CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies

1 Contacts and URLs

Website: http://universaldependencies.org/conll17

Email contact: ud.conll.shared.task.2017@gmail.com

2 Description of the Shared Task

Ten years ago, two CoNLL shared tasks were a major milestone for parsing research in general and dependency parsing in particular. For the first time dependency treebanks in more than ten languages were available for learning parsers; many of them were used in follow-up work, evaluating parsers on multiple languages became a standard; and multiple state-of-the art, open-source parsers became available, facilitating production of dependency structures to be used in downstream applications. While the 2006 & 2007 tasks were extremely important in setting the scene for the following years, there were also limitations that complicated application of their results: 1. gold-standard tokenization and tags in the test data moved the tasks away from real-world scenarios, and 2. incompatible annotation schemes made cross-linguistic comparison impossible. CoNLL 2017 will pick up the threads of the pioneering tasks and address the two issues just mentioned.

The focus of the 2017 task is learning syntactic dependency parsers that can work in a real-world setting, starting from raw text, and that can work over many typologically different languages, even surprise languages for which there is little or no training data, by exploiting a common syntactic annotation standard. This task has been made possible by the Universal Dependencies initiative (UD, http://universaldependencies.org/), which has developed treebanks for 40+ languages with cross-linguistically consistent annotation and recoverability of the original raw texts. For the Shared Task, the annotation scheme called “Universal Dependencies version 2”, or “UD v2” for short will be used.

Participating systems will have to find labeled syntactic dependencies between words, i.e. a syntactic *head* for each word, and a *label* classifying the type of the dependency relation. There will be multiple test sets in various languages but all data sets will adhere to the common annotation style of UD v2. Participants will parse raw text where no gold-standard pre-processing (tokenization, lemmas, morphology) is available. However, there are at least two open-source pipelines (UDPipe, https://ufal.mff.cuni.cz/udpipe/, and SyntaxNet, https://www.tensorflow.org/versions/r0.9/tutorials/syntaxnet/index.html) that the participants can run instead of training their own models for any steps preceding the dependency analysis. We will even provide variants of the test data that have been preprocessed by UDPipe. We believe that this makes the task reasonably accessible for everyone.

There will be no separate open and closed tracks. Instead, we will include every system in a single track, which will be formally closed, but where the list of permitted resources is rather broad.

The task is open to everyone. The organizers rely, as is usual in large shared tasks, on the honesty of all participants who might have some prior knowledge of part of the data that will eventually be used for evaluation, not to unfairly use such knowledge. The only exception is the chair of the organizing team, who cannot submit a system, and who will serve as an authority to resolve any disputes concerning ethical issues or completeness of system descriptions.

3 Data

The task will only utilize resources that are publicly available, royalty-free, and under a license that is free at least for non-commercial usage (e.g., CC BY-SA or CC BY-NC-SA).

3.1 Treebanks

Training/development data will be taken from the Universal Dependencies release 2.0. They will be available for many languages but the exact set of languages will be only known at release time. Participants are not allowed to use any of the previous UD releases (not only is the UD 1.* annotation style incompatible, it is also not guaranteed that the training-development-test data split remains identical). The UD release 2.0 will not contain the test data, which will be only published at the beginning of the shared task test phase.

As is usual in UD, there may be more than one training/development treebank for certain languages. Typically, the additional treebanks come from a different source and their text is from a different domain. There will be separate test data for each such treebank, and treebanks will be evaluated separately as if they were different languages. Nevertheless, participants are free to use any or all training/development treebanks/languages when they train the parser for any target treebank.

In addition, there will be test sets for which no corresponding training/development data sets exist. These additional test sets will be of two types: **1. parallel test sets** for selected known languages. Texts that have not been previously released (not even as part of a UD 1.* treebank) will be manually annotated according to the UD v2 guidelines and used in evaluation. The participating systems will know the language code, they will be thus able to pick the model trained on data from the same language; but the domain will probably be different from their training data. On the other hand, the domain of all these additional test sets (in languages where they are provided) will be identical, as they are parallel texts translated from one source. **2.** The second type of additional test sets are **surprise languages,** which have not been previously released in UD. Names of surprise languages and a small sample of gold-standard data in these languages will be published shortly before the beginning of the evaluation phase. The point of having surprise languages is to encourage participants to pursue truly multilingual approaches to parsing. However, participants who do not want to focus on the surprise languages can run a simple delexicalized parser, as predicted POS tags will be provided.

