Automatically Selecting the Best Dependency Annotation Design with Dynamic Oracles

Guillaume Wisniewski[1], **Ophélie Lacroix**[2] and **François Yvon**[1]
[1]LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, F-91405 Orsay, France
[2]Siteimprove, Sankt Annæ Plads 28, DK-1250 Copenhagen, Denmark
wisniews@limsi.fr, ola@siteimprove.com, yvon@limsi.fr

Abstract

This work introduces a new strategy to compare the numerous conventions that have been proposed over the years for expressing dependency structures and discover the one for which a parser will achieve the highest parsing performance. Instead of associating each sentence in the training set with a single gold reference, we propose to consider a set of references encoding alternative syntactic representations. Training a parser with a dynamic oracle will then automatically select among all alternatives the reference that will be predicted with the highest accuracy. Experiments on the UD corpora show the validity of this approach.

1 Introduction

Multiple annotation conventions have been proposed over the years for representing dependency structures (Hajič et al., 2001; De Marneffe et al., 2014). The divergence between annotation guidelines can result from the theoretical linguistic principles governing the choices of head status and dependency inventories, the tree-to-dependency conversion scheme or arbitrary decisions regarding closed class words, such as interjections or discursive markers, the syntactic role of which is debatable. Several works have shown that the choice of a dependency structure can have a large impact on parsing performance (Silveira and Manning, 2015; de Lhoneux and Nivre, 2016; Kohita et al., 2017) and on the performance of downstream applications (Elming et al., 2013).

A natural way to decide which syntactic representation is the best is to choose the one for which a standard parser will achieve the highest parsing performance (Schwartz et al., 2012; Husain and Agrawal, 2012; Noro et al., 2005). Implementing this general principle faces two challenges: *i)* defining a learning criterion that can predict which dependency structure will be the easiest to learn *ii)* finding a way to explore a potentially large number of annotation schemes that describe all combinations of several design decisions.

This work shows that the dynamic oracle of Goldberg and Nivre (2013) can straightforwardly uncover the most *learnable* dependency representation among a predefined set of possible references.[1] Rather than associating each sentence in the training set to a single reference, we propose to consider a set of references encoding alternative syntactic representations. Training a parser with a dynamic oracle will then automatically select among all alternatives the reference that will be predicted with the highest accuracy.

This article is organized as follows: we first review standard structural transformations studied in the literature that will be used to build a treebank annotated with multiple references (§2). We then show how the dynamic oracle of Goldberg and Nivre (2013) can be used to train a parser when each sentence is associated to a set of references and explain how it can be used to define a learnability criteria (§3). An experimental evaluation of our approach is presented in §4.

2 Dependency Transformations

In this section, we explain how to automatically transform the reference UD treebanks (Nivre et al., 2016), to build corpora in which each sentence is annotated by a set of possible trees.

The UD project aims at developing cross-linguistically consistent treebank annotations for many languages by harmonizing annotation schemes between languages and converting existing treebanks to this new scheme. Several recent papers (Kohita et al., 2017; de Lhoneux and Nivre, 2016; Silveira and Manning, 2015; Popel et al.,

[1]Contrary to unsupervised parsing, our approach does not aim at discovering a dependency structure and rather relies on the existence of several hand-crafted references.

2013) have investigated whether the choices made to increase the sharing of structures between languages hurt parsing performance and have identified a variety of choice points in which more than one design could be advocated. Most of these points are related to the issue of headness: contrary to most works in theoretical linguistic, UD assumes that function words should be categorically subordinated to content words to maximize the similarity of dependency trees across languages (Osborne and Maxwell, 2015).

The alternative representations we consider are summarized in Table 1. They mostly consist in demoting the lexical head and making it dependent on a functional head. We designed a set of handcrafted rules[2] to convert dependencies between these two schemes. Each application of a rule creates a new tree in the set of references that is being built. As shown in Figure 1, the resulting set of references encodes all possible combinations of the considered transformations.

