Stanza: A Python Natural Language Processing Toolkit for Many Human Languages

Peng Qi* **Yuhao Zhang*** **Yuhui Zhang**
Jason Bolton **Christopher D. Manning**
Stanford University
Stanford, CA 94305
{pengqi, yuhaozhang, yuhuiz}@stanford.edu
{jebolton, manning}@stanford.edu

Abstract

We introduce Stanza, an open-source Python natural language processing toolkit supporting 66 human languages. Compared to existing widely used toolkits, Stanza features a language-agnostic fully neural pipeline for text analysis, including tokenization, multi-word token expansion, lemmatization, part-of-speech and morphological feature tagging, dependency parsing, and named entity recognition. We have trained Stanza on a total of 112 datasets, including the Universal Dependencies treebanks and other multilingual corpora, and show that the same neural architecture generalizes well and achieves competitive performance on all languages tested. Additionally, Stanza includes a native Python interface to the widely used Java Stanford CoreNLP software, which further extends its functionality to cover other tasks such as coreference resolution and relation extraction. Source code, documentation, and pretrained models for 66 languages are available at https://stanfordnlp.github.io/stanza/.

1 Introduction

The growing availability of open-source natural language processing (NLP) toolkits has made it easier for users to build tools with sophisticated linguistic processing. While existing NLP toolkits such as CoreNLP (Manning et al., 2014), FLAIR (Akbik et al., 2019), spaCy[1], and UDPipe (Straka, 2018) have had wide usage, they also suffer from several limitations. First, existing toolkits often support only a few major languages. This has significantly limited the community's ability to process multilingual text. Second, widely used tools are sometimes under-optimized for accuracy either due to a focus on efficiency (*e.g.*, spaCy) or use of less powerful models (*e.g.*, CoreNLP), potentially misleading downstream applications and insights obtained from them. Third, some tools assume input text has been tokenized or annotated with other tools, lacking the ability to process raw text within a unified framework. This has limited their wide applicability to text from diverse sources.

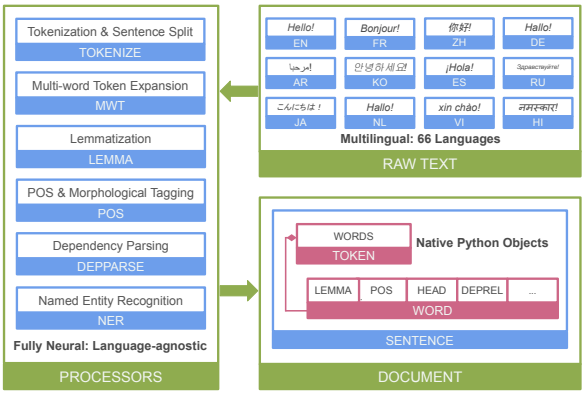

Figure 1: Overview of Stanza's neural NLP pipeline. Stanza takes multilingual text as input, and produces annotations accessible as native Python objects. Besides this neural pipeline, Stanza also features a Python client interface to the Java CoreNLP software.

We introduce Stanza[2], a Python natural language processing toolkit supporting many human languages. As shown in Table 1, compared to existing widely-used NLP toolkits, Stanza has the following advantages:

- **From raw text to annotations.** Stanza features a fully neural pipeline which takes raw text as input, and produces annotations including tokenization, multi-word token expansion, lemmatization, part-of-speech and morphological feature tagging, dependency parsing, and named entity recognition.
- **Multilinguality.** Stanza's architectural design is language-agnostic and data-driven, which allows us to release models support-

*Equal contribution. Order decided by a tossed coin.
[1] https://spacy.io/
[2] The toolkit was called StanfordNLP prior to v1.0.0.

System	# Human Languages	Programming Language	Raw Text Processing	Fully Neural	Pretrained Models	State-of-the-art Performance
CoreNLP	6	Java	✓		✓	
FLAIR	12	Python		✓	✓	✓
spaCy	10	Python	✓		✓	
UDPipe	61	C++	✓		✓	✓
Stanza	66	Python	✓	✓	✓	✓

Table 1: Feature comparisons of Stanza against other popular natural language processing toolkits.

ing 66 languages, by training the pipeline on the Universal Dependencies (UD) treebanks and other multilingual corpora.

