Improving and Simplifying Template-Based Named Entity Recognition

Murali Kondragunta **Olatz Perez-de-Viñaspre** **Maite Oronoz**
HiTZ Center - IXA research group,
University of the Basque Country
{mkondragunta001}@ikasle.ehu.eus
{olatz.perezdevinaspre, maite.oronoz}@ehu.eus

Abstract

With the rise in larger language models, researchers started exploiting them by pivoting the downstream tasks as language modeling tasks using prompts. In this work, we convert the Named Entity Recognition task into a seq2seq task by generating the synthetic sentences using templates. Our main contribution is the conversion framework which provides faster inference. In addition, we test our method's performance in resource-rich, low resource and domain transfer settings. Results show that our method achieves comparable results in the resource-rich setting and outperforms the current seq2seq paradigm state-of-the-art approach in few-shot settings. Through the experiments, we observed that the negative examples play an important role in model's performance. We applied our approach over BART and T5-base models, and we notice that the T5 architecture aligns better with our task. The work is performed on the datasets in English language.

1 Introduction

Named Entity Recognition (NER) is traditionally approached as a sequence labeling task where a tag is predicted for each token. For a sentence with l tokens, the output would be l tags, usually, in the Inside–outside–beginning (IOB) tagging format (Ramshaw and Marcus, 1995).

Traditionally in probabilistic classification, Language Models (LMs) "compute the probability of a label, conditioned on the text" (Eisenstein, 2018), that is, they quantify the likelihood of a sentence to pertain to a language. Contextualized word representations are used nowadays as pre-trained language models that have shown to be effective in natural language processing tasks as co-reference resolution, gender resolution, etc.

Current language modeling approaches can be broadly divided into two types:

1. Auto-Regressive: Models that generate predictions, in our case pieces of text, using previous predictions. Architectures like Generative Pre-trained Transformers (i.e GPT3) (Brown et al., 2020) follow this approach predicting the next word given a sequence of prior words.

2. Sentence-Reconstruction: In this approach, the input sentence is corrupted by either randomly dropping words (Devlin et al., 2018), spans (Joshi et al., 2019) or permutating the word order (Lewis et al., 2019) and the model is trained to reconstruct the original sentence. Architectures like Bidirectional Encoder Representations from Transformers or BERT models (Devlin et al., 2018) and T5 (Raffel et al., 2020) follow this approach.

Mou et al. (2016) and Dai and Le (2015) suggested transferring the knowledge learned during pre-training to downstream tasks but with limited success. However, with the advances in neural networks, (Howard and Ruder, 2018) proved that it is possible by successfully fine-tuning a pretrained LSTM (Hochreiter and Schmidhuber, 1997) language model on a downstream task. After pre-training, the model's last layer, which is used to predict the next word, is replaced with a new layer for a downstream task. This way, information from the remaining layers can help the model learn new tasks better.

Instead of introducing a new layer at the end of pre-trained models, Schick and Schütze (2020) converted the downstream task examples into a cloze-question format to leverage the knowledge acquired during the pre-training. Later, the pre-trained model is further fine-tuned in a method called Pattern Exploitative Training (PET). Downstream tasks like sentiment analysis would be much simpler to adapt since one can just append a template at the end of the review and expect the model to predict words related to the corresponding label.

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop, pages 79–86

For instance, if we are predicting the polarity of a movie review, we append a template to the review as follows *Overall, the movie is* $< mask >$ and fine-tune the language model to predict $< mask >$ with the words corresponding to the sentiment. However, `PET` cannot be directly applied to Named Entity Recognition (NER) since the outputs must be constrained to phrases within the sentence. *It is difficult (if not impossible) to provide a single task description which allows the language model to assign a label to each token in the input text* (Gatta et al., 2021). Gatta et al. (2021) handled this issue by appending a template, filled by each word and entity type, to the sentence where the model must predict if the word in the template is either beginning, inside or outside an entity type. The template is generated for all combinations of words in the sentence and entity types.

