A Systematic Comparison of Syntactic Representations of Dependency Parsing

Guillaume Wisniewski
LIMSI, CNRS, Univ. Paris-Sud
Université Paris-Saclay
91 405 Orsay, France
guillaume.wisniewski@limsi.fr

Ophélie Lacroix
DIKU, University of Copenhagen
University Park 5
2100 Copenhagen
lacroix@di.ku.dk

Abstract
We compare the performance of a transition-based parser in regards to different annotation schemes. We propose to convert some specific syntactic constructions observed in the universal dependency treebanks into a so-called more standard representation and to evaluate parsing performances over all the languages of the project. We show that the “standard” constructions do not lead systematically to better parsing performance and that the scores vary considerably according to the languages.

1 Introduction
Many treebanks have been developed for dependency parsing, following different annotations conventions. The divergence between the guidelines can results from both the theoretical linguistic principles governing the choices of head status and dependency inventories or to improve the performance of down-stream applications (Elming et al., 2013). Therefore it is difficult to compare parsing performance across languages or even across the different corpora of a single language.

Two projects of unified treebanks have recently emerged: the HamleDT (Zeman et al., 2014) and the Universal Dependency Treebank (UDT) (McDonald et al., 2013). They aim at harmonizing annotation schemes (at the level of PoS-tags and dependencies) between languages by converting existing treebanks to the new scheme. These works have led to the creation of the Universal Dependencies (UD) project (Nivre et al., 2016) that gathers treebanks for more than 45 languages (v1.3).

The UD annotation scheme has been designed to facilitate the transfer of annotations across languages: similar syntactic relations are represented by similar syntactic structures in different languages, and relations tend to hold between content words rather than through function words. However, (Schwartz et al., 2012) showed that, for English, some of the choices made to increase the sharing of structures between languages actually hurts parsing performance. Since then the UD scheme has been hypothesized to be sub-optimal for (monolingual) parsing.

In this work, we propose to systematically compare the parsing performance of alternative syntactic representations over all the languages of the UD project. We design a set of rules to automatically modify the representation of several syntactic constructions of the UD to alternative representations proposed in the literature (§ 3) and evaluate whether these transformations improve parsing performance or not (§ 4). Further we try to relate the choice of the syntactic representation to different measure of learnability to see if it is possible to predict which representation will achieve the best parsing performance.

2 Related Work
Since (Nilsson et al., 2006) many works have shown that well-chosen transformations of syntactic representations can greatly improve the parsing accuracy achieved by dependency parsers. (Schwartz et al., 2012) shows that “selecting one representation over another may affect parsing performance”. Focusing on English, they compare parsing performance through several alternatives and conclude that parsers prefer attachment via function word over content-word attachments. They argue that the learnability of a representation, estimated by the accuracy within this representation is a good criterion for selecting a syntactic representation among alternatives.

More recently, (de Lhoneux and Nivre, 2016)

1 Source code to transform between the various dependency structures we consider can be downloaded from https://perso.limsi.fr/wisniews/recherche/#dependency-transformations
studies the representation of verbal constructions to see if parsing works better when auxiliaries are the head of auxiliary dependency relations, which is not the case in UD. They highlight that the parsing benefits from the disambiguation of PoS tags for main verbs and auxiliaries in UD PoS tagset even if the overall parsing accuracy decreases.

To the best of our knowledge, (Rosa, 2015) is the only work to study the impact of the annotation scheme on the performance of transferred parsers. It compares the Prague annotation style used in the HamleDT (Zeman et al., 2014) with the Stanford style (De Marneffe and Manning, 2008) that has inspired the UD guidelines and shows that Prague style results in better parsing performance. Nevertheless — with a particular focus on the adposition attachment case — the Stanford style is advantageous for delexicalized parsing transfer.

Finally, (Silveira and Manning, 2015) performs an analysis very similar to ours and find that, for English, UD is a good parsing representation. More recently, (Kohita et al., 2017) shows that it is possible to improve parsing performance for a wide array of language by converting the dependency structure back-and-forth.

3 Conversion

We consider several alternatives to the UD annotation scheme. Most have been proposed by (Schwartz et al., 2012) or have been discussed when defining annotations of the UD (e.g. when abandoning the so-called “standard” scheme of the UDT for the content-head scheme now used in the UD). The transformations are summarized in the upper part of Table 1. We omit the transformation of verb groups that is already analyzed in detail in (de Lhonneux and Nivre, 2016). In contrast to most works analyzing the impact of annotation conventions, the alternative representations we consider are defined by selecting dependencies according to their label and transforming them rather than by modifying the tree-to-dependency conversion scheme. It is therefore possible to apply them to any language of the UD initiative.