- **State-of-the-art performance.** We evaluate Stanza on a total of 112 datasets, and find its neural pipeline adapts well to text of different genres, achieving state-of-the-art or competitive performance at each step of the pipeline.

Additionally, Stanza features a Python interface to the widely used Java CoreNLP package, allowing access to additional tools such as coreference resolution and relation extraction.

Stanza is fully open source and we make pretrained models for all supported languages and datasets available for public download. We hope Stanza can facilitate multilingual NLP research and applications, and drive future research that produces insights from human languages.

2 System Design and Architecture

At the top level, Stanza consists of two individual components: (1) a fully neural multilingual NLP pipeline; (2) a Python client interface to the Java Stanford CoreNLP software. In this section we introduce their designs.

2.1 Neural Multilingual NLP Pipeline

Stanza's neural pipeline consists of models that range from tokenizing raw text to performing syntactic analysis on entire sentences (see Figure 1). All components are designed with processing many human languages in mind, with high-level design choices capturing common phenomena in many languages and data-driven models that learn the difference between these languages from data. Moreover, the implementation of Stanza components is highly modular, and reuses basic model architectures when possible for compactness. We highlight the important design choices here, and refer the reader to Qi et al. (2018) for modeling details.

(fr) L'Association *des* Hôtels
(en) The Association of Hotels
(fr) Il y a *des* hôtels en bas de la rue
(en) There are hotels down the street

Figure 2: An example of multi-word tokens in French. The *des* in the first sentence corresponds to two syntactic words, *de* and *les*; the second *des* is a single word.

Tokenization and Sentence Splitting. When presented raw text, Stanza tokenizes it and groups tokens into sentences as the first step of processing. Unlike most existing toolkits, Stanza combines tokenization and sentence segmentation from raw text into a single module. This is modeled as a tagging problem over character sequences, where the model predicts whether a given character is the end of a token, end of a sentence, or end of a multi-word token (MWT, see Figure 2).[3] We choose to predict MWTs jointly with tokenization because this task is context-sensitive in some languages.

Multi-word Token Expansion. Once MWTs are identified by the tokenizer, they are expanded into the underlying syntactic words as the basis of downstream processing. This is achieved with an ensemble of a frequency lexicon and a neural sequence-to-sequence (seq2seq) model, to ensure that frequently observed expansions in the training set are always robustly expanded while maintaining flexibility to model unseen words statistically.

POS and Morphological Feature Tagging. For each word in a sentence, Stanza assigns it a part-of-speech (POS), and analyzes its universal morphological features (UFeats, *e.g.*, singular/plural, 1st/2nd/3rd person, etc.). To predict POS and UFeats, we adopt a bidirectional long short-term memory network (Bi-LSTM) as the basic architecture. For consistency among universal POS (UPOS),

[3] Following Universal Dependencies (Nivre et al., 2020), we make a distinction between *tokens* (contiguous spans of characters in the input text) and syntactic *words*. These are interchangeable aside from the cases of MWTs, where one token can correspond to multiple words.

treebank-specific POS (XPOS), and UFeats, we adopt the biaffine scoring mechanism from Dozat and Manning (2017) to condition XPOS and UFeats prediction on that of UPOS.

Lemmatization. Stanza also lemmatizes each word in a sentence to recover its canonical form (*e.g.*, *did*→*do*). Similar to the multi-word token expander, Stanza's lemmatizer is implemented as an ensemble of a dictionary-based lemmatizer and a neural seq2seq lemmatizer. An additional classifier is built on the encoder output of the seq2seq model, to predict *shortcuts* such as lowercasing and identity copy for robustness on long input sequences such as URLs.