Cui et al. (2021) approached the problem by converting the NER identification task into a seq2seq task by generating synthetic target sentences. Figure 1 illustrates their data creation process. N-grams generated from the sentence are substituted in the template to generate positive and negative examples. The authors fine-tune BART language model (Lewis et al., 2019) to train a seq2seq model, which inputs the natural source sentence (sentence before the "$\rightarrow$" symbol in Figure 1) and outputs the synthetic target sentence (sentence after the "$\rightarrow$" symbol). At inference, the fine-tuned model is used to score the synthetic sentences generated from all the N-grams. A major limitation of this approach is the N-grams part because, during inference, for a given sentence, all the synthetic target sentences generated from all the N-grams ranging from n = 1 to 8 must be checked against all the entity types. This situation exacerbates with a larger sentence and more number of entity types in a task. Avoiding the problem of generating all possible N-grams, (Yan et al., 2021) proposed a pointer-based seq2seq (BARTNER) framework, which converts NER sub-tasks to a unified sequence generation task and predicts entities from the input sentences and the corresponding type indexes. LightNER (Chen et al., 2021) introduced prompt-tuning to the attention mechanism of BARTNER and achieved promising improvement in low-resource scenarios.

Our **contribution** is to avoid the usage of N-grams by pivoting our task from a sequence labeling task into a seq2seq task that aligns with the pre-training objective of certain generative models like T5, BART, etc. Specifically, we leverage templates to convert the target entities into sentences. Figure 2 illustrates the conversion process. Our approach performs better in low-resource settings and is comparable to existing works in resource-rich setting. In comparison to the recent works, our approach is faster than `TemplateBART` (Cui et al., 2021) in terms of real-time inference. Although BARTNER does inference in a single step, we observed that our approach yields better results at the cost of linear inference speed, proportional to the number of entity types. At the same time, our approach is simpler than LightNER.

We test our method's performance in resource-rich, low resource and domain transfer settings to prove its robustness. We approximate low resource setting by considering few-shot scenarios of the datasets. Results show that our method achieves comparable results in the resource-rich setting and outperforms the current seq2seq paradigm state-of-the-art approach in few-shot settings.

2 Related Work

To adapt the fine-tuning approach to the NER task, the standard norm has been to use architectures like LSTMs (Hochreiter and Schmidhuber, 1997), CNNs (LeCun et al., 1989), and Transformers (Vaswani et al., 2017) to extract token-level features which are later passed on for classification into the corresponding entity classes. In the final layer, the softmax function (Strubell et al., 2017; Chiu and Nichols, 2015; Cui and Zhang, 2019) or the Conditional Random Fields algorithm (CRF) (Lample et al., 2016; Ma and Hovy, 2016; Luo et al., 2019) are used for classification.

With the rise of powerful LMs, there has been a lot of interest in exploiting them for downstream tasks. ULMFit (Howard and Ruder, 2018) was the first approach to successfully fine-tune a pre-trained LSTM language model to a downstream task. However, this approach does not completely exploit the information LMs have learned during pre-training. Schick and Schütze (2020) converted the downstream tasks into cloze questions to probe LMs and leverage the knowledge learned during pre-training. Simultaneously, Brown et al. (2020) showed that, with large LMs (170 Billion parameters large), it is possible to probe information in a few-shot approach without having to train the full model. These methods gave rise to a new research field called Pattern Engineering (Liu et al., 2021).

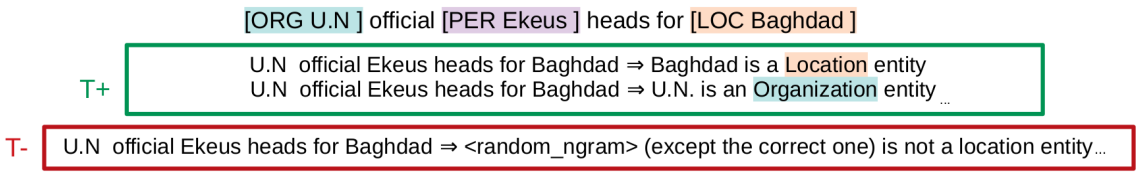

Figure 1: Template based NER approach by Cui et al. (2021)

.

