Does Universal Dependencies need a parsing representation? An investigation of English

Natalia Silveira
Stanford University
Department of Linguistics
Stanford, CA
natalias@stanford.edu

Christopher Manning
Stanford University
Department of Linguistics and
Department of Computer Science
Stanford, CA
manning@stanford.edu

Abstract

This paper investigates the potential of defining a parsing representation for English data in Universal Dependencies, a crosslingual dependency scheme. We investigate structural transformations that change the choices of headedness in the dependency tree. The transformations make auxiliaries, copulas, subordinating conjunctions and prepositions heads, while in UD they are dependents of a lexical head. We show experimental results for the performance of MaltParser, a data-driven transition-based parser, on the product of each transformation. While some transformed representations favor performance, inverting the transformations to obtain UD for the final product propagates errors, in part due to the nature of lexical-head representations. This prevents the transformations from being profitably used to improve parser performance in that representation.

1 Introduction

There is a considerable amount of research suggesting that the choice of syntactic representation can have an impact on parsing performance, in constituency (Klein and Manning, 2003; Bikel, 2004; Petrov et al., 2006; Bengoetxea and Gojenola, 2009) as well as dependency (Nilsson et al., 2007; Nilsson et al., 2006; Schwartz et al., 2012) parsing. Recently, this has led designers of dependency representations (Marneffe et al., 2014) to suggest the use of an alternative parsing representation to support the performance of statistical learners.

While it is clear that, at the limit, trivializing a linguistic representation in order to make it easier to parse is undesirable – for example, by making each word depend on the previous one – there certainly exists a variety of choice points in which more than one type of design is defensible. In the dependency tradition, semantic and syntactic criteria have been recognized to motivate headedness, and there are well-known examples of conflicts between those criteria (Nilsson et al., 2006). Here we investigate four syntactic constructions that are *loci* of such conflicts: verb groups, prepositional phrases, copular clauses and subordinate clauses. The baseline representation is Universal Dependencies (Nivre et al., 2015), a multilingual dependency scheme that strongly prefers lexical heads. For each target construction, structural transformations are defined that demote the lexical head and make it dependent on a functional head.

We show experimental results for the performance of MaltParser, a data-driven transition-based parser, on the product of each transformation. While some transformed representations are in fact easier to learn, error propagation when inverting the transformations to obtain UD prevents them from being profitably used to improve parser performance in that representation.

2 Related work

Schwartz et al. (2012) is a systematic study of how representation choices in dependency annotation schemes affect their learnability for parsing. The choice points investigated, much like in the current paper, relate to the issue of headedness. The experiments look at functional versus content heads in six constructions: (a) coordination structures (where the head can be a conjunction or one of the conjuncts), (2) infinitives (the verb or the marker *to*), (3) nominal phrases (the determiner, if any, or the noun), (4) nominal compounds (the first noun or the last), (5) prepositional phrases (the preposition or its complement) and (6) verb groups (the main verb, or the highest modal, if any). Each combination of these binary

Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015), pages 310–319,
Uppsala, Sweden, August 24–26 2015.

choices is tested with 5 different parsers, which represent different paradigms in dependency parsing.The edges in the representation are unlabeled, unlike the common practice in NLP. The results show a learnability bias towards a conjunct in (1), a noun in (3), and a preposition in (5) in all the parsers. Furthermore, a bias towards the modal heads in (6) and towards the head-initial representation in (4) is seen with some parsers. No significant results are found for (2).

In Ivanova et al. (2013), the authors run a set of experiments that provide a comparison of (1) 3 dependency schemes, (2) 3 data-driven dependency parsers and (3) 2 approaches to POS-tagging in a parsing pipeline. The comparison that is relevant here is (1). The dependency representations compared are the basic version of Stanford Dependencies (Marneffe and Manning, 2008), and two versions of the CoNLL Syntactic Dependencies (Johansson and Nugues, 2007). For all parsers and in most experiments (which explore several pipelines with different POS-tagging strategies), SD is easier to label (i.e., label accuracy scores are higher) and CoNLL is easier to structure (i.e., unlabeled attachment scores are higher). In terms of LAS, MaltParser (Nivre et al., 2007) performs best of the 3 parsers with SD, and MSTParser (McDonald et al., 2006) performs best with CoNLL.

