Choosing a new dependency parser for Sparv
Technical report

Sasha Berdicevskis, 2020-06-03

Brief summary

I suggest we use Stanza (aka Stanford NLP, the official implementation of the well-known Dozat-Manning parser). It yields an LAS increase of about 6 points compared to the current Sparv.

Background

See my report on choosing taggers for Sparv. The data and scripts for this report will be put here: https://github.com/spraakbanken/parsing

Format

This report is exclusively about the Mamba-Dep format (the dependency format used by Nivre et al. when they created the Swedish treebank, converting the original Mamba). Lars and Marcus decreed that it should be the first priority. While I understand that we probably should continue to provide this format (it is reasonable to assume that our users will expect it), I do not think we should spend too much effort on it, since:

- it is known that the corpus contains numerous errors and inconsistencies (see Dan's report and my observations below);
- as far as I can see, the format is not being maintained and developed, there is no community, and its creators have abandoned it in favour of UD;
- it is not very well documented (and unlikely to become);
- it does not seem likely that more treebanks will be annotated using Mamba-Dep (cf. steadily growing UD).

To sum up, Mamba-Dep seems a legacy format to me, and it makes more sense to concentrate on more viable formats like UD and Koala.

Dataset

The only treebank in Mamba-Dep is Talbanken, more precisely, the version that is included in the Swedish Treebank (STB). Comparing this version with UD, I discovered that there are significant differences in sentence segmentation. In addition, Talbanken-STB has numerous omissions (if compared to the "original" Talbanken05) both on sentence level and, more importantly, on token level. They (or at least many of them) have been corrected in UD, but not in STB. Compare the numbers: Talbanken-STB contains 6160 sentences and 96346 tokens, while Talbanken-UD contains 6026 sentences (due to different segmentation) and 96819 tokens (mostly due to corrected omissions).

There is a standard STB split, but it is two-part (train vs test, no dev). I resplit train into train and dev, approximating the standard UD split as closely as possible (perfect mapping is impossible for the reasons outlined above), and I label this new split Talbanken-SBX. I preserve the automatic lemmatization provided by Sparv (again, in the UD version some of the lemmatization errors have been corrected; a semi-automatic mapping of these corrections on Talbanken-SBX might be possible).

As with taggers, we can provide both the "full" model (trained on the whole Talbanken) and the "evaluable" model.

Parsers and results

I use LAS as evaluation measure. Peter et al. list quite a few alternative measures in evaluation report (2019), but it's unclear if any of them is better as a general-purpose measure.

In addition, some of them (MLAS, BLEX) cannot be applied directly without additional adaptation. Results are listed in Table 1. I experimented a little with different embeddings (fasttext, different versions of word2vec) without much effect.

	LAS	Embeddings	Comment
Maltparser	78.39	none	baseline (≈ currentSparv)
UDPipe	74.43	fasttext (trimmed)	much better results on UD (fine-tuning problem?)
L2R	83.49	word2vec	currently not really usable
Stanza	**84.48**	word2vec	

Table 1. Parser evaluation

BASELINE: MALTPARSER. The current Sparv uses the swemalt model, trained on full Talbanken using MaltParser 1.7.2. Obviously, its performance cannot be directly evaluated, and the exact settings used to train this model are unknown. I train MaltParser (optimized via MaltOptimizer) on Talbanken-SBX-train+dev and evaluate its performance on test (LAS 78.39). I also tried the latest MaltParser (1.9.2), that does not change anything.

UDPIPE. While UDPipe 1.2 yields good results when tested on UD (LAS 84.23), I was not able to achieve more than 74.43 on Mamba-Dep. I didn't perform an exhaustive grid search, so it may be a question of fine-tuning, but I tried tweaking all the basic hyperparameters without much effect.

L2R. This is an interesting transition-based parser (designed to be fast) that yields good results (83.49). The functionality, however, is currently limited: it is impossible to save models or parse without training. In addition, outdated versions of Python and Pytorch are used, which may lead to problems.

STANZA. Formerly known as Stanford NLP, it features the official implementation of the Dozat-Manning parser (aka dozat). Decent speed (on GPU), best results (84.48). It has been explicitly written to work with UD, but it did not take much effort to apply it to Mamba-Dep instead, On UD, it reaches 85.91; higher results for SBX are unlikely.

I did not perform any error analysis or learning-curve analysis: again, I think it's better to save this effort for the other formats (which will certainly yield different results).

References

Dan's report on parsing experiments: https://github.com/spraakbanken/parsing-experiments/blob/master/Parsing-and-tagging-at-SB.md

Fasttext embeddings: https://fasttext.cc/docs/en/crawl-vectors.html, trimmed version: https://github.com/spraakbanken/parsing-experiments/tree/master/udpipe1

L2R: https://github.com/danifg/Left2Right-Pointer-Parser/

Peter et al's report on evaluation: https://gup.ub.gu.se/publication/281222?lang=sv

Stanza: https://stanfordnlp.github.io/stanza/index.html

Swemalt model: http://www.maltparser.org/mco/swedish_parser/swemalt.html

Talbanken05: https://cl.lingfil.uu.se/~nivre/research/Talbanken05.html

UDPipe: http://ufal.mff.cuni.cz/udpipe

Word2vec (CONLL17 = CommonCrawl + Wikipedia): http://vectors.nlpl.eu/repository/#

Word2vec (Göteborgsposten): https://www.ida.liu.se/divisions/hcs/nlplab/swectors/