Prague at EPE 2017: The UDPipe System

Milan Straka, Jana Straková, Jan Hajič

Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Charles University
{straka, strakova, hajic}@ufal.mff.cuni.cz

EPE 2017, 20th September 2017
EPE 2017

- you already know

UDPipe

- a trainable pipeline which performs sentence segmentation, tokenization, POS tagging, lemmatization and dependency parsing
- models for all 50 languages of UD 2.0
- easily trainable using data in CoNLL-U format
Introduction

EPE 2017

- you already know

UDPipe

- a trainable pipeline which performs sentence segmentation, tokenization, POS tagging, lemmatization and dependency parsing
- models for all 50 languages of UD 2.0
- easily trainable using data in CoNLL-U format
http://ufal.mff.cuni.cz/udpipe

- open-source Mozilla Public License (MPL)
- models under CC BY-SA-NC license
- bindings for C++, Python, Perl, Java, C#

```
pip install ufal.udpipe
cpan UFAL::UDPipe
```

- REST web service
  - http://lindat.mff.cuni.cz/service/udpipe/

I brought my car in on a Sunday to replace a shredded tire.

![UDPipe tree diagram]

---

**Introduction**

**UD**

**Tokenizer**

**Tagger**

**Parser**

**Results**

Milan Straka, Jana Straková, Jan Hajič

Prague at EPE 2017: The UDPipe System
He gave no indication on the value of the highest bid.
- fully trainable from CoNLL-U training data
  - CoNLL-U v1 allows reconstruction of spaces between tokens in one sentence
  - CoNLL-U v1 does not provide markup for paragraph and document boundaries, which are often indicated by visual layout and/or spacing

Keep in touch, / Mike / Michael J. McDermott

i have two options / using the metro or the air france bus / can anybody tell me if the metro runs directly ...

- CoNLL-U v2 does include markup for paragraph and document boundaries, but only document boundaries are marked in English UD 2.0 data
Tokenizer Architecture

- bidirectional character-level GRU network
- predicts break type after every character
  - no break
  - token break
  - sentence break
suffix based guesser predicting most frequent (UPOS, XPOS, FEATS) candidates from data

<table>
<thead>
<tr>
<th>SUFFIX</th>
<th>UPOS</th>
<th>XPOS</th>
<th>FEATS</th>
<th>LEMMA RULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>-ing</td>
<td>VERB</td>
<td>VBG</td>
<td>VerbForm=Ger</td>
<td>remove ing</td>
</tr>
<tr>
<td></td>
<td>NOUN</td>
<td>NN</td>
<td>Number=Sing</td>
<td>keep unchanged</td>
</tr>
<tr>
<td></td>
<td>VERB</td>
<td>VBG</td>
<td>VerbForm=Ger</td>
<td>remove ing, append e</td>
</tr>
<tr>
<td></td>
<td>ADJ</td>
<td>JJ</td>
<td>Degree=Pos</td>
<td>keep unchanged</td>
</tr>
<tr>
<td></td>
<td>PROP</td>
<td>NNP</td>
<td>Number=Sing</td>
<td>keep unchanged</td>
</tr>
<tr>
<td></td>
<td>VERB</td>
<td>VBG</td>
<td>VerbForm=Ger</td>
<td>remove ting</td>
</tr>
<tr>
<td></td>
<td>VERB</td>
<td>VBG</td>
<td>VerbForm=Ger</td>
<td>remove ping</td>
</tr>
</tbody>
</table>

to generate candidates for a given form
- all candidates for the same form in the training data
- several most-frequent candidates using the suffix guesser

disambiguated by averaged perceptron utilizing a predefined rich set of feature templates and Viterbi decoding of order 3
very similar to tagger, but utilize \((UPOS, \text{lemma rule})\) candidates

\textit{lemma rule} is the shortest formula generating lemma from a given form, using any combination of

- remove a specific prefix
- remove a specific suffix
- append a prefix
- append a suffix
Transition-Based Dependency Parsing

- **initial configuration**

- **transitions modify the configuration**

- **final configuration**
- fast neural dependency parser inspired by Chen and Manning (2014)
- several transition systems
  - projective (arc standard)
  - partially non-projective (arc2)
  - fully non-projective (swap)
- multiple oracles
  - static oracles for all systems
  - dynamic oracle for arc standard system
  - search-based oracle for all systems
<table>
<thead>
<tr>
<th>Run name</th>
<th>Run</th>
<th>Description</th>
<th>Tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>UD2.0 En/UDPipe/20</td>
<td>0</td>
<td>UDPipe 1.2, UD 2.0 English data, UDPipe tokenizer, beam size 20</td>
<td>204.5k</td>
</tr>
<tr>
<td>UD2.0 En/EPE/20</td>
<td>1</td>
<td>UDPipe 1.2, UD 2.0 English data, EPE provided tokenizer, beam size 20</td>
<td>204.5k</td>
</tr>
<tr>
<td>UD2.0 EnMerged/UDPipe/20</td>
<td>2</td>
<td>UDPipe 1.2, UD 2.0 English + English LinES + English ParTUT data, UDPipe tokenizer, beam size 20</td>
<td>292.2k</td>
</tr>
<tr>
<td>UD2.0 EnMinus/UDPipe/5</td>
<td>3</td>
<td>UDPipe 1.1, first 95% of UD 2.0 English, UDPipe tokenizer, beam size 5</td>
<td>192.5k</td>
</tr>
<tr>
<td>UD1.2 En/UDPipe/5</td>
<td>4</td>
<td>UDPipe 1.0, UD 1.2 English data, UDPipe tokenizer, beam size 5</td>
<td>204.5k</td>
</tr>
<tr>
<td>Stanford-Paris</td>
<td>6</td>
<td>UD v1 enhanced dependencies, WSJ+Brown+GENIA data</td>
<td>1692.0k</td>
</tr>
</tbody>
</table>
## Extrinsic Results

