PublishInCovid19 at WNUT 2020 Shared Task-1: Entity Recognition in Wet Lab Protocols using Structured Learning Ensemble and Contextualised Embeddings

Janvijay Singh and Anshul Wadhawan
Flipkart Private Limited
{janvijay.singh,anshul.wadhwan}@flipkart.com

Abstract

In this paper, we describe the approach that we employed to address the task of Entity Recognition over Wet Lab Protocols - a shared task in EMNLP WNUT-2020 Workshop. Our approach is composed of two phases. In the first phase, we experiment with various contextualised word embeddings (like Flair, BERT-based) and a BiLSTM-CRF model to arrive at the best-performing architecture. In the second phase, we create an ensemble composed of eleven BiLSTM-CRF models. The individual models are trained on random train-validation splits of the complete dataset. Here, we also experiment with different output merging schemes, including Majority Voting and Structured Learning Ensembling (SLE). Our final submission achieved a micro F1-score of 0.8175 and 0.7757 for the partial and exact match of the entity spans, respectively. We were ranked first and second, in terms of partial and exact match, respectively.

1 Introduction

Entity Recognition (aka entity extraction or chunking) involves detection (begin and end boundaries) and classification of entities mentioned in unstructured text into pre-defined categories. It is one of the foundational sub-task of several Information Extraction (Hanafiah and Quix, 2014) (IE) and Natural Language Processing (NLP) pipelines. Hence, errors introduced during the extraction of entities can propagate further and degrade the performance of the complete IE or NLP pipeline. In the domains of experimental biology, the growing complexity of experiments has resulted in a need to automate wet laboratory procedures. Such an automation will be useful in avoiding human errors introduced in the wet lab protocols and thereby will enhance the reproducibility of experimental biological research.

To achieve this reproducibility, some of the previous research works have focussed on defining machine-readable formats for writing wet lab protocols (King et al., 2009; Ananthanarayanan and Thies, 2010; Vasilev et al., 2011). However, the vast majority of today's protocols are written in natural language with jargon and colloquial language constructs that emerge as a byproduct of ad-hoc protocol documentation. This motivates the need for machine reading systems that can interpret the meaning of these natural language instructions, to enhance reproducibility via semantic protocols (e.g. the Aquarium project) and enable robotic automation (Bates et al., 2017) by mapping natural language instructions to executable actions. In order to enable research on interpreting natural language instructions, with practical applications in biology and life sciences, an annotated database (Kulkarni et al., 2018) of wet lab protocols was introduced.

The first step in interpreting natural language lab protocols is to extract entities, followed by identification of relations between them. To address the research focussing on entity recognition over Wet Lab Protocols a shared task (Tabassum et al., 2020) was introduced at EMNLP WNUT-2020 Workshop. The task was based on the annotated database (Kulkarni et al., 2018) of wet lab protocols. We tackle this task in two phases. In the first phase, we experiment with various contextualised word embeddings (like Flair, BERT-based) and a BiLSTM-CRF model to arrive at the best-performing architecture. In the second phase, we create an ensemble composed of eleven BiLSTM-CRF models. The individual models are trained on random train-validation splits of the complete dataset. Here, we also experiment with different output merging schemes, including Majority Voting and SLE.

The rest of the paper is structured as follows: Section 2 states the task definition. Section 3 describes the specifics of our methodology. Section 4 explains the experimental setup and the results,

and Section 5 concludes the paper.

2   Task Definition

The steps involved in any lab procedure are specified by lab protocols. These protocols have several characteristics like noise, density and domain specificity. Any process that can automatically or semi-automatically convert protocols into a format that machine recognizes advantages biological research. In this task, system entries for entity recognition on a dataset of lab protocols are invited. Since the protocols are written manually by lab technicians and researchers, they are subject to spelling errors and non standard language.

The data provided in the task is made available in two formats:

2.1   CoNLL format

In this format, each line represents the named entity in the following manner:

$$<word >+ "\backslash t"+ <NE >$$

An empty line denotes the end of a sentence.

2.2   Standoff format

The standoff format contains each protocol represented by two separate files. One file, with .txt extension, contains protocols in text format, while the other file, with .ann extension, contains protocol annotations. The two files are linked by using a simple file naming convention wherein their base name is the same, i.e. the file name without the extension is the same. For example,the annotation file named as protocol_17.ann contains annotations for the file protocol_17.txt.