3.1 From Simple Conversions...

The syntactic relations that we transform are mostly represented with only one dependency which can be identified by its label. In this case the conversion simply consists in inverting the role of the tokens involved in the main dependency representing the syntactic relation: the dependent becomes the head and the head becomes the dependent. Given an original dependency \( w_i \sim \sim w_j \) in which \( w_i \) is the head (i.e. \( w_i \) receive a dependency from another word \( w_k \)); \( i \) the dependency is replaced by \( w_i \sim \sim w_j \), \( j \) the former head of \( w_i \), named \( w_k \), become the new head of \( w_j \). These transformations applies to relations such as the clause subordinates (mark), the determiners (dot) or the case markings (case).

3.2 ...to Non-Projectivity...

However, more than two tokens are frequently involved in the sub-structure carried by the dependency in question. In that case, the conversion may create non-projective dependencies (i.e. crossing between dependencies). Figure 1 illustrates this problem. Let \( w_i \sim \sim w_j \) be the original dependency we want to invert, \( w_j \) being the head and \( w_i \) the dependent. If the head \( w_i \) has a child \( w_k \), i.e. there is a \( w_k \) such as \( w_i \sim \sim w_k \), and the tokens are ordered such as \( k < j < i \) or \( i < j < k \) then a crossing between the dependencies\(^2\) will appear when inverting the role of \( w_i \) and \( w_j \). To avoid introducing a non-projectivity, it is necessary to attach the former child \( w_k \) of \( w_i \) to \( w_j \).

![Figure 1](image)

Figure 1: Cases of non-projectivity caused by conversion, and correction. The main (bold) dependency \( w_i \sim \sim w_j \) is the one to invert. When inverting, \( w_j \) becomes the root of the sub-structure.

3.3 ...and Particular Cases

**Noun Sequences** For noun sequences (me, name and goeswith), we systematically consider the first word of the sequence as the head, and, when the sequence contains several words, attach each word to its preceding word, while, in UD guidelines, noun sequences are annotated in a flat, head-initial structure, in which all words in the name modify the first one (see Figure 3.3).

\(^2\)A crossing generally appears between the dependency going from \( w_i \) to his child \( w_k \) and the root dependency, now
### Table 1: Annotation scheme in the UD treebanks and standard alternatives.

| Syntactic Functions | Annotation Scheme |
|---------------------|-------------------|
| **Clause subordinates** | **UD relations** | **UD** | **Alternative** |
|                     | mwe | to read | to read |
| Detectors           | det  | the book | the book |
| Noun sequences      | mwe+goeswith, name | John Jr. Doe | John Jr. Doe |
| Case marking        | case | of Earth | of Earth |
| Coordinations       | cc+conj | me and you | me and you |
| Copulas             | cop+auxpass | is nice | is nice |
| Verb groups         | root+aux | have been done | have been done |

**Copulas** In copula constructions (cop and auxpass dependencies), the head of the dependency is generally the root of the sentence (or of a subordinate clause). The transformation of a copula dependency \( w_i \sim w_j \) between the the i-th and j-th word of the sentence consists in inverting the dependency (as for mark and case), making \( w_j \) the root of the sentence and attaching all words that were modifying \( w_i \) to \( w_j \) with a dependency not related to nouns such as det, amod, or nmod. The last step allows us to ensure the coherence of the annotations (with respect, for instance, to the final punctuation).

**Coordinations** For coordinating structures (cc and conj dependencies), in the UD scheme, the first conjunct\(^3\) is taken as the head of the coordination and all the other conjuncts depend on it via the conj relation, and each coordinating conjunction\(^4\) is attached to the first conjunct with a cc relation.\(^5\) As an alternative, we define the first coordinating conjunction as the head and attach all conjuncts to it (see Figure 3).

---

\(^3\)Typically a noun for instance (but could also be a verb or an adjective) for which the incoming dependency could be labeled with dobj, root, amod, etc.

\(^4\)Often PoS-tagged with a CONJ such as and, or, etc.

\(^5\)Recall that we are considering the version 1 guidelines; the definition of the cc relation has changed in version 2.
UD, resulting in the creation of 266 transformed corpora, 44 of which were identical to the original corpora as the transformation can not be applied (e.g. there are no multi-word expressions in Chinese). These corpora are not included in the different statistics presented in this Section.

