Parsing Experiments 2023

Overview of available parsers

UDPipe

Charles University, Milan Straka and Jana Straková. There are two versions available. UDPipe 1.2 (2017) is a standalone tool that can be retrained on custom data. UDPipe 2 (2018) is not so easy to retrain, but it is not impossible, as the sources are available from Github. We typically rely on Milan to train the models that we then use through the web interface (or through REST API). But the performance of UDPipe 2 is significantly better than that of UDPipe 1.2. UDPipe 3 is being developed and it will be a trainable standalone tool again, but it is not available yet. Models for UDPipe 1.2 are small and fast (parsing 1000 words/sec on a single CPU). Models for UDPipe 2 are larger and slower (parsing 60 words/sec using 1 CPU thread, 300 words with 8 threads, and 2000 words on a GPU).

UDPipe 1.2 is a transition-based parser (like Goldberg, or previously Malt parser and others). In contrast, UDPipe 2 is a graph-based biaffine attention parser proposed by Dozat et al. (2017).

Web: https://ufal.mff.cuni.cz/udpipe

On-line demo: https://lindat.mff.cuni.cz/services/udpipe/

Pretrained models: UD 2.5 for UDPipe 1.2, UD 2.6 and 2.10 for UDPipe 2. Only treebanks that have official training data in UD, with at least 1000 training words. The models on UD 2.10 (https://github.com/ufal/udpipe/blob/udpipe-2/docs/models_ud210.md) have tokenization trained on W2C "for treebanks that do not contain original plain text version" – what does it mean? Models are trained on UD 2.10 data, multilingual BERT, and RobeCzech. The models on UD 2.6 (https://github.com/ufal/udpipe/blob/udpipe-2/docs/models_ud26.md) differ in that they do not use RobeCzech. (Note that in the 2017 paper by Milan Straka, not even multilingual BERT is mentioned; only pretrained static embeddings, trained embeddings, and character embeddings. The ability to use contextual vector representation must have been added to UDPipe later.)

What it does: sentence segmentation, tokenization + word segmentation, lemmatization, UPOS tagging, XPOS tagging, morphological features, basic UD parsing. Prediction of morphology is done before parsing (at least that is the case with UDPipe 1.2) and parsing uses morphology as its input. Therefore, the parsing accuracy goes up if the input has gold-standard tokenization and morphology. UDPipe 2 does not do segmentation and tokenization, so these parts would have to be taken from UDPipe 1.2 or another tool.

Bibtex reference:

```
@InProceedings{straka2017udpipe,
 author    = {Straka, Milan and Strakov\'{a}, Jana},
 title     = {Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with
UDPipe},
 booktitle = {Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing
from Raw Text to Universal Dependencies},
 month     = {August},
 year      = {2017},
 address   = {Vancouver, Canada},
 publisher = {Association for Computational Linguistics},
 pages     = {88–99},
 url       = {https://aclanthology.org/K17-3009/},
}

@InProceedings{straka2018udpipe,
 title     = {{UDP}ipe 2.0 Prototype at {C}o{NLL} 2018 {UD} Shared Task},
 author    = {Straka, Milan},
 booktitle = {Proceedings of the {C}o{NLL} 2018 Shared Task: Multilingual
Parsing from Raw Text to Universal Dependencies},
 month     = oct,
 year      = {2018},
 address   = {Brussels, Belgium},
 publisher = {Association for Computational Linguistics},
 url       = {https://aclanthology.org/K18-2020/},
 doi       = {10.18653/v1/K18-2020},
 pages     = {197--207},
}
```

Running at ÚFAL: `/home/zeman/nastroje/udpipe/udpipe-1.2.0-bin/bin-linux64/udpipe` (it is in Dan’s $PATH)

Models: http://hdl.handle.net/11234/1-3131

```
curl --remote-name-all
https://lindat.mff.cuni.cz/repository/xmlui/bitstream/handle/11234/1-3131{/udpipe-ud-2.5-191206-reproducible_training.zip,/czech-pdt-ud-2.5-191206.udpipe,/czech-fictree-ud-2.5-191206.udpipe,/czech-cac-ud-2.5-191206.udpipe,/czech-cltt-ud-2.5-191206.udpipe}
```

```
udpipe --tokenize --tag --parse czech-cltt-ud-2.5-191206.udpipe
/net/data/universal-dependencies-2.5/UD_Czech-CLTT/cs_cltt-ud-test.txt >
cs_cltt-ud-2.5-test-parsed.conllu
```

```
eval.py -v /net/data/universal-dependencies-2.5/UD_Czech-CLTT/cs_cltt-ud-test.conllu cs_cltt-ud-2.5-test-parsed.conllu
```