The test set for each treebank will contain at least 10,000 words. There is no upper limit on the test size (the largest test set is currently ~170K). Gold-standard annotation of the test data will only be published after the evaluation of the shared task.

Participants will receive training+development data with gold-standard tokenization, sentence segmentation, POS tags and dependency relations; for some languages also lemmas and/or morphological features. The size of these data sets will vary according to availability. For some languages, they may be as small as the test set (or even smaller), for others it may be ten times larger than the test set, and for the surprise languages it will be close to zero. One subset of the data will be formally designated as the development set, but participants will be free to use it also for training their final system.

3.2 Raw Data

We will provide additional raw data for the languages of the shared task, useful, for example, for producing word embeddings. These data sets will be taken from CommonCrawl and automatically sorted by a language recognizer. They may not be available for all languages (note that UD contains also some classical languages such as Ancient Greek). For convenience, we will provide a variant of this data pre-processed by UDPipe, and also pre-computed word embedding vectors for those participants who want to use them but do not want to tweak their own settings of the word-to-vector software.

3.3 Parallel Data

To support multi-lingual and cross-lingual approaches and model transfers, participants will be allowed to use data from the OPUS parallel corpus (http://opus.lingfil.uu.se/). We will not redistribute these data sets, participants are simply referred to the OPUS website.

3.4 Call for Additional Data

Instead of organizing a separate open track we encourage the participants to report additional data they want to use (see below for deadlines and time schedule). If the data sets are relevant to the task and meet the public availability condition, they will be added to the list of resources available to all participants.

4 Evaluation of Participating Systems

All systems will be required to generate valid output in the CoNLL-U format for all test sets. They will know the language and treebank code of the test set, but they must respond even to unknown language/treebank codes (for which there are no training data). The systems will be able to select either raw text as input, or the file pre-processed by UDPipe. Every system must produce valid output for every test set.

The evaluation will focus on dependency relations, i.e., the index of the head node and the dependency label. POS tags, lemmas and morphological features are not counted in the main evaluation metric, although the systems are free to predict them too. On the other hand, word segmentation must be reflected in the metric because the systems do not have access to gold-standard segmentation, and identifying the words is a prerequisite for dependency evaluation.

The evaluation starts by aligning the system-produced words to the gold standard ones (see the Appendix for details). Once the words are aligned, we will compute the Labeled Attachment Score (LAS) as the main scoring metric. Systems will be ranked by a macro-average over all test sets.

Labeled Attachment Score (LAS) is a standard evaluation metric in dependency parsing: the percentage of words that are assigned both the correct syntactic head and the correct dependency label. For scoring purposes, only universal dependency labels will be taken into account, which means that language-specific subtypes such as "acl:relcl" (relative clause), a subtype of the universal relation "acl" (adnominal clause), will be truncated to "acl" both in the gold standard and in the parser output in the evaluation. (Parsers can still choose to predict language-specific subtypes if it improves accuracy.) In our configuration, the standard LAS score will also have to be modified to take word segmentation mismatches into account. A dependency is therefore scored as correct only if both nodes of the relation match existing gold-standard nodes. Precision P is the number of correct relations divided by the number of system-produced nodes; recall R is the number of correct relations divided by the number of gold-standard nodes. We then define LAS as the F_1 score = 2PR / (P+R).

Besides the central metric and one overall ranking of the systems, we will evaluate the systems along various other dimensions and we may publish additional rankings for sub-tasks (e.g., performance on the surprise languages). The evaluation script will be publicly available at the shared task website.

We plan to use the Tira platform (http://www.tira.io/) to evaluate the participating systems. Therefore, participants will submit systems, not parsed data, allowing us to keep unreleased test data hidden until after the task has been completed.

5 Timeline

- December 11: Announcement of the shared task and set up of the shared task website. Registration for the Shared Task open.
- January 10: Deadline for suggesting additional data by participants (registration necessary)
- February 20 … trial data for a few languages will be available
- February 28: Task Registration deadline. Participants have to register to setup their evaluation space and other data, and get access to task data later.
- March 1 … training and development data will be released
- May 1 … surprise languages will be announced and small sample data released
- May 8 – 12 … test phase
- May 15 … results will be announced
- May 26 … submission of system description papers
- June 2 … reviews due
- June 9 … final papers due
- August 3 – 4 … CoNLL, Vancouver, Canada