Dependency Parsing. Stanza parses each sentence for its syntactic structure, where each word in the sentence is assigned a syntactic head that is either another word in the sentence, or in the case of the root word, an artificial root symbol. We implement a Bi-LSTM-based deep biaffine neural dependency parser (Dozat and Manning, 2017). We further augment this model with two linguistically motivated features: one that predicts the *linearization* order of two words in a given language, and the other that predicts the typical distance in linear order between them. We have previously shown that these features significantly improve parsing accuracy (Qi et al., 2018).

Named Entity Recognition. For each input sentence, Stanza also recognizes named entities in it (*e.g.*, person names, organizations, etc.). For NER we adopt the contextualized string representation-based sequence tagger from Akbik et al. (2018). We first train a forward and a backward character-level LSTM language model, and at tagging time we concatenate the representations at the end of each word position from both language models with word embeddings, and feed the result into a standard one-layer Bi-LSTM sequence tagger with a conditional random field (CRF)-based decoder.

2.2 CoreNLP Client

Stanford's Java CoreNLP software provides a comprehensive set of NLP tools especially for the English language. However, these tools are not easily accessible with Python, the programming language of choice for many NLP practitioners, due to the lack of official support. To facilitate the use of CoreNLP from Python, we take advantage of the existing server interface in CoreNLP, and implement a robust client as its Python interface.

When the CoreNLP client is instantiated, Stanza will automatically start the CoreNLP server as a local process. The client then communicates with the server through its RESTful APIs, after which annotations are transmitted in Protocol Buffers, and converted back to native Python objects. Users can also specify JSON or XML as annotation format. To ensure robustness, while the client is being used, Stanza periodically checks the health of the server, and restarts it if necessary.

3 System Usage

Stanza's user interface is designed to allow quick out-of-the-box processing of multilingual text. To achieve this, Stanza supports automated model download via Python code and pipeline customization with processors of choice. Annotation results can be accessed as native Python objects to allow for flexible post-processing.

3.1 Neural Pipeline Interface

Stanza's neural NLP pipeline can be initialized with the `Pipeline` class, taking language name as an argument. By default, all processors will be loaded and run over the input text; however, users can also specify the processors to load and run with a list of processor names as an argument. Users can additionally specify other processor-level properties, such as batch sizes used by processors, at initialization time.

The following code snippet shows a minimal usage of Stanza for downloading the Chinese model, annotating a sentence with customized processors, and printing out all annotations:

```
import stanza
# download Chinese model
stanza.download('zh')
# initialize Chinese neural pipeline
nlp = stanza.Pipeline('zh', processors='tokenize,
    pos,ner')
# run annotation over a sentence
doc = nlp('斯坦福是一所私立研究型大学。')
print(doc)
```

After all processors are run, a `Document` instance will be returned, which stores all annotation results. Within a `Document`, annotations are further stored in `Sentence`s, `Token`s and `Word`s in a top-down fashion (Figure 1). The following code snippet demonstrates how to access the text and POS tag of each word in a document and all named entities in the document:

```
# print the text and POS of all words
for sentence in doc.sentences:
    for word in sentence.words:
        print(word.text, word.pos)

# print all entities in the document
print(doc.entities)
```

Stanza is designed to be run on different hardware devices. By default, CUDA devices will be used whenever they are visible by the pipeline, or otherwise CPUs will be used. However, users can force all computation to be run on CPUs by setting `use_gpu=False` at initialization time.

3.2 CoreNLP Client Interface

The CoreNLP client interface is designed in a way that the actual communication with the backend CoreNLP server is transparent to the user. To annotate an input text with the CoreNLP client, a `CoreNLPClient` instance needs to be initialized, with an optional list of CoreNLP annotators. After the annotation is complete, results will be accessible as native Python objects.

This code snippet shows how to establish a CoreNLP client and obtain the NER and coreference annotations of an English sentence:

```
from stanza.server import CoreNLPClient

# start a CoreNLP client
with CoreNLPClient(annotators=['tokenize','ssplit
    ','pos','lemma','ner','parse','coref']) as
    client:
    # run annotation over input
    ann = client.annotate('Emily said that she
     liked the movie.')
    # access all entities
    for sent in ann.sentence:
        print(sent.mentions)
    # access coreference annotations
    print(ann.corefChain)
```

With the client interface, users can annotate text in 6 languages as supported by CoreNLP.

