Split-NER: Named Entity Recognition via Two Question-Answering-based Classifications

Jatin Arora
Nuro Inc.
jarora@nuro.ai

Youngja Park
IBM T.J. Watson Research Center
young_park@us.ibm.com

Abstract

In this work, we address the NER problem by splitting it into two logical sub-tasks: (1) *Span Detection* which simply extracts mention spans of entities, irrespective of entity type; (2) *Span Classification* which classifies the spans into their entity types. Further, we formulate both sub-tasks as question-answering (QA) problems and produce two leaner models which can be optimized separately for each sub-task. Experiments with four cross-domain datasets demonstrate that this two-step approach is both effective and time efficient. Our system, `SplitNER` outperforms baselines on *OntoNotes5.0*, *WNUT17* and a cybersecurity dataset and gives on-par performance on *BioNLP13CG*. In all cases, it achieves a significant reduction in training time compared to its QA baseline counterpart. The effectiveness of our system stems from fine-tuning the BERT model twice, separately for span detection and classification. The source code can be found at `github.com/c3sr/split-ner`.

1 Introduction

Named entity recognition (NER) is a foundational task for a variety of applications like question answering and machine translation (Li et al., 2020a). Traditionally, NER has been seen as a sequence labeling task where a model is trained to classify each token of a sequence to a predefined class (Carreras et al., 2002, 2003; Chiu and Nichols, 2016; Lample et al., 2016; Ma and Hovy, 2016; Devlin et al., 2019; Wan et al., 2022).

Recently, there has been a new trend of formulating NER as span prediction problem (Stratos, 2017; Li et al., 2020b; Jiang et al., 2020; Ouchi et al., 2020; Fu et al., 2021), where a model is trained to jointly perform span boundary detection and multi-class classification over the spans. Another trend is to formulate NER as a question answering (QA) task (Li et al., 2020b), where the model is given a sentence and a query corresponding to each entity type. The model is trained to understand the query and extracts mentions of the entity type as answers. While these new frameworks have shown improved results, both approaches suffer from a high computational cost: span-based NER systems consider all possible spans (i.e., n^2 (quadratic) spans for a sentence with n tokens) and the QA-based system multiplies each input sequence by the number of entity types resulting in $N \times T$ input sequences for N sentences and T entity types.

In this work, we borrow the effectiveness of span-based and QA-based techniques and make it more efficient by breaking (splitting up) the NER task into a two-step pipeline of classification tasks. In essence, our overall approach comes under the span-based NER paradigm, and each sub-task is formulated as a QA task inspired by the higher accuracy offered by the QA framework. The first step, *Span Detection* performs token-level classification to extract mention spans from text, irrespective of entity type and the second step, *Span Classification* classifies the extracted spans into their corresponding entity type, thus completing the NER task. Unlike other span-based NER techniques which are quadratic in terms of sequence length, our *Span Detection* process is linear. Compared to other QA-based techniques which query for all entity types in each sentence, our *Span Classification* queries each sentence only once for each entity mention in the sentence. This makes it highly efficient for datasets with large number of entity types like *OntoNotes5.0*.

2 Method

Figure 1 illustrates how our two-step `SplitNER` system works. *Span Detection Model* is entity-agnostic and identifies all mention spans irrespective of entity type. The extracted spans are passed to *Span Classification Model* which reanalyses them in the sentence structure and classifies them into an entity type. Both models use BERT-