Udify

Charles University, Dan Kondratyuk and Milan Straka. One model trained on all UD languages (paper reports on UD 2.3, excluding text-less treebanks, resulting in 124 treebanks of 75 languages) and the multilingual BERT (104 languages, mostly Wikipedia, large overlap with the languages in UD), applicable to all UD languages. Training needs a GPU, lots of memory, and reportedly takes over 20 days (edit: Milan now says that 20 days were needed with old types of GPU; at present it would be just a few days). Udify is supposed to be advantageous for languages with little or no training data. But it did show improvement also on some larger languages.

Multitask training of UPOS, features (as one tag), lemmas and labeled dependencies. For the latter, the graph-based biaffine attention parser by Dozat and Manning (2016); Dozat et al. (2017) is used.

Web (Github): https://github.com/Hyperparticle/udify

Pretrained model: UD 2.3. See http://hdl.handle.net/11234/1-3042 (759 MB).

What it does: lemmatization, UPOS tagging, morphological features, basic UD parsing. It does not do XPOS tagging because the tags are not standardized across treebanks. Prediction of morphology is done together with parsing. It is a multitask training, meaning that missing or bad training morphology may still impact parsing quality. Udify does not do sentence segmentation, tokenization and word segmentation, so if we want an end-to-end parser, we have to combine it e.g. with UDPipe 1.2.

Running at ÚFAL: It requires certain versions of certain libraries (see the file `requirements.txt`), which in turn require old Python (they are not available for newer versions of Python). We can use `/opt/python/3.6.3`. However, I am unable to install packages with that version of Python on the ÚFAL network: “pip is configured with locations that require TLS/SSL, however the ssl module in Python is not available.” Eventually I was able to run Udify after copying Federica’s virtual environment with Python 3.7.15 and all the required libraries.

Bibtex reference:

```
@inproceedings{kondratyuk2019udify,
 author    = {Kondratyuk, Dan and Straka, Milan},
 title     = {75 Languages, 1 Model: Parsing {Universal Dependencies}
Universally},
 booktitle = {Proceedings of the 2019 Conference on Empirical Methods in Natural
Language Processing},
 month     = nov,
 year      = {2019},
 address   = {Hong Kong, China},
 publisher = {Association for Computational Linguistics},
 pages     = {2779–2795},
 doi       = {10.18653/v1/D19-1279},
 url       = {https://aclanthology.org/D19-1279/},
}
```

Stanza

Stanford University, Peng Qi et al., now maintained by John Bauer. To just use the pretrained models, one can install it via pip for Python 3.6 or newer (pip install stanza), then download the pretrained models, then use it. To train one’s own models (see https://stanfordnlp.github.io/stanza/training_and_evaluation.html), the sources from two Stanza Github repositories must be cloned. If installed via pip, then it works as a Python library and we must write a simple Python script to actually apply it on a text file and obtain the CoNLL-U file. There is less explicit control over which models we are using. Stanza may even automatically download and use new models if they become available on their server (but this behavior can be turned off).

Graph-based biaffine neural dependency parser (Dozat and Manning, 2017).

Web: https://stanfordnlp.github.io/stanza/

On-line demo: http://stanza.run/

Pretrained models: UD 2.8 (including languages that have only a test set in UD). Original paper reports on UD 2.5 + wor2vec embeddings from the CoNLL 2017 shared task, or fastText embeddings for languages that don't have the former.

What it does: lots of things, morphosyntactic analysis from raw text, also named entity recognition and sentiment analysis for a significant number of languages. Additionally a client Python interface to Stanford CoreNLP (which is in Java), adding coreference resolution and relation extraction (but this is only for 6 languages).

