Report in Algorithm-Focused Tasks: Named Entity Recognition problem

Thi-Trang Nguyen * * HCIR lab, SeoulTech
trangnguyen.hust117@gmail.com

***Abstract*— The report is organized as follows: Section I introduces some important work in NER. Section II presents my implementation and results I got. Section III describes my application.**

I. A SURVEY ON RECENT ADVANCES IN NAMED ENTITY RECOGNITION

A. *Definition*

Named Entity Recognition (NER) is a field of computer science and natural language processing (NLP) that deals with identifying and classifying named items in unstructured text. Named entities are specific words or phrases that refer to real-world objects such as people, organizations, locations, dates, quantities, and gene and protein names in the biomedical domain. By formal context, given a sequence of tokens denoted $T = (t_1, t_2, ..., t_N)$, NER helps produce a collection of tuples (I_s, I_e, l), where s and e are both integers that be within the interval $[1, N]$; Is and Ie correspond to the beginning and the ending indices of a named entity mention respectively, and l denotes the type of entity from a predefined set of categories.
For instance, given the sentence *"Barack Obama was born in Honolulu."*, the tokens *"Barack Obama"* are identified as the person's name and *"Honolulu"* as the location by NER

Q1:*Can you summarize the overall research and state of the art you learned in no more than one page?*
Answer: Although large-scale Language Models (LLM) have achieved SOTA performances on a variety of NLP tasks, its performance on NER is still significantly below supervised baselines. This is due to the gap between the two tasks the NER and LLMs: the former is a sequence labeling task in nature while the latter is a text-generation model.
Approaches based on contextualized embeddings, such as ELMo [1], Flair [2], BERT [3], and XLM-R [4] have been consistently raising the state-of-the-art for various structured prediction tasks such as Named Entity Recognition.

1, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [3]
BERT is designed to pre-train deep bidirectional representations from the unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications. The question-answering example in Figure 1 will serve as a running example for the Fine-tuning step.
There are two approaches to applying BERT to the CoNLL-2003 Named Entity Recognition (NER) task [5]. All of the BERT results presented so far have used the fine-tuning approach, where a simple classification layer is added to the pre-trained model, and all parameters are jointly fine-tuned on a downstream task. However, the Feature-based approach, where fixed features are extracted from the pretrained model, has certain advantages. Table I shows the performance of BERT on CoNLL-2003 dataset. It can easily be seen that the baseline BERT still outperforms the large language model approach GPT3-NER [6]

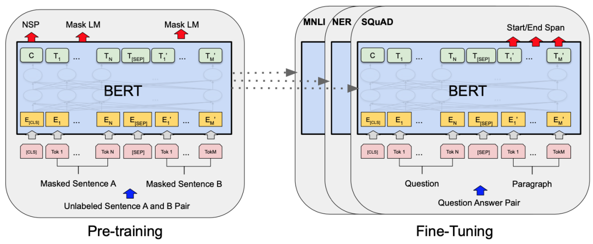

Figure 1. Overall pre-training and fine-tuning procedures for BERT [3]. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different downstream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

Table I
CoNLL-2003 Named Entity Recognition results. Hyperparameters were selected using the Dev set. The reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.

Model	Dev F1	Test F1
ELMo [1]	95.7	92.2
Fine-tuning approach		
$BERT_{LARGE}$	96.6	92.8
$BERT_{BASE}$	96.4	92.4
Feature-based approach ($BERT_{BASE}$)		
Embeddings	91	-
Second-to-last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 layers	95.5	-
ACE [7]	-	93
ACE + document-context	-	**94.6**
GPT-3 + entity-level embedding [6]	-	90.91

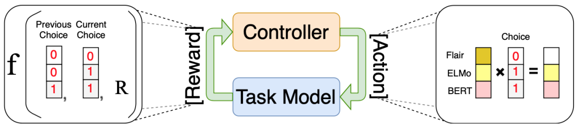

Figure 2. The main paradigm of ACE [7] approach is shown in the middle, where an example of reward function is represented in the left and an example of a concatenation action is shown in the right

Table II

Performance of joint models on PhoATIS (Vietnamese). The "Intent Acc." and "Sent. Acc." columns denote intent detection accuracy and sentence accuracy, respectively.