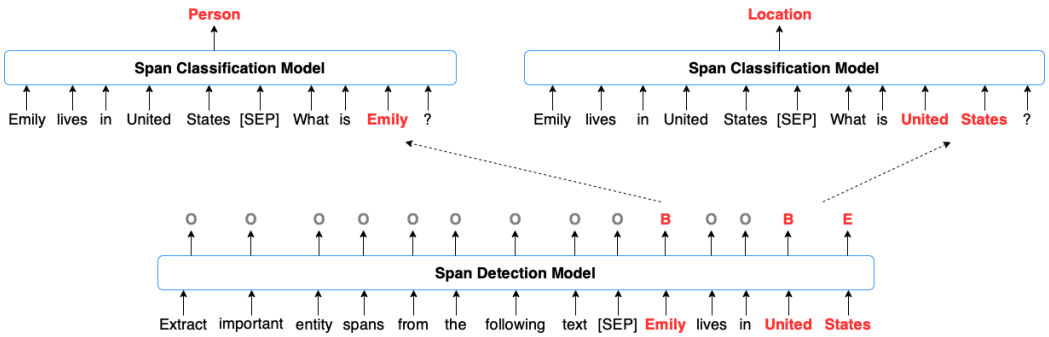


Figure 1: SplitNER System Overview. *Span Detection Model* identities two mention spans (*Emily* and *United States*) from the input (*"Emily lives in United States"*), and *Span Classification Model* assigns each span to their entity type.

base as their underlying architecture and are designed as QA tasks. Hence, moving forward, we may sometimes explicitly call our system as SplitNER(QA-QA) to distinguish it from other variants we experiment with.

2.1 Span Detection

Given a sentence $\mathcal{S}$ as a n-length sequence of tokens, $\mathcal{S} = \langle w_1, w_2 \dots w_n \rangle$, the goal is to output a list of spans $\langle s, e \rangle$, where $s, e \in [1, n]$ are *start* and *end* indices of a mention. We formulate this as a QA task classifying each token using BIOE scheme[1]. Since the goal is to detect spans irrespective of their entity type, we use a generic question, "*Extract important entity spans from the following text*", prefixed with input sentence (see Figure 1)[2].

A well-known problem in pipeline systems is error propagation. Inaccurate mention boundaries will lead to incorrect entity type classification. We observed that such boundary detection errors happen mostly for domain-specific terms which occur rarely and do not have a good semantic representation in the underlying BERT model. However, these domain specific terms often share patterns at character-level (e.g., chemical formulas). Thus we add character sequences and intrinsic orthographic patterns as additional features along with the BERT embeddings. The character and pattern features are shown to produce better word representations (Carreras et al., 2002; Limsopatham and Collier, 2016; Boukkouri et al., 2020; Lange et al., 2021).

Character Sequence Feature To learn character-level representation of each token, we use five one-dimensional CNNs with kernel sizes from 1 to 5, each having 16 filters and 50 input channels. Each token output from WordPiece Tokenizer is fed to the five CNN models simultaneously, which produce a 50-dimensional embedding for each character. These are max-pooled and the outputs from the CNNs are concatenated and passed through a linear layer with ReLU activation to get a 768-dimensional character-level representation of the token. Figure 2a shows the process.

Orthographic Pattern Feature To capture the intrinsic orthographic patterns (or word shapes) of entity mentions at the sub-word level, we map all uppercase tokens to a single character, U, all lowercase tokens to L, all digit tokens to D. If a token contains a mix of uppercase, lowercase and digits, we map each lowercase character to l, uppercase to u and digit to d. Special characters are retained and BERT's special tokens, "[CLS]" and "[SEP]", are mapped to C and S respectively.

We use 3 CNNs with the same setup as character sequence with kernel sizes of 1 to 3. Note that a contextual learning layer is needed to capture patterns in mentions spanning multiple tokens. Thus, we pass the pattern-level embeddings for all tokens to a bidirectional LSTM with 256 hidden dimensions as shown in Figure 2b. Finally, the character and pattern features are concatenated with the BERT output for the token and fed to a final classifier layer as shown in Figure 3.

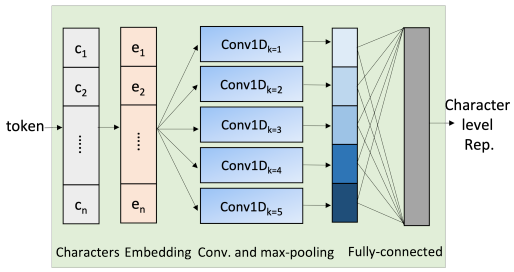
2.2 Span Classification

Given a sentence $\mathcal{S} = \langle w_1, w_2 \dots w_n \rangle$ and a span $\langle s, e \rangle$, this step determines the entity type for the span. Existing QA-based NER methods take the target entity type as the question (e.g., *"Where is* person*?*) and return the corresponding mentions in the sentence. On the contrary, our model takes a mention as the question (e.g., *"What is* Emily*?*) and outputs its entity type.

