Evaluating Dependency Parsing: Robust and Heuristics-Free Cross-Annotation Evaluation

Reut Tsarfaty
Uppsala University
Sweden

Joakim Nivre
Uppsala University
Sweden

Evelina Andersson
Uppsala University
Sweden

Abstract

Methods for evaluating dependency parsing using attachment scores are highly sensitive to representational variation between dependency treebanks, making cross-experimental evaluation opaque. This paper develops a robust procedure for cross-experimental dependency parsing evaluation based on deterministic unification-based operations for harmonizing different representations and a refined notion of tree edit distance for evaluating parse hypotheses relative to multiple gold standards. We demonstrate that, for different conversions of the Penn Treebank into dependencies, apparent performance trends observed when comparing parsing results obtained in isolation may change or dissolve completely when parse hypotheses are normalized and brought into the same common ground.

1 Introduction

Data-driven dependency parsing has seen a considerable surge of interest in recent years, where dependency parsers are trained on and evaluated for parsing sentences in English (Yamada and Matsumoto, 2003; Nivre and Scholz, 2004; McDonald et al., 2005) as well as other languages (Nivre et al., 2007). The evaluation metric traditionally associated with dependency parsing is based on scoring labeled or unlabeled attachment decisions, whereby each correctly identified pair of head-dependent words is counted towards the success of the parser (Buchholz and Marsi, 2006). As it turns out, however, such evaluation procedures are sensitive to the annotation choices in the data on which the parser was trained.

Different annotation schemes often make different assumptions with respect to how linguistic content is represented in a treebank (Rambow, 2010). The consequence of such annotation discrepancies is that when we compare parsing results across different experiments, even ones that use the same parser and the same set of sentences, we may be comparing trees that express the same linguistic content differently. The gap between results in different experiments then may not reflect a true gap in parser performance but an arbitrary difference in a linguistic annotation choices.

Different methods have been proposed to make dependency parsing results comparable across experiments, from picking a single gold standard for all experiments in the language (Buchholz and Marsi, 2006), to neutralizing the direction of the arcs in the tree (Schwartz et al., 2011), to articulating a standard set of correct dependencies to which the parser output should be converted (Carroll et al., 1998; Candito et al., 2010). All of these methods have their pitfalls. Picking a gold standard skews the results in favor of parsers which were trained on it. Neutralizing the direction of arcs may not solve all possible discrepancies. Transforming sets of arcs to “correct dependencies” typically requires the use of heuristic rules and runs the risk of distorting correct information, for which the parser may be penalized.

This paper proposes a robust evaluation procedure for comparing parsing results across data sets which is based on a fully deterministic and heuristic-free three-phase strategy. First, we convert the annotated dependency trees into functional trees, fully-ordered and label trees which retain all the informa-
tion provided by the arc-structure and grammatical function labels in the original trees. Next, we use a set of unification-based operations to formally compute, for each sentence in the data, the common denominator of the different gold standards, containing all and only the linguistic content that is shared between the functional trees in the different schemes. Finally, we use a refined version of tree edit distance (Zhang and Shasha, 1989) to compute the normalized distance from the overall gold tree to the parse hypotheses, discarding all edit operations that reflect other treebanks’ annotation variants.

We apply the proposed procedure to comparing dependency parsing results across different experiments which use the Penn Treebank data converted into dependencies according to different sets of linguistic assumptions. We show that even when starting off with the same data and the same parser, different conversion schemes yield different experimental results. While selecting a single gold standard for evaluation seriously increases the apparent performance gap, our procedure shows that results across different conversions are on a part with one another when compared and normalized against their shared linguistic content. We conclude then that empirical comparison of parsing results should be a well thought through enterprise, and suggest ways to extend the proposed pipeline for additional cross-experimental evaluation scenarios.

2 The Challenge: Treebank Theories

Dependency treebanks contain information about the grammatically meaningful elements in the utterance and the grammatical relations between them. While the formal representation in a dependency treebank is well-defined according to current standards (Küblier et al., 2009), there are different ways in which the trees can be used to express syntactic content (Rambow, 2010). Consider, for instance, algorithms for converting the trees in the Penn treebank (Marcus et al., 1993) into dependency structures. Different conversion algorithms implicitly make different assumptions about how to represent the sentences in the data. When multiple conversion algorithms are applied to the same data (cf. (de Marneffe et al., 2006; Johansson and Nugues, 2007; Choi and Palmer, 2010)), we end up with differing dependency trees for the same sentences. Let us demonstrate common discrepancies that annotation or conversion idiosyncrasies may yield.