Bibtex reference:

```
@inproceedings{qi2020stanza,
 author    = {Qi, Peng and Zhang, Yuhao and Zhang, Yuhui and Bolton, Jason and
Manning, Christopher D.},
 booktitle = {Proceedings of the 58th Annual Meeting of the Association for
Computational Linguistics: System Demonstrations},
 title     = {Stanza: A {Python} Natural Language Processing Toolkit for Many
Human Languages},
 month     = jul,
 year      = {2020},
 address   = {Online},
 publisher = {Association for Computational Linguistics},
 pages     = {101--108},
 doi       = {10.18653/v1/2020.acl-demos.14},
 url       = {https://aclanthology.org/2020.acl-demos.14/},
}
```

Trankit

University of Oregon, Minh Van Nguyen, Viet Dac Lai, Amir Pouran Ben Veyseh, and Thien Huu Nguyen. Transformer-based toolkit for NLP. Uses XLM-Roberta and claims to be better than Stanza thanks to that. Trankit authors say it is memory-efficient thanks to a novel technique they use. Nevertheless, it is slow. A GPU should be used not only for training but also for parsing. Otherwise, if the PDT test set is parsed on a CPU on the cluster, it takes 1:40 h (compare to 6 minutes needed by Stanza and 3 minutes needed by UDPipe 2).

On-line demo: http://nlp.uoregon.edu/trankit

Documentation: https://trankit.readthedocs.io/en/latest/

Pretrained models: UD 2.5. The authors publish (https://trankit.readthedocs.io/en/latest/performance.html#universal-dependencies-v2-5) scores for "Trankit[large]" (using XLM-Roberta large) and "Trankit[base]" (using XLM-Roberta base). Apparently, since version 1.0.0, Trankit uses the large models and does not offer the possibility to use the base ones.

Bibtex reference:

```
@inproceedings{nguyen2021trankit,
  title     = {Trankit: A Light-Weight Transformer-based Toolkit for
Multilingual Natural Language Processing},
  author    = {Nguyen, Minh Van and Lai, Viet Dac and Pouran Ben Veyseh, Amir
and Nguyen, Thien Huu},
  booktitle = {Proceedings of the 16th Conference of the European Chapter of the
Association for Computational Linguistics: System Demonstrations},
  month     = apr,
  year      = {2021},
  address   = {Online},
  publisher = {Association for Computational Linguistics},
  url       = {https://aclanthology.org/2021.eacl-demos.10},
  note      = {updated version (October 2021) at
https://arxiv.org/abs/2101.03289},
  doi       = {10.18653/v1/2021.eacl-demos.10},
  pages     = {80--90},
}
```

UUParser

Uppsala University. This was the first parser to use treebank embeddings for combined models.

Web (Github): https://github.com/UppsalaNLP/uuparser

Pretrained models: No pretrained models seem to be available.

Bibtex reference:

```
@inproceedings{stymne2018heterogeneous,
 author    = {Stymne, Sara and de Lhoneux, Miryam and Smith, Aaron and Nivre,
Joakim},
 title     = {Parser Training with Heterogeneous Treebanks},
 booktitle = {Proceedings of the 56th Annual Meeting of the Association for
Computational Linguistics},
 month     = jul,
 year      = {2018},
 address   = {Melbourne, Australia},
 publisher = {Association for Computational Linguistics},
 pages     = {619--625},
 doi       = {10.18653/v1/P18-2098},
 url       = {https://aclanthology.org/P18-2098/},
}

@inproceedings{smith2018uuparser,
 author    = {Smith, Aaron and Bohnet, Bernd and de Lhoneux, Miryam and Nivre,
Joakim and Shao, Yan and Stymne, Sara},
 title     = {82 Treebanks, 34 Models: Universal Dependency Parsing with Cross-
Treebank Models},
 booktitle = {Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing
from Raw Text to Universal Dependencies},
 month     = oct,
 year      = {2018},
 address   = {Bruxelles, Belgium},
 publisher = {Association for Computational Linguistics},
 url       = {https://aclanthology.org/K18-2011/},
}
```

HOPS “honest parser of sentences”

Loïc Grobol and Benoît Crabbé. A graph-based dependency parser inspired by Dozat and Manning (2017)’s biaffine graph parser. Contrary to Dozat, the parser performs its own tagging and can use several lexers such as FastText, BERT, FlauBert and others.

Web: https://hopsparser.readthedocs.io/en/stable/

Pretrained models: Only for French and Old French.

