The same applies to *Measure-Type*, *Numerical* and *Speed* that are less than 2% of the test set each. This is in line with the results reported by Dodge et al. (2020) which shows that results can vary a lot across the seeds when a small amount of data is available. Indeed, as these entities are quite rare, a simple misclassification can have a high impact on the macro F_1-score. That being said, the micro F_1-scores seem relatively stable across all the pretrained models.

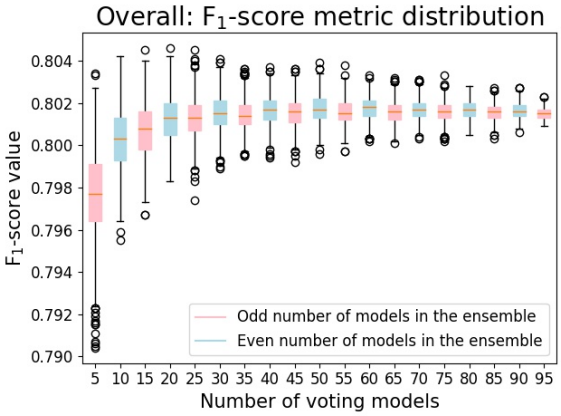

Figure 2: Micro F_1-score distribution by number of models used in an ensemble (sample size of 1000) on the test set.

5.1 Ensemble results analysis

In this section, we will try to analyse the results we observe when sampling on different ensemble model compositions. These results are exclusively computed on the test set. The idea behind this experiment is to try to understand the behaviour of some metrics when adding more models.

Figures 2 to 4 show the F_1-score, the recall and precision distributions with respect to the number of models taken in a given ensemble, respectively.

The first thing we notice from Figures 2 to 4 is that the more the number of models taken into account in an ensemble grows, the more the metrics variance tends to be smaller and steadier.

When looking at Figure 3 and 4, we clearly see that odd number of voters has a positive impact on the recall while it looks like it has a negative impact on the precision. For the moment, this is unclear to us why this behaviour can be observed; however, we think it could be linked to the majority rule we introduced in our voting strategy where majority is easier to reach in an odd system. When looking closely at Figure 2, it appears that the "odd/even number effect" tends to cancel out when the number of voters increases and even number of voter getting slightly better results.

In Figure 3, there is clearly a positive slope that seems to flatten at the end, which means that the more models we have in our ensemble, the higher recall we should expect. Conversely, this trend doesn't seem that clear for precision (Figure 4) where it looks like we have a positive relation with odd numbers of voters, a negative one with even number of voters which at the end seem to converge

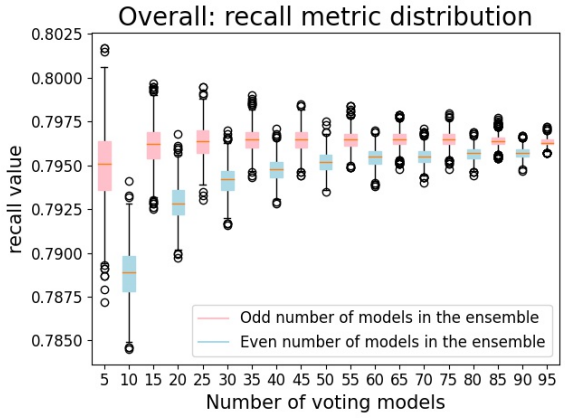

Figure 3: Micro recall distribution by number of models used in an ensemble (sample size of 1000) on the test set.

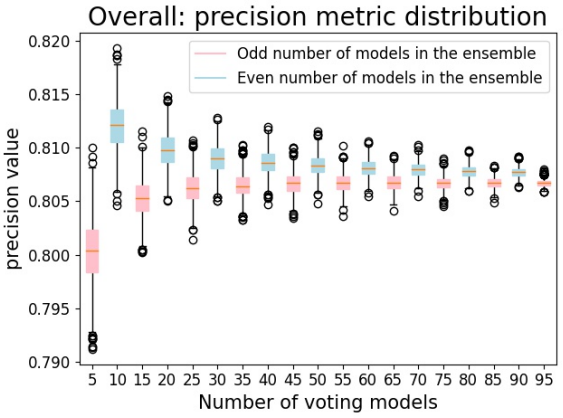

Figure 4: Micro precision distribution by number of models used in an ensemble (sample size of 1000) on the test set.

into a flat trend for both of them. However, in both figures, as already mentioned, the variance of their respective metrics seems to get steadier and smaller when adding more models in the ensemble composition.