3.3 Interactive Web-based Demo

To help visualize documents and their annotations generated by Stanza, we build an interactive web demo that runs the pipeline interactively. For all languages and all annotations Stanza provides in those languages, we generate predictions from the models trained on the largest treebank/NER dataset, and visualize the result with the Brat rapid annotation tool.[4] This demo runs in a client/server architecture, and annotation is performed on the server side. We make one instance of this demo publicly available at `http://stanza.run/`. It can also be run locally with proper Python libraries installed.

[4] https://brat.nlplab.org/

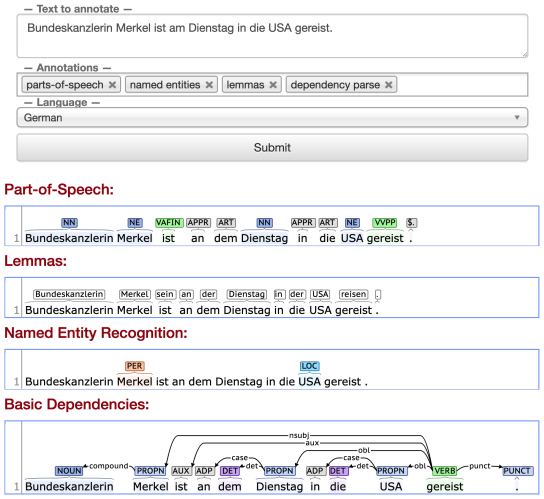

Figure 3: Stanza annotates a German sentence, as visualized by our interactive demo. Note *am* is expanded into syntactic words *an* and *dem* before downstream analyses are performed.

An example of running Stanza on a German sentence can be found in Figure 3.

3.4 Training Pipeline Models

For all neural processors, Stanza provides command-line interfaces for users to train their own customized models. To do this, users need to prepare the training and development data in compatible formats (i.e., `CoNLL-U` format for the Universal Dependencies pipeline and `BIO` format column files for the NER model). The following command trains a neural dependency parser with user-specified training and development data:

```
$ python -m stanza.models.parser \
    --train_file train.conllu \
    --eval_file dev.conllu \
    --gold_file dev.conllu \
    --output_file output.conllu
```

4 Performance Evaluation

To establish benchmark results and compare with other popular toolkits, we trained and evaluated Stanza on a total of 112 datasets. All pretrained models are publicly downloadable.

Datasets. We train and evaluate Stanza's tokenizer/sentence splitter, MWT expander, POS/UFeats tagger, lemmatizer, and dependency parser with the Universal Dependencies v2.5 treebanks (Zeman et al., 2019). For training we use 100 treebanks from this release that have non-copyrighted training data, and for treebanks that do not include development data, we randomly split out 20% of

Treebank	System	Tokens	Sents.	Words	UPOS	XPOS	UFeats	Lemmas	UAS	LAS
Overall (100 treebanks)	Stanza	99.09	86.05	98.63	92.49	91.80	89.93	92.78	80.45	75.68
Arabic-PADT	Stanza	**99.98**	80.43	**97.88**	**94.89**	**91.75**	**91.86**	**93.27**	**83.27**	**79.33**
	UDPipe	**99.98**	**82.09**	94.58	90.36	84.00	84.16	88.46	72.67	68.14
Chinese-GSD	Stanza	**92.83**	98.80	**92.83**	**89.12**	**88.93**	**92.11**	**92.83**	**72.88**	**69.82**
	UDPipe	90.27	**99.10**	90.27	84.13	84.04	89.05	90.26	61.60	57.81
English-EWT	Stanza	**99.01**	**81.13**	**99.01**	**95.40**	**95.12**	**96.11**	**97.21**	**86.22**	**83.59**
	UDPipe	98.90	77.40	98.90	93.26	92.75	94.23	95.45	80.22	77.03
	spaCy	97.30	61.19	97.30	86.72	90.83	–	87.05	–	–
French-GSD	Stanza	**99.68**	**94.92**	**99.48**	**97.30**	–	**96.72**	**97.64**	**91.38**	**89.05**
	UDPipe	**99.68**	93.59	98.81	95.85	–	95.55	96.61	87.14	84.26
	spaCy	98.34	77.30	94.15	86.82	–	–	87.29	67.46	60.60
Spanish-AnCora	Stanza	**99.98**	**99.07**	**99.98**	**98.78**	**98.67**	**98.59**	**99.19**	**92.21**	**90.01**
	UDPipe	99.97	98.32	99.95	98.32	98.13	98.13	98.48	88.22	85.10
	spaCy	99.47	97.59	98.95	94.04	–	–	79.63	86.63	84.13