Within each annotation file, individual annotations connect to different parts of text through character offsets. For example, in the document starting as "Put 3.68 g of NaCl", the text "Put" is denoted by the offset range 0..3. It is evident from the above example that all offsets are 0 indexed and include the character at the start offset and exclude the character at the end offset. All text files have the file extension .txt and contain the text of original documents provided as inputs to the system. The encoding used in the protocol text files which are stored as plain text files is UTF-8 (an extension of ASCII). Each line in the protocol text file denotes a single step in the protocol. Hence, all steps in the entire protocol are separated by newline characters. The first line in every file indicates the protocol's name/title.

3   Methodology

This section talks about the core methodology we adopted to tackle the given problem. The process pipeline involves providing contextualised word embeddings as input to the BiLSTM-CRF model, followed by a Structured learning Ensemble approach. Each of the these modules have been described in detail in the below subsections.

3.1   Embeddings

We experiment with two types of contextualised word embeddings, BERT and Flair based, which we discuss in detail in the below subsections.

3.1.1   BERT

Neural models based on transformers (Vaswani et al., 2017) have excelled in most NLP tasks. The primary components in their architecture being the self attention blocks and feed forward layers, these models have been proven successful in providing a significant boost to state-of-the-art results. The major difference between transformers and RNN based models (Li et al., 2018) is that transformers do not rely on recurrence mechanisms to establish relations and dependencies in the input sequence, by making use of self attention at each input time step instead. Attention can be interpreted as a technique to map a query and a set of key-value pairs to an output, where the query, keys, values and output are all vectors. As far as self attention is concerned, a separate feed forward layer is used to formulate the query, key and value vectors for each vector in the input sequence. For every input vector, the score for attention is calculated using a compatibility function which takes as input the input keys and query vector. These attention scores are used to denote the weights of a weighted sum of value vectors, which is the output of self attention technique. Another technique widely used is the multi headed attention technique in which several modules of these self attention blocks work over the input sequence. The encoder module in the transformer's architecture contains 6 identical layers each having two sublayers - position wise densely connected feed forward network and multi headed self attention layers. These sublayers are wrapped around with residual connections. Layer normalisation follows the above module. BERT pre-trains bidirectional representations by jointly utilizing both right and left contexts across all layers with the help of a multi layer encoder module.

These pre-trained BERT representations are then fine tuned as per the required task by appending a separate output layer depending on the task to be performed.

For every token, the summation of the corresponding token, segment and position embeddings is carried out to produce BERT's input representation. The training process for BERT involves Masked Language Modelling (Nozza et al., 2020) and Next Sentence Prediction (Shi and Demberg, 2019), both of which are unsupervised prediction tasks. BERT representation for each token in the input text is then fed to the appended densely connected layers to produce the output labels for the token as part of the fine tuning process. The predictions produced are independent of the surrounding predictions produced.

We experimented with different variations of BERT models (Devlin et al., 2018) for generating word embeddings. All the listed model types have 12 layers, 12 attention heads and 110M parameters.

BERT-base-cased : This model is trained on cased English text of general domain like Wikipedia text and BooksCorpus.

BioBERT (Lee et al., 2019) : BioBERT is a language representation model pre-trained on the domain of biomedical data. The pre-training process for BioBERT involves initializing weights with those of BERT which is pre-trained on general domain corpora, followed by pre-training BioBERT with biomedical data corpora like PMC full-text articles and PubMed abstracts.

PubMedBERT (Gu et al., 2020) : The base architecture of PubMedBERT is the same as an uncased BERT base model. The model is pre-trained on full PubMed Central articles and PubMed abstracts. The pre-training process for this model involves direct pre-training on biomedical text from scratch. Thus, the weights are not initialized with those of BERT as was in the case of BioBERT. The pre-training corpus contains 14 million PubMed abstracts with 3 billion words, 21 GB of textual data in total. Another version of the same model is pre-trained on additional data of full text PubMed Central articles, with the total textual data containing 16.8 billion words and 107 GB in size.