**Lexical vs. Functional Head Choice.** In linguistics, there is a distinction between lexical heads and functional heads. A lexical head carries the semantic gist of a phrase while a functional one marks its relation to other parts of the sentence. The two kinds of heads may or may not coincide in a single wordform (Zwicky, 1993). A common example is prepositional phrases, such as the phrase “on Sunday”. This phrase has two possible analyses, selecting a lexical head (1a), and a functional one (1b).

```
(1a) Sunday  (1b) on
on        poly
```

Similar choices are found in phrases which contain functional elements such as determiners, coordination markers, subordinating elements, etc.

**Multi-Headed Constructions.** Some phrases are considered to have multiple lexical heads, for instance coordinated structures. Since dependency-based formalisms require us to represent all content as binary relations, there are different ways we could represent such constructions. Take, for instance the coordination of nominals below. We can choose between a functional head (1a), a lexical head (2b)-(2c). We can further choose between a flat representation in which the first conjunct is a single head (2b), or a nested structure, where each conjunct/marker is the head of the following element (2c). All three alternatives empirically exist. Example (2a) reflects the structures in the CoNLL 2007 shared task data (Nivre et al., 2007), and Johansson and Nugues (2007) use structures like (2b). Example (2c) reflects a choice defined by Mel’čuk (1988).

```
(2a) and  (2b) earth (2c) earth

  cc coord  cc coord  coord
earth wind fire wind and fire

```

**Complex Determiners.** When noun phrases include complex determiners — such as the phrase “only three boys” below — there are different ways to consider the division into phrases, which dictate different arc structures and labeling of the depen-
dency trees. One way to represent the set of dependencies in this phrase is to view “only” as an adverbal head for the noun “three boys”, as in (3a).

A different way to view this phrase is to treat “only five” as a complex quantifier for which “boys” is a lexical head, and “only” modifies “three”, as in (3b).

\[(3a)\text{boys} \quad (3b)\text{only}
\]

In standard settings, two different experiments that use data sets that reflect different annotation decisions report results that are compared against their own annotation choices. In such cases the empirical results are not comparable across experiments.

A simple way to make results comparable across experiments is to evaluate all results against a single gold standard. In such cases, however, parses from different experiments will be penalized on learning specific annotation decisions. Consider, for instance, the example in table 1. If we choose gold1 as our overall gold standard, then parse2 results will be considered lower even though both parsers produced perfect output compared to their own gold data.

The study of (Schwartz et al., 2011) takes a different approach, and considers different directions of head-dependent relations as equivalent. However, even in this case, the error will be reflected in the attachment to the verbal head. For instance, for the phrase “arrive on sunday”, attachment-based evaluation will fail to match arrive→on against arrive→suniday. In the evaluation scheme of (Schwartz et al., 2011) this is fixed through letting a node pick a child or a grandchild as its dependent. However, neutralizing such decisions runs the risk of losing linguistically significant information.

The study of (Schwartz et al., 2011) takes a different approach, and considers different directions of head-dependent relations as equivalent. However, even in this case, the error will be reflected in the attachment to the verbal head. For instance, for the phrase “arrive on sunday”, attachment-based evaluation will fail to match arrive→on against arrive→suniday. In the evaluation scheme of (Schwartz et al., 2011) this is fixed through letting a node pick a child or a grandchild as its dependent. However, neutralizing such decisions runs the risk of losing linguistically significant information. For “I saw him leave”, for example, their procedure will fail to distinguish the erroneous (4a) from (4b).

\[(4a) \text{saw} \quad (4b) \text{said}
\]

Other methods proposed in the literature include heuristics for converting outputs into standard sets of dependencies (Cer et al., 2010) or task-based evaluation (Buyko and Hahn, 2010). However heuristic rules may become increasingly complex, may not cover all possible discrepancies up front, and may introduce annotation noise of their own. Task-based evaluation may be sensitive to the particular implementation of the embedding task. Beyond that, such heuristics and task-based procedures are developed almost exclusively for English. Other languages typically lack such resources. What we need is therefore a fully deterministic and formally precise strategy that can be applied to any set of annotated dependency trees, to consolidate the shared linguistic content in the different gold standards, and to compare parses against them through a sound metrics.

In order to tackle this challenge we require three fundamental components: (i) abstracting away from annotation peculiarities (here, specific bi-lexical attachments deterministically convert to functional tree representations), (ii) generalizing theory-specific structures into a single coherent gold standard (here, using unification-based operations to combine information from different treebanks), and (iii) defining a sound evaluation metric that takes into account not only the the gold standard and the parse trees, but also the different gold standards we started with (here, through tree edit-distances). The remainder of this paper details the components of a fully deterministic and sound end-to-end evaluation procedure that responds to these desiderata.

3 The Proposal: Cross-Annotation Evaluation in Three Simple Steps

In this section we fully elaborate on the three components necessary for implementing our proposed evaluation strategy. First, we define functional trees as the common space of formal objects, and define deterministic conversion procedures from sets of di-
rectional dependencies. Then we define a set of unification-based operations on functional trees that to compute, for every pair of corresponding trees with the same yield, a single structure that resolves inconsistencies among alternatives and combine the information they contain. Finally, we define scores based on tree edit-distance, refined to consider the distance from parses to the overall gold and from this gold set and to different annotation alternatives.