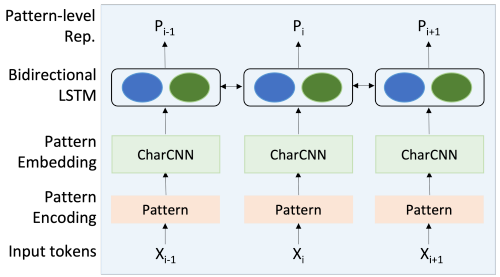
During training, we create a training sample for

[1] All experiments in this paper use BIOE scheme but the approach is generalizable to other schemes like BIOES.

[2] Other similar question texts / no question text also gives similar results as shown in ablation study in Appendix A.



(a) Character Sequence Feature Learning



(b) Orthographic Feature Learning

Figure 2: Architecture for Character and Pattern Feature Learning

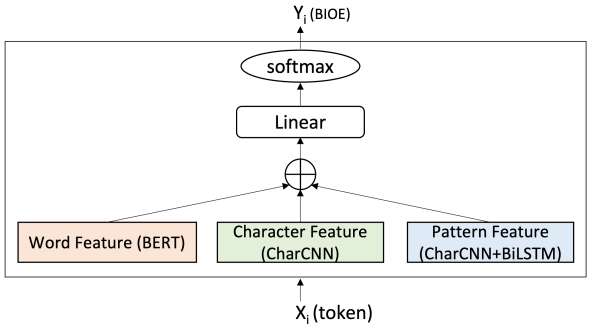


Figure 3: Token-level Schematic Model for *Span Detection Model*

each labeled entity mention in a sentence. During inference, the model gets the mention spans from *Span Detection Model* as its input. An input sample is created by appending the mention span text as *"What is [mention]?"* to the input sentence (see top diagrams in Figure 1 for example). This is fed to a BERT model and the pooled sequence embedding is fed to a fully connected layer and converted into a probability distribution over the entity types.

3 Experimental Results

We demonstrate the effectiveness of our method in terms of performance and latency.

3.1 Datasets

Table 1 shows our datasets, including three public benchmark datasets, *BioNLP13CG* (Pyysalo et al., 2015), *OntoNotes5.0* (Weischedel et al., 2013), and *WNUT17* (Derczynski et al., 2017), and a private dataset[3] (*CTIReports*) from the cybersecurity domain which contains news articles and technical reports related to malware and security threats. These datasets cover not only the traditional whole-word entities like PERSON but also entity types with non-word mentions (e.g., chemical formulas) and very long mentions (e.g., URLs).

Dataset	*Type*	*Density*	*Train*	*Dev*	*Test*
BioNLP13CG	16	3.59	3,033	1,003	1,906
CTIReports	8	0.63	38,721	6,322	9,837
OntoNotes5.0	18	1.36	59,924	8,528	8,262
WNUT17	6	0.68	3,394	1,009	1,287

Table 1: Dataset Summary. *Type* and *Density* show number of entity types and average number of mentions per sentence. *Train*, *Dev*, *Test* show number of sentences.

3.2 Experimental Setup

We implement our baselines and our proposed system, `SplitNER` in `pytorch` using `transformers` (Wolf et al., 2019). All models are trained on *Nvidia Tesla V100* GPUs and use BERT-base architecture. We use pretrained RoBERTa-*base* (Liu et al., 2019) backbone for all experiments with *OntoNotes5.0* corpus following Ye et al. (2022); Zhu and Li (2022) and use SciBERT-*scivocab-uncased* (Beltagy et al., 2019) for *BioNLP13CG* since this dataset has chemical formulas and scientific entities[4]. For *WNUT17*[5] and *CTIReports*, we use BERT-*base-uncased* (Devlin et al., 2019). Note that our model is a general two-step NER framework which has the performance benefits of QA-based and span-based approaches with efficiency. It can work with any BERT-based pretrained backbones.

The training data is randomly shuffled, and a batch size of 16 is used with post-padding. The maximum sequence length is set to 512 for

[3]The dataset curation procedure, entity types and their distribution is described in detail in Appendix C.