In Nilsson et al. (2006), the authors investigate the effects of two types of input transformation on the performance of MaltParser. Those two types are: structural transformations, of the same nature of those investigated in the present paper; and projectivization transformations, that allow non-projective structures to be represented in a way that can be learned by projective-only parsing algorithms, and then transformed into the non-projective representation at the end. Of interest here are the structural transformations, which in their work target coordinated phrases and verb groups. The data and baseline representation come from the Prague Dependency Treebank (PDT) version 1.0 (Hajic et al. 2001). The PDT's representation of coordination is so different from UD's that the results cannot be expected to carry over. The verb group transformation, on the other hand, is almost identical to the auxhead transformation proposed here. In the PDT, auxiliary verbs never have dependents. Other dependents of the main verb are attached to the first verb of the verb group if they occur anywhere before the last verb; otherwise, they are attached to the left verb. In the reverse transformation, all dependents of auxiliaries go back to the main verb. All the transformations reported in the paper prove helpful for the parser. In the case of verb groups, which is of particular interest here, the labeled attachment score goes up by 0.14% (in a test set of 126k tokens).

Following up on the previous paper, (Nilsson et al., 2007) investigates the same transformations applied to different datasets and under distinct parsing algorithms, to understand if they generalize across languages and parsing strategies. The representations for the different languages studied are similar to the PDT's representation. With respect to the structural transformations, the authors find that there are, again, small gains from converting the representations of coordination and verb groups. However, in their experiments, graph-based MSTParser, unlike transition-based MaltParser, does not perform better on the transformed input.

3 Background

3.1 Universal Dependencies

The baseline representation to which transformations are applied in this set of experiments is the UD representation, which was developed to allow for parallel annotation across languages. It is based on Stanford Dependencies (Marneffe and Manning, 2008), a widely used representation for English. In order to preserve some flexibility for language-specific annotation, UD has a two-layer architecture. The universal layer is common to all languages, and it aims to capture phenomena at a level that highlights crosslinguistic commonalities. However, the need for parallelism with other languages often imposes a high level of abstraction on the annotation, which might be undesirable when working in a monolingual setting. For that reason, the representation can be extended with language-specific relations as needed, via inheritance. This makes it straightforward to informatively harmonize annotations across languages, since they already use the same dependency types at the universal level. At the same time, it allows enough expressivity for capturing detail that may be important for a specific language, but difficult to port to others.

UD inherits from SD the concern with usefulness for relation extraction, in addition to crosslinguistic parallelism. Both of those motivate a

radical stance on headedness: lexical heads are adopted across the board. The idea is that, because syntax competes with morphology, grammatical functions that are performed by function words in one language may be performed by bound morphemes in another. If those function words are allowed to enter contentful relations (such as predicate-argument relations), the structures assigned in the presence of such words will be very different than the structures assigned when those words give way to bound morphemes. This has been the primary motivation for the most important change from SD to UD for English, which is the new treatment of prepositional phrases. While in SD a functional-head representation was adopted, with prepositions heading nouns, in UD a lexical representation is adopted, and the complement in the prepositional phrase depends on the preposition. This allows more parallelism with languages in which the functions of some English prepositions (such as *of*) are performed by case morphemes.

3.2 The English Web Treebank corpus

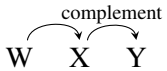
The corpus used in all of this paper's experiments is the UD-annotated English Web Treebank (EWT) corpus (Silveira et al., 2014). The EWT consists of 254,830 word tokens (16,624 sentences) of text, and was released by the Linguistic Data Consortium in 2012. The text is manually annotated for sentence- and word-level tokenization, as well as part-of-speech tags and constituency structure in the Penn Treebank scheme. The data comprises five domains of web text: blog posts, BBC newsgroup threads, emails from the Enron corpus, Amazon reviews and answers from Yahoo! answers.