Figures 6 to 8 show the same metrics while trying to isolate the effect of adding a new pretrained model versus the effect of adding an already taken pretrained model with a new random fully connected layer initialization.

In order to understand the setting of this experiment, we first build a matrix (see Figure 5) where each column is a pretrained model and each row is a fine-tuned version of it. We then compare the performances of ensemble models based on combinations of columns to those of the ensemble models based on combinations of rows. In Figures 6 to 8, the $x-$axis represents the number of row or columns taken into account.

For instance, the first two boxplots are computing metrics distributions of ensembles taking either one row or one column as an ensemble, the following two boxplots will take a combination of either two rows or two columns as ensemble and so on up to 9 rows/columns combinations.

More precisely, the first pink boxplot will compose an ensemble taking one column of models, namely, all the BERT$_{base}$ models to begin with, then all the BERT$_{large}$ models and so on until it computes the metrics for an ensemble composed with all the XLNet$_{large}$ models. Then, the second pink boxplot will take the composition of 2 columns, for example, it will first compute an ensemble with all the BERT$_{base}$ models and all the BERT$_{large}$ models, then another with all the

$$\begin{bmatrix} \text{BERT}_{\text{base}_1} & \text{BERT}_{\text{large}_1} & \cdots & \text{XLNet}_{\text{base}_1} & \text{XLNet}_{\text{large}_1} \\ \text{BERT}_{\text{base}_2} & \text{BERT}_{\text{large}_2} & \cdots & \text{XLNet}_{\text{base}_2} & \text{XLNet}_{\text{large}_2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \text{BERT}_{\text{base}_9} & \text{BERT}_{\text{large}_9} & \cdots & \text{XLNet}_{\text{base}_9} & \text{XLNet}_{\text{large}_9} \\ \text{BERT}_{\text{base}_{10}} & \text{BERT}_{\text{large}_{10}} & \cdots & \text{XLNet}_{\text{base}_{10}} & \text{XLNet}_{\text{large}_{10}} \end{bmatrix}$$

Figure 5: Matrix where each column represents a pretrained model and each row represents a fine-tuned model with a new random initialization of the fully connected layer.

BERT$_{base}$ models and all XLNet$_{base}$ models and so on until it computes an ensemble containing all the XLNet$_{base}$ and XLNet$_{large}$ models.

On the other hand, the blue boxplots will compose ensembles with combination of rows. This means that the first blue boxplot will first compute an ensemble composed of the first row (BERT$_{base_1}$, BERT$_{large_1}$, ..., XLNet$_{base_1}$, XLNet$_{large_1}$), then of the second row (BERT$_{base_2}$, BERT$_{large_2}$, ..., XLNet$_{base_2}$, XLNet$_{large_2}$) and so on until it computes an ensemble with the last row (BERT$_{base_{10}}$, BERT$_{large_{10}}$, ..., XLNet$_{base_{10}}$, XLNet$_{large_{10}}$). In the same manner, the second blue boxplot will compute ensembles composed by the combinations of two rows. First, all the models in the first and second rows, then, all the models in the first and third rows and so on until it computes an ensemble composed with all the models of the last two rows.

In this setting, as the maximum number of possible combinations of row is $252 = \binom{10}{5}$, we were able to compute all the possible combinations instead of sampling them. As we have the same number of pretrained models as fine-tuned versions, we end up with the same number of possible com-

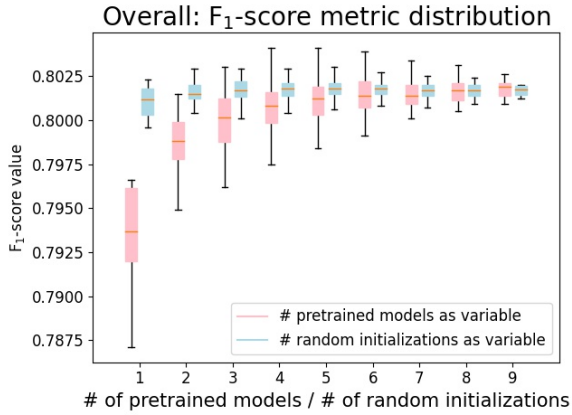

Figure 6: Micro F_1-score distribution using ensemble composed of either 1 to 9 different pretrained models (each time with 10 different fine-tuning) vs. 1 to 9 different fine-tuning using all the pretrained models.