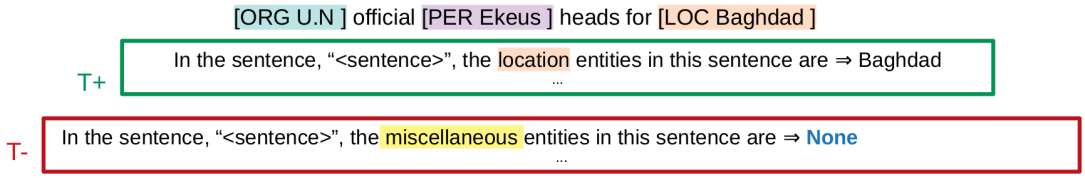

Figure 2: Our approach.

However, until recently, most prompt-based approaches were not designed for Named Entity Recognition. Cui et al. (2021) were the first to adopt the template-based approach for Sequence Labeling. They convert the task into a seq2seq task and use language models as a scoring function for each span. As mentioned before, Sainz et al. (2022) filter the n-grams by labels provided by a linguistic analyzer in a textual entailment task.

3 Methodology

We consider NER as a language model generation task where the input is a text written in natural language, appended by synthetic sentence generated by a template and the output is also a synthetic sentence but mentioning the corresponding entities. (see Section 3.1). We start this section by introducing the template creation process in Section 3.1, and then explain the training and inference processes in Sections 3.3 and 3.2, respectively. In later sections, we speak about the different assumptions made for different resource settings.

3.1 Template creation

Let us consider a dataset with n entity types. For each training instance, n synthetic sentences (corresponding to each entity type) are appended to the source sequence i.e. input text and the corresponding entities are provided as target sequences. In case of multiple entities for an entity type, they are separated by a delimiter in the output. Figure 2 explains our approach. The input template is as follows,

In the sentence, $<$ *sentence* $>$*, the* $<$ *entity* $-$ *type* $>$ *entities are*

In this template $<$ *sentence* $>$ is substituted by a sentence from the training set and $<$ *entitytype* $>$ by an entity type. The training instance shown in the figure has three entity types, namely, ORG (organization), PER (person), and LOC (location). So, we will have 3 positive examples, one for each entity type. One such positive example is shown as T^{+}. Since MISC (miscellaneous) type entities are not present, we consider it as a negative example (T^{-}) and the output is made as "None" denoting that there are no entities for the entity type mentioned in the input.

3.2 Inference

During inference, for a given sentence, templates generated for each entity type are passed through the model for extracting entities. This approach invokes the model only n times whereas Cui et al. (2021)'s approach invokes the model $n \times k$ times, where k is the number of N-grams.

3.3 Training

As we are leveraging pre-trained language models as the base models, we fine-tune them on the downstream seq2seq task. During fine-tuning, the model is trained to predict the correct named entities word-by-word. In case of negative samples, the model must only output the word `None`. In the example in Figure 2, the system has generated the named entity "Baghdad" for the location (LOC) entity type and the label "None" for the "miscellaneous" type, or negative example. It is possible that the generative models may output partial entities

or arbitrary text. In such cases, we consider the output as no match at all.

Considering the success of contrastive learning (Chen et al., 2020), we believe that the negative examples help the models in understanding the task better, especially when the training instances are less. Therefore, we consider all the n sequences (including positives and negatives) generated per each training instance for training. In later sections (Section 4.1), we evaluate our hypothesis by comparing the impact of negative examples against model performance. Experiments have been performed on different datasets with different entity classes (see Section 4).