3.1.2 Flair

[1] Flair embeddings are pre-trained Contextualised Word Embeddings (CWE) provided in the Flair

---

[1] https://github.com/flairNLP/flair

NLP framework. In contrast to classical work embeddings like GloVe, the Flair CWE concatenate two context vectors based on the left and right sentence context of the word to it. These context vectors are computed using two recurrent neural models. One of the character language model is trained from left to right while the other is trained from right to left. Flair CWEs have been applied successfully to sequence tagging tasks such as Named Entity Recognition and Part of Speech Tagging. Since this shared task is closely related to Bio-medical domain, we have used "pubmed" variant of Flair CWEs in all our experiments.

3.2 BiLSTM-CRF Model

The ability of Recurrent Neural Networks (RNNs) (Yadav and Bethard, 2018) to execute the same function at each time step, allowing parameters to be shared across the input sequence, make them highly suitable for sequential input data . Useful information from each time step is forwarded to further time steps in the form of a hidden vector, which is utilized to make a prediction at each of the future steps. However, RNNs face the issue of vanishing gradients in case of large input sequences. To solve this issue of vanishing gradients, (Long Short Term Memory) LSTM (Hochreiter and Schmidhuber, 1997) was introduced. The presence of gating mechanisms in LSTMs makes sure that long range dependencies are captured appropriately. While LSTMs utilize only past time steps to make a prediction, Bidirectional LSTM (BiLSTM) (Schuster and Paliwal, 1997) utilizes information from past as well as future time steps. In our case, the output embeddings are fed to the BiLSTM layer, which outputs a vector for each word in the input sequence. Since the task under consideration has labels which have dependencies among themselves, such as an intermediate_label following a start_label, we need to consider these dependencies in our modelling approach. For this, a linear chain (Conditional Random Fields) CRF layer (Sutton and McCallum, 2010) is appended to the BiLSTM layer. Due to utilization of transition matrices for output labels, a linear chain CRF is able to learn inter label dependencies, if any, among the output labels.

3.3 Ensemble Process

We created eleven randomly shuffled splits of training and validation data, and fine tuned our final model on these eleven splits to produce eleven sets

of predictions. We then merged these predictions following two merging techniques, Majority Voting and Structured Learning Ensemble (SLE), thus comparing the performance of the two merging functions. In our experiments, we provide a fair comparison of the above two combination techniques, i.e. Majority Voting technique and SLE.

Given N number of ensembles and x as the input example, $\{y_1, y_2, ..., y_N\}$ being the predictions from N different models are merged to produce the final prediction y. The ensemble methods for structured output classification and multiclass classification differ in the way they merge the predicted results of the base models.

The merging techniques have been described below:

3.3.1 Majority voting

For every entity predicted, we choose the mode i.e. the most frequently occurring entity among the eleven predictions (Adejo and Connolly, 2017). Thus, the entity which has the maximum number of votes wins.

Mathematically, the above process of majority voting scheme to produce the final predictions can be denoted in the below manner :

$$\mathbf{y} = \langle majority\,\{(\mathbf{y}_1)_1, (\mathbf{y}_2)_1, \ldots, (\mathbf{y}_N)_1\}\rangle$$
$$\ldots\ldots\ldots\ majority\,\{(\mathbf{y}_1)_L, (\mathbf{y}_2)_L, \ldots, (\mathbf{y}_N)_L\}\rangle$$

where L is the length of all predictions.

3.3.2 Structured Learning Ensemble (SLE)

Due to the presence of correlations and intrinsic structures in the output labels, we speculated that the majority voting scheme would not suffice for our problem. (Nguyen and Guo, 2007) proposed a technique to combine the predictions considering the correlations of the output labels. Named as weighted transition combination, the algorithm involves construction of (L-1) transition matrices of size ($|\Sigma|$ x $|\Sigma|$) , where Σ is the set of all possible labels. Apart from this, it also involves construction of a transition matrix T^k which provides the number of transitions at the kth position as follows:

$$T^k(t_i, t_j) = count_k(t_i, t_j), \forall 1 \le k \le (L-1)$$

where $count_k(t_i, t_j)$ denotes the number of times the label t_j occurs after t_i at the kth position in the set of predicted sequences $\{y_1, y_2, ..., y_N\}$. Also, a stateweight vector is constructed that denotes the

number of times label t_i occurs at position k in the predicted sequences.

$$U^k(t_i) = count_k(t_i), \forall 1 \le k \le L$$

The predicted sequence of SLE is given by:

$$\mathbf{y} = argmax_{\mathbf{y}} \prod_{k=1}^{L-1} T^k(y_k, y_{k+1}) \prod_{k=1}^{L} U^k(y_k)$$

The computation involved in the argmax calculation of the above equation is similar to Viterbi dynamic programming approach.

