HIT-SCIR at MRP 2020: Transition-based Parser and Iterative Inference Parser

Longxu Dou, Yunlong Feng, Yuqiu Ji, Wanxiang Che, Ting Liu

Research Center for Social Computing and Information Retrieval

Harbin Institute of Technology

Overview of Our Techniques

- **Rank 3rd according to ALL-F1**
- **Submission models:**
  - **Transition-based Parser for Flavor (1) (UCCA、EDS、PTG)**
  - **Iterative Inference Parser for Flavor (2) (AMR、DRG)**

System	UCCA	EDS	PTG	AMR	DRG	ALL
Hitachi	0.75	0.94	0.89	0.82	0.93	0.86
UFAL	0.76	0.93	0.88	0.80	0.94	0.86
HIT-SCIR	0.75	0.87	0.84	0.70	0.89	0.81
HUJI-KU	0.73	0.80	0.54	0.52	0.63	0.64
ISCAS	0.06	0.86	0.18	0.61	0.69	0.48
TJU-BLCU	0.10	0.49	0.21	0.30	0.40	0.30

- **Cross-Framework**

System	UCCA	PTG	AMR	DRG	ALL
UFAL	0.81	0.91	0.78	0.90	0.85
Hitachi	0.79	0.87	0.80	0.93	0.85
HIT-SCIR	0.80	0.78	0.49	0.68	0.69
HUJI-KU	0.75	0.58	0.45	0.62	0.60

- **Cross-Lingual**

Transition-based Parser

- **Treat parsing as a sequence of actions, such as *SHIFT, REDUCE, EDGE*.**
  - **Flexible in predicting the anchor information, using the *NODE* ops.**

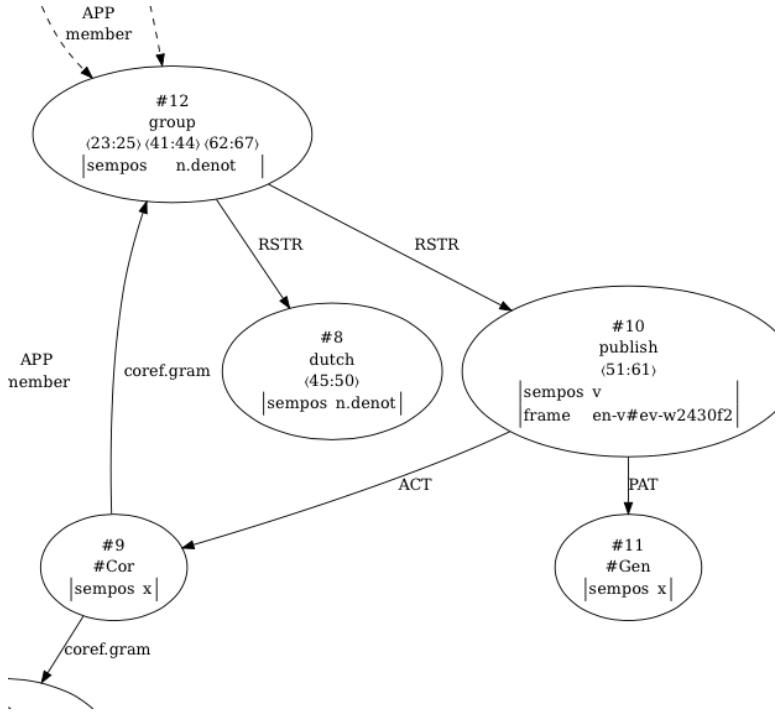
- **We use this system to parse the UCCA\EDS\PTG.**
  - **Adopt HIT-SCIR-2019 for UCCA/EDS.**
  - **We present a new transition-based parser for PTG.**

PTG

- We propose a new transition-based arc-eager parser for PTG.

- PTG is not a DAG (directed acyclic graph).
- Resolve cycle caused by coref.gram edge
  - Reverse edge

- Two classes of Node
  - #node: the label comes from a fixed vocab
  - node: aligns to the tokens of sentence