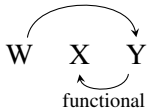
This corpus was hand-annotated with dependency relations following an evolving version of Stanford Dependencies. The UD annotation was obtained from the SD annotation, partly via automatic conversions, and partly via manual revisions. The result is the first human-checked large-scale gold standard for syntactic dependency annotation of English text. The first version annotated with the UD representation was released in 2015 (Nivre et al., 2015)[1].

[1] http://universaldependencies.github.io/docs/

4 Structural transformations

All the transformations studied in this paper have the same underlying structure: they involve a content word which is a head by semantic criteria, and a functional word which is a head by syntactic criteria. They reverse those heads' roles in relation to each other, and in relation to the outer structure in which they are embedded. One head is the promoted head, and it stands in an appropriate relation with some element from outside the construction (e.g., *dobj* in the case of a noun phrase). The other (candidate) head is the demoted head, and it should be attached to the promoted counterpart. So we have:

(1) a. 

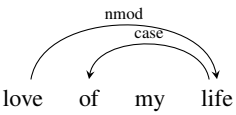
b. 

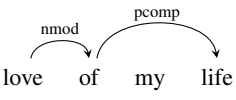
In the simplest case, transformations of this kind can be inverted with no loss, which adds interest: there is no need to allow the parser to orient design decisions. The linguistic representation can be transformed for parser training, and the parser output can go through the inverse transformation for general use. (This is the approach taken in Nilsson et al. (2006).) In other (common) cases, however, there may be important difficulties, which are discussed below. Four constructions are studied here: prepositional phrases, verb groups, copular clauses, and embedded clauses with overt complementizers.

4.1 The *casehead* transformation

To illustrate in some detail, let us examine the case of prepositional phrases. Take, for example, the sentence in 2a. The lexical-head representation, which UD adopts, chooses *life* as the promoted head and *of* as the demoted head, as shown in 2b. The functional representation, shown in 2c, swaps those roles.

(2) a. I found the love of my life

b. 

c. 

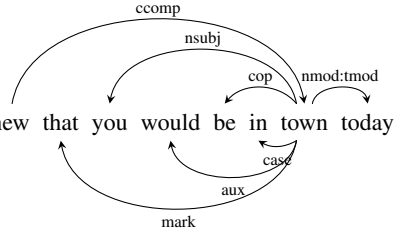
This is a particularly interesting example, because there already is evidence in the literature (Schwartz et al., 2012) that making prepositions the heads – that is, adopting the functional-head representation – can yield better parser performance. This will be called the *casehead* transformation. As mentioned above, the label *case* is used in UD for prepositions in prepositional phrases; it is also used for the genitive marker *'s*, but here the transformation is not applied to that marker. The other transformations are *auxhead*, *cophead* and *markhead*. All are named after the labels used in UD for the dependencies attaching the function word promoted by the transformation to its lexical head.

4.2 Other transformations

The sentence below exemplifies uses of the labels *aux*, *cop* and *mark*. Each transformation generates a different tree for this sentence.

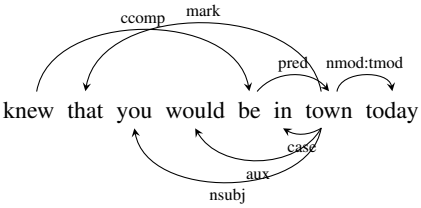
It will be clear from the examples in this section that, when the functional head is promoted, the way in which the dependents of the (now demoted) lexical head are handled can have important consequences. Illustrated first are the simplest versions of each transformation, where no dependents of the lexical heads are moved. In 4.3, alternatives will be discussed. In 3a is the UD representation of a sentence that has all the target constructions for which transformations are defined.