4 Experiments

Our experimentation strategy is distributed in two phases. In the first phase, we experiment with various architectures and their specifications by varying the type of pre-trained model, deciding layers to freeze i.e. complete fine-tuning or contextual word embeddings, varying type and size of final layer in order to arrive at the best performing model. We trained each of our model architectures on the train split and identified the checkpoint which worked best using the validation split. We reported the final numbers on the test split. For each model, we train three different models with random seed values and then report averaged f1 scores to ensure that improvements are not the result of randomisation. A configuration of concatenated contextual word embeddings from PubmedBERT and Flair, followed by 2 BiLSTM layers with 512 dimensional hidden size and a CRF layer in the end worked best. In the second phase, we train individual models on random splits of train + validation sets. In order to merge the outputs of individual models, we experiment with two output merging schemes namely Majority Voting and Structured Learning Ensemble (SLE). Finally, we report the results on the test dataset.

In the following sub-sections, we describe the dataset, system settings, evaluation metrics, results and a brief error analysis for our final submitted system.

4.1 Dataset

Wet Lab Protocol (WLP) dataset consists of 615 unique protocols from 623 protocols released by (Kulkarni et al., 2018). It excludes the following 8 duplicate protocols:

protocol 45 (duplicate of protocol 441)
protocol 459 (duplicate of protocol 310)

	train_data	dev_data	test_data	test_data_2020
Measure-Type	857	329	272	731
Numerical	838	262	231	520
Size	262	124	114	238
Seal	210	92	64	119
Speed	626	241	167	240
Location	3929	1407	1327	1670
Temperature	1594	492	532	760
Amount	3438	1102	1193	1238
Method	1605	545	582	1077
pH	67	37	62	66
Generic-Measure	487	136	143	176
O	53690	18454	17925	26012
Action	12368	4057	4140	5439
Mention	257	84	56	145
Concentration	1333	427	537	705
Reagent	11142	3646	4004	5079
Time	2399	751	870	959
Modifier	4593	1554	1601	3476
Device	1752	618	468	911

Table 1: Frequency of various entity-types in different dataset splits.

	#protocols	#sentences
train_data	370	8444
dev_data	122	2839
test_data	123	2862
test_data_2020	111	3562

Table 2: Statistics of different dataset splits.

	Vocabulary	OOV (wrt ref)
train_data	7397	-
dev_data	4082	1148
test_data	3946	982
test_data_2020	5718	2461

Table 3: Out-of-Vocabulary statistics.

protocol 464 (duplicate of protocol 46)

protocol 480 (duplicate of protocol 473)

protocol 482 (duplicate of protocol 474)

protocol 483 (duplicate of protocol 475)

protocol 484 (duplicate of protocol 476)

protocol 621 (duplicate of protocol 570)

After discarding the duplicate protocols, the remaining 615 unique protocols are re-annotated in brat by 3 annotators with 0.75 inter-annotator agreement, measured by span-level Krippendorff's α. The annotators not only added the missing entity-relations but also rectified the inconsistencies.

The detailed class-wise statistics pertaining to each of the dataset splits provided in the task are shown in Table 1. Corresponding number of protocols and sentences are provided in Table 2. Here, train_data denotes the training dataset, dev_data denotes the validation dataset, test_data denotes the test dataset and test_data_2020 denotes the surprise test dataset. The surprise dataset was not revealed before the evaluation window.

Table 3 presents the total number of words, words absent in reference and words present in reference for each dataset. Reference varies according to the dataset being considered. For validation dataset and test dataset, training dataset is the reference. For surprise dataset, all data i.e. the union of training dataset, validation dataset and test dataset is considered as the reference. There is no reference in case of training dataset.