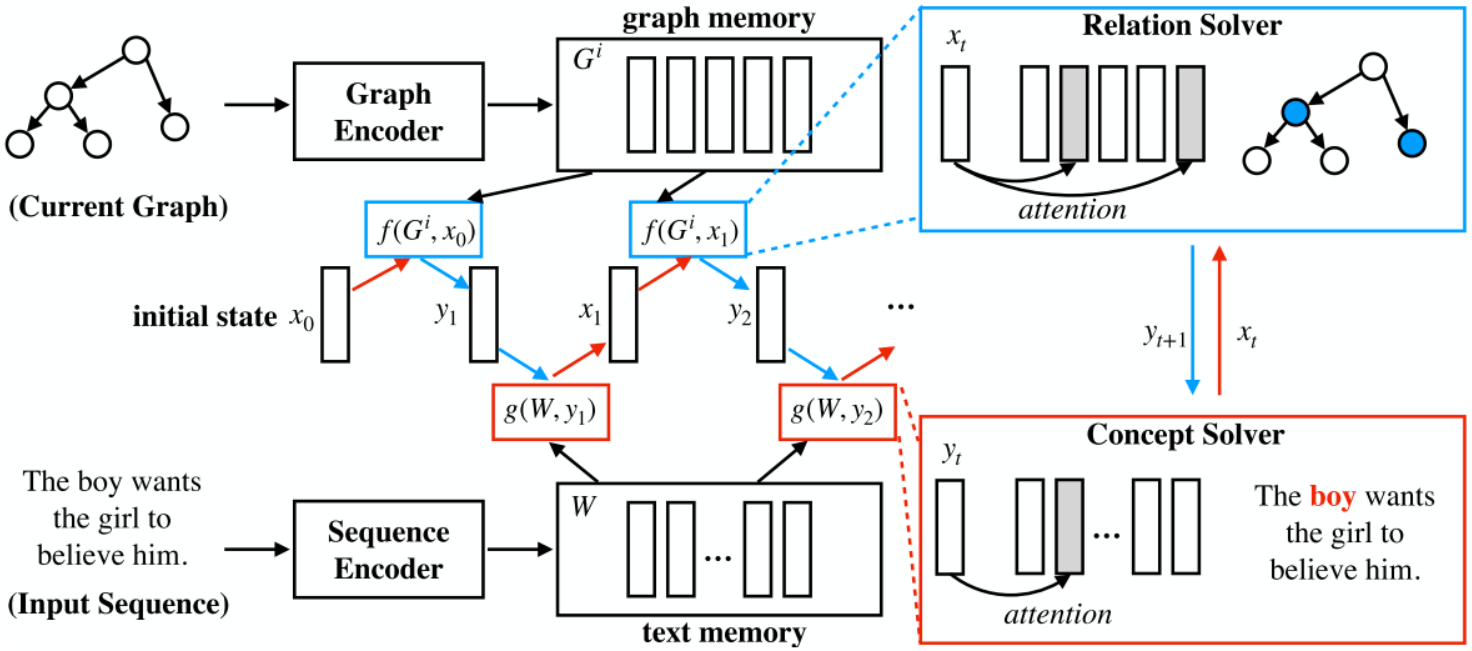
PTG Transition System

- #Action/ #Token = 4.656 (Train)

Before Transition					Transition	After Transition					Condition
Stack	List	Buffer	Nodes	Edges		Stack	List	Buffer	Nodes	Edges	
S	L	$x \mid B$	V	E	SHIFT	$S \mid L \mid x$	$\emptyset$	B	V	E	concept(x)
$S \mid x$	L	B	V	E	REDUCE	S	L	B	V	E	
$S \mid x$	L	$y \mid B$	V	E	RIGHT-EDGE$_X$	$S \mid x$	L	$y \mid B$	V	$E \cup \{(x,y)_X\}$	
$S \mid y$	L	$x \mid B$	V	E	LEFT-EDGE$_X$	$S \mid y$	L	$x \mid B$	V	$E \cup \{(x,y)_X\}$	
$S \mid y$	L	$x \mid B$	V	E	SELF-EDGE$_X$	$S \mid y$	L	$x \mid B$	V	$E \cup \{(x,x)_X\}$	
$S \mid x$	L	B	V	E	PASS	S	$x \mid L$	B	V	E	
S	L	$x \mid B$	V	E	DROP	$S \mid L$	$\emptyset$	B	V	E	token(x)
S	L	$x \mid B$	V	E	NODE$_X$	S	L	$y \mid x \mid B$	$V \cup \{y_{label=X}\}$	E	token(x)
S	L	$x \mid B$	V	E	NODE-ROOT$_X$	S	L	$y \mid x \mid B$	$V \cup \{y_{label=X}\}$	E	root(x)
S	L	$x \mid B$	V	E	TERMINAL-NOLABEL	S	L	$y \mid x \mid B$	$V \cup \{y\}$	E	
S	L	$x \mid B$	V	E	TERMINAL$_X$	S	L	$y \mid x \mid B$	$V \cup \{y_{label=X}\}$	E	
[root]	$\emptyset$	$\emptyset$	V	E	FINISH	$\emptyset$	$\emptyset$	$\emptyset$	V	E	

Table 3: The transition set of PTG parser. We write the **Stack** with its top to the right, the **Buffer** with its head to the left and the **List** with its head to the left. The elements in **Stack** and **List** are all concept nodes. Indicator function token(x) means x is a token of the sentence, while concept(x) means it's a concept node. root(x) indicates x is the top node.

Iterative Inference Parser



- **Treat parsing as a series of dual decisions on the input sequence and the incrementally constructed graph**

Cai, ACL2020, AMR Parsing via Graph-Sequence Iterative Inference

Rule-based Tagger

- We propose a rule-based tagger for EDS/PTG property prediction.

Training

- Count the co-occurrence of node label, upos, dep and property.

Infering

- Select the property based on the co-occurrence statistics.
  - If the triple (node label, upos, dep) is not found, we backoff to the tuple (upos, dep).

	Oracle Node Label				Rule Node Label			
	Accuracy	P	R	F1	Accuracy	P	R	F1
Train	0.9059	0.9868	0.8388	0.9068	0.8912	0.9843	0.9133	0.9474
Dev	0.9107	0.9872	0.8449	0.9105	0.8893	0.9838	0.9128	0.9470

PTG Property Score on Train/Dev

Conclusion

- **Submission systems:**
  - **Transition-based Parser for Flavor (1) including UCCA, EDS, and PTG.**
  - **Iterative Inference Parser for Flavor (2) including AMR and DR.**

- **Our Contribution:**
  - **Rule-based tagger for EDS/PTG property prediction.**
  - **Transition-based parser for PTG graph.**

- **Our code: https://github.com/DreamerDeo/HIT-SCIR-CoNLL2020**