(3) a. We knew that you would be in town today



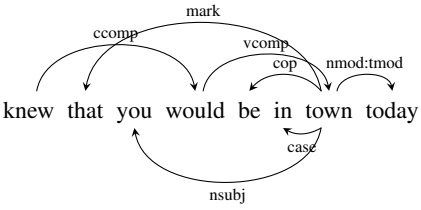
The label *cop* is used for the verb *be* in copular clauses. In relation to other dependency schemes, UD makes an unusual choice here, inherited from SD: instead of attaching the subject and other clausal dependents to the copular verb, and making the predicate itself a dependent of that verb, the representation takes the nonverbal predicate as the head and attaches the verb and clausal dependents to it. In the present example, the predicate is a prepositional phrase, but since those are also represented with lexical heads, the head of the entire copular clause is the noun *town*. In terms of crosslinguistic adequacy, it pays off, since this structure creates a parallel between English and languages where no copular verbs are used for this type of predication. Note that even the auxiliary is attached to the predicate rather than the copular verb. The simple *cophead* transformation, in which none of the dependents of the lexical head are moved to the functional head with its promotion, yields the tree in 4a.

(4) a. We knew that you would be in town today



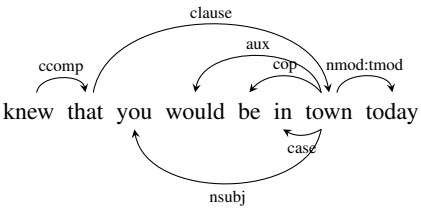
In English, the label *aux* is used to attach modals and traditional auxiliaries. In the case of auxiliary *be*, the label *auxpass* is used, to encode voice information directly in the dependency tree. These dependents are always attached to the predicate, which is why here the head of *would* is *town*. The simple *auxhead* transformation results in the tree depicted in 5a.

(5) a. We knew that you would be in town today



The label *mark* is used for subordinating conjunctions in embedded clauses, and additionally for the infinitive marker *to*. It is always attached to the predicate, much like *aux*. The simple *mark* transformation is illustrated in 6a.

(6) a. We knew that you would be in town today



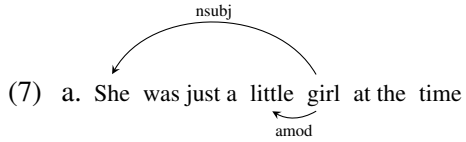
Note that in all cases, the labels used for the demoted head in the transformations are not part of the UD label set. The *auxhead* transformation is also used for *auxpass* dependencies; in those cases, the complement is called *vcomppass*. This is to avoid making the transformed representation artificially easier by eliminating the voice distinction.

4.3 Handling of dependents in transformations

The examples of simplified transformations given above make it apparent that transformations can introduce undesired nonprojectivity, and may sometimes result in representations that are linguistically objectionable. Both of those are reasons why it may be desirable to move the dependents of the lexical head when it is demoted. But exactly which dependents to move is an important question, due to the fact that modifier attachment in a dependency representation can be inevitably ambiguous, as shown below.

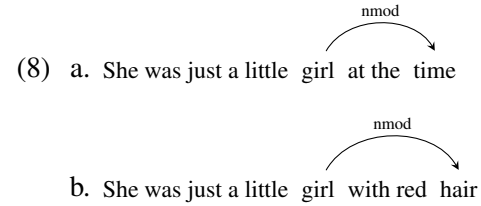
In UD, all the nodes in dependency graphs are words, and therefore all edges must be relations between words. However, syntactic relations can occur not only between words, but between constituents, at different levels. In UD, modifiers of constituents are indistinguishable from modifiers of the constituents head.

This has an important consequence for the distinction between functional-head and lexical-head representations. In the light of a theory of syntax in the style of Minimalist Grammar, one may argue that no two constituents share the same functional head. However, it is clear that the same lexical item can be the lexical head of multiple constituents that contain one another. Consider the example in 7a.