Bibtex reference:

```
@inproceedings{grobol2021hops,
 title      = {{Analyse en dépendances du français avec des plongements
contextualisés}},
 author     = {Grobol, Loïc and Crabbé, Benoît},
 booktitle  = {{Actes de la 28ème Conférence sur le Traitement Automatique des
Langues Naturelles}},
 eventtitle = {{TALN-RÉCITAL 2021}},
 venue      = {Lille, France},
 url        = {https://hal.archives-ouvertes.fr/hal-03223424},
 pdf = {https://hal.archives-ouvertes.fr/hal-03223424/file/HOPS_final.pdf},
}
```

Embeddings used/usable by the parsers

M-BERT (Multilingual BERT)

Produced by Google, trained on 104 languages (Wikipedia), unbalanced in terms of language-sample size. (There are multiple versions. They recommend the new, cased, with some normalizations especially for non-Latin scripts, and with two extra languages – the original M-BERT had only 102 languages.)

https://github.com/google-research/bert/blob/master/multilingual.md

Jacob Devlin and Slav Petrov. Last commit October 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of NAACL.

XLM-RoBERTa (XLM-R)

Multilingual. Two versions: base and large. Trained on 2.5T (terabytes? terawords?) of data from 100 languages filtered from CommonCrawl.

https://pytext.readthedocs.io/en/master/xlm_r.html

Bibtex reference:

```
@article{conneau2019xlmr
  title = {Unsupervised Cross-lingual Representation Learning at Scale},
  author = {Alexis Conneau and Kartikay Khandelwal
      and Naman Goyal and Vishrav Chaudhary and Guillaume Wenzek
      and Francisco Guzm\'an and Edouard Grave and Myle Ott
      and Luke Zettlemoyer and Veselin Stoyanov
  },
  journal={},
  year = {2019},
  url = {https://arxiv.org/pdf/1911.02116.pdf}
}
```

RobeCzech

RobeCzech: Czech RoBERTa, a Monolingual Contextualized Language Representation Model. Milan Straka, Jakub Náplava, Jana Straková and David Samuel. TSD 2021. https://link.springer.com/chapter/10.1007/978-3-030-83527-9_17

Trained on 4.9G Czech tokens (SYN v4, Czes, Czech part of W2C, Czech Wikipedia).

Bibtex reference:

```
@InProceedings{straka2021robeczech,
 author="Straka, Milan
 and N{\'a}plava, Jakub
 and Strakov{\'a}, Jana
 and Samuel, David",
 editor="Ek{\v{s}}tein, Kamil
 and P{\'a}rtl, Franti{\v{s}}ek
 and Konop{\'i}k, Miloslav",
 title="RobeCzech: Czech RoBERTa, a Monolingual Contextualized Language
Representation Model",
 booktitle="Text, Speech, and Dialogue",
 year="2021",
 publisher="Springer International Publishing",
 address="Cham",
 pages="197--209",
 isbn="978-3-030-83527-9"
}
```

Evaluation on Czech UD treebanks

Comparison of the Czech treebanks

PDT (Prague Dependency Treebank)

Lidové noviny + Mladá Fronta + Českomoravský Profit + Vesmír, 1993–1994

daily news, business weekly, popular science

87K sentences, 1.5M words total (train+dev+test)

CAC (Czech Academic Corpus / Korpus věcného stylu)

non-fiction, 1971–1985

24K sentences, 493K words

CLTT (Czech Legal Text Treebank)

The Accounting Act (Zákon o účetnictví)

1K sentences, 36K words

FicTree

fiction, from Czech National Corpus, 1991–2007

12K sentences, 166K words

PUD (Parallel Universal Dependencies)

online news + Wikipedia, translated from en/de/fr/it/es, around 2016

1K sentences, 18K words (test only!)