Table 2: Neural pipeline performance comparisons on the Universal Dependencies (v2.5) test treebanks. For our system we show macro-averaged results over all 100 treebanks. We also compare our system against UDPipe and spaCy on treebanks of five major languages where the corresponding pretrained models are publicly available. All results are F_1 scores produced by the 2018 UD Shared Task official evaluation script.

the training data as development data. These treebanks represent 66 languages, mostly European languages, but spanning a diversity of language families, including Indo-European, Afro-Asiatic, Uralic, Turkic, Sino-Tibetan, etc. For NER, we train and evaluate Stanza with 12 publicly available datasets covering 8 major languages as shown in Table 3 (Nothman et al., 2013; Tjong Kim Sang and De Meulder, 2003; Tjong Kim Sang, 2002; Benikova et al., 2014; Mohit et al., 2012; Taulé et al., 2008; Weischedel et al., 2013). For the WikiNER corpora, as canonical splits are not available, we randomly split them into 70% training, 15% dev and 15% test splits. For all other corpora we used their canonical splits.

Training. On the Universal Dependencies treebanks, we tuned all hyper-parameters on several large treebanks and applied them to all other treebanks. We used the word2vec embeddings released as part of the 2018 UD Shared Task (Zeman et al., 2018), or the fastText embeddings (Bojanowski et al., 2017) whenever word2vec is not available. For the character-level language models in the NER component, we pretrained them on a mix of the Common Crawl and Wikipedia dumps, and the news corpora released by the WMT19 Shared Task (Barrault et al., 2019), except for English and Chinese, for which we pretrained on the Google One Billion Word (Chelba et al., 2013) and the Chinese Gigaword corpora[5], respectively. We again applied the same hyper-parameters to models for all languages.

Universal Dependencies Results. For performance on UD treebanks, we compared Stanza (v1.0) against UDPipe (v1.2) and spaCy (v2.2) on treebanks of 5 major languages whenever a pretrained model is available. As shown in Table 2, Stanza achieved the best performance on most scores reported. Notably, we find that Stanza's language-agnostic architecture is able to adapt to datasets of different languages and genres. This is also shown by Stanza's high macro-averaged scores over 100 treebanks covering 66 languages.

NER Results. For performance of the NER component, we compared Stanza (v1.0) against FLAIR (v0.4.5) and spaCy (v2.2). For spaCy we reported results from its publicly available pretrained model whenever one trained on the same dataset can be found, otherwise we retrained its model on our datasets with default hyper-parameters, following the publicly available tutorial.[6] For FLAIR, since their downloadable models were pretrained

[5] https://catalog.ldc.upenn.edu/LDC2011T13

[6] https://spacy.io/usage/training#ner
Note that, following this public tutorial, we did not use pretrained word embeddings when training spaCy NER models, although using pretrained word embeddings may potentially improve the NER results.