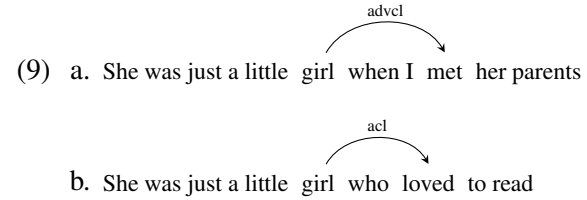
Here *girl* has dependents on two levels: as a nominal head, it has an adjectival dependent, *little*. As a nominal predicate, which is the lexical head of a copular clause, it has a subject dependent, 'she'. The entire clause and the noun phrase which constitutes its main predicate share a lexical head in UD. Because modifiers at both levels will be attached to that shared lexical head, it is not possible to determine from the structure alone what constituent is being modified by a dependent.

These distinctions are often very subtle and irrelevant for practical applications; but UD's radical adoption of lexical heads creates some cases where the distinctions are clear and very meaningful. Perhaps the clearest case is that of copulas with nominal predicates. In UD, we have trees like 8a and 8b.

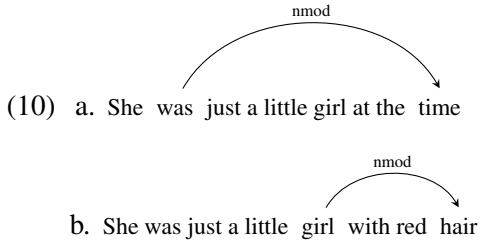


In 8a, the prepositional phrase is a modifier of the predicate. In a constituent representation, its parent would not be the NP/DP node. But in 8b, clearly the modifier is in the nominal domain. In UD, the head is the noun *girl*, because it is both the head of the nominal constituent, and the head of the clausal constituent (since it is the lexical head of the copula).

If these were clausal modifiers, UD would offer an opportunity for disambiguation in the type system: clausal dependents of a noun are typed *acl* (see 9b), but clausal dependents of a predicate are typed *advcl* (as in 9a). Prepositional phrases, nonetheless, are uniformly labeled *nmod*.



The ambiguity of the representation on this point is a consequence of the choice of representing lexical heads. Attachment would be distinct if heads were functional, because then the clausal constituent and the nominal constituent would each have its own distinct head. The functional-head representation would give us the two distinct structures in 10a and 10b.

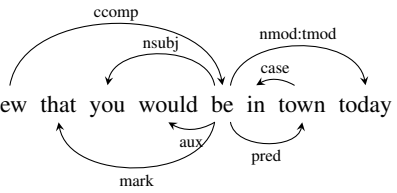


This poses a problem in the context of structural transformations, because, in converting from an ambiguous representation to a non-ambiguous one (lexical-head to functional-head, in this case), it is not necessarily simple, or possible, to resolve the ambiguity in order to obtain the correct parsing representation. (The same issue arises with coordinated constituents in Nilsson et al. (2006).)

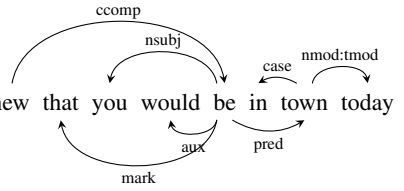
More generally, this highlights the fact that it may be harmful to blindly reattach the dependents of a lexical head to a functional head in a transformation, and careful handling of dependents may be necessary.

In an attempt to address these difficulties, 3 versions of each transformation were tested. It should be noted that dependents which are known to attach to heads rather than constituents are never moved – these are *mwe*, *compound*, *goeswith*, *name* and *foreign*. In the *simple* version, which has been illustrated above, none of the dependents of the lexical head are moved when the functional head is promoted. In the *full* version, all dependents of the lexical head are moved, except those which are known to modify nouns exclusively (in UD, these are *amod*, *acl*, *appos*, *det*, *case*, *nummod*). In the *partial* version, which is design to minimize nonprojectivity, all dependents of the lexical head which occur to the left of the functional head are moved when that head is promoted, and all other dependents are left attached to the lexical head. So now for each *Xhead* transformation, we have $Xhead_s$, $Xhead_f$ and $Xhead_p$. To provide a comparison with $copheads_s$, which was shown in 4a, $copheads_f$ and $copheads_p$ are illustrated in 11a and 11b, respectively.