4.2 System Settings

While training individual models of our final ensemble, we rely on concatenated word representations from PubMedBERT and Flair. We train the BiLSTM-CRF based model with 3 BiLSTM layer each of hidden size 512 using a patience-based strategy. With this strategy, after every epoch of

Hyperparameter	Value
Embedding	Flair + PubMedBERT
Final layer type	BiLSTM
Final layer hidden size	512
# Final layers	2
CRF	✓
Patience epochs	3
max epochs	30
Initial learning rate	0.1
Mini batch size	32
Merging scheme	SLE

Table 4: System Settings for the final model.

training, we compute the F1-score on validation split and if the metric doesn't improve continuously for "patience" number of epochs, we reduce the learning rate by half. We ultimately stop the training when either the learning rate diminishes to 0.0001 or the epoch number reaches a maximum limit. We have utilised hugging-face[2] BERT APIs and Flair Framework(Akbik et al., 2019) to train our model. We ran our experiments on a single NVIDIA V100 GPU. It took around 2.5 hours to train each individual model of our final submitted ensemble. Table 4 summarises the hyperparameters which we employed to train our models.

4.3 Evaluation Metrics

Assuming that P and T represent the set of predicted and ground-truth entities for a particular word in the protocol text. Then, precision, recall and F1-score for the entity prediction of the considered word is defined as follows:

$$Precision = \frac{|P \bigcap T|}{|P|}$$

$$Recall = \frac{|P \bigcap T|}{|T|}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

There were two criteria for evaluation metrics in the task, partial match and exact match. In case of partial match, P intersection T will include all entities whose types match and boundaries match partially, i.e. there is some overlap in the boundaries. However, in case of exact match, for an entity to be included in the intersection set, it must have the same type as well as exact same boundaries.

---

[2]https://huggingface.co/transformers/

4.4 Results and Error Analysis

Our approach involved working in two phases, first in which we experiment with different model architectures and the second in which we experiment with two output merging schemes. The results of our experiments in Phases 1 and 2 are summarised in Table 5 and 6 respectively. In Table 5, we present the micro-F1 and macro-F1 scores for different model architectures we experiment with by varying the base model, fine tuning implementation, type and specifications of final layer and CRF layer addition. Table 6 presents the micro-F1 scores on the test set when we experiment with the number of ensembles, i.e. on merging different number of prediction sets.

For our final submission to WNUT Shared Task-1, we employed an ensemble of eleven individual models. Each of these models was trained on a random train-validation split of original train + validation + test dataset. Our ensemble achieved a micro-F1 score of 0.8175 and 0.7757 for the partial and exact match of entity boundaries, respectively. We achieved highest micro-recall score among all the participating teams. In Table 7, we report the top-10 confusions which our model makes while assigning entity type to different words. Results of the final submission on surprise test set are summarised in Table 8. Upon close inspection of predicted outputs on test split, we identified the following error patterns in the model predictions:

- From Table 7, we can see that model dominantly gets confused while identifying the begin and intermediate tags for class Reagent. Upon inspection of the predictions, we identified that such errors were more common when the Reagent class in validation/test set was unseen in training examples. We can come up with a dictionary based approach to improve the precision of tags specifically for the Reagent class.

- Modifier entity type modifies the semantics of some other entity type, so for a word to be Modifier or not is highly dependent on context and modified entity. But since our model fails to over-rely on context for recognition of certain entities, Modifier entity-type often gets confused with Other type.

- For the entities corresponding to numerical values like Concentration, Amount, Size and

| Base Model | Finetuning | Final Layer | | | CRF | micro-F1 | macro-F1 |
		Type	#layers	#dim			
Bert-base-cased	✓	Dense	1	-	✗	78.78	71.06
Bert-base-cased	✗	BiLSTM	1	128	✓	80.56	73.41
BioBERT	✗	BiLSTM	1	128	✓	81.01	73.81
PubmedBERT	✗	BiLSTM	1	128	✓	81.36	73.67
Flair	✗	BiLSTM	1	128	✓	81.63	74.84
PubmedBERT + Flair	✗	BiLSTM	1	128	✓	81.71	75.22
PubmedBERT + Flair	✗	BiLSTM	2	374	✓	82.06	75.18
PubmedBERT + Flair	✗	BiLSTM	2	512	✓	82.28	75.57

Table 5: Results of experiments to identity the best architecture specification.