[4]We also experimented with BioBERT(Lee et al., 2020) (`dmis-lab/biobert-base-cased-v1.1`) which gives similar trends. But SciBERT outperforms in our experiments.

[5]BERTweet(Nguyen and Vu, 2020) model can also be used for *WNUT17*. We expect it to give same trends with even better performance figures.

Model	*BioNLP13CG*	*CTIReports*	*OntoNotes5.0*	*WNUT17*
SplitNER(QA-QA)	86.75	**74.96**	**90.86**	**57.25**
SplitNER($QA_{NoCharPattern}$-QA)	86.70	74.05	90.58	56.24
SplitNER(SeqTag-QA)	86.08	73.84	90.30	56.10
Single(QA)	86.68	71.70	89.02	43.45
Single(SeqTag)	**87.08**	72.36	88.64	44.97

Table 2: NER Performance Comparison (mention-level F1). SplitNER(QA-QA) is our proposed method.

Model	*BioNLP13CG*		*CTIReports*		*OntoNotes5.0*		*WNUT17*	
	Train	Inf.	Train	Inf.	Train	Inf.	Train	Inf.
SplitNER(QA-QA)	241.2	57.7	**1,455.7**	120.0	**3,007.8**	183.0	122.9	26.0
Single(QA)	1,372.8	323.3	8,771.0	551.6	73,818.4	2,227.8	568.2	91.2
Single(SeqTag)	**102.2**	**25.2**	6,425.9	**86.4**	9,181.1	**105.0**	**101.3**	**18.6**

Table 3: Comparison of training and inference (Inf.) latency in seconds.

OntoNotes5.0[6] and to 256 for all other datasets. For model optimization, we use cross entropy loss for span detection and dice loss(Li et al., 2020c) for span classification. All other training parameters are set to defaults in `transformers`.

3.3 Performance Evaluation

We compare our method SplitNER(QA-QA) with the following baselines and variants. (1) Single(SeqTag): The standard single-model sequence tagging NER setup which classifies each token using BIOE scheme. (2) Single(QA): The standard single-model QA-based setup which prefixes input sentences with a question describing the target entity type (e.g., *Where is the* person *mentioned in the text?*); (3) SplitNER(SeqTag-QA): A variant of our model which uses sequence tagging for span detection with our QA-based *Span Classification Model*; (4) SplitNER($QA_{NoCharPattern}$-QA): This model is the same as our method but without the additional character and pattern features. All other baselines use character and pattern features for fair comparison. We trained all models with 5 random seeds and report the mean mention-level Micro-F1 score in Table 2. As can be seen, SplitNER(QA-QA) outperforms all baselines on three cross-domain datasets and gives comparable results on *BioNLP13CG*. We present further ablation studies on individual components of our system in Appendix A and a qualitative study in Appendix B.

[6]Sentences in *OntoNotes5.0* are found to be longer and with maximum sequence length set to 256, lots of sentences get truncated. Hence we select a larger limit of 512.

3.4 Latency Evaluation

We compare the latency of our method, SplitNER(QA-QA) and the two single-model NER methods. Table 3 shows the training and inference times. Training time is measured for one epoch and averaged over 10 runs. For a fair comparison, we report the training latency for our system as the sum of span detection and classification even though they can be trained in parallel.

The results show that, compared to Single(QA), our method is 5 to 25 times faster for training and about 5 times faster for inference, and it is especially beneficial for large datasets with many entity types. Compared to Single(SeqTag), our method is slightly slower but achieves much better F1 scores (Table 2). These results validate SplitNER(QA-QA)'s effectiveness in achieving the balance between performance and time efficiency.

4 Related Work

In recent years, deep learning has been increasingly applied for NER (Torfi et al., 2020; Li et al., 2020a), a popular architecture being CNN-LSTM-CRF (Ma and Hovy, 2016; Xu et al., 2021) and BERT (Devlin et al., 2019). Li et al. (2020b,c) propose a QA-based setup for NER using one model for both span detection and classification. Li et al. (2020b); Jiang et al. (2020); Ouchi et al. (2020); Fu et al. (2021); Zhu and Li (2022) perform NER as a span prediction task. However, they enumerate all possible spans in a sentence leading to quadratic complexity w.r.t. sentence length. Our model does a token-level classification and hence is linear.