Language	Corpus	# Types	Stanza	FLAIR	spaCy
Arabic	AQMAR	4	**74.3**	74.0	–
Chinese	OntoNotes	18	**79.2**	–	–
Dutch	CoNLL02	4	89.2	**90.3**	73.8
	WikiNER	4	**94.8**	**94.8**	90.9
English	CoNLL03	4	92.1	**92.7**	81.0
	OntoNotes	18	88.8	**89.0**	85.4*
French	WikiNER	4	**92.9**	92.5	88.8*
German	CoNLL03	4	81.9	**82.5**	63.9
	GermEval14	4	85.2	**85.4**	68.4
Russian	WikiNER	4	**92.9**	–	–
Spanish	CoNLL02	4	**88.1**	87.3	77.5
	AnCora	4	**88.6**	88.4	76.1

Table 3: NER performance across different languages and corpora. All scores reported are entity micro-averaged test F_1. For each corpus we also list the number of entity types. $*$ marks results from publicly available pretrained models on the same dataset, while others are from models retrained on our datasets.

on dataset versions different from canonical ones, we retrained all models on our own dataset splits with their best reported hyper-parameters. All test results are shown in Table 3. We find that on all datasets Stanza achieved either higher or close F_1 scores when compared against FLAIR. When compared to spaCy, Stanza's NER performance is much better. It is worth noting that Stanza's high performance is achieved with much smaller models compared with FLAIR (up to 75% smaller), as we intentionally compressed the models for memory efficiency and ease of distribution.

Speed comparison. We compare Stanza against existing toolkits to evaluate the time it takes to annotate text (see Table 4). For GPU tests we use a single NVIDIA Titan RTX card. Unsurprisingly, Stanza's extensive use of accurate neural models makes it take significantly longer than spaCy to annotate text, but it is still competitive when compared against toolkits of similar accuracy, especially with the help of GPU acceleration.

5 Conclusion and Future Work

We introduced Stanza, a Python natural language processing toolkit supporting many human languages. We have showed that Stanza's neural pipeline not only has wide coverage of human languages, but also is accurate on all tasks, thanks to its language-agnostic, fully neural architectural design. Simultaneously, Stanza's CoreNLP client extends its functionality with additional NLP tools.

Task	Stanza		UDPipe	FLAIR	
	CPU	GPU	CPU	CPU	GPU
UD	10.3×	3.22×	4.30×	–	–
NER	17.7×	1.08×	–	51.8×	1.17×

Table 4: Annotation runtime of various toolkits relative to spaCy (CPU) on the English EWT treebank and OntoNotes NER test sets. For reference, on the compared UD and NER tasks, spaCy is able to process 8140 and 5912 tokens per second, respectively.

For future work, we consider the following areas of improvement in the near term:

- Models downloadable in Stanza are largely trained on a single dataset. To make models robust to many different genres of text, we would like to investigate the possibility of pooling various sources of compatible data to train "default" models for each language;
- The amount of computation and resources available to us is limited. We would therefore like to build an open "model zoo" for Stanza, so that researchers from outside our group can also contribute their models and benefit from models released by others;
- Stanza was designed to optimize for accuracy of its predictions, but this sometimes comes at the cost of computational efficiency and limits the toolkit's use. We would like to further investigate reducing model sizes and speeding up computation in the toolkit, while still maintaining the same level of accuracy.
- We would also like to expand Stanza's functionality by adding other processors such as neural coreference resolution or relation extraction for richer text analytics.

Acknowledgments

The authors would like to thank the anonymous reviewers for their comments, Arun Chaganty for his early contribution to this toolkit, Tim Dozat for his design of the original architectures of the tagger and parser models, Matthew Honnibal and Ines Montani for their help with spaCy integration and helpful comments on the draft, Ranting Guo for the logo design, and John Bauer and the community contributors for their help with maintaining and improving this toolkit. This research is funded in part by Samsung Electronics Co., Ltd. and in part by the SAIL-JD Research Initiative.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Association for Computational Linguistics.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics.

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*. Association for Computational Linguistics.

Darina Benikova, Chris Biemann, and Marc Reznicek. 2014. NoSta-D named entity annotation for German: Guidelines and dataset. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. Technical report, Google.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *International Conference on Learning Representations (ICLR)*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*.

