HLT@SUDA at SemEval-2019 Task 1: UCCA
Graph Parsing as Constituent Tree Parsing

Wei Jiang, Zhenghua Li, Yu Zhang, Min Zhang
School of Computer Science and Technology,
Soochow University, China
2019.6.6
After graduation, John gave everything up.

- **P**: process
- **A**: participant
- **H**: linked scene
- **C**: center
- **R**: relator
- **N**: connector
- **L**: scene linker
- **U**: punctuation
- **F**: function unit
Our Approaches

- The main approach, inspired by the pseudo non-projective dependency parsing approach of Nivre and Nilsson (2005)
  - Convert UCCA to constituent tree
  - Utilize minimal span-based parser
  - Remote recovery as a new task (multi-task learning)
- Use of BERT
- Cross-lingual Parsing (French)
Pseudo non-projective dependency parsing (Nivre and Nilsson, 2005)

- Converting non-projective trees into projective trees
- Then recover.

("Only one of them concerns quality.")
Pseudo non-projective dependency parsing (Nivre and Nilsson, 2005)

- Converting non-projective trees into projective
  - Complex labels
- Projective dependency parsing
- Then recover via labels

```
AuxZ
  `--- AuxP
        `--- Sb
                `--- AuxZ
                        `--- AuxP
                                             `--- R
                                                  `--- R
                                                       `--- N4
                                                            `--- Z:

R
Z
(Out-of
 nich
them

VB
je
is

T
jen
only

C
jedna
one-FEM-SG

R
na
to

N4
kvalitu
quality

Z:
.

("Only one of them concerns quality.")
```
Convert UCCA to constituent tree

- Step 1: Remove remote edges
- Step 2: Handle discontinuous nodes
- Step 3: Push labels into nodes

After graduation, John gave everything up.
Convert UCCA to constituent tree

- Step1: Remove remote edges

Delete remote edges and add “-remote” on the label of the primary edge to mark remote nodes.
Convert UCCA to constituent tree

- Step 2: Handle discontinuous nodes

“ancestor1” means the true parent node 2 is the first ancestor of the pseudo parent node 4

After graduation, John gave everything up
Convert UCCA to constituent tree

- Step 2: Handle discontinuous nodes
  - Another situation

Node 3 is not an ancestor

```
Node 3 is not an ancestor
```
Convert UCCA to constituent tree

- Step 2: Handle discontinuous nodes
  - Considering the skewed distribution, we only keep “ancestor1”

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Dev</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>ancestor1</td>
<td>1460</td>
<td>149</td>
<td>1609</td>
<td>91.3</td>
</tr>
<tr>
<td>ancestor2</td>
<td>96</td>
<td>19</td>
<td>115</td>
<td>6.5</td>
</tr>
<tr>
<td>ancestor3</td>
<td>21</td>
<td>0</td>
<td>21</td>
<td>1.2</td>
</tr>
<tr>
<td>Other</td>
<td>16</td>
<td>2</td>
<td>18</td>
<td>1.0</td>
</tr>
</tbody>
</table>
After graduation, John gave everything up.

Convert UCCA to constituent tree

- Step3: Push labels into nodes
After graduation, John gave everything up.
Minimal span-based parser

- Input:
  \[ x_i = e_{w_i} \oplus e_{t_i} \oplus \cdots \]
  including the embeddings of POS, named entity tags and dependency labels

- Encoder: two cascaded bidirectional LSTM layers

- Span representation:
  \[ r_{i,j} = (f_j - f_i) \oplus (b_i - b_j) \]

- Top-down decoding
Minimal span-based parser

label scores
split scores

MLPs

span representation

Top-down Parsing

BiLSTMs

\[ \cdots \]

\[ x_{i-1} \]
\[ x_i \]
\[ x_{i+1} \]

\[ \cdots \]
After graduation, John gave everything up.
Recovery from constituent tree

- Reverse the three steps
  - Step1: Move labels to edges.
  - Step2: Move edges according to “-ancestor1”.
  - Step3: Recover remote edges according to “-remote”.
Recovery from constituent tree

- Step 1: Move labels to edges

After graduation, John gave everything up
Recovery from constituent tree

- Step 2: Move edges according to “-ancestor1”

After graduation, John gave everything up
Recovery from constituent tree

- Step 3: Recover remote edges according to “-remote”

How to find the correct remote edge and label?