Model	Intent Acc.	Slot F1	Sent Acc.
JointIDSF [9]	97.62	94.98	86.25
JointBERT + CRF [9]	97.40	94.75	85.55
JointIDSF	**97.64**	95.06	**86.56**
JointBERT + CRF	97.42	**95.18**	86.33

2, ACE: Automated Concatenation of Embeddings for Structured Prediction [7]

Recent work found that better word representations can be obtained by concatenating different types of embeddings. However, the selection of embeddings to form the best-concatenated representation usually varies depending on the task and the collection of candidate embeddings, and the ever-increasing number of embedding types makes it a more difficult problem. In this work, the Automated Concatenation of Embeddings (ACE) is proposed to automate the process of finding better concatenations of embeddings for structured prediction tasks, based on a formulation inspired by recent progress on neural architecture search. They focus on *searching for better word representations rather than better model architectures*. Figure 2 explains the approach of ACE. This work is the current state of the art on CoNLL-2013 by 94.6% with additional document context.

3, Joint models

Spoken Language Understanding (SLU) has received much attention in recent years with the proliferation of portable devices, smart speakers, and the evolution of personal assistants, such as Amazon's Alexa, Apple's Siri, Google's Assistant, and Microsoft's Cortana. NER is a key component known well as *Slot filling*, and the other is *Intent Detection*.

JointBERT+ CRF [8] employs a shared BERT encoder and two separate decoders for intent detection and slot filling that are structured on top of the encoder. This work evaluates on ATIS dataset which includes audio recordings of people making flight reservations. The other work JointIDSF [9] extends the recent JointBERT+CRF model with an intent-slot attention layer to explicitly incorporate intent context information into slot filling via "soft" intent label embedding (As in Figure 3). Experimental results on the Vietnamese dataset namely PhoATIS show that the proposed model significantly outperforms JointBERT+CRF as shown in Table II.

Table III

Performance of joint models on ATIS (English). The "Intent Acc." and "Sent. Acc." columns denote intent detection accuracy and sentence accuracy, respectively.

Model	Intent Acc.	Slot F1	Sent Acc.
JointBERT + CRF [8]	97.9	96	88.6
JointBERT + CRF	**97.53**	**95.64**	87.45
JointIDSF	97.31	95.61	**87.79**

II. Implementation

Q2: *Why do you choose this paper? Can you present your implementation and how it works? Are you able to reproduce the results of the paper? Observe, analyze, and interpret the results.*

For this report, I choose the JointIDSF model for implementation because the approach using pretrains is still highly effective for the labeling task. Secondly, the JointIDSF model is an extension of JointBERT by adding an attention layer to enrich the information between the Intent detection and Slot filling (NER) tasks.

In the implementation part, there are two components to pay attention to in this code. Firstly, in the architecture part, an additional attention layer is created to incorporate the information intent context information into slot filling via "soft" intent label embedding after getting the representation information from the encoder (Figure 3). Second, the loss function is defined by the sum of Intent's loss and Slot's loss in the Training part. For both JointIDSF and JointBERT+CRF, I employ the AdamW optimizer and set the batch size to 32 due to the limitation of my local memory computer. The Adam initial learning rate equals $5e-5$ and the mixture weight $\lambda = 0.15$ and $\lambda = 0.5$ for PhoATIS and ATIS datasets, respectively.