(11) a.



b.



Note that, in $copheads_f$, 'today' is moved and becomes a dependent of *be*, the promoted head; in contrast, in $copheads_p$, that dependent remains attached to the lexical head *town*, since it does not occur to the left of the promoted head. If the sentence was *We knew that today you would be in town*, the two transformations would have identical results.

5 Experiments

The experiments in this paper fit the following template: a version of the training and development data from the EWT corpus was used to optimize a MaltParser model. Then that model was used to parse the test set (of 25k tokens) and evaluated on the gold standard. In the 12 experiments where the training data had undergone a transformation, the output of the parser was converted back into the original UD representation with the inverse transformation, so that it could be compared to the actual gold standard.

An important concern with this type of experiment is that the default feature sets for the algorithms may be implicitly biased towards a particular type of representation. Therefore, it was crucial to explore different hyperparameters and feature sets. This was done in two steps. The MaltParser model was obtained via an optimization heuristic: MaltOptimizer (Ballesteros and Nivre, 2012) was used on the different versions of the training set to obtain models optimized for the different transformation. This generates 13 models: one for the baseline, and one for each of the three versions of the four transformations. In a second step, all 13 representations of the dev set data were parsed with all the 13 models that MaltOptimizer produced in the previous step. Note that MaltOptimizer did not use the dev set. The model that performed best on the dev set for each transformation was chosen. Interestingly, it came out that the best-performing model for a representation was never the one recommended by MaltOptimizer for that representation. For all the representations, the models that effectively performed best on the dev set consistently used the *stackproj* algorithm, coupled with different pseudo-projectivization strategies. Throughout this procedure, the metric being maximized was the labeled attachment score (excluding punctuation), which seems to be the crucial measure of performance for most client applications.

Each *Xhead* transformation targets a different construction, and the frequency of those in the data varies. Additionally, the three versions of the transformations change the data to different extents. To give a measure of these differences, Table 1 shows the percentage of tokens in the training data that are changed by a transformation, for all 12 transformations.

These counts make it clear that, in the case of *casehead* and *markhead*, there is little difference between the *partial* and *simple* transformations. This is because in the case of these transforma-

	full	*partial*	*simple*
auxhead	21.15%	13.62%	08.37%
casehead	21.83%	19.17%	18.37%
cophead	11.62%	08.33%	04.94%
markhead	15.75%	08.04%	07.83%

Table 1: Percentage of tokens changed with relation to the gold standard by each transformation.

	full	*partial*	*simple*
auxhead	05.80%	05.49%	41.55%
casehead	06.51%	06.22%	31.57%
cophead	05.84%	05.13%	07.52%
markhead	09.71%	05.14%	10.41%
baseline		05.13%	

Table 2: Percentage of non-projective dependencies per version of the data.

tions, the lexical head is unlikely (in English) to have dependents which occur to the left of the functional head.

The transformations are also very different in terms of how much non-projectivity they introduce. Table 2 shows how that proportion changes with each transformation, which helps understand their performance.

The labeled attachment scores of the best-performing models for each representation on the test set are given in Table 3. These results were obtained by comparing the output of parsers trained on transformed representation to a transformed version of the gold-standard test set. These scores will be referred to as the *within-representation performance*. Statistical significance was assessed using Dan Bikel's parsing evaluation comparator[2], at the 0.05 significance level.