3.4 Experimental settings

As mentioned earlier, we test our approach in three different scenarios. The following sub-sections explain the experimental settings of each scenario.

3.4.1 Resource-Rich setting

In this setting, we evaluate if it is worth adopting this approach when there is access to abundance of data.

3.4.2 Low Resource setting

In this setting, we measure the impact of the amount of examples against model's performance. We follow Cui et al. (2021)'s evaluation settings for better comparison. Specifically, we check the model's performance on different input data sizes m = {10, 20, 50, 100, 200, 500}, where m stands for number of instances per each entity type.

3.4.3 Domain Transfer

Since the input template and target sequences look like natural sentences, the model trained on one dataset should be able to adapt to another dataset with different entity types. The above hypothesis is made under the hypothesis that, with the first dataset, model understands the task format better.

We evaluate this hypothesis by training a model on a resource-rich dataset and later fune-tuning it on a low-resource dataset. In our case, it would mean, fine-tuning a pretrained model on a resource-rich dataset and further fine-tuning it on a low-resource dataset.

3.5 Adding special tokens

One-way of pre-training is to reconstruct the corrupted sentence by predicting the dropped spans (Joshi et al., 2019; Raffel et al., 2020). For instance, given a sentence, *I'm looking <X> to the party this <Y>*, the model is trained to predict the following text, *<X> forward <Y> weekend.*

In order to make our downstream task closer to the pre-training task of de-noising corrupted sentences, we append a special token to the source sentence and prepend the same token on the target side to mimic the pre-training style. In the results, we call this method as "Ours+ (T5 pretraining-style)".

Ours: *In the sentence,* $<$ *sentence* $>$*, the* $<$ *entity* $-$ *type* $>$ *entities are*

Ours + (T5 pretraining-style): *In the sentence,* $<$ *sentence* $>$*, the* $<$ *entity* $-$ *type* $>$ *entities are* $<$ *X* $>$

4 Results

For benchmarking, we use CoNLL03 dataset for resource-rich setting and MIT restaurant (Liu et al., 2013), MIT movie (Liu et al., 2013) and ATIS (Hakkani-Tur et al., 2016) datasets are used for low-resource and domain transfer settings. See Table 4 in Appendix for more details. ATIS dataset contains highly imbalanced entity types with one entity type containing around 3705 instances while another type occurring only once. All the scores reported in this section are an average of 5 runs.

We conjecture that our hypothesis aligns better with T5 considering the closeness to its pre-training task. In order to evaluate this, we train T5-base and BART-large (Lewis et al., 2019) on CoNLL03 (Tjong Kim Sang and De Meulder, 2003) and check the performance. Table 2 shows that T5-base performs better than BART-large. It is interesting to note that T5-base, with 3x fewer parameters, performs better.

From here on, we will be using T5-base model for the experiments. **The improvements reported can be attributed to T5 but the inference speed is due to our approach.** For instance, when tested on `Titan XP` GPU with the same pre-trained model, T5, our approach takes 2 minutes to complete the inference whereas `TemplateBART` (Cui et al., 2021) approach takes 68 minutes.[1].

Resource-rich setting: Table 2 shows that, when tested in a resource-rich setting, (CoNLL03),

[1]We use a batch size of 32 and TemplateBART use a dynamic batch size varying between 5 to 40 with an average batch size of 31