Behrang Mohit, Nathan Schneider, Rishav Bhowmick, Kemal Oflazer, and Noah A Smith. 2012. Recall-oriented learning of named entities in Arabic Wikipedia. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the Twelfth International Conference on Language Resources and Evaluation (LREC'20)*.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R Curran. 2013. Learning multilingual named entity recognition from Wikipedia. *Artificial Intelligence*, 194:151–175.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics.

Milan Straka. 2018. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics.

Mariona Taulé, M. Antònia Martí, and Marta Recasens. 2008. AnCora: Multilevel annotated corpora for Catalan and Spanish. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. European Language Resources Association (ELRA).

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. OntoNotes release 5.0. *Linguistic Data Consortium*.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics.

Daniel Zeman, Joakim Nivre, Mitchell Abrams, Noëmi Aepli, Željko Agić, Lars Ahrenberg, Gabrielė Aleksandravičiūtė, Lene Antonsen, Katya Aplonova, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, Victoria Basmov, Colin

Batchelor, John Bauer, Sandra Bellato, Kepa Bengoetxea, Yevgeni Berzak, Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Agnė Bielinskienė, Rogier Blokland, Victoria Bobicev, Loïc Boizou, Emanuel Borges Völker, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Kristina Brokaitė, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Tatiana Cavalcanti, Gülşen Cebiroğlu Eryiğit, Flavio Massimiliano Cecchini, Giuseppe G. A. Celano, Slavomír Čéplö, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Alessandra T. Cignarella, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Elvis de Souza, Arantza Diaz de Ilarraza, Carly Dickerson, Bamba Dione, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Hanne Eckhoff, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Olga Erina, Tomaž Erjavec, Aline Etienne, Wograine Evelyn, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Kazunori Fujita, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta González Saavedra, Bernadeta Griciūtė, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Mika Hämäläinen, Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Johannes Heinecke, Felix Hennig, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Takumi Ikeda, Radu Ion, Elena Irimia, Ọlájídé Ishola, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Markus Juutinen, Hüner Kaşıkara, Andre Kaasen, Nadezhda Kabaeva, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Tolga Kayadelen, Jessica Kenney, Václava Kettnerová, Jesse Kirchner, Elena Klementieva, Arne Köhn, Kamil Kopacewicz, Natalia Kotsyba, Jolanta Kovalevskaitė, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Lucia Lam, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phương Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Maria Livovina, Yuan Li, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Sarah McGuinness, Gustavo Mendonça, Niko Miekka, Margarita Misirpashayeva, Anna Missilä, Cătălin Mititelu, Maria Mitrofan, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Keiko Sophie Mori, Tomohiko Morioka, Shinsuke Mori, Shigeki Moro, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Robert Munro, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Yoshihiro Nikaido, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Atul Kr. Ojha, Adédayọ̀ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Guilherme Paulino-Passos, Angelika Peljak-Łapińska, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Daria Petrova, Slav Petrov, Jason Phelan, Jussi Piitulainen, Tommi A Pirinen, Emily Pitler, Barbara Plank, Thierry Poibeau, Larisa Ponomareva, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Peng Qi, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Ivan Riabov, Michael Rießler, Erika Rimkutė, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Rosca, Olga Rudina, Jack Rueter, Shoval Sadde, Benoît Sagot, Shadi Saleh, Alessio Salomoni, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Dage Särg, Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Hiroyuki Shirasu, Muh Shohibussirri, Dmitry Sichinava, Aline Silveira, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos, Carolyn Spadine, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Shingo Suzuki, Zsolt Szántó, Dima Taji, Yuta Takahashi, Fabio Tamburini, Takaaki Tanaka, Isabelle Tellier, Guillaume Thomas, Liisi Torga, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Andrius Utka, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Abigail Walsh, Jing Xian Wang, Jonathan North Washington, Maximilan Wendt, Seyi Williams, Mats Wirén, Christian Wittern, Tsegay Woldemariam, Tak-sum Wong, Alina Wróblewska, Mary Yako, Naoki Yamazaki, Chunxiao Yan, Koichi Yasuoka, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Manying Zhang, and Hanzhi Zhu. 2019. Universal Dependencies 2.5. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.