Moreover, I employ the pre-trained language models namely PhoBERT as the encoder in the Vietnamese dataset, and Roberta-base as the encoder in the English dataset. I train for 50 epochs and calculate the score of

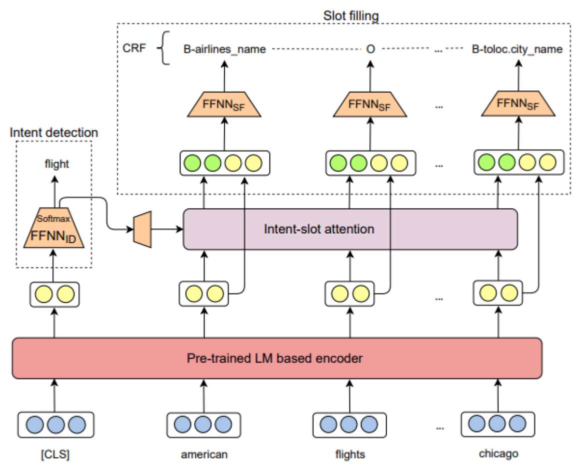

Figure 3. Illustration of JointIDSF [9]. NER is a key component known well as Slot filling, and the other is Intent Detection.

the intent accuracy for intent detection and the F1-score (in %) for slot filling after each training epoch on the validation. My result is provided in Table II and III ([9] provided their results on average over 5 runs with 5 different random seeds).

III. Application

Q3: *Can you present your extension and how it works? Why is it useful? Any ideas to make it even more useful?*
Answer: Yes, I can. First I deploy an API by Flask, and then I connect to Google Extension using js. The user visits a webpage in Chrome, your extension will read the text that needs to extract entities, then send the request to my server. Last, the results that the text is extracted entities back to the extension.
Additionally, I also created a simple webpage using Gradio, which can display extracting entities in the Flight domain in both Vietnamese and English. My video which is attached to this report provides details about my extension and my simple webpage

Q4.1: *What have you learned from the exercise above?*
Answer: I met an issue with reading requests into API in the process of connecting my extension, then I took 2 hours to figure it out. Interestingly, I have learned how to deploy a simple Google extension because I just concentrated on working with the model implementation and validation before. Moreover, I also found an open-source Python package namely Gradio that allows quickly build a demo or web application for the machine learning models, API, or any arbitrary Python function.

Q4.2: *Do you think the results were satisfactory for practical usage in the application of browser extension? How can we adapt/improve the techniques to improve?*
Answer: Regarding the performance aspect of the model, the slot filling (NER) task can accommodate the expansion of practical applications for a specific domain (Flight domain). But for multi-domain applications, more diverse data sources and a more effective feature extraction approach are needed. For example, for this problem, it is necessary to exploit two-way information flow (intent-slots/slots-intent) to improve both intention detection and slots-filling tasks instead of only using one-way information as in the original research.

Q4.3: *What new applications can be built using this problem? Propose a cool application you can build that does not exist today.*
Answer: As I mentioned in the Survey part, this problem plays a key role in Spoken Language Understanding. We can build Assistant systems in the Service industry such as Teaching, Flights, and Restaurants.

References

[1] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, M. Walker, H. Ji, and A. Stent, Eds. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 2227–2237. [Online]. Available: https://aclanthology.org/N18-1202

[2] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf, "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)*, W. Ammar, A. Louis, and N. Mostafazadeh, Eds. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 54–59. [Online]. Available: https://aclanthology.org/N19-4010

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[4] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," 2020.

[5] E. F. Tjong Kim Sang and F. De Meulder, "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*, 2003, pp. 142–147. [Online]. Available: https://aclanthology.org/W03-0419

[6] S. Wang, X. Sun, X. Li, R. Ouyang, F. Wu, T. Zhang, J. Li, and G. Wang, "Gpt-ner: Named entity recognition via large language models," 2023.

[7] X. Wang, Y. Jiang, N. Bach, T. Wang, Z. Huang, F. Huang, and K. Tu, "Automated concatenation of embeddings for structured prediction," 2021.

[8] Q. Chen, Z. Zhuo, and W. Wang, "Bert for joint intent classification and slot filling," 2019.

[9] M. H. Dao, T. H. Truong, and D. Q. Nguyen, "Intent detection and slot filling for vietnamese," *CoRR*, vol. abs/2104.02021, 2021. [Online]. Available: https://arxiv.org/abs/2104.02021