Xu et al. (2021) propose a Syn-LSTM setup leveraging dependency tree structure with pre-

trained BERT embeddings for NER. Yan et al. (2021) propose a generative framework leveraging BART (Lewis et al., 2020) for NER. Yu et al. (2020) propose a biaffine model utilizing pretrained BERT and FastText (Bojanowski et al., 2017) embeddings along with character-level CNN setup over a Bi-LSTM architecture. All of these models report good performance on *OntoNotes5.0*, however, using BERT-*large* architecture. Nguyen and Vu (2020) propose the BERTweet model by training BERT on a corpus of English tweets and report good performance on *WNUT17*. Wang et al. (2021) leverage external knowledge and a cooperative learning setup. On *BioNLP13CG*, Crichton et al. (2017) report 78.90 F1 in a multi-task learning setup and Neumann et al. (2019) report 77.60 using the SciSpacy system. `SplitNER(QA-QA)` outperforms both of these by a large margin.

5 Conclusion

Using the QA-framework for both span detection and span classification, we show that this division of labor is not only effective but also significantly efficient through experiments on multiple cross-domain datasets. Through this work, we open up the possibility of breaking down other complex NLP tasks into smaller sub-tasks and fine-tuning large pretrained language models for each task.

Limitations

Our proposed approach requires to train two independent classification models. While the models can be trained in parallel, this requires larger GPU memory. For the experiments, we trained two BERT-base models, which have around 220M trainable parameters when trained in parallel. This requires almost twice the GPU memory compared to a single BERT-base NER model, having around 110M trainable parameters.

Owing to a pipeline-based structure, the overall performance of our system is upper bounded by the performance of *Span Detection Model* which has lots of potential for improvement. On dev set, we find that around 30% of errors for *OntoNotes5.0* and *BioNLP13CG*, and around 22% errors on *WNUT17* are just due to minor boundary detection issues. Their entity types are being detected correctly. We henceforth encourage the research community to design architectures or new training objectives to detect mention boundaries more effectively. Currently, in our *Span Detection Model*, all entity mentions are grouped into a single class. As a potential future work, we expect to get even better performance by a hierarchical extension of our setup. At the top level, we can detect mentions belonging to some crude categories and gradually break them down into more fine-grained categories.

Acknowledgements

This work was conducted under the IBM-Illinois Center for Cognitive Computing Systems Research (C3SR[7]), while the first author was a graduate student at University of Illinois Urbana-Champaign and an intern at IBM. We are also very grateful to Prof. Jiawei Han, University of Illinois Urbana-Champaign, for his continued guidance, support and valuable feedback throughout this work.

References

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Hicham Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, and Pierre Zweigenbaum. 2020. Characterbert: Reconciling elmo and bert for word-level open-vocabulary representations from characters. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6903–6915.

Xavier Carreras, Lluís Màrquez, and Lluís Padró. 2002. Named entity extraction using AdaBoost. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Xavier Carreras, Lluís Màrquez, and Lluís Padró. 2003. Learning a perceptron-based named entity chunker via online recognition feedback. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*.

Jason P. C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Trans. Assoc. Comput. Linguistics*, 4:357–370.

Gamal Crichton, Sampo Pyysalo, Billy Chiu, and Anna Korhonen. 2017. A neural network multi-task learning approach to biomedical named entity recognition. *BMC bioinformatics*, 18(1):1–14.

[7] https://www.c3sr.com

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the wnut2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147.

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, NAACL-HLT*, pages 4171–4186.

Jinlan Fu, Xuanjing Huang, and Pengfei Liu. 2021. Spanner: Named entity re-/recognition as span prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP*, pages 7183–7195.

Zhengbao Jiang, Wei Xu, Jun Araki, and Graham Neubig. 2020. Generalizing natural language analysis through span-relation representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 2120–2133. Association for Computational Linguistics.

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

Lukas Lange, Heike Adel, Jannik Strötgen, and Dietrich Klakow. 2021. FAME: feature-based adversarial meta-embeddings for robust input representations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 8382–8395.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2020a. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020b. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 5849–5859. Association for Computational Linguistics.

Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. 2020c. Dice loss for data-imbalanced NLP tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 465–476. Association for Computational Linguistics.

Nut Limsopatham and Nigel Collier. 2016. Bidirectional LSTM for named entity recognition in Twitter messages. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 145–152.

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. *arXiv preprint arXiv:1907.11692*.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics ACL*.

Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. Scispacy: Fast and robust models for biomedical natural language processing. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 319–327.

Dat Quoc Nguyen and Thanh Vu. 2020. Bertweet: A pre-trained language model for english tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14.

Hiroki Ouchi, Jun Suzuki, Sosuke Kobayashi, Sho Yokoi, Tatsuki Kuribayashi, Ryuto Konno, and Kentaro Inui. 2020. Instance-based learning of span representations: A case study through named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 6452–6459.

Sampo Pyysalo, Tomoko Ohta, Rafal Rak, Andrew Rowley, Hong-Woo Chun, Sung-Jae Jung, Sung-Pil Choi, Jun'ichi Tsujii, and Sophia Ananiadou. 2015. Overview of the cancer genetics and pathway curation tasks of bionlp shared task 2013. *BMC bioinformatics*, 16(10):1–19.

Karl Stratos. 2017. Entity identification as multitasking. In *Proceedings of the 2nd Workshop on Structured Prediction for Natural Language Processing, SPNLP@EMNLP*, pages 7–11.

Amirsina Torfi, Rouzbeh A Shirvani, Yaser Keneshloo, Nader Tavvaf, and Edward A Fox. 2020. Natural language processing advancements by deep learning: A survey. *arXiv preprint arXiv:2003.01200*.

Juncheng Wan, Dongyu Ru, Weinan Zhang, and Yong Yu. 2022. Nested named entity recognition with span-level graphs. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL*, pages 892–903. Association for Computational Linguistics.

Xinyu Wang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021. Improving named entity recognition by external context retrieving and cooperative learning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 1800–1812.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*, 23.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing.

Lu Xu, Zhanming Jie, Wei Lu, and Lidong Bing. 2021. Better feature integration for named entity recognition. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3457–3469.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various ner subtasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 5808–5822.

Deming Ye, Yankai Lin, Peng Li, and Maosong Sun. 2022. Packed levitated marker for entity and relation extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4904–4917.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476.

Enwei Zhu and Jinpeng Li. 2022. Boundary smoothing for named entity recognition. *arXiv preprint arXiv:2204.12031*.

A Performance Ablations

Here, we study the individual components of our system, `SplitNER(QA-QA)` in detail. First, we investigate the effectiveness of the additional character and pattern features for span detection. As we can see from Table 4, the character and pattern features improve the NER performance for all datasets.

We also study the effect of the character and pattern features separately. Table 5 shows this ablation study on the *BioNLP13CG* dataset. As we can see, adding the character feature or the pattern feature alone makes a small change in the performance. Interestingly, the character feature helps with recall, while the pattern features improves precision, and, thus, adding them together improves both precision and recall. However, adding part-of-speech (POS) in addition to the character and pattern features shows little impact on the performance.

Next, we compare dice loss and cross-entropy loss for their effectiveness in handling the class imbalance issue in span classification. As shown in Table 6, dice loss works better for imbalanced data confirming the results found in Li et al. (2020c).

Finally, we experimented with different question sentences in *Span Detection Model* to check if BERT is giving any importance to the query part. As shown in Table 7, different queries do have a minor impact but as expected, the model mostly learns not to focus on the query part as can be seen by the comparable results with <*empty*> query.

A.1 Discussions

From the results of the experiments described in Section 3 together with the ablation studies, we make the following observations:

- As shown in Table 2, `SplitNER(QA-QA)` outperforms both the sequence tagging and QA-based baselines on three cross-domain datasets and performs on-par on *BioNLP13CG*.
- The division of labor allows each model to be optimized for its own sub-task. Adding character and pattern features improves the accuracy of *Span Detection Model* (Table 4). However, adding these same features in *Span Classification Model* was found to deteriorate the performance. Similarly, dice loss improves the performance for *Span Classification Model* (Table 6), but no such impact was observed for *Span Detection Model*.