After graduation, John gave everything up
Recovery of remote edges

- Consider all other non-terminal nodes as candidate remote parents.
- Given a remote node A (pointed by “-remote”) and a candidate remote parent B, following (Dozat and Manning, 2017)
  - First apply two MLPs to produce \( r_{i,j}^A \) and \( r_{i',j'}^B \)
  - Biaffine computation
    \[
    s(A \leftarrow B) = \left[ r_{i,j}^A \right]^T w r_{i',j'}^B
    \]
  - \(|s|\) is the number of the label set, including a “NOT-PARENT” label.
Recovery from constituent tree

- Recover remote edges according to “-remote”

After graduation, John gave everything up.
Recovery of remote edges

- How to train the recovery model?
- Multi-task learning framework
Use of BERT for open tracks

- Bert - contextualized word representations as extra input features (Devlin et al., 2018)
- **Weighted summation** of the last four transformer layers’ outputs and then multiply a task-specific weight parameter following ELMo (Peters et al., 2018)
- Multi-lingual Bert for En/De/Fr for simplicity.
Use of BERT (base)

Bert Encoder composed with 12 transformer layers

\[ h_{\text{BERT}} = \gamma_{\text{task}} \sum_{j=1}^{4} s_{-j}^{\text{task}} h_{-j}^{\text{BERT}} \]

\[ x_i = x_i \oplus h_i^{\text{BERT}} \]
Cross-lingual Parsing

- Language embedding
  - Training sentences: 4K En + 5K De + 15 Fr
  - Multi-lingual Bert
    \[ x_i = x_i \oplus e_{\text{lang=en/fr/de}} \]

\[
\begin{align*}
\text{Remote Recovery} & \quad \text{Constituent Parsing} \\
\text{MLPs and Biaffines} & \quad \text{MLPs} \\
\text{Shared BiLSTMs} & \quad \text{...} \\
& \quad \text{...} \\
& \quad \text{...} \\
& \quad \text{...}
\end{align*}
\]
## Word embedding settings

<table>
<thead>
<tr>
<th>Methods</th>
<th>F1 score</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Primary</td>
<td>Remote</td>
<td>Avg</td>
</tr>
<tr>
<td>Single-language models on English</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random emb</td>
<td>0.778</td>
<td>0.542</td>
<td>0.774</td>
<td></td>
</tr>
<tr>
<td>pretrained emb (no finetune)</td>
<td>0.790</td>
<td>0.494</td>
<td>0.785</td>
<td></td>
</tr>
<tr>
<td>pretrained emb</td>
<td>0.794</td>
<td>0.535</td>
<td><strong>0.789</strong></td>
<td></td>
</tr>
<tr>
<td>Single-language models on German</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random emb</td>
<td>0.817</td>
<td>0.549</td>
<td>0.811</td>
<td></td>
</tr>
<tr>
<td>pretrained emb (no finetune)</td>
<td>0.829</td>
<td>0.544</td>
<td>0.823</td>
<td></td>
</tr>
<tr>
<td>pretrained emb</td>
<td>0.831</td>
<td>0.536</td>
<td><strong>0.825</strong></td>
<td></td>
</tr>
<tr>
<td>Multilingual models on French</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random emb</td>
<td>0.688</td>
<td>0.343</td>
<td>0.681</td>
<td></td>
</tr>
<tr>
<td>pretrained emb</td>
<td>0.673</td>
<td>0.174</td>
<td>0.665</td>
<td></td>
</tr>
</tbody>
</table>