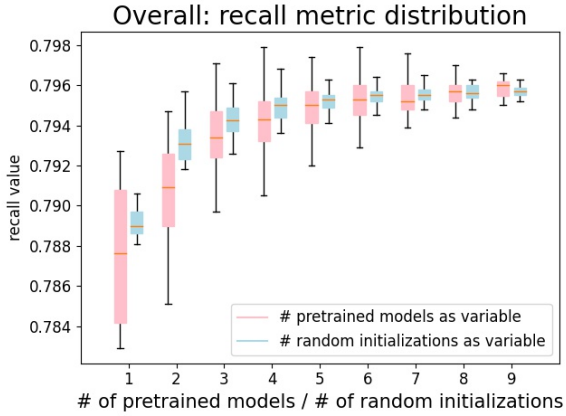

Figure 7: Micro recall distribution using ensemble composed of either 1 to 9 different pretrained models (each time with 10 different fine-tuning) vs. 1 to 9 different fine-tuning using all the pretrained models.

binations of ensemble. That being said, for each number of models taken into account in an ensemble, this allows us to compare the pink boxplot with the blue one in a more convenient manner.

It is worth noting that the more we increase the number of columns and rows present in an ensemble model, the more they share a certain number of models. For example, at 9, the pink boxplot shows the distribution of the metrics for all the possible ensemble models containing 9 columns of models (90 models out of 100), while the blue boxplot shows the same metrics for 9 rows of models (also 90 models out of 100). At this point, it is expected to see both boxplots converging as they both share 64 models predictions out of 90.

Focusing on the left part of Figure 6, we clearly see the benefits of using more pretrained models. First, it shows better results with only an ensemble of 10 different pretrained models. Then, it really looks steadier as the F_1-score distribution is much narrower than the ensemble composed of multiple fine-tuning of the same pretrained model.

When looking at Figure 7, we see that the major difference between both distributions are the variances of the recall distributions, indeed, taking different pretrained models tends to retrieve important passages more systematically. The trend of both selection strategies seems to be increasing, in other words, in both cases, the more we add models, the more we retrieve important passages.

Finally, it is interesting to see in Figure 8 that the precision begins quite high and tends to decrease when we add more fine-tuned models. Conversely, when taking more pretrained models, it seems the

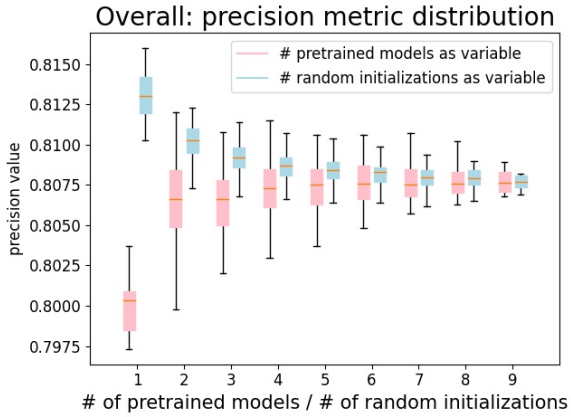

Figure 8: Micro precision distribution using ensemble composed of either 1 to 9 different pretrained models (each time with 10 different fine-tuning) vs. 1 to 9 different fine-tuning using all the pretrained models.

precision has a positive relation to the number of models we use. As explained before, this relation is also due to the fact that we share more and more models in both ensemble selection strategies.

This analysis helped us to understand a bit more about what was happening behind our majority of votes strategy, it would be interesting to take notes of some of the observed behaviours and try to devise new strategies accordingly.

5.2 Official results

The official results in terms of Precision, Recall, and F_1 on the test 2020 set is shown in Table 5. Each team was allowed to submit only one run. Our submitted run was based on the ensemble model described in sections 4.2 and 4.3. Our BiTeM team achieved the highest precision score in both ex-

Team Name	Exact Match			Partial Match		
	P	R	F_1	P	R	F_1
B-NLP	77.95	63.93	70.25	84.85	69.59	76.46
BIO-BIO	78.49	71.06	74.59	83.16	75.29	79.03
BiTeM	**84.73**	72.25	**77.99**	**88.72**	75.66	81.67
DSC-IITISM	64.20	57.07	60.42	68.52	60.90	64.49
Fancy Man	76.21	71.76	73.92	81.15	76.41	78.71
IBS	74.26	62.55	67.90	79.72	67.15	72.89
Kabir	78.79	72.20	75.35	83.73	76.73	80.08
KaushikAcharya	73.68	63.98	68.48	79.31	68.87	73.73
mahab	50.19	52.96	51.54	55.09	58.14	56.57
mgsohrab	83.69	70.62	76.60	87.95	74.22	80.50
PublishInCovid19	81.36	**74.12**	77.57	85.74	**78.11**	**81.75**
SudeshnaTCS	74.99	71.43	73.16	79.73	75.95	77.80
IITKGP	77.00	72.93	74.91	81.76	77.43	79.54

Table 5: Official results on Test 2020.

act match and partial match evaluation reaching 84.73% and 88.72%, respectively, and F_1-score in exact match evaluation reaching 77.99% among 13 teams. The F_1-score of our model in partial match (81.67%) was slightly lower than the best F_1-score (81.75%).