Formal Preliminaries. Let $T$ be a finite set of terminal symbols and let $L$ be a set of grammatical relation labels. A dependency graph $d$ is a directed graph which consists of nodes $V_d$ and arcs $A_d \subseteq V_d \times V_d$. We assume that all nodes in $V_d$ are labeled by terminal symbols, i.e., we assume a function $\text{label} : V_d \rightarrow T$. A well-formed dependency graph $d = (V_d, A_d)$ for a sentence $S = t_1, t_2, ..., t_n$ is any dependency graph that is a directed tree originating out of a node $v_0$ labeled $t_0 = \text{ROOT}$, and spans all terminals in the sentence, that is, $\{\text{label}(v) | v \in V_d\} = \{t_0, ..., t_n\}$. For simplicity, we always assume that a node $v$ is indexed according to the position of the terminal in $S$ that labels it, that is, for every $v_i \in V_d : \text{label}(v_i) = t_i$.

In a labeled dependency graph, arcs in $A_d$ are further labeled by elements of $L$, i.e., we assume a function $\text{label} : A_d \rightarrow L$ from arcs to the labels’ set and refer to an arc $(v_i, v_j)$ for which $\text{label}(v_i, v_j) = l$ as $l(v_i, v_j)$. When there is no confusion, we abbreviate $l(t_i, t_j)$ to refer to the label of an arc for which $\text{label}(v_i) = t_i$ and $\text{label}(v_j) = t_j$.

We further define two auxiliary functions on nodes in dependency trees. The function $\text{subtree} : V_d \rightarrow \mathcal{P}(V_d)$ assigns to each $v \in V_d$ the set of nodes that are dominated by it, where domination is the reflexive transitive closure of the arc relation $A_d$. The function $\text{span} : V_d \rightarrow T \cup \{\text{ROOT}\}$ assigns for each $v_i \in V_d$ a set of terminals such that $\text{span}(v_i) = \{\text{label}(v) | v \in \text{subtree}(v_i)\}$.

A dependency tree $d = (V_d, A_d)$ is projective if all arcs in it are projective. An arc $(v_i, v_j)$ is projective if there is a directed path from $v_i$ to all the nodes between the two endpoints of the arc. If $d$ is projective, than for all $v \in V_d$ it holds that the terminals in $\text{span}(v)$ form an ordered sequential segment of $S$.

For the purpose of the current discussion we assume that all trees contain projective dependencies only.

**Step 1: Functional Representation** Our first goal is to define a representation format that keeps all functional relationships that are represented in the dependency trees intact, but remains neutral with respect to the directionality of head-dependent relations. To do so we define functional trees — ordered labeled trees which, instead of head-to-head relations, represent the relations between the chunks they head. We can obtain functional trees from dependency trees using the following procedure:

- Relabel each node $v \in V_d$ with the label of its incoming arc.
  - $\text{relabel}(v) = l(u, v)$
- In case $|\text{span}(v)| > 1$ add a daughter designating the lexical head:
  - $V_d := V_d \cup \{u\}$
  - $A_d := A_d \cup \{(v, u)\}$
  - $\text{relabel}(u) = \ast$
- For each node $v$ such that $|\text{span}(v)| = 1$, add a daughter relabeled with its own terminal:
  - $V_d := V_d \cup \{u\}$
  - $A_d := A_d \cup \{(v, u)\}$
  - if $(\text{relabel}(v)! = \ast)$
    - $\text{relabel}(u) := \text{label}(v)$
  - else
    - $\text{relabel}(u) := \text{label}(\text{parent}(v))$

That is to say, we label all nodes with spans greater than 1 with the grammatical function of their head, and we add a node $u$ as a daughter designating the head itself. This gives us a constituency-like representation of the dependency trees, labeled with functional information, and realizing the linguistic assumptions reflected in the dependency trees. Heads are labeled as wildcards, which are compatible with any, more specific, functional characterization of their role inside of the phrase. So, examples (1)-(3) get transformed into (4a)-(4b) respectively.

\[
\begin{array}{c}
\text{prep} \\
\text{on} \quad \text{Sunday} \\
\end{array}
\quad ...
\quad \begin{array}{c}
\ast \\
\ast \\
\text{pobj} \\
\text{on} \quad \text{Sunday} \\
\end{array}
\]
Step 2: Formal Operations on Trees

The intu-...
Unlike unification, generalization can never fail. For every pair of trees there exists a tree that is more general than both: in the extreme case, pick the completely flat structure over the yield, which is more general than any other structure. For (6a)-(6b), we get then that \((6a) \cap (6b) = \ldots\)

\[
\begin{array}{ccc}
| & & * \\
adv & quant & \\
| & only & three \\
& boys & \\
\end{array}
\]

The generalization of two functional trees provides us with one structure that reflects the common and consistent content of the two trees. These structures thus provide us with a formally well-defined gold standard for the cross-treebank evaluation.

