i-Parser: Interactive Parser Development Kit for Natural Language Processing

Gaku Morio, Hiroaki Ozaki, Yuta Koreeda, Terufumi Morishita, Toshinori Miyoshi

1Research and Development Group, Hitachi, Ltd., Kokubunji, Tokyo, Japan
2Research and Development Group, Hitachi America, Ltd., Santa Clara, CA, USA

gaku.morio.vn@hitachi.com, hiroaki.ozaki.yu@hitachi.com, yuta.koreeda@hal.hitachi.com,
terufumi.morishita.wp@hitachi.com, toshinori.miyoshi.pd@hitachi.com

Abstract

This demonstration paper presents i-Parser, a novel development kit that produces high-performance semantic parsers. i-Parser converts training graphs into sequences written in a context-free language, then our proposed model learns to generate the sequences. With interactive configuration and visualization, users can easily build their own parsers. Benchmark results of i-Parser showed high performances of various parsing tasks in natural language processing.

Introduction

Parsing text into a graph is key technology for dialog systems, document analysis, and so on. In fact, there exist abundant graph representations to deal with different needs. For example, Abstract Meaning Representation (AMR; Banarescu et al. (2013)) is designed to represent high-level sentence meaning as a graph, and it is useful for dialog systems. Elementary Dependency Structures (EDS; Oepen and Lønning (2006)) based on English head-driven phrase structure grammar requires node-to-input anchoring. Math word problems (Wang, Liu, and Shi (2017)) are aimed at generating a computation tree from a natural language math problem. However, most of these graphs have been parsed by a task-specific system due to differences in graph forms.

In this paper, we present a novel development kit, i-Parser, that allows developers to easily implement desirable parsers. Figure 1 shows an overview of i-Parser. Users only need to provide training data and graph specifications. The system interactively generates a setup file by asking users graph outline questions such as Tree or DAG? With the setup file, i-Parser Converter universally transforms a given graph into our proposed Plain Graph Notation (PGN) sequence. Finally, i-Parser Model that leverages Transformers (Vaswani et al. 2017) is trained to generate PGN sequences.

i-Parser supports many graph variants without any task-specific implementations (e.g., transition design). i-Parser also provides out-of-the-box visualization and hyperparameter tuning tools. This allows developers to benefit from faster development cycles.

Figure 1: An overview of i-Parser. Top (training): We convert a graph into a PGN sequence, then we train the Transformer encoder-decoder. Bottom (inference): We predict the PGN for the input text to obtain an output graph.

Results of semantic graph parsing and math word problems show that i-Parser achieves high performances. 1

System

i-Parser Converter  This tool converts a graph into a PGN sequence to represent graph variants in a unified text-based notation. For example, an AMR graph (Figure 1; bottom right) can be represented as PGN, as shown in Figure 1 (bottom middle). The i-Parser Converter can also reconstruct a graph from a predicted PGN sequence.

The input graph can be in CoNLL-U or MRP (i.e., JSON) formats (Oepen et al. 2019). We also allow command options as follows:

```
converter.py --read mrp --write lark 
--write_rule {RULE} --head_is_source 
--encode_processor embed input > output 
where --read specifies an input codec and 
--encode_processor specifies the PGN decora-
```

1Detailed model description is presented at CoNLL 2020 shared task (Ozaki et al. 2020). This paper mainly presents the user interface, training system and visualization of i-Parser.
Interactive Configuration i-Parser provides an interactive configuration tool (Figure 2) that enables beginners to implement their own parsers easily. After the users answer 10–20 simple questions, i-Parser constructs a new parser by generating configurations of the i-Parser Converter and training scripts of the i-Parser Model. Example questions include Use default postprocessor?

Visualization i-Parser provides a visualization server (Figure 3) that offers developers a way to debug their models. When given input sentence(s), the system parses and draws a graph in a web browser interface.

Hyperparameter Tuner i-Parser provides a hyperparameter tuning system powered by Optuna (Akiba et al. 2019) that allows developers to build a strong parser without manual configurations. This tuning system parses a JSON file (Figure 4) in which tunable parameters are registered as environmental variables, applying Bayesian optimization.

Benchmark We developed nine parsers using our i-Parser prototype and achieved the top performance in the CoNLL 2020 shared task (Oepen et al. 2020). We obtained F-scores of 81.54 for AMR, 93.56 for EDS, 88.73 for PTG (Hajič et al. 2012), 80.44 for Chinese AMR (Li et al. 2016), and 93.36 for German DRG (Bos et al. 2017), which are ranked the first amongst all the teams. Furthermore, we applied i-Parser to math word problems (Zhang et al. 2020b,a) where the parser had to generate an arithmetic expression tree. i-Parser achieved 77.5% solution accuracy with Math23K. These results show the effectiveness of our system.

Online Demonstration Scenario In the upcoming AAAI conference, we will show how i-Parser works. We will present its interactive configuration and how it monitors training. The visualization system will also be a key part of our demonstration. In the visualization, we will showcase many state-of-the-art parsing results.

Conclusion We presented a novel parser development kit, i-Parser. The interactive system allows developing new parsers with an easy configuration. Our system can contribute to extending parsing applications in the natural language processing field.

Acknowledgments We appreciate Prof. Dr. Naoaki Okazaki at Tokyo Institute of Technology for his helpful comments. Computational resource of AI Bridging Cloud Infrastructure (ABCI) provided by the National Institute of Advanced Industrial Science and Technology (AIST) was used.
References