5.3 Results of the ensemble model on test 2020 data

The precision, recall, and F_1-score results of all entities and *Action* on the test 2020 in the exact match evaluation is represented in Table 6. The best F_1-score was achieved for *pH*. *Size* was the most difficult entity for detection.

5.4 Error analysis

Figure 9 shows the normalized confusion matrix for the predictions (exact match) of the ensemble model on the test 2020 data. As we can see, more than 78% of *Size* predictions are mislabelled as *Amount*. This can be due to the few number of training instances of *Size* entity. As we can see in the following examples, *50 mL* can refer to both *Size* and *Amount* depending on the context. In the first example, *50 mL* refers to *Amount* and in the second example, it refers to *Size*.

Example 5.4.1 *Add more NEB –no $\beta-$mercaptoethanol to final volume of 50 mL.*

Example 5.4.2 *Transfer the aqueous phase to a.new 50 mL Falcon tube.*

About 17% of the *Device* predictions are mislabelled as *Location* that can be due to the inconsistencies in the annotation process, for example *magnetic rack* is annotated as *Device* in a few protocols (protocol 0680, protocol 0683, protocol 0685), and as *Location* in others (protocol 32148, protocol 33630). Here are two examples of *magnetic rack* annotated as *Location* and *Device*, respectively.

Entity	Precision	Recall	F_1
Action	90.09	82.29	86.01
Amount	77.09	89.47	82.82
Concentration	86.76	88.16	87.45
Device	80.38	56.00	66.01
Generic-Measure	55.65	37.87	45.07
Location	69.59	68.98	69.28
Measure-Type	73.87	46.00	56.70
Mention	67.32	74.10	70.55
Method	61.49	35.77	45.23
Modifier	83.02	42.66	56.36
Numerical	65.32	38.49	48.44
Reagent	82.58	82.54	82.56
Seal	81.58	78.15	79.83
Size	63.64	17.80	27.81
Speed	86.38	86.38	86.38
Temperature	91.68	83.27	87.27
Time	92.58	87.66	90.05
pH	96.72	90.77	93.65

Table 6: The precision, recall, and F_1-score of the ensemble model for all the entities and *action* on the test 2020.

Example 5.4.3 *Place samples on magnetic rack, and incubate for 5 mins on the rack. Remove supernatant.*

Example 5.4.4 *Place the tube on a magnetic rack.*

Similarly *freezer* is annotated interchangeably as *Location* and *Device*. *Generic-Measure* is mostly confused with *Concentration* label (20.4%), and *Method* is mostly confused by *Action*. About 12% of *Numerical* is annotated as *Concentration*.

6 Conclusion

With almost no preprocessing, we have seen that current pretrained language models seem to be quite efficient in any NER task (Copara et al., 2020a,b). By analysing our voting strategy, we have also demonstrated the strengths as well as the weaknesses of such ensemble models. For instance, it looks like the more models we use, the more the performances tend to be high and stable, however, it appears that new pretrained model brings more information than fine-tuning again a pretrained model with a new fully connected weights random initialization.

