UDPipe

Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0

Milan Straka and Jana Straková, Charles University
{straka,strakova}@ufal.mff.cuni.cz

UDPipe 1.1 and 1.2

Updated versions capable of handling UD 2.0.

Tokenization

- UD utilizes three level tokenization: sentences, tokens and words
- sentences and tokens breaks are predicted jointly using trainable bidirectional GRU recurrent network
- if the token breaks are not marked in the data, they can be generated automatically utilizing arbitrary plain text in given language
- tokens are split into words by rules obtained from the training data
  - full token rules
  - token suffix rules

  zum → zu + dem
  ∗łem → ∗ł + em
- fast inference (direct C++ implementation, caching in the network)
- **UDPipe 1.1 changes**
  - spaces in tokens are allowed (but only if present in data)
  - paragraph and document boundaries are both produced and used during training (not to predict sentence breaks at their end)
  - allow perfect reconstruction of spaces

    `SpacesBefore=\n\n|SpacesAfter=\s\t\s`
  - token ranges in the original text `TokenRange=42:45`

Morphological Analysis

- two guessers generating from a suffix (and sometimes also prefix)
  - UPOSTAG & XPOSTAG & FEATS triples
  - UPOSTAG & LEMMA pairs
- lemmas generated either absolutely or relatively to the word form

SUFFIX	UPOS	XPOS	FEATS	LEMMA RULE
	VERB	VBG	VerbForm=Ger	remove ing
	NOUN	NN	Number=Sing	keep unchanged
	VERB	VBG	VerbForm=Ger	remove ing, append e
-ing	ADJ	JJ	Degree=Pos	keep unchanged
	PROPN	NNP	Number=Sing	keep unchanged
	VERB	VBG	VerbForm=Ger	remove ting
	VERB	VBG	VerbForm=Ger	remove ping

POS Tagging and Lemmatization

- old-school averaged perceptron with Viterbi order 3 decoding

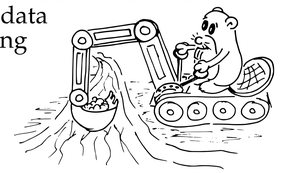
Parsing

- transitional-based parser based on simple NN classifier
- transition systems for non-projective and projective trees
- novel **search-based oracle** usable with any transition system
- dynamic oracle for the projective system
- very fast, but unfortunately not SoTA performance

CoNLL 2017 UD Shared Task

UDPipe 1.1 Baseline System

- tokenizer, POS tagger, lemmatizer and parser provided to the participants of the CoNLL 2017 UD Shared Task
- trained without using development data
- preprocessed test sets available during TIRA evaluation
- for surprise languages, jackknifed POS tags on test sets also provided
- **13th** out of 33 contestant systems



UDPipe 1.2 Participant System

- full training data used, with hyperparameter search on dev data
- slightly larger hyperparameters
  - tokenizer GRU dimension (24→64), form embeddings (50→64)
- merging treebanks of the same language
  - enriching training data of small treebanks utilizing other treebanks of the same language
  - append only ¼, ½, 1 or 2 times the size of the original treebank due to often inconsistent annotation in different treebanks
- joint segmentation and parsing
  - difficult segmentation in treebanks with missing punctuation
  - choose segmentation maximizing dependency trees logprob
- **8th** out of 33 contestant systems

Search-based Oracle Evaluation

- the overall effect of search-based oracle on various transition systems across all UD 2.0 treebanks

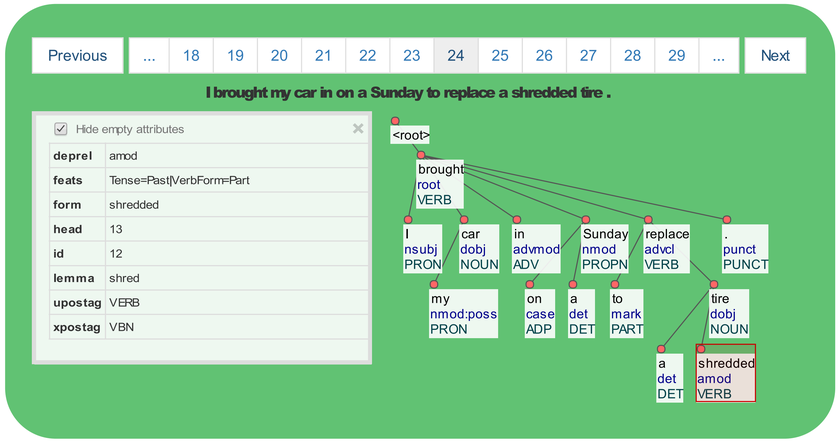
Transition system and oracle	No search-based oracle		Search-based oracle	
	UAS	LAS	UAS	LAS
Arc standard system with static oracle	74.29	68.27	74.80	68.87
Arc standard system with dynamic oracle	75.31	69.36	75.40	69.51
Swap system with static lazy oracle	74.73	68.76	75.16	69.27
Link2 system with static oracle	74.79	68.76	75.21	69.29
Any system, static oracle	74.72	68.71	75.21	69.31
Any system, any oracle	75.27	69.31	75.38	69.52

Future Work

- more accurate neural network models
  - deeper, RNNs, biaffine, both pretrained&learned embeddings, ...
- add multithreaded and GPU support for both training and inference

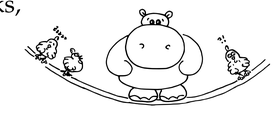
UDPipe Implementation

- easily trainable from CoNLL-U file `ufal.mff.cuni.cz/udpipe`
  - hyperparameter search support
- CC BY-SA-NC models for UD 2.0 treebanks (50 languages)
- efficient standalone C++ implementation under MPL
- precompiled Linux/Windows/OS X binaries
- Python package on PyPI `pip install ufal.udpipe`
- Perl package on CPAN, precompiled Java and C# binding
- REST service (both running and a provided server), web application `http://lindat.mff.cuni.cz/services/udpipe/`



Model Size and Runtime Performance

- displayed as a median for all UD 2.0 treebanks, together with the 5th and 95th percentile
- complete model consists of a tokenizer with dimension 64, a tagger, a lemmatizer and a parser with beam size 5



Model configuration	Model size [MB]	Model speed [kwords/s]
Tokenizer dim 24	0.04 (0.03–0.15)	27.7 (20–37)
Tokenizer dim 64	0.20 (0.19–0.31)	6.0 (4.9–8.6)
Tagger&lemmatizer	9.4 (2.3–24.8)	6.5 (2.1–14)
Parser beam size 1	3.2 (1.9–6.9)	14.9 (12–19)
Parser beam size 5		2.7 (2.2–3.6)
Complete model	13.2 (4.4–31.9)	1.7 (1.2–2.3)

Demo and Download: `http://ufal.mff.cuni.cz/udpipe`

Acknowledgements: This work has been partially supported and has been using language resources and tools developed, stored and distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071), also partially supported by OP VVV projects CZ.02.1.01/0.0/0.0/16_013/0001781 and CZ.02.2.69/0.0/0.0/16_018/0002373, and by SVV project number 260 453.