Evaluation of UDPipe 1.2 models pretrained on UD 2.5

Evaluation on test data from UD 2.5. Standard LAS, ignoring relation subtypes. I did the evaluation myself using the official UD `eval.py` script but the numbers on the diagonal (training and test data from the same treebank) are identical to those published at https://ufal.mff.cuni.cz/udpipe/1/models#universal_dependencies_25_models, lines labeled “raw text”.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	83.95	81.86	65.43	80.47	79.65
CAC	73.89	82.84	65.94	73.88	73.08
CLTT	46.92	56.31	75.43	36.44	47.54
FicTree	65.40	70.83	58.09	82.00	64.50

Evaluation on test data from UD 2.6. Results on PUD are slightly better than on PUD 2.5. Results on all other test sets are slightly worse than with UD 2.5. What are the changes in the gold standard data between UD 2.5 (November 2019) and 2.6 (May 2020)? According to the change logs, genitive, dative and instrumental nominals are now considered oblique while previously they could be objects (this affects PDT, CAC, CLTT, FicTree, PUD). Additional fixes in CLTT should make it more similar to the other treebanks: fixed annotation of “to je”, relative clauses distinguished from other adnominal clauses.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	83.37	81.36	64.98	79.54	79.99
CAC	73.51	82.39	65.62	73.29	73.50
CLTT	46.88	56.16	74.95	36.37	47.92
FicTree	65.20	70.54	57.96	81.16	64.99

Evaluation on test data from UD 2.8 (May 2021). Very similar to evaluation on UD 2.6. Change logs: Fixed bug – question marks were replaced by asterisks (PDT); adjusted treatment of double lemmas like “m`metr” (PDT, CAC); fixed bug: SpaceAfter=No should not occur at the end of paragraph (PDT); “§” is now SYM instead of NOUN (PDT, CAC, CLTT); fixed recognition of clauses with passive participles (PDT, CAC, CLTT, FicTree); restored some prepositions and other missing words (CAC); fixed some annotation errors, improved tokenization of multi-word terms, fixed tokenization of “je-li, není-li” (CLTT).

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	83.32	81.31	64.70	79.52	79.99
CAC	73.47	82.41	65.51	73.28	73.50
CLTT	46.86	56.19	74.67	36.36	47.92
FicTree	65.21	70.59	57.92	81.16	64.99

Evaluation on test data from UD 2.10 (May 2022). Improved scores on PUD, even better than with UD 2.5! Almost identical everywhere else. Change logs: Added VerbForm=Part | Voice=Pass to long forms of passive participles (PDT, CAC, PUD). Added VerbForm=Vnoun to verbal nouns (PDT, CAC, PUD). The verb "být" is now AUX in all contexts (PDT, CAC, CLTT, FicTree, PUD). Merged PRON/DET "sám/samý" (PDT, CAC, CLTT, FicTree, PUD). The word "každý" is now DET instead of ADJ (PUD).

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	83.31	81.31	64.70	79.51	80.04
CAC	73.46	82.41	65.51	73.28	73.55
CLTT	46.86	56.19	74.67	36.36	47.96
FicTree	65.21	70.59	57.92	81.16	65.05

Evaluation on test data from UD 2.11 (November 2022). The only visible change is on CLTT, which is also the only treebank that changed since UD 2.10. Change log: New train-dev-test split! Some sentences that were in training data in UD 2.5 are now in test data, so the old model should not be evaluated on the new data at all!

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	83.31	81.31	66.46	79.51	80.04
CAC	73.46	82.41	67.30	73.28	73.55
CLTT	46.86	56.19	81.11	36.36	47.96
FicTree	65.21	70.59	56.21	81.16	65.05

Evaluation of UDPipe 1.2 models trained by me on UD 2.5

If we go the default way of training a UDPipe model, without tweaking the parameters, we will not get a model identical to the pretrained one. To get an identical model, one should download the zip file "reproducible training", which is distributed together with the pretrained models, and follow the steps described there. The table below is evaluated on the test sets from UD 2.5, models are trained on UD 2.5 with dev as heldout data and with the default parameters, and the scores are lower than those achieved with the pretrained models.

```
grep cs_ udpipe-ud-2.5-191206-reproducible_training/models-ud-2.5/params_*
```

```
embedding_form_file=../ud-2.5-embeddings/cs_pdt.skip.forms.50.vectors
```

These are pretrained word embeddings in `word2vec` textual format. Each treebank has a different file and the sizes seem to reflect the sizes of the treebanks. I have not found a corresponding statement in the documentation but my hypothesis is that these embeddings were precomputed (separate from parser training) but still only on the UD data. From the UDPipe manual: "Note that pre-training word embeddings even on the UD data itself improves the accuracy. We use `word2vec` with `-cbow 0 -size 50 -window 10 -negative 5 -hs 0 -sample 1e-1 -threads 12 -binary 0 -iter 15 -min-count 2` options to pre-train on the UD data after converting it to to the horizontal format using `udpipe --output=horizontal`)." Otherwise one could also use external embeddings, e.g., those from Common Crawl as supplied for the CoNLL 2017 shared task.