Source	Method	MIT Movies						MIT Restaurant						ATIS		
		10	20	50	100	200	500	10	20	50	100	200	500	10	20	50
None	TemplateBART	37.3	48.5	52.2	56.3	62.0	74.9	46	57.1	58.7	60.1	62.8	65	71.7	79.4	92.6
	LightNER	41.7	57.8	73.1	78.0	80.6	84.8	48.5	58.0	62.0	70.8	75.5	80.2	76.3	85.3	92.8
	BARTNER*	41.1	54.0	67.7	NA	NA	NA	44.0	56.0	64.0	NA	NA	NA	77.7	86.1	93.4
	Ours	**53.29**	**61.89**	**75.75**	**79.99**	**81.61**	**83.08**	47.09	60.25	**67.02**	72.7	74.15	75.69	**91.9**	**93.8**	**94.58**
	Ours2	52.1	60.43	75.23	78.3	81.07	82.47	**51.34**	**61.97**	65.86	**73.64**	**76.31**	**77.24**	**92.75**	**93.57**	**94.66**
CoNLL03	TemplateBART	42.4	54.2	59.6	65.3	69.6	80.3	53.1	60.3	64.1	67.3	72.2	75.7	77.3	88.9	93.5
	LightNER	62.9	75.6	78.8	82.2	84.5	85.7	58.1	67.4	69.5	73.7	78.4	81.1	86.9	89.4	93.9
	Ours	62.27	72.24	76.67	78.98	81.72	82.89	56.55	64.73	69.84	73.39	75.46	74.41	**93.34**	**94.25**	**95**
	Ours2	62.07	70.52	75.64	78.84	81.09	82.56	59.03	64.06	67.72	74.21	75.47	75.94	92.9	93.83	94.68

Table 1: Few-shot results (including domain-transfer settings) on various input sizes. The first column "Source" refers to the domain-transfer setting where we first train on CoNLL03 dataset or from scratch (None) and later fine-tune on the current dataset. The scores reported are an average of 5 runs with a standard deviation around 2-3. "Ours" and "Ours2" refer to the two training styles followed in section 3.5. BARTNER* scores are taken from the publication. Hence, couldn't get its scores on size=100, 200 and 500.

Models	P	R	F
LightNER	92.39	93.48	92.93
TemplateBART	90.51	93.34	91.90
Ours (T5-base)	91	89	90
Ours (BART-large)	90	80	85

Table 2: CoNLL03 results: Comparing our approach with TemplateBART and LightNER

`TemplateBART` and `LightNER` perform better than our approach. We conclude that our approach is not a value addition when there is access to enough labeled data.

Low Resource setting: In the few shot setting, we check the model's performance on different input data sizes m = {10, 20, 50, 100, 200, 500}, where m stands for number of instances per each entity type. Table 1 shows that our approaches perform comparatively better.

Domain Transfer: Initially, we fine-tune the T5 model on the full CoNLL03 dataset and later fine-tune it on the few-shot scenarios of MIT restaurant, MIT movie and ATIS datasets. Table 1 shows that our approaches indeed leverage the knowledge gained from one task to learn another. However, it only performs comparably to LightNER.

4.1 Ratio of negative samples

In case of resource-rich setting (Table 2), we can observe that the precision of the system is comparable to other baselines. We believe that this is due to the large number of negative examples we are considering while training. As a result, model is producing less false positives.

In order to check the hypothesis that more negative examples can be helpful, we experimented with different negative-positive ratio sentences per each training instance. Table 3 shows that the negative samples are extremely important in few-shot scenarios, especially when m (number of instances per entity type) is small. We can also observe that learning is saturated quickly when m is bigger. For instance, for m=200 & 500, there is no considerable improvement despite the increase in negative samples.

-ve to +ve ratio	10	20	50	100	200	500
1:1	18.32	42.97	57.58	67.06	71.83	75.43
2:1	26.25	53.95	62.88	70.55	75.13	77.24
3:1	34.32	55.91	64.68	71.1	75.37	77.2

Table 3: Effect of negative samples on model's performance. Scores reported on MIT restaurant. 10 indicates 10 instances for each entity types. The scores reported are an average of 5 runs with a standard deviation around 2-3.

5 Conclusions and Future Work

In this work, we project NER as a seq2seq task by generating synthetic sentences from templates and show that our approach is faster than the current state-of-the-art. We also show that our approach works better in few-shot and domain transfer scenarios especially when the input size is small, although the approach can be used in other scenarios. While performing the experiments, we found T5 to better align with our task and also observed the importance of negative examples.