Appendix: Data format and evaluation details

The CoNLL-U data format, used for Universal Dependencies treebanks, is described in more detail at http://universaldependencies.org/format.html. It is deliberately similar to the CoNLL-X format that was used in the CoNLL 2006 Shared Task and has become a de-facto standard since then. Each word has its own line and there are tab-separated columns for word form, lemma, POS tag etc. For instance, the following snippet encodes the English sentence *They buy and sell books.*

1	They	they	PRON	PRP	Case=Nom|Number=Plur	2	nsubj	_	_
2	buy	buy	VERB	VBP	Number=Plur|Person=3|Tense=Pres	0	root	_	_
3	and	and	CONJ	CC	_	2	cc	_	_
4	sell	sell	VERB	VBP	Number=Plur|Person=3|Tense=Pres	2	conj	_	_
5	books	book	NOUN	NNS	Number=Plur	2	dobj	_	SpaceAfter=No
6	.	.	PUNCT	.	_	2	punct	_	_

Syntactic words vs. multi-word tokens

However, there are a few important extensions w.r.t. the CoNLL-X format. Perhaps the most important is the notion of *syntactic words* vs. *multi-word tokens*. It makes the tokenization step in UD harder than the relatively simple procedure called tokenization in other areas of NLP. For instance, German *zum* is a contraction of the

preposition *zu* “to”, and the article *dem* “the”. In UD it is a multi-word token consisting of two syntactic words, *zu* and *dem*. These syntactic words are nodes in dependency relations. Learning this is harder than separating punctuation from words, because a contraction is not a pure concatenation of the participating words. The CoNLL-U format uses two different mechanisms here: punctuation that is conventionally written adjacent to a word is a separate single-“word” token, and an attribute in the last column tells that there was no whitespace character between the punctuation symbol and the word. On the other hand, the contraction is a multi-word token which has a separate line starting with range of following syntactic words that belong to it. Consider a German phrase *zur Stadt, zum Haus* “to the city, to the house”. The corresponding CoNLL-U section could look like this:

```
1-2   zur    _   _      _   _   _   _     _   _
1     zu     _   ADP    _   _   3   case  _   _
2     der    _   DET    _   _   3   det   _   _
3     Stadt  _   NOUN   _   _   0   root  _   SpaceAfter=No
4     ,      _   PUNCT  _   _   3   punct _   _
5-6   zum    _   _      _   _   _   _     _   _
5     zu     _   ADP    _   _   7   case  _   _
6     dem    _   DET    _   _   7   det   _   _
7     Haus   _   NOUN   _   _   3   conj  _   _
```

We will not evaluate whether the system correctly generated the range lines (1-2 zur and 5-6 zum, respectively), nor whether it generated the SpaceAfter=No attribute. But we will have to align the nodes (syntactic words) output by the system to those in the gold standard data. Thus if the system fails to recognize *zur* as a contraction and outputs

```
1     zur
2     Stadt
3     ,
```

we will treat any relations going to or from the node *zur* as incorrect. The same will happen with the node “Stadt,”, should the system fail to separate punctuation from the word *Stadt*.

If the system wrongly splits the word *Haus* and outputs

```
7-8   Haus
7     Hau
8     das
```

relations involving either *Hau* or *das* will be considered incorrect.

Even if the system recognizes *zur* as contraction but outputs wrong syntactic word forms, the tokens will be considered incorrect:

```
1-2   zur
1     zur
2     der
```

Relations involving node 1 are incorrect but relations involving node 2 may be correct.

Aligning system words with the gold standard

Easy part: suppose there are no multi-word tokens (contractions). Both token sequences (gold, system) share the same underlying text (minus whitespace). Tokens can be represented as character ranges. We can find intersections of system character ranges with gold character ranges and find the alignment in one run.

Now let’s assume there are multi-word tokens. They may contain anything, without any similarity to the original text; however, the data still contains the original surface form and we know which part of the underlying text they correspond to. So we only have to align the individual words between a gold and a system multi-word token. We use the LCS (longest common subsequence) algorithm to do that.

Sentence boundaries will be ignored during token alignment, i.e. the entire test set will be aligned at once. The systems will have to perform sentence segmentation in order to produce valid CoNLL-U files but the sentence boundaries will be evaluated only indirectly, through dependency relations. A dependency relation that goes across a gold sentence boundary is incorrect. If on the other hand the system generates a false sentence break, it will not be penalized directly, but there will necessarily be at least one gold relation that the system missed; not getting points for such relations will be an indirect penalization for wrong sentence segmentation.