How long it takes on the cluster (CPU only, default models without pretrained embeddings): Training of a default model on FicTree took 1:52:03 hours (17:49:35 – 19:41:38). Training of a default model on CAC took 7:02:05 hours (Saturday 17:49:36 – Sunday 00:51:41). Training of a default model on PDT took 16:58:32 (Saturday 17:51:34 – Sunday 10:50:06).

Model \ Test	**PDT**	**CAC**	**CLTT**	**FicTree**	**PUD**
PDT	78.19	77.76	60.89	75.59	75.75
CAC	70.73	78.95	64.78	69.87	70.75
CLTT	45.57	55.23	73.68	35.04	46.12
FicTree	64.20	69.08	57.07	80.23	63.72

Compare with the results of the pretrained models, table copied from above:

Model \ Test	**PDT**	**CAC**	**CLTT**	**FicTree**	**PUD**
PDT	83.95	81.86	65.43	80.47	79.65
CAC	73.89	82.84	65.94	73.88	73.08
CLTT	46.92	56.31	75.43	36.44	47.54
FicTree	65.40	70.83	58.09	82.00	64.50

Also compare with the results of UDPipe 1.2 trained by Milan for the CoNLL shared task 2017 (official results copied from http://universaldependencies.org/conll17/results-las.html), possibly using the big embeddings and other tricks. **Note however that these models were trained and tested on UD 2.0, so the comparability to the above results on UD 2.5 is questionable!** To provide more context, the second row contains performance published with the models at https://ufal.mff.cuni.cz/udpipe/1/models#universal_dependencies_20_models.

Model \ Test	**PDT**	**CAC**	**CLTT**		**PUD**
ÚFAL-UDPipe 1.2	83.19	84.40	76.69		79.67
Pretr UD 2.0 (published)	83.2	82.7	76.6		

Evaluation of UDPipe 2 models pretrained on UD 2.6

Evaluation on test data from UD 2.5.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	91.17	90.40	69.41	88.78	85.62
CAC	86.17	91.73	75.03	85.98	82.78
CLTT	79.53	87.81	88.82	75.13	75.86
FicTree	87.82	91.32	74.96	90.89	84.27

Evaluation on test data from UD 2.6. This is the best match because the models were trained on this release.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	92.03	91.11	69.96	90.52	87.12
CAC	87.02	92.48	75.83	87.74	84.25
CLTT	80.34	88.47	89.67	76.71	77.23
FicTree	88.66	92.05	75.73	92.66	85.73

Evaluation on test data from UD 2.8.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	91.95	91.08	69.72	90.50	87.12
CAC	86.95	92.44	75.59	87.72	84.25
CLTT	80.26	88.45	89.24	76.69	77.23
FicTree	88.59	91.99	75.43	92.64	85.73

Evaluation on test data from UD 2.10.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	91.93	91.08	69.72	90.49	87.17
CAC	86.93	92.44	75.59	87.71	84.30
CLTT	80.24	88.45	89.24	76.67	77.29
FicTree	88.57	91.99	75.43	92.63	85.78

Evaluation on test data from UD 2.11.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	91.93	91.08	73.80	90.49	87.17
CAC	86.93	92.44	76.86	87.71	84.30
CLTT	80.24	88.45	87.64	76.67	77.29
FicTree	88.57	91.99	77.28	92.63	85.78

Evaluation of UDPipe 2 models pretrained on UD 2.10

Evaluation on test data from UD 2.5.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	92.46	92.33	72.36	90.92	86.70
CAC	87.66	93.69	76.05	85.91	85.02
CLTT	81.71	89.39	87.70	80.92	78.10
FicTree	89.10	93.19	75.48	93.00	85.60

Evaluation on test data from UD 2.6.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	93.40	93.17	73.08	92.82	88.23
CAC	88.60	94.59	76.83	87.77	86.53
CLTT	82.60	90.23	88.52	82.68	79.57
FicTree	90.04	94.06	76.25	94.89	87.11