#ensembles	MajV	SLE
3	82.32	82.50
5	82.52	82.68
7	82.52	82.58
9	82.55	82.64
11	82.60	82.74

Table 6: micro-F1 on test-set after ensembling.

P_Label	T_Label	Count
O	B-Modifier	324
B-Modifier	O	287
O	I-Modifier	247
B-Reagent	I-Reagent	180
I-Reagent	B-Reagent	112
B-Modifier	B-Reagent	122
O	B-Action	137
O	I-Reagent	115
O	I-Method	112
B-Action	O	190

Table 7: Top-10 errors occurring in model predictions.

	Exact Match	Partial Match
Precision	81.36	85.74
Recall	74.12	78.11
Micro-F1	77.57	81.75

Table 8: Final results on surprise-test dataset.

Numerical, model often gets confused among such entities. The main reason which we suspect is that to classify these entities, the model should over-rely on context and not on the token corresponding to the entity itself. Since tokens can be shared across different classes. e.g. 1.5 ml microcentrifuge tube; Preds: B-Amount I-Amount B-Location I-Location; True Label: B-Size I-Size B-Location I-Location;

5 Conclusion and Future Work

Through this paper, we showcased our approach to tackle the Shared Task 1 in EMNLP WNUT-2020 Workshop which involved Entity Recognition over Wet Lab Protocols. We solved the task in two phases. The first phase involved experimenting with different contextualised word embeddings like BERT and Flair, and a BiLSTM-CRF model to find the best performing model configuration for the problem at hand. In the second phase, we create an ensemble consisting of eleven BiLSTM-CRF models. We train individual models on randomly shuffled train-validation splits of the complete dataset. Also, we experiment with different merging techniques like Majority Voting and Structured Learning Ensemble (SLE). Our end solution achieved a micro F1-score of 0.8175 and 0.7757 in the partial and exact match categories, respectively. We were ranked first and second in partial and exact match categories respectively. In the future, we wish to explore the idea of employing rule-based approach to overcome the shortcomings of current solution.

References

Olugbenga Adejo and Thomas Connolly. 2017. Predicting student academic performance using multi-

model heterogeneous ensemble approach. *Journal of Applied Research in Higher Education*, 10:00–00.

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota. Association for Computational Linguistics.

Vaishnavi Ananthanarayanan and William Thies. 2010. Biocoder: A programming language for standardizing and automating biology protocols. *Journal of biological engineering*, 4(1):1–13.

Maxwell Bates, Aaron J Berliner, Joe Lachoff, Paul R Jaschke, and Eli S Groban. 2017. Wet lab accelerator: a web-based application democratizing laboratory automation for synthetic biology. *ACS synthetic biology*, 6(1):167–171.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

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.

Novita Hanafiah and Christoph Quix. 2014. Entity recognition in information extraction. In *Intelligent Information and Database Systems*, pages 113–122, Cham. Springer International Publishing.

S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

Ross D King, Jem Rowland, Stephen G Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N Soldatova, et al. 2009. The automation of science. *Science*, 324(5923):85–89.

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 (NAACL)*.

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*.

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2018. A survey on deep learning for named entity recognition.

Nam Nguyen and Yunsong Guo. 2007. Comparisons of sequence labeling algorithms and extensions. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 681–688, New York, NY, USA. Association for Computing Machinery.

Debora Nozza, Federico Bianchi, and Dirk Hovy. 2020. What the [mask]? making sense of language-specific bert models.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

Wei Shi and Vera Demberg. 2019. Next sentence prediction helps implicit discourse relation classification within and across domains. 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 5790–5796, Hong Kong, China. Association for Computational Linguistics.

Charles Sutton and Andrew McCallum. 2010. An introduction to conditional random fields.

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

Viktor Vasilev, Chenkai Liu, Traci Haddock, Swapnil Bhatia, Aaron Adler, Fusun Yaman, Jacob Beal, Jonathan Babb, Ron Weiss, and Douglas Densmore. 2011. A software stack for specification and robotic execution of protocols for synthetic biological engineering.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Vikas Yadav and Steven Bethard. 2018. A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158, Santa Fe, New Mexico, USA. Association for Computational Linguistics.