For each configuration (i.e. a language and a transformation), a dependency parser is trained on the original data annotated with UD convention (denoted UD) and the transformed data (denoted transformed). Parsing performance is estimated using the usual Unlabeled Attachment Score (UAS, excluding punctuation). Reported scores are averaged over three trainings.

4.2 Parser

We use our own implementation of the arc-eager dependency parser with a dynamic oracle and an averaged perceptron (Aufrant and Wisniewski, 2016), using the features described in (Zhang and Nivre, 2011) which have been designed for English. Preliminary experiments show that similar results are achieved with other implementation of transition-based parsers (namely with the MaltParser (Nivre, 2003)).

5 Results

Figure 4 shows the distribution of differences in UAS between a parser trained on the original data and a parser trained on the transformed data (positive differences indicates corpora for which the UD annotation scheme results in better predictions). As expected, the annotation scheme has a large impact on the quality of the prediction, with an average difference in scores of 0.66 UAS points and variations as large as 8.1 UAS points.

However, contrary to what is usually believed, the UD scheme appears to achieve, in most cases, better prediction performance than the proposed transformations: in 58.1% of the configurations, the parser trained and evaluated on transformed data is outperformed by the parser trained on the original UD data. More precisely, the difference in UAS is negative in 93 configurations and positive in 129 configurations. Table 2 details for each transformation the percentage of languages for which the UD scheme results in better predictions. The cc dependency (conjunction), and to a lesser extent the det dependency, are easier to learn in the UD scheme than in the proposed transformed scheme. On the contrary, the choice of the cop and name structure in the UD results in large losses for many languages. For the other variations considered, the learnability of the scheme highly depends on the language. Table 3 shows the configurations with the largest positive and negative differences in scores.

Table 2: Number of times, for each transformation, a parser trained and evaluated on UD data outperforms a parser trained and evaluated on transformed data.

| Transfo. | UAS(trans.) | UAS(UD) |
|----------|-------------|---------|
| case | 44.74% | mark 58.33% | det 80.56% |
| cc | 89.47% | mwe 50.00% | name 45.83% |
| cop | 25.00% |

Table 3: Languages and transformations with the highest UAS difference.

| Lang. | Transfo. | UAS(trans.) | UAS(UD) |
|-------|----------|-------------|---------|
| nl | copule | 69.82% | 67.73% |
| fi | copule | 66.59% | 64.30% |
| et | copule | 70.38% | 67.95% |
| la | copule | 59.34% | 56.47% |
| sl | copule | 79.69% | 76.75% |

Analysis To understand the empirical preferences of annotation schemes we consider several measures of the ‘learnability’ and ‘complexity’ of a treebank:
A metric is said coherent if it scores the syntactic structure that achieves the best parsing performance higher than its variation. Table 4 reports the numbers of times, averaged over languages and transformations, that each metric is coherent.

Contrarily to what has been previously reported, the considered metrics are hardly able to predict which annotation scheme will result in the best parsing performance. Several reasons can explain this result. First, it is the first time, to the best of our knowledge that these metrics are compared on such a wide array of languages. It is possible that these metrics are not as language-independent as can be expected. Second, as our transformations are directly applied on the dependency structures rather than when converting the dependency structure from a constituency structure, it is possible that some of their transformations are erroneous and the resulting complexity metric biased.

### 6 Conclusion

Comparing the performance of parsers trained and evaluated on UD data and transformed data, it appears that the UD scheme leads mainly to better scores and that measures of learnability and complexity are not sufficient to explain the annotation preferences of dependency parsers.

### 7 Acknowledgement

Ophélie Lacroix is funded by the ERC Starting Grant LOWLANDS No. 313695.

### References

Lauriane Aufrant and Guillaume Wisniewski. 2016. PanParser: a Modular Implementation for Efficient Transition-Based Dependency Parsing. Technical report, LIMSI-CNRS, March.

Miryam de Lhoneux and Joakim Nivre. 2016. Should have, would have, could have. investigating verb group representations for parsing with universal dependencies. In Proceedings of the Workshop on Multilingual and Cross-lingual Methods in NLP, pages 10–19, San Diego, California, June. Association for Computational Linguistics.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. The stanford typed dependencies representation. In Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation, pages 1–8. Association for Computational Linguistics.