Evaluation on test data from UD 2.8.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	93.50	93.32	73.39	92.85	88.23
CAC	88.70	94.73	77.09	87.80	86.53
CLTT	82.69	90.37	89.29	82.71	79.57
FicTree	90.13	94.21	76.36	94.91	87.11

Evaluation on test data from UD 2.10. This is the best match because the models were trained on this release. **These are the best Czech parsing results so far.**

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	93.50	93.32	73.39	92.85	88.28
CAC	88.69	94.73	77.09	87.79	86.58
CLTT	82.69	90.37	89.29	82.70	79.62
FicTree	90.13	94.21	76.36	94.91	87.17

Evaluation on test data from UD 2.11.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	93.50	93.32	76.02	92.85	88.28
CAC	88.69	94.73	77.39	87.79	86.58
CLTT	82.69	90.37	87.87	82.70	79.62
FicTree	90.13	94.21	77.31	94.91	87.17

Evaluation of the Udify multilingual model pretrained on UD 2.3

These are the scores published by Kondratyuk and Straka (2019) in Table 7 in Appendix, presumably evaluated also on UD 2.3. The scores are probably achieved on input with gold-standard sentence segmentation and tokenization because Udify does not do these steps itself.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
All	92.88	92.41	89.96	92.77	87.95

Evaluation on UD 2.3. This is the best match because the model was trained on this release. Since Udify does not do its own sentence segmentation, tokenization and word segmentation, we used UDPipe 1.2 with the model pretrained on PDT UD 2.5 to preprocess the input text, then applied Udify to re-estimate the lemmas, UPOS tags, features, and dependency relations.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
All	92.17	91.60	73.65	91.37	86.56

Evaluation on UD 2.5.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
All	92.13	91.52	73.50	91.40	86.54

Evaluation on UD 2.6.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
All	91.39	90.88	72.76	90.03	86.89

Evaluation on UD 2.8.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
All	91.30	90.83	72.75	90.01	86.89

Evaluation on UD 2.10.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
All	91.29	90.83	72.75	90.00	86.94

Evaluation on UD 2.11.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
All	91.29	90.83	75.47	90.00	86.94

Evaluation of Stanza models pretrained on UD 2.8

Evaluation on test data from UD 2.5.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	89.28	87.66	72.60	85.62	85.06
CAC	78.25	87.84	73.51	76.50	78.07
CLTT	60.75	68.48	82.39	51.42	62.56
FicTree	78.14	82.56	68.06	89.01	78.31

Evaluation on test data from UD 2.6.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	90.09	88.31	73.37	87.20	86.49
CAC	79.00	88.49	74.32	77.82	79.42
CLTT	61.11	68.83	83.24	51.83	63.50
FicTree	78.80	83.10	68.68	90.70	79.55

Evaluation on test data from UD 2.8. This is the best match because the models were trained on this release.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	90.18	88.46	73.71	87.23	86.49
CAC	79.08	88.64	74.48	77.86	79.42
CLTT	61.16	68.93	83.93	51.86	63.50
FicTree	78.86	83.19	68.90	90.73	79.55

Evaluation on test data from UD 2.10.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	90.17	88.46	73.71	87.22	86.54
CAC	79.07	88.64	74.48	77.84	79.48
CLTT	61.16	68.93	83.93	51.86	63.56
FicTree	78.85	83.19	68.90	90.71	79.61

Evaluation on test data from UD 2.11.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	90.17	88.46	75.69	87.22	86.54
CAC	79.07	88.64	75.01	77.84	79.48
CLTT	61.16	68.93	87.36	51.86	63.56
FicTree	78.85	83.19	68.87	90.71	79.61

Evaluation of Trankit models pretrained on UD 2.5

Evaluation on test data from UD 2.5. This is the best match because the models were trained on this release.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	93.18	91.94	72.30	90.95	87.44
CAC	87.74	93.29	73.73	90.68	85.24
CLTT	76.86	82.92	88.01	73.10	77.38
FicTree	87.30	89.76	70.21	93.84	85.56

Evaluation on test data from UD 2.6.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	92.39	91.24	71.50	89.46	87.75
CAC	87.08	92.59	73.12	89.51	85.59
CLTT	76.35	82.56	87.23	72.22	77.81
FicTree	86.57	89.06	69.45	92.39	85.89

Evaluation on test data from UD 2.8.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	92.30	91.21	71.51	89.44	87.75
CAC	87.00	92.56	73.23	89.49	85.59
CLTT	76.30	82.54	86.82	72.19	77.81
FicTree	86.52	89.04	69.62	92.37	85.89

Evaluation on test data from UD 2.10.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	92.29	91.21	71.51	89.43	87.80
CAC	86.98	92.56	73.23	89.48	85.64
CLTT	76.29	82.54	86.82	72.19	77.86
FicTree	86.51	89.04	69.62	92.35	85.94

Evaluation on test data from UD 2.11.