Our system has been compared against seq2seq systems (TemplateNER, LightNER and BARTNER). But Instruction-NER (Wang et al., 2022) leverages auxiliary training and obtains better results than our system. For instance, in MIT movies few shot scenario (n=10), our approach has 53 F1 score and instructionNER 65 F1 score. In future, it would be interesting to explore automatic template generation along with auxiliary training.

6 Acknowledgments

We would like to thank the members of IXA research group for the discussion and feedback. The first author is supported by the Erasmus Mundus Master's Programme "Language and Communication Technologies".

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. CoRR, abs/2005.14165.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A simple framework for contrastive learning of visual representations.

Xiang Chen, Lei Li, Shumin Deng, Chuanqi Tan, Changliang Xu, Fei Huang, Luo Si, Huajun Chen, and Ningyu Zhang. 2021. Lightner: A lightweight tuning paradigm for low-resource NER via pluggable prompting. CoRR, abs/2109.00720.

Jason P. C. Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. CoRR, abs/1511.08308.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using BART. CoRR, abs/2106.01760.

Leyang Cui and Yue Zhang. 2019. Hierarchically-refined label attention network for sequence labeling. CoRR, abs/1908.08676.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. CoRR, abs/1511.01432.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. CoRR, abs/1810.04805.

Jacob Eisenstein. 2018. Natural language processing.

Valerio La Gatta, Vincenzo Moscato, Marco Postiglione, and Giancarlo Sperlì. 2021. Few-shot named entity recognition with cloze questions. CoRR, abs/2111.12421.

Dilek Hakkani-Tur, Gokhan Tur, Celikyilmaz Asli, Yun-Nung Chen, Jianfeng Gao, li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. CoRR, abs/1801.06146.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019. Spanbert: Improving pre-training by representing and predicting spans. CoRR, abs/1907.10529.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. CoRR, abs/1603.01360.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. 1989. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4):541–551.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. CoRR, abs/1910.13461.

Jingjing Liu, Panupong Pasupat, Scott Cyphers, and Jim Glass. 2013. Asgard: A portable architecture for multilingual dialogue systems. In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pages 8386–8390.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. CoRR, abs/2107.13586.

Ying Luo, Fengshun Xiao, and Hai Zhao. 2019. Hierarchical contextualized representation for named entity recognition. CoRR, abs/1911.02257.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. CoRR, abs/1603.01354.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in NLP applications? CoRR, abs/1603.06111.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140):1–67.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In Third Workshop on Very Large Corpora.

Oscar Sainz, Haoling Qiu, Oier Lopez de Lacalle, Eneko Agirre, and Bonan Min. 2022. Zs4ie: A toolkit for zero-shot information extraction with simple verbalizations.

Timo Schick and Hinrich Schütze. 2020. Exploiting cloze questions for few-shot text classification and natural language inference. CoRR, abs/2001.07676.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate sequence labeling with iterated dilated convolutions. CoRR, abs/1702.02098.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003, pages 142–147.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. CoRR, abs/1706.03762.

Liwen Wang, Rumei Li, Yang Yan, Yuanmeng Yan, Sirui Wang, Wei Wu, and Weiran Xu. 2022. Instructionner: A multi-task instruction-based generative framework for few-shot ner.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various NER subtasks. CoRR, abs/2106.01223.

7 Appendix

Dataset	Number of entity types	# Training	# Validation	# Testing	Max	Min
CoNLL	4	14041	3250	3453	7141	3451
MIT Restaurant	8	6128	1225	1522	3031	581
MIT Movie	12	7821	1564	2444	3510	92
ATIS	79	4233	634	894	3705	1

Table 4: Datasets: Number of sentences in train, validation and testing splits in each dataset. # refers to number of sentences. The "Max" and "Min" columns refer to the entity types with maximum and minimum occurrences.