<table>
<thead>
<tr>
<th>UDPipe run</th>
<th>Event extraction</th>
<th>Negation resolution</th>
<th>Opinion analysis</th>
<th>Overall score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-UD2.0 En/UDPipe/20</td>
<td>43.58</td>
<td>58.83</td>
<td>59.79</td>
<td>54.07</td>
</tr>
<tr>
<td>1-UD2.0 En/EPE/20</td>
<td>45.54</td>
<td>61.62</td>
<td>61.00</td>
<td><strong>56.05</strong></td>
</tr>
<tr>
<td>2-UD2.0 EnMerged/UDPipe/20</td>
<td>44.25</td>
<td>59.95</td>
<td>58.71</td>
<td>54.30</td>
</tr>
<tr>
<td>3-UD2.0 EnMinus/UDPipe/5</td>
<td>42.70</td>
<td>59.95</td>
<td>58.90</td>
<td>53.85</td>
</tr>
<tr>
<td>4-UD1.2 En/UDPipe/5</td>
<td>43.22</td>
<td><strong>50.85</strong></td>
<td>58.53</td>
<td>50.86</td>
</tr>
<tr>
<td><strong>Stanford-Paris, run 6</strong></td>
<td><strong>50.23</strong></td>
<td>66.16</td>
<td>65.14</td>
<td><strong>60.51</strong></td>
</tr>
</tbody>
</table>

Higher numbers are better.

- overall score of 56.05, lacking behind nearly all other participant systems
- overall scores better systems 56.23, 56.24, 56.65, 56.81, 58.57, 60.51
- best results for EPE-provided tokenizer
- UD 2.0 offers only 200k of English training data
- we emphasize that even though the EPE 2017 shared task focused on English language only, UDPipe is trained in a language agnostic manner for 50 languages without any adaptation for English other than setting up the hyperparameters of the artificial neural networks
## Intrinsic Results

<table>
<thead>
<tr>
<th>Row</th>
<th>Data</th>
<th>Plain text processing</th>
<th>Using gold tokenization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Words</td>
<td>Sents</td>
</tr>
<tr>
<td>0-UD2.0 En/UDPipe/20</td>
<td>UD 2.0 En</td>
<td>99.0</td>
<td>75.3</td>
</tr>
<tr>
<td></td>
<td>UD 2.0 EnMerged</td>
<td>98.9</td>
<td>79.5</td>
</tr>
<tr>
<td></td>
<td>UD 1.2 En</td>
<td>99.0</td>
<td>75.3</td>
</tr>
<tr>
<td>1-UD2.0 En/EPE/20</td>
<td>UD 2.0 En</td>
<td>96.2</td>
<td>59.9</td>
</tr>
<tr>
<td></td>
<td>UD 2.0 EnMerged</td>
<td>97.8</td>
<td>71.0</td>
</tr>
<tr>
<td></td>
<td>UD 1.2 En</td>
<td>96.2</td>
<td>59.9</td>
</tr>
<tr>
<td>2-UD2.0 EnMerged/UDPipe/20</td>
<td>UD 2.0 En</td>
<td>99.0</td>
<td>75.3</td>
</tr>
<tr>
<td></td>
<td>UD 2.0 EnMerged</td>
<td>98.9</td>
<td>79.5</td>
</tr>
<tr>
<td></td>
<td>UD 1.2 En</td>
<td>99.0</td>
<td>75.3</td>
</tr>
<tr>
<td>3-UD2.0 EnMinus/UDPipe/5</td>
<td>UD 2.0 En</td>
<td>98.7</td>
<td>73.2</td>
</tr>
<tr>
<td></td>
<td>UD 2.0 EnMerged</td>
<td>98.8</td>
<td>78.6</td>
</tr>
<tr>
<td></td>
<td>UD 1.2 En</td>
<td>98.7</td>
<td>73.2</td>
</tr>
<tr>
<td>4-UD1.2 En/UDPipe/5</td>
<td>UD 2.0 En</td>
<td>98.4</td>
<td>72.3</td>
</tr>
<tr>
<td></td>
<td>UD 2.0 EnMerged</td>
<td>98.7</td>
<td>77.8</td>
</tr>
<tr>
<td></td>
<td>UD 1.2 En</td>
<td>98.4</td>
<td>72.3</td>
</tr>
</tbody>
</table>

Higher numbers are better.
Observations

Tokenization Issues
- UDPipe tokenizer of lower quality – EPE-provided tokenizer improves score by 2 points in EPE 2017
- opposite results in intrinsic evaluation on UD 2.0 test set

Merged English Treebanks
- merged UD 2.0 English treebanks show improvements in extrinsic evaluation, even if they have inconsistent XPOS tags and show performance drop in intrinsic evaluation

Negation Resolution Drop of Run 4
- poor Unicode character handling by UDPipe 1.0 tokenizer as no Unicode in UD 1.2 English training data
Conclusions

- evaluation of language-agnostic UDPipe pipeline in the EPE 2017 shared task

Immediate Future Work

- when the paragraph boundaries are annotated in the UD data, does the trained sentence segmenter achieve better performance?
- can a rule-based English tokenizer also improve the results?
- what effect would larger training data (like WSJ) have?
- what performance would a state-of-the-art dependency parser attain using the UD 2.0 data only?
Questions?