Model \ Test	PDT	CAC	CLTT	FicTree	PUD
PDT	92.29	91.21	74.23	89.43	87.80
CAC	86.98	92.56	75.18	89.48	85.64
CLTT	76.29	82.54	88.71	72.19	77.86
FicTree	86.51	89.04	71.65	92.35	85.94

Summary and comparison of parsers

Best LAS on Czech test sets: UDPipe 2 pretrained on UD 2.10: 94.91 (FicTree 2.10). Trankit pretrained on UD 2.5: 93.84 (FicTree 2.5). Udify pretrained on all treebanks from UD 2.3: 92.88 (PDT 2.3). UDPipe 2 pretrained on UD 2.6: 92.66 (FicTree 2.6). Stanza pretrained on UD 2.8: 90.73 (FicTree 2.8). UDPipe 1.2 pretrained on UD 2.5: 83.95 (PDT 2.5).

Parser architecture (transition-based vs. graph-based): UDPipe 1.2: transition-based. UDPipe 2: graph-based. Udify: graph-based. Stanza: graph-based. Trankit: graph-based.

Vector representation: UDPipe 1.2: pretrained static word embeddings on UD data (it could use also embeddings pretrained on larger raw texts). UDPipe 2: Straka (2017) only static embeddings; models pretrained on UD 2.6 use multilingual BERT; models pretrained on UD 2.10 use multilingual BERT and RobeCzech. Udify: multilingual BERT (104 languages). Stanza: only static embeddings: word2vec from CoNLL 2017 + fastText for new languages. Trankit: XLM-Roberta “large”.

Speed of parsing (CPU):

(Sizes of test sets: PDT 174 000 words; CAC 11 000 words; CLTT 11 000 words; FicTree 17 000 words; PUD 19 000 words.)

udpipe1 pdt 23 by pdt 25 … 4:15 min

udpipe2 pdt 23 by pdt 26 … 1:51 min (but server runs on GPU)

udpipe2 pdt 23 by pdt210 … 1:43 min

udify pdt 25 by all 23 … 1:25:40 hours

udify pdt 26 by all 23 … 1:25:00 hours

udify cac 25 by all 23 … 6:35 min

udify cac 28 by all 23 … 5:55 min

udify cltt 28 by all 23 … 5:20 min

udify cltt 210 by all 23 … 5:20 min

udify fictree 28 by all 23 … 7:33 min

udify fictree 210 by all 23 … 8:41 min

udify pud 26 by all 23 … 7:51 min

udify pud 210 by all 23 … 8:54 min

stanza pdt 23 by pdt 28 … 7:09 min

stanza pdt 23 by cac 28 … 5:49 min

trankit pdt 23 by pdt 25 … 1:17:05 hours

trankit pdt 23 by cltt 25 … 1:36:31 hours

trankit cac 23 by pdt 25 … 3:51 min

trankit cltt 23 by pdt 25 … 2:27 min

trankit fictree 23 by pdt 25 … 6:43 min

trankit pud 23 by pdt 25 … 11:20 min

How to use it: UDPipe 1.2: download binary, download models from Lindat, run it. UDPipe 2: download lightweight Python script that communicates with Lindat via the REST API. The models are on the server. It is fast and easy but a good internet connection is needed. Udify: clone Github repository, install required modules (Python 3.7.15), download multilingual model from Lindat, preprocess raw text to CoNLL-U by something else, run predict.py. Stanza: pip install stanza (Python 3.11), construct and use pipeline in it, save as CoNLL-U. Automatic download of models. Trankit: pip install trankit (Python 3.7.15), construct and use pipeline in it, result can be saved as JSON but if you want CoNLL-U, you probably have to implement writing yourself (documentation does not say anything). Automatic download of models.

How to train it

LAS on some other languages?