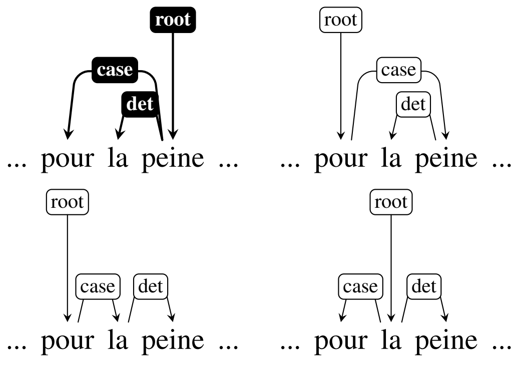


Figure 1: Examples of all the annotations generated by applying the rules of Table 1. The UD reference is in solid black.

3 Training a Dependency Parser with Multiple References

Dynamic Oracle In a transition-based parser (Nivre, 2008), a parse is computed by performing a sequence of *transitions* building the parse tree in an incremental fashion. A partially built dependency tree is represented by a *configuration* c; when in c, applying a transition t results in the parser moving to a new configuration denoted $c \circ t$.

At each step of the parsing process, every possible transition is scored by a classifier (e.g. a linear model), given a feature representation of c and

[2]A more detailed description of the transformations can be found in (Wisniewski and Lacroix, 2017). The source code is freely available on the first author web site.

Algorithm 1: Training on one sentence with multiple references (see text for notations).

```
Input: $W$ the input sentence, $\mathcal{T}$ the set of gold trees
1  $c \leftarrow \textsc{Initial}(W)$
2  while $\neg\textsc{Terminal}(c)$ do
3      $\textsc{Correct} \leftarrow \{t \mid \exists T \in \mathcal{T}, \textsc{Oracle}(t, c, T) = 0\}$
4      $t_p \leftarrow \arg\max_{t \in \textsc{Legal}(c)} \mathbf{w} \cdot \phi(c, t)$
5      $t_o \leftarrow \arg\max_{t \in \textsc{Correct}(c)} \mathbf{w} \cdot \phi(c, t)$
6      if $t_p \notin \textsc{Correct}$ then
7          $\textsc{Update}\,(\mathbf{w}, \phi(c, t_o), \phi(c, t_p))$
8          $t_{\text{next}} \leftarrow t_o$
9      else
10         $t_{\text{next}} \leftarrow t_p$
11     $\mathcal{T} \leftarrow \{T \in \mathcal{T} \mid \textsc{Oracle}(t_{\text{next}}, c, T) = 0\}$
12     $c \leftarrow c \circ t_{\text{next}}$
```

model parameters $\mathbf{w}$; the score of a *derivation* (a sequence of transitions) generating a given parse tree is the sum of its transition scores. Parsing thus amounts to finding, starting from the initial configuration $\textsc{Initial}(W)$, the derivation having the highest score, typically using greedy or beam search.[3]

Algorithm 1 formalizes the training procedure when the dynamic[4] oracle of Goldberg and Nivre (2013) is used: for each sentence, a parse tree is built incrementally and at each step, if the predicted transition prevents the creation of a gold dependency, the parameters are updated, according, for instance, to the perceptron rule (l.7). Erroneous transitions can efficiently be found using the $\textsc{Oracle}(t, c, T)$ function formally defined in (Goldberg and Nivre, 2013) as computing the number of dependencies of a gold parse tree T that can no longer be predicted when a transition t is applied in configuration c.

During training, it often happens that several transitions are equally good: in such situations, the training algorithm breaks ties among oracle transitions according to the model current prediction (l.5). As suggested in the imitation learning literature (Daumé III and Marcu, 2005; Ross and Bagnell, 2010), this strategy enables to sample those configurations that will be the most similar to the ones seen when predicting a new dependency tree:

[3]In this work we only consider greedy parsers. Extending the proposed approach to beam parsers would prevent discarding a reference because one of the its transition is too hard to predict (i.e. has a very low score), which would, intuitively, results in even better predictions.

[4]Algorithm 1 only uses the non-deterministic property of the oracle and not its completeness. Even if we use the most common term, 'dynamic oracle', our approach only requires the former property.