Our interest here is not to guide the design of

[2]http://pauillac.inria.fr/ seddah/compare.pl

	full	*partial*	*simple*
auxhead	84.71%	85.21%*	84.59%
casehead	84.46%	85.31%*	85.11%*
cophead	85.11%*	85.31%*	84.49%
markhead	84.29%*	84.94%	85.10%*
baseline		84.69%	

Table 3: Labeled accuracy scores for within-representation evaluations. The scores marked with * have a significant difference from the baseline.

	full	*partial*	*simple*
auxhead	84.37%	84.84%	84.43%
casehead	84.13%*	84.91%	84.86%
cophead	84.28%*	84.53%	84.03%*
markhead	84.27%*	84.89%	85.00%
baseline		84.69%	

Table 4: Labeled accuracy scores for evaluations on UD. The scores marked with * have a significant difference from the baseline.

a new representation, but rather to find strategies that will improve parser performance for the existing UD representation. For this reason, we also present results on the actual UD representation. These results are obtained by transforming the output of a parser with the inverse of the transformation applied to the training data, and comparing that to the actual gold standard annotation. The labeled accuracy scores are in Table 4.

6 Discussion

These results show that, in the case of UD, tree transformations do not seem to improve parser performance if the output needs to be converted back to UD. There are no significant positive results in Table 4, and in fact a few of the transformations have a significant negative impact on the score. Interestingly, this holds even for some representations which have better within-representation performance than the baseline.

In terms of within-representation performance, the most successful transformations were $cophead_p$ and $casehead_p$. The $cophead_p$ transformation makes the representation of copular clauses more parallel to that of other clauses in UD: it moves dependents from a nonverbal predicate to a copular verb (excluding those with labels that apply exclusively to noun modifiers or head-level modifiers). With this transformation, verbs are uniformly viewed as the heads of clauses, making the representation more predictable. The effect of this transformation is particularly notable because it is the one that affects the fewest tokens, as shown in Table 1. The results on $casehead_p$ shown in Table 3 are consistent with Schwartz et al. (2012). This transformation shortens dependency lengths, which benefits the transition-based parser. Dependency edges with $length < 5$ constitute 81.93% of the total in the $casehead_p$ data, and 81.73% in the $casehead_s$

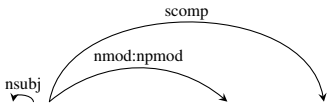
data. These numbers are up from 80.35% in the baseline.

6.1 Inverting transformations

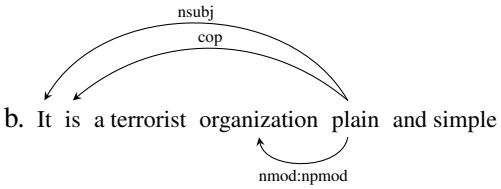
An obvious trend in these results is that attachment scores consistently decrease when the output of a parser trained on transformed input is inverted back to UD. On perfectly annotated data, there is no distortion: for all 12 operations proposed here, using the inverse transformation on a transformed gold standard reverts all the changes and gives back the original data. However, parser errors are not always handled well by transformations. The reason for this is that the different representations yielded in these operations reflect attachment decisions differently. The differences can skew the evaluation results.

All transformations target constructions including 2 crucial edges, as seen before: one between the promoted head and the demoted head, and another – the attachment edge – coming from the outside of the construction to the promoted head. In functional-head representations, the attachment edge can be correct even if the lexical head is wrongly identified, as in 12a, which is an actual parser error on the $cophead_f$-transformed data. The inverted version is shown in 12b.

(12) a.



b.



When this tree is converted back to the lexical-head representation, the attachment edge, which in this case is simply *root*, is moved to the wrongly-identified lexical head. While in 12a the root of the sentence was identified correctly, in the inverted version it is wrong; one error in the functional-head representation turns into two in its lexical-head counterpart.

Another issue that arises in inverting transformations is that, when dependents are moved from the functional head to the lexical head, errors may be amplified. This is also seen here. The phrase *plain and simple* was wrongly identified as a predicate. With the inversion of the transformation, the subject of the sentence, which was correct in 12a, is moved and made a dependent of the false predicate. This type of error propagation is the reason why the *simple* transformations have the smallest differences between the score on the transformed gold standard and the score on the inverted parsed output.