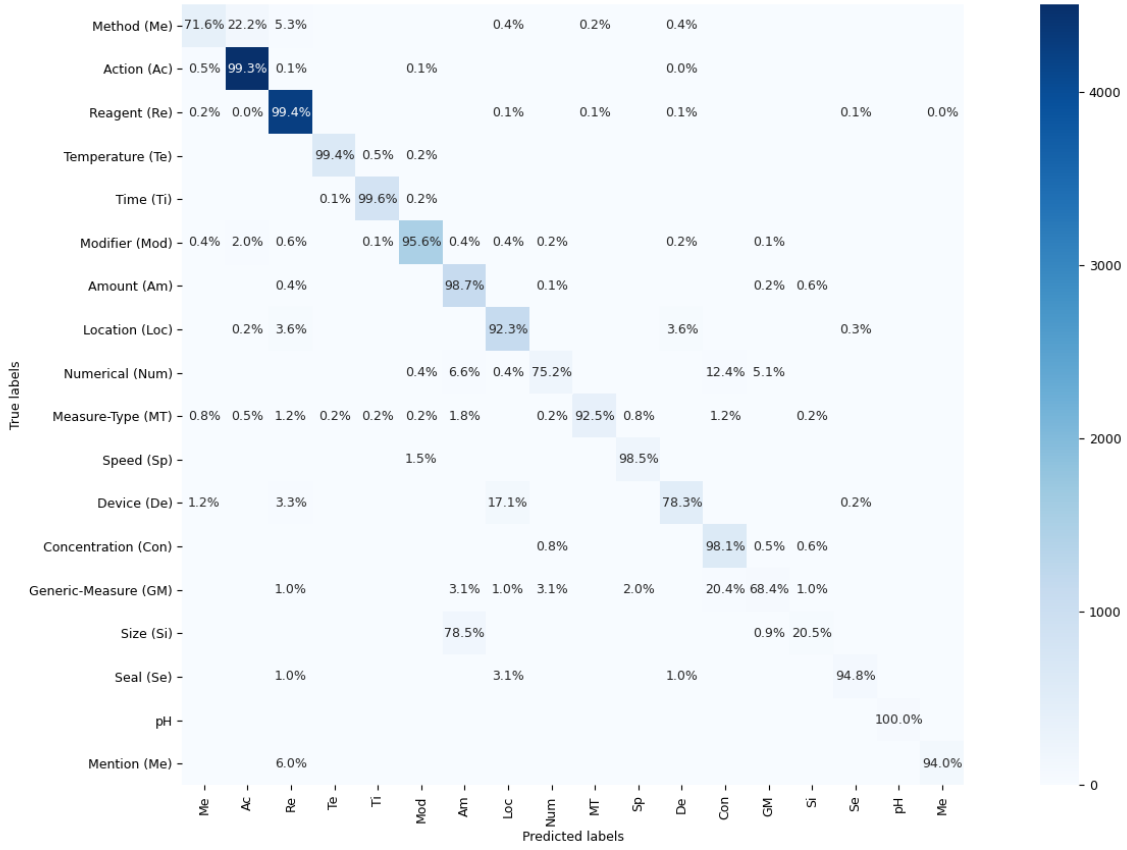

Figure 9: Normalized Confusion matrix for the ensemble model on the test 2020 data.

With this voting strategy, our submission achieved the best exact match overall F_1-score of the competition. This clearly shows the power of such models. With almost no knowledge on the topic of wet laboratory protocols required, we think that those models open opportunity to out-of-field researchers.

In future work, it would be interesting to improve the number of pretrained models selection and explore bootstrapping instead of fine-tuning multiple times the same pretrained model. It would also be interesting to see if some preprocessing tweaks could help us to improve the detection performance of *Speed* where our models were outperformed by the baseline.

References

Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly Available Clinical BERT Embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78.

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 (EMNLP-IJCNLP)*, pages 3606–3611.

Jenny Copara, Julien Knafou, Nona Naderi, Claudia Moro, Patrick Ruch, and Douglas Teodoro. 2020a. Contextualized French language models for biomedical named entity recognition. In *Actes de la 6e conférence conjointe Journées d'Études sur la Parole (JEP, 33e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition). Atelier DÉfi Fouille de Textes*, pages 36–48, Nancy, France. ATALA et AFCP.

Jenny Copara, Nona Naderi, Julien Knafou, Patrick Ruch, and Douglas Teodoro. 2020b. Named entity recognition in chemical patents using ensemble of contextual language models. *arXiv preprint arXiv:2007.12569*.

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: Human Language*

Technologies, Volume 1 (Long and Short Papers), pages 4171–4186.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2020. Domain-specific language model pretraining for biomedical natural language processing.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of ACL*.

Qiao Jin, Bhuwan Dhingra, William Cohen, and Xinghua Lu. 2019. Probing biomedical embeddings from language models. In *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*, pages 82–89.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Chaitanya Kulkarni, Wei Xu, Alan Ritter, and Raghu Machiraju. 2018. An annotated corpus for machine reading of instructions in wet lab protocols. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 97–106, New Orleans, Louisiana. Association for Computational Linguistics.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270.

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):1234–1240.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Jeniya Tabassum, Sydney Lee, Wei Xu, and Alan Ritter. 2020. WNUT-2020 Task 1 Overview: Extracting Entities and Relations from Wet Lab Protocols. In *Proceedings of EMNLP 2020 Workshop on Noisy User-generated Text (WNUT)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Advances in neural information processing systems*, pages 5753–5763.