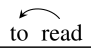
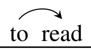
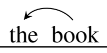
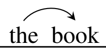
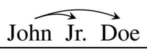
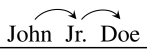
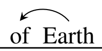
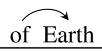
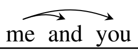
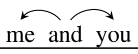
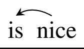
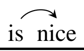
Syntactic Functions		Annotation Scheme	
Relation	**UD labels**	**UD**	**Alternative**
Clause subordinates	`mark`	 to read	 to read
Determiners	`det`	 the book	 the book
Noun sequences	`mwe+goeswith`, `name`	 John Jr. Doe	 John Jr. Doe
Case marking	`case`	 of Earth	 of Earth
Coordinations	`cc+conj`	 me and you	 me and you
Copulas	`cop+auxpass`	 is nice	 is nice

Table 1: Annotation schemes in the UD treebanks and standard alternatives.

it is a way to let the parser explore more specifically the part of the search space it prefers and is more likely to see at test time (Aufrant et al., 2017). Using a dynamic oracle usually results in substantial improvements in accuracy compared to static oracles.

Considering Multiple References Implementing the training algorithm described above only requires the ability to detect whether a transition will cause an erroneous dependency. It can naturally be extended to the case of multiple references: a transition is considered correct as long as it can predict at least one of the gold trees; when moving to a new configuration, trees that can no longer be generated are removed from the set of references, in order to make sure the parser will not mix the dependencies of two gold trees (l.11).

Upon full completion of parsing, there will remain only one surviving reference that has been selected *according to the model current predictions*. This reference corresponds to the dependency structure that is the most similar to the hypothesis the parser would have predicted at test time and can therefore be described as the reference the parser prefers: intuitively, Algorithm 1 will thus identify the reference that will be predicted with the highest accuracy.

4 Experiments

Data We separately apply to the 7 dependencies considered the transformations described in Section 2 on the 38 languages of the UD project (v1.3), resulting in 266 transformed corpora.[5] To evaluate the ability of the proposed method to identify the 'best' dependency structure, we consider fully as well as partially transformed sentences: a sentence with n dependencies of interest will generate 2^n references.

For each condition (i.e. a language and a transformation), a dependency parser is trained using (a) the original data annotated with UD convention, (b) 'transformed' data in which each sentence is associated to a reference in which all dependencies of interest have been transformed and (c) the data associated with a set of reference containing all the partially transformed references (including the original and transformed references).

Parser We use our own implementation of an arc-eager unlabeled dependency parser with a dynamic oracle and an averaged perceptron, using the features described in (Zhang and Nivre, 2011) which have been designed for English and have not been adapted to the specificities of the other languages.[6] Training stops when the UAS estimated on the validation set has converged.

Impact of Transformations Figure 2 shows the distribution of differences in UAS between a parser trained on the original data (setting (a)) and a parser trained on the transformed data (setting (b)). To evaluate the proposed transformations, we follow the approach introduced in (Schwartz et al.,

[5]44 transformed corpora were identical to the original corpora as the transformation can not be applied (e.g. there are no multi-word expression in Chinese).

[6]Note that the proposed approach can be apply to other transition systems and classifiers.

2012) consisting in comparing the original and the transformed data on their respective references.

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. These results show that, contrary to general belief (Schwartz et al., 2012; Kohita et al., 2017), the UD scheme is not sub-optimal for monolingual parsing: the difference in UAS is negative in 93 conditions and positive in 129. Table 2 details for each dependency the when the UD scheme results in better predictions.

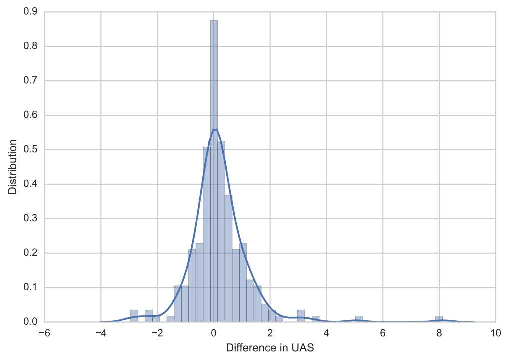


Figure 2: Distribution of differences between the UAS achieved on the UD and transformed corpora. Positive values indicate better prediction performance with UD annotations.