Jakob Elming, Anders Johannsen, Sigrid Klerke, Emanuele Lapponi, Hector Martinez Alonso, and

### Table 4: Number of times a given learnability measure is able to predict which annotation scheme will result in the best parsing performance.

| metric                  | percentage |
|-------------------------|------------|
| distance                | 43.6%      |
| predictability          | 64.8%      |
| derivation complexity   | 62.6%      |
| derivation perplexity   | 61.2%      |

[6] Similarly to (Søgaard and Haulrich, 2010) we consider a trigram language model but use a Witten-Bell smoothing as many corpora were too small to use a Kneser-Ney smoothing. As for the derivation complexity, the words are ordered according to an oracle prediction of the reference structure.
Anders Søgaard. 2013. Down-stream effects of tree-to-dependency conversions. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 617–626, Atlanta, Georgia, June. Association for Computational Linguistics.

Dan Gusfield. 1997. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, New York, NY, USA.

Ryosuke Kohita, Hiroshi Noji, and Yuji Matsumoto. 2017. Multilingual back-and-forth conversion between content and function head for easy dependency parsing. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 1–7, Valencia, Spain, April. Association for Computational Linguistics.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pages 122–131, Prague, Czech Republic, June. Association for Computational Linguistics.

Ryan McDonald, Joakim Nivre, Yvonne Quirimbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal Dependency Annotation for Multilingual Parsing. In Proceedings of ACL 2013, the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 92–97, Sofia, Bulgaria, August.

Jens Nilsson, Joakim Nivre, and Johan Hall. 2006. Graph transformations in data-driven dependency parsing. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, pages 257–264, Sydney, Australia, July. Association for Computational Linguistics.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzae, Masayuki Asahara, Aitziber Atxuza, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Yevgeni Berzak, Riyaz Ahmad Bhat, Cristina Bosco, Gosse Bouma, Sam Bowman, Gülşen Çebirolu Eryiit, Giuseppe G. A. Celano, Čar Čoltekin, Miriam Connor, Marie-Catherine de Marneffe, Arantzta Díaz de Ibarra, Kaja Dobrovoljc, Timothy Dozat, Kira Drogozina, Tomaž Erjavec, Richard Farkas, Jennifer Foster, Daniel Galbraith, Sebastian Garza, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gokirmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Normunds Grūzitis, Bruno Guillaume, Jan Hajić, Dag Haug, Barbora Hladká, Radu Ion, Elena Irimia, Anders Johansen, Hüner Kaškara, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Jessica Kenney, Simon Krek, Veronika Laippala, Lucia Lam, Alessandro Lenci, Nikola Ljubesić, Olga Lyshevskaya, Teresa Lynn, Aibek Mazahinov, Christopher Manning, Catálina Máranduc, David Mareček, Héctor Martínez Alonso, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missišša, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Keiko Sophie Mori, Shunsuke Mori, Kardi Muischnek, Nina Mustafina, Kaili Miurisep, Vitaly Nikolaev, Hanna Nurmia, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Čenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalnina, Prokopis Prokopidis, Tina Puolakainen, Sampo Pyysalo, Loganathan Ramsamy, Laura Rituma, Rudolf Rosa, Shadi Saleh, Baiba Saulite, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Natalie Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Kiril Simov, Aaron Smith, Carolyn Spadine, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jing Xian Wang, Jonathan North Washington, Zdeněk Zabokrtský, Daniel Zeman, and Hanzi Zhu. 2016. Universal dependencies 1.3. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Joakim Nivre. 2003. An Efficient Algorithm for Projective Dependency Parsing. In Proceedings of IWPT 2003, the 8th International Workshop on Parsing Technologies, Nancy, France.

Rudolf Rosa. 2015. Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015), chapter Multi-source Cross-lingual Delexicalized Parser Transfer: Prague or Stanford??. pages 281–290. Uppsala University, Uppsala, Sweden.

Roy Schwartz, Omri Abend, and Ari Rappoport. 2012. Learnability-based syntactic annotation design. In Proceedings of COLING 2012, pages 2405–2422, Mumbai, India, December. The COLING 2012 Organizing Committee.

Natalia Silveira and Christopher Manning. 2015. Does universal dependencies need a parsing representation? an investigation of english. Depling 2015, 310.

Anders Søgaard and Martin Haulrich. 2010. On the derivation perplexity of treebanks. In Proceedings of Treebanks and Linguistic Theories 9.

Daniel Zeman, Ondřej Dušek, David Mareček, Martin Popel, Loganathan Ramsamy, Jan Štěpánek, Zdeněk Zabokrtský, and Jan Hajič. 2014. Hamledt: Harmonized multi-language dependency treebank. Language Resources and Evaluation, 48(4):601–637.
Yue Zhang and Joakim Nivre. 2011. Transition-based Dependency Parsing with Rich Non-local Features. In Proceedings of ACL 2011, the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 188–193, Portland, Oregon, USA, June. Association for Computational Linguistics.