1. Pre-trained vs. Random?
2. Fine-tuning?
Bert is helpful, 10% for French

<table>
<thead>
<tr>
<th>Methods</th>
<th>F1 score</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Primary</td>
<td>Remote</td>
<td>Avg</td>
</tr>
<tr>
<td><strong>Single-language models on English</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pretrained emb</td>
<td>0.794</td>
<td>0.535</td>
<td></td>
<td>0.789</td>
</tr>
<tr>
<td>bert</td>
<td>0.821</td>
<td>0.593</td>
<td></td>
<td>0.817</td>
</tr>
<tr>
<td>pretrained emb $\oplus$ bert</td>
<td>0.825</td>
<td>0.603</td>
<td></td>
<td>0.821</td>
</tr>
<tr>
<td><strong>Single-language models on German</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pretrained emb</td>
<td>0.831</td>
<td>0.536</td>
<td></td>
<td>0.825</td>
</tr>
<tr>
<td>bert</td>
<td>0.842</td>
<td>0.610</td>
<td></td>
<td>0.837</td>
</tr>
<tr>
<td>pretrained emb $\oplus$ bert</td>
<td>0.849</td>
<td>0.628</td>
<td></td>
<td>0.844</td>
</tr>
<tr>
<td><strong>Multilingual models on French</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pretrained emb</td>
<td>0.673</td>
<td>0.174</td>
<td></td>
<td>0.665</td>
</tr>
<tr>
<td>bert</td>
<td>0.796</td>
<td>0.524</td>
<td></td>
<td>0.789</td>
</tr>
</tbody>
</table>
Other trails

• Not helpful
  • Provided POS, NE, parsing features
  • SA encoder to replace BiLSTM (not like in constituent parsing)

• Things probably work (future trials)
  • Cross-lingual word embeddings
  • CharLSTM word representations (besides word embeddings)
  • Language-specific BERT (large?)
## Final results (open vs. closed)

<table>
<thead>
<tr>
<th>Tracks</th>
<th>Teams</th>
<th>F1 score</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Primary</td>
<td>Remote</td>
<td>Avg</td>
<td></td>
</tr>
<tr>
<td>English-Wiki_closed</td>
<td>HLT@SUDA</td>
<td>0.779</td>
<td>0.733</td>
<td>0.724</td>
<td><strong>0.774</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>baseline</td>
<td>0.522</td>
<td>0.472</td>
<td>0.728</td>
<td></td>
<td></td>
</tr>
<tr>
<td>English-Wiki_open</td>
<td>HLT@SUDA</td>
<td>0.810</td>
<td>0.802</td>
<td>0.800</td>
<td><strong>0.805</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNUY-PekingU</td>
<td>0.588</td>
<td>0.666</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>English-20K_closed</td>
<td>HLT@SUDA</td>
<td>0.736</td>
<td>0.682</td>
<td>0.672</td>
<td><strong>0.727</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>baseline</td>
<td>0.312</td>
<td>0.237</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>English-20K_open</td>
<td>HLT@SUDA</td>
<td>0.777</td>
<td>0.746</td>
<td>0.739</td>
<td><strong>0.767</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNUY-PekingU</td>
<td>0.392</td>
<td>0.457</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>German-20K_closed</td>
<td>HLT@SUDA</td>
<td>0.838</td>
<td>0.802</td>
<td>0.797</td>
<td><strong>0.832</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNUY-PekingU</td>
<td>0.592</td>
<td>0.593</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>German-20K_open</td>
<td>HLT@SUDA</td>
<td>0.854</td>
<td>0.845</td>
<td>0.841</td>
<td><strong>0.849</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNUY-PekingU</td>
<td>0.641</td>
<td>0.660</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>French-20K_open</td>
<td>CNUY-PekingU</td>
<td>0.800</td>
<td>0.779</td>
<td>0.796</td>
<td><strong>0.800</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HLT@SUDA</td>
<td>0.645</td>
<td>0.438</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Note: F1 score values are rounded to two decimal places.*
Conclusions

• We propose a simple graph-based UCCA parsing approach.
  • Converting graph into tree.
• BERT (base) is extensively helpful, but the improvement is not very large for English/German.
• Language embedding is helpful for resource-low languages (French) in the cross-lingual scenario.
Thank you for your attention!
Questions?