`case`	44.7%	`mark`	58.3%	`det`	80.5%
`cc`	89.4%	`mwe`	50.0%	`name`	45.8%
`cop`	25.0%				

Table 2: Percentage of times a parser trained and evaluated on UD data (setting (a)) outperforms a parser trained and evaluated on transformed data (setting (b)).

Training with Multiple References To assess the impact of training with multiple references (setting (c)), we first evaluate the capacity of Algorithm 1 to consistently select a single annotation scheme during training. We count, in each conditions, the number of times the reference that has survived training was following the original scheme and the number of times it was following the transformed scheme. For 74.7% of the conditions, the reference that has survived training was following the same annotation scheme for more than 70% of the training examples. This observation proves the ability of the parser to commit itself to a single annotation scheme.

Learnability Criterion The training procedure proposed in this article was designed to uncover the dependency structure that will optimize parsing accuracy. In this section we evaluate whether this goal is achieved, by counting the number of conditions in which the annotation scheme that has survived training the most often (in setting (c)) is indeed the one that achieves the best performance on the test set, as evaluated by testing a parser in settings (a) and (b).

We will consider, as baselines, two measures of the 'learnability' of a treebank, the *predictability* of an annotation scheme (Schwartz et al., 2012) and the *derivation perplexity* (Søgaard and Haulrich, 2010). Contrary to our approach, these two measures aims at deciding which of two annotations schemes will achieve the best parsing accuracy without actually training and testing a parser. The predictability is defined as the entropy of the conditional distribution of the dependent PoS knowing the head PoS. The derivation perplexity is the perplexity of 3-gram language model estimated on a corpus in which the words of a sentence appear in the order in which they are attached to their head.[7]

Table 3 reports the number of times, averaged over languages and transformations, that each measure of learnability is able to predict which of two competing annotation schemes will yield the best parsing performance. These results clearly show that the approach we propose to evaluate the 'learnability' of an annotation scheme outperforms existing criteria and is able to select the annotation convention that achieves the highest parsing performance.

metric	learnability
predictability	64.8%
derivation complexity	62.6%
`multiple references`	76.3%

Table 3: Number of times a given learnability measure is able to predict which annotation scheme will result in the best parsing performance. '`multiple references`' corresponds to the approach proposed in this work.

[7]Similarly to (Søgaard and Haulrich, 2010), we consider a trigram language model but use Witten-Bell smoothing as many corpora are too small to use Kneser-Ney smoothing.

5 Conclusion

This work introduces a new strategy to compare the numerous representations that have been proposed over the years for expressing dependency structures and discover the one that is easiest to learn. Experiments with the popular transition-based parser on the UD corpora show the validity of the proposed approach.

In future work, we would like to evaluate the impact of annotation conventions on other kind of parsers and to find the properties of a dependency tree that facilitate its prediction. We also plan to find ways to easily annotate sentences with multiple references (e.g. by indicating that the head of word can be chosen arbitrarily) and eliminate the constraint that references should be trees.

Acknowledgments

This work has been partly funded by the *Agence Nationale de la Recherche* under ParSiTi project (ANR-16-CE33-0021) and MultiSem project (ANR-16-CE33-0013).

References

Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2017. Don't stop me now! Using global dynamic oracles to correct training biases of transition-based dependency parsers. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 318–323. http://www.aclweb.org/anthology/E17-2051.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: approximate large margin methods for structured prediction. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*. ACM Press, New York, NY, USA, pages 169–176. https://doi.org/http://doi.acm.org/10.1145/1102351.1102373.

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*. Association for Computational Linguistics, San Diego, California, pages 10–19. http://www.aclweb.org/anthology/W16-1202.

Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal stanford dependencies: a cross-linguistic typology. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA), Reykjavik, Iceland.