Even when the parser does correctly identify the lexical head as a dependent of the functional head, another source of complications is that it may identify additional "lexical heads" (i.e., dependents with the label reserved for the lexical head, such as *pred* in the case of *cophead*). In this implementation, we do not use any heuristics to try to identify if one of the candidates is the actual lexical head, and which. This can also lead to errors, as now the inverse transformation may erroneously move dependents.

As a counterpoint, one should note that inverting the transformations to obtain a lexical-head representation is also, in a way, forgiving: there are no distinctions between attachment to the functional or to the lexical head, because the inversion moves all dependents of the functional head to the lexical head. This eliminates a plausible source of errors – and some linguistic information, making UD the poorer representation here. Nevertheless, errors of this types seem to be outnumbered by others that are introduced or amplified by inverting these transformations.

These problems help explain why the results reported here, with respect to prepositional phrases and verb groups, and suggest different directions than those reported in Schwartz et al. (2012): in that paper, the results of different parsers are evaluated against different versions of the gold standard. Here, since the main concern is the design of a parsing representation that is meant simply as an intermediary step, all output has to be evaluated against the same gold standard. This creates an opportunity for losses that did not exist in the experiments of Schwartz et al. (2012).

7 Conclusion

Although there have been cases in the literature in which small gains in performance were obtained from invertible structural transformations on dependency trees, similar transformations do not seem to yield any significant gain for UD in English. This occurs despite the fact that these tree operations can result in performance improve-

ments, as is evident from the within-representation scores of some of the transformed datasets. Nevertheless, because of the nature of lexical- and functional-head representations, the inversion of the transformations on the parser output can and does amplify errors. Due to these difficulties, it is not immediately possible to exploit structures transformations for the benefit of UD.

We believe that other styles of tree transformations may yield gains for parser performance on UD; specifically, ones designed in the style of the node-merging and -splitting that has been used in constituency parsing since Klein and Manning (2003). That investigation is left for future work.

References

Miguel Ballesteros and Joakim Nivre. 2012. MaltOptimizer: An Optimization Tool for MaltParser. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 58–62, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kepa Bengoetxea and Koldo Gojenola. 2009. Exploring Treebank Transformations in Dependency Parsing. In *Proceedings of the International Conference RANLP-2009*, pages 33–38, Borovets, Bulgaria. Association for Computational Linguistics.

Daniel M. Bikel. 2004. Intricacies of collins' parsing model. *Comput. Linguist.*, 30(4):479–511, December.

Angelina Ivanova, Stephan Oepen, and Lilja vrelid. 2013. Survey on parsing three dependency representations for English. *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 31–37.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In Joakim Nivre, Heiki-Jaan Kalep, Kadri Muischnek, and Mare Koit, editors, *NODALIDA 2007 Proceedings*, pages 105–112, Tartu, Estonia. University of Tartu.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marie-Catherine Marneffe and Christopher D. Manning. 2008. The Stanford Typed Dependencies Representation. *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8.

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. *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 216–220, Stroudsburg, PA, USA. Association for Computational Linguistics.

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 the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 257–264, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jens Nilsson, Joakim Nivre, and Johan Hall. 2007. Generalizing Tree Transformations for Inductive Dependency Parsing. *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 968–975.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Glsen Eryigit, Sandra Kbler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Joakim Nivre, Cristina Bosco, Jinho Choi, Marie-Catherine de Marneffe, Timothy Dozat, Richárd Farkas, Jennifer Foster, Filip Ginter, Yoav Goldberg, Jan Hajič, Jenna Kanerva, Veronika Laippala, Alessandro Lenci, Teresa Lynn, Christopher Manning, Ryan McDonald, Anna Missilä, Simonetta Montemagni, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Maria Simi, Aaron Smith, Reut Tsarfaty, Veronika Vincze, and Daniel Zeman. 2015. Universal dependencies 1.0.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 433–440, Stroudsburg, PA, USA. Association for Computational Linguistics.

Roy Schwartz, Omri Abend, and Ari Rappoport. 2012. Learnability-based syntactic annotation design. In *COLING*, volume 24, pages 2405–2422.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A Gold

Standard Dependency Corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.