Span Detection Features	*BioNLP13CG*			*CTIReports*			*OntoNotes5.0*			*WNUT17*		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
`+CharPattern`	**91.43**	90.70	**91.06**	**80.59**	77.21	**78.86**	**92.17**	**92.83**	**92.50**	**73.38**	**44.25**	**55.21**
`-CharPattern`	90.31	**91.03**	90.67	79.65	**77.77**	78.70	91.96	92.79	92.37	72.63	44.06	54.85

Table 4: *Span Detection Model* performance with and without character and pattern features.

Features	P	R	F1
`Base Model`	90.31	91.03	90.67
`+Char`	89.85	91.45	90.64
`+Pattern`	91.29	90.22	90.75
`+Char+Pattern`	**91.43**	**90.70**	**91.06**
`+Char+Pattern+POS`	91.14	90.64	90.89

Table 5: *Span Detection Model* performance for *BioNLP13CG* with different feature sets. `Base Model` does not use character and pattern features.

Dataset	Dice Loss	Cross Entropy Loss
BioNLP13CG	**94.27**	94.04
CyberThreats	**87.84**	87.58
OntoNotes5.0	**96.74**	96.50
WNUT17	**73.40**	73.31

Table 6: Span classification performance comparison

Question Type	F1
Extract important entity spans from the following text.	**90.67**
Where is the entity mentioned in the text?	90.38
Find named entities in the following text.	90.32
<empty>	90.48

Table 7: *Span Detection Model* performance for *BioNLP13CG* using different questions. <empty> denotes an empty question sentence. All experiments were done using `Base Model` in Table 5.

- Span detection using the QA setting is slightly more effective than the sequence tagging setup as done in `SplitNER(SeqTag-QA)` (Table 2).
- Our model has more representative power than the baseline approaches, because it leverages two BERT models, each working on their own sub-tasks.
- It also leverages the QA framework much more efficiently than the standard single-model QA system (Table 3). The margin of improvement is more pronounced when the data size and number of entity types increase.
- The training time for our model in Table 3 considers *Span Detection Model* and *Span Classification Model* being trained sequentially. However, the two components can be trained in parallel, reducing the overall train time significantly. The sequential execution is necessary only at inference time.
- *WNUT17* has a diverse range of rare and emerging entities crudely categorized into 6 entity types. A single-model NER system may get confused and try to learn sub-optimal entity-specific extraction rules. Our task segregation allows *Span Detection Model* to form generalized extraction rules which is found to be more effective as shown in Table 2.
- As a sidenote, all the models built in this work outperform the previously published approaches on *BioNLP13CG* (Table 2), thus setting new state-of-the-art results. The credit goes to the SciBERT model and the additional character and pattern features.

B Qualitative Analysis

Table 8 shows some sample predictions by our method, `SplitNER(QA-QA)` and compares them with our single-model NER baseline, `Single(QA)`. From the results, we observe that:

- `SplitNER(QA-QA)` is better in detecting emerging entities and out-of-vocabulary (OOV) terms (e.g., movie titles and softwares). This can be attributed to *Span Detection Model* being stronger in generalizing and sharing entity extraction rules across multiple entity types.
- `Single(QA)` gets confused when entities have special symbols within them (e.g., hyphens and commas). Our character and orthographic pattern features help handle such cases well.
- `Single(QA)` model develops a bias towards more common entity types (e.g., PERSON) and misclassifies rare entity mentions when they occur in a similar context. `SplitNER(QA-QA)` handles such cases well thanks to the dedicated *Span Classification Model* using dice loss.

Category	Example Sentence
General Detection	CVS selling their own version of ... *CVS* selling their own version of ...
Emerging Entities	Rogue One create a plot hole in Return of the Jedi *Rogue One* create a plot hole in *Return of the Jedi*
Scientific Terms	Treating EU - 6 with anti-survivin antisense ... Treating *EU - 6* with anti-survivin antisense ...
Boundary	Hotel Housekeepers Needed in Spring , *TX* ... Hotel Housekeepers Needed in *Spring , TX* ...
OOV Terms	Store SQL database credentials in a webserver Store *SQL* database credentials in a webserver
Entity Type	Why do so many kids in *Digimon* wear gloves? Why do so many kids in *Digimon* wear gloves?