Jakob Elming, Anders Johannsen, Sigrid Klerke, Emanuele Lapponi, Hector Martinez Alonso, and 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*. Association for Computational Linguistics, Atlanta, Georgia, pages 617–626. http://www.aclweb.org/anthology/N13-1070.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the Association for Computational Linguistics* 1:403–414. https://transacl.org/ojs/index.php/tacl/article/view/145.

Jan Hajič, Barbora Vidová-Hladká, and Petr Pajas. 2001. The Prague Dependency Treebank: Annotation Structure and Support. In *Proceedings of the IRCS Workshop on Linguistic Databases*. University of Pennsylvania, Philadelphia, USA, pages 105–114.

Samar Husain and Bhasha Agrawal. 2012. Analyzing parser errors to improve parsing accuracy and to inform tree banking decisions. *Linguistic Issues in Language Technology* 7.

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*. Association for Computational Linguistics, Valencia, Spain, pages 1–7. http://www.aclweb.org/anthology/E17-2001.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Comput. Linguist.* 34(4):513–553. https://doi.org/10.1162/coli.07-056-R1-07-027.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Yevgeni Berzak, Riyaz Ahmad Bhat, Cristina Bosco, Gosse Bouma, Sam Bowman, Gülşen Cebirolu Eryiit, Giuseppe G. A. Celano, Çar Çöltekin, Miriam Connor, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Tomaž Erjavec, Richárd

Farkas, Jennifer Foster, Daniel Galbraith, Sebastian Garza, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gokirmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Normunds Grūzītis, Bruno Guillaume, Jan Hajič, Dag Haug, Barbora Hladká, Radu Ion, Elena Irimia, Anders Johannsen, Hüner Kaşkara, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Jessica Kenney, Simon Krek, Veronika Laippala, Lucia Lam, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Keiko Sophie Mori, Shunsuke Mori, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Vitaly Nikolaev, Hanna Nurmi, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalnia, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Loganathan Ramasamy, Laura Rituma, Rudolf Rosa, Shadi Saleh, Baiba Saulīte, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Natalia 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 Žabokrtský, Daniel Zeman, and Hanzhi Zhu. 2016. Universal dependencies 1.3. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague. http://hdl.handle.net/11234/1-1699.

Tomoya Noro, Chimato Koike, Taiichi Hashimoto, Takenobu Tokunaga, and Hozumi Tanaka. 2005. Evaluation of a japanese cfg derived from a syntactically annotated corpus with respect to dependency measures. In *Proceedings of the Fifth Workshop on Asian Language Resources (ALR-05) and First Symposium on Asian Language Resources Network (ALRN)*. http://aclweb.org/anthology/I/I05/I05-4002.pdf.

Timothy Osborne and Daniel Maxwell. 2015. A historical overview of the status of function words in dependency grammar. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*. Uppsala University, Uppsala, Sweden, Uppsala, Sweden, pages 241–250. http://www.aclweb.org/anthology/W15-2127.

Martin Popel, David Mareček, Jan Štpánek, Daniel Zeman, and Zdnk Žabokrtský. 2013. Coordination structures in dependency treebanks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 517–527. http://www.aclweb.org/anthology/P13-1051.

Stephane Ross and J. Andrew (Drew) Bagnell. 2010. Efficient reductions for imitation learning. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*. pages 661–668.

Roy Schwartz, Omri Abend, and Ari Rappoport. 2012. Learnability-based syntactic annotation design. In *Proceedings of COLING 2012*. The COLING 2012 Organizing Committee, Mumbai, India, pages 2405–2422. http://www.aclweb.org/anthology/C12-1147.

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*.

Guillaume Wisniewski and Ophélie Lacroix. 2017. A systematic comparison of syntactic representations of dependency parsing. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*. Association for Computational Linguistics, Gothenburg, Sweden, pages 146–152. http://www.aclweb.org/anthology/W17-0419.

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*. Association for Computational Linguistics, Portland, Oregon, USA, pages 188–193. http://www.aclweb.org/anthology/P11-2033.