Codes: https://github.com/SUDA-LA/ucca-parser
<table>
<thead>
<tr>
<th>Tracks</th>
<th>Training</th>
<th></th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Closed</td>
<td>Open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>En-Wiki</td>
<td>4113</td>
<td>5132</td>
<td>514</td>
<td>515</td>
</tr>
<tr>
<td>En-20K</td>
<td>0</td>
<td>5132</td>
<td>0</td>
<td>492</td>
</tr>
<tr>
<td>Ge-20K</td>
<td>5211</td>
<td>6360</td>
<td>651</td>
<td>632</td>
</tr>
<tr>
<td>Fr-20K</td>
<td>15</td>
<td>547</td>
<td>238</td>
<td>239</td>
</tr>
</tbody>
</table>

Table 1: Sentence number in training, dev, and test sets.
Use all layers of Bert may help?

- Gamma: 1.0539
- $s_{-1, -2, -3, -4}$: 0.07 0.04 0.08 0.081
The use of Bert

Replacing word embedding with the BERT representation is useful on English (2.8% increase) and German (1.2% increase).

Concatenating pre-trained word embeddings with BERT outputs leads is also beneficial.

<table>
<thead>
<tr>
<th>Methods</th>
<th>F1 score</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Primary</td>
<td>Remote</td>
<td>Avg</td>
</tr>
<tr>
<td>Single-language models on English</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random emb</td>
<td>0.778</td>
<td>0.542</td>
<td>0.774</td>
</tr>
<tr>
<td>pretrained emb (no finetune)</td>
<td>0.790</td>
<td>0.494</td>
<td>0.785</td>
</tr>
<tr>
<td>pretrained emb</td>
<td>0.794</td>
<td>0.535</td>
<td>0.789</td>
</tr>
<tr>
<td>bert</td>
<td>0.821</td>
<td>0.593</td>
<td>0.817</td>
</tr>
<tr>
<td>pretrained emb⊕bert</td>
<td>0.825</td>
<td>0.603</td>
<td>0.821</td>
</tr>
<tr>
<td>official baseline (closed)</td>
<td>0.745</td>
<td>0.534</td>
<td>0.741</td>
</tr>
<tr>
<td>official baseline (open)</td>
<td>0.753</td>
<td>0.514</td>
<td>0.748</td>
</tr>
<tr>
<td>Single-language models on German</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random emb</td>
<td>0.817</td>
<td>0.549</td>
<td>0.811</td>
</tr>
<tr>
<td>pretrained emb (no finetune)</td>
<td>0.829</td>
<td>0.544</td>
<td>0.823</td>
</tr>
<tr>
<td>pretrained emb</td>
<td>0.831</td>
<td>0.536</td>
<td>0.825</td>
</tr>
<tr>
<td>bert</td>
<td>0.842</td>
<td>0.610</td>
<td>0.837</td>
</tr>
<tr>
<td>pretrained emb⊕bert</td>
<td>0.849</td>
<td>0.628</td>
<td>0.844</td>
</tr>
<tr>
<td>official baseline (closed)</td>
<td>0.737</td>
<td>0.46</td>
<td>0.731</td>
</tr>
<tr>
<td>official baseline (open)</td>
<td>0.797</td>
<td>0.587</td>
<td>0.792</td>
</tr>
<tr>
<td>Multilingual models on French</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random emb</td>
<td>0.688</td>
<td>0.343</td>
<td>0.681</td>
</tr>
<tr>
<td>pretrained emb</td>
<td>0.673</td>
<td>0.174</td>
<td>0.665</td>
</tr>
<tr>
<td>bert</td>
<td>0.796</td>
<td>0.524</td>
<td>0.789</td>
</tr>
<tr>
<td>official baseline (open)</td>
<td>0.523</td>
<td>0.016</td>
<td>0.514</td>
</tr>
</tbody>
</table>