Table 8: Qualitative comparison of `SplitNER(QA-QA)` and `Single(QA)` systems. For each category, the first line shows the result of `Single(QA)`, and the second line shows the result of `SplitNER(QA-QA)`. The words in italics are the entity mentions extracted by the systems color-coded as ORG, CREATIVE WORK, GENE, LOCATION and PRODUCT.

C *CTIReports* Dataset

The *CTIReports* dataset is curated from a collection of 967 documents which include cybersecurity news articles and white papers published online by reputable companies and domain knowledge experts. These documents usually provide deep analysis on a certain malware, a hacking group or a newly discovered vulnerability (like a bug in software that can be exploited). The documents were published between 2016 and 2018. We split the dataset into the train, development, and test sets as shown in Table 9.

A team of cybersecurity domain experts labeled the dataset for the following 8 entity types. These types were selected based on the STIX (Structured Threat Information Expression) schema which is used to exchange cyber threat intelligence. For more detailed information about the 8 types, please refer the STIX documentation[8].

- CAMPAIGN: Names of cyber campaigns that describe a set of malicious activities or attacks over a period of time.
- COURSE OF ACTION: Tools or actions to take in response to cyber attacks.
- EXPLOIT TARGET: Vulnerabilities that are targeted for exploitation.
- IDENTITY: Individuals, groups or organizations.

[8] https://stixproject.github.io/releases/1.2

	Train	Test	Dev	Total
# documents	667	133	167	967
# sentences	38,721	9,837	6,322	54,880
# tokens	465,826	119,613	92,788	678,227

Table 9: Summary of the *CTIReports* corpus showing the number of documents, sentences and tokens in each dataset.

- INDICATOR: Objects that are used to detect suspicious or malicious cyber activity such as domain name, IP address and file names.
- MALWARE: Names of malicious codes used in cyber crimes.
- RESOURCE: Tools that are used in cyber attacks.
- THREAT ACTOR: Individuals or groups that commit cyber crimes.

Table 10 and Table 11 show the statistics of the entity types in the corpus and some sample mentions of these types respectively.

Entity Type	Train	Dev	Test
CAMPAIGN	247	27	85
COURSE OF ACTION	1,938	779	329
EXPLOIT TARGET	5,839	1,412	1,282
IDENTITY	6,175	1,262	1,692
INDICATOR	3,718	1,071	886
MALWARE	4,252	776	1,027
RESOURCE	438	91	114
THREAT ACTOR	755	91	144

Table 10: The number of mentions for each entity type in the train, development and test sets

Entity Type	Examples
CAMPAIGN	“Operation Pawn Storm”, “The Mask” “MiniDuke”, “Woolen-Goldfish” “Ke3chang”
COURSE OF ACTION	“Trojan.Poweliks Removal Tool” “HPSBHF03535”, “TDSSKiller” “cd chktrust -i FixTool.exe” “http://www.ubuntu.com/usn/usn-2428-1” “Initial Rapid Release version June 15, 2015 revision 02”
EXPLOIT TARGET	“CVE-2015-8431”, “Adobe Flash Player” ”Ubuntu”, “Windows”, “CGI.pm” “version 20.0.0.306 and earlier”
IDENTITY	“Symantec”, “Jon DiMaggio”, “Belgium” “Kaspersky Lab”, “RSA”
INDICATOR	“C:\WINDOWS\assembly\GAC_MSIL” “hxxp://deschatz-army.net”, “67.23.112.226” “b4b483eb0d25fa3a9ec589eb11467ab8”
MALWARE	“ChewBacca”, “SONAR.AM.E.J!g13” “Trojan.Poweliks”, “BlackHole” “TDL3”, “LockyZeus” “JS/TrojanDownloader.Nemucod”
RESOURCE	“IRC”, “Tor”, “DroidPlugin”, “Onion” “PowerShell”, “Google Play” “Free Pascal 2.7.1.”, “Teamviewer”
THREAT ACTOR	“ProjectSauron”, “Strider”, “Ogundokun” “APT28”, “APT 28”, “Fancy Bear” “Pro_Mast3r”, “Equation Group”

Table 11: Sample entity mentions for each type in the *CTIReports* corpus