**Step 3: Measuring Distances** Figure 1 summarizes the steps in the evaluation procedure we defined so far. We start off with two versions of the treebank, TB1 and TB2, which are parsed separately and provide their own gold standards and parse hypotheses. After all structures have been converted into functional trees, we compute the generalization of the all versions of the gold functional trees for each sentence in the data, as the overall gold standard.

For \((6a)-(6b)\), we get then that \((6a) \cap (6b) = \ldots\)

\[
\begin{array}{ccc}
| & & * \\
adv & quant & \\
| & only & three \\
& boys & \\
\end{array}
\]

We assume that each operation is assigned a cost. Attachment scores of a dependency parse tree formally take the error to be the cost of all edit operations that are required to turn a parse tree into its gold standard. Turning the error rate into a measure of success requires normalizing the error with respect to the overall size of the dependency tree and subtracting it from a unity.

Here we apply the same idea, of defining error by tree edit distance and normalizing it relative to the fully ordered labeled trees.\(^2\) For \(\pi\) an ordered labeled tree, we can define the following operations:

- **relabel-node** change the label of node \(v\) in \(\pi\)
- **delete-node** delete a non-root node \(v\) in \(\pi\) with parent \(u\), making the children of \(v\) the children of \(u\), inserted in the place of \(v\) as a subsequence in the left-to-right order of the children of \(u\).
- **insert-node** insert a node \(v\) as a child of \(u\) in \(\pi\) making it the parent of a consecutive subsequence of the children of \(u\).

An edit script \(ES(\pi_1, \pi_2) = \{e_0, e_1, \ldots, e_k\}\) between \(\pi_1\) and \(\pi_2\) is a set of edit operations required for turning \(\pi_1\) into \(\pi_2\). Now, assume that we are given a cost function defined for each edit operation. The cost of \(ES(\pi_1, \pi_2)\) is the sum of the costs of the operations in the script. An optimal edit script is an edit script between \(\pi_1\) and \(\pi_2\) of minimum cost.

\[
ES^*(\pi_1, \pi_2) = \min_{ES(\pi_1, \pi_2)} \sum_{e \in ES(\pi_1, \pi_2)} \text{cost}(e)
\]

The tree edit distance problem is defined to be the problem of finding the optimal edit script and computing the corresponding distance (Bille, 2005).

A simple way to calculate the error of a parse \(\delta\) would be to define it as the edit distance between the parse hypothesis \(\pi_1\) and the gold standard \(\pi_2\).

\[
\delta(\pi_1, \pi_2) = \text{cost}(ES^*(\pi_1, \pi_2))
\]

The distances in figure 1 will then be as follows.

\[
\delta_{1-3} = \delta(\text{parse1}, \text{gold3})
\]

\[
\delta_{2-3} = \delta(\text{parse2}, \text{gold3})
\]

\(^2\)This idea can be used regardless of our pipeline, for better constituency-based metrics. This is left for future research.

---

\(1\)Generalization is an associative operation. Proof omitted.
However, in such cases the parser may still get penalized for assigning structure that is not a part of the generalization, as learned from the original treebank. To solve this, we refine the distance between the parse and the new gold tree to discard edit operations on substructures that are not reflected in the new gold standard. We record the operations in the edit script turning an old gold to the new gold that were used for calculating $\delta$, and discard their cost.

$$\delta_{\text{new}}(\text{parse}_1, \text{gold}_3) = \delta(\text{parse}_1, \text{gold}_3) - \text{cost}(ES^*(\text{parse}_1, \text{gold}_3) \cap ES^*(\text{gold}_1, \text{gold}_3))$$

Now, if gold1, gold2 and gold3 are identical, then $ES^*(\text{parse}_1, \text{gold}_3) \cap ES^*(\text{gold}_2, \text{gold}_3) = \emptyset$ and we fall back on simple tree edit distance $\delta_{\text{new}}(\text{parse}_1, \text{gold}_3) = \delta(\text{parse}_1, \text{gold}_3)$. When parse1 and gold1 are identical, i.e., the parser produced perfect output with respect to its own scheme, then $ES^*(\text{parse}_1, \text{gold}_3) = ES^*(\text{gold}_1, \text{gold}_3)$ and thus $\delta_{\text{new}}(\text{parse}_1, \text{gold}_3) = \delta(\text{parse}_1, \text{gold}_3) - \text{cost}(ES^*(\text{parse}_1, \text{gold}_1))$. Since by definition $\delta(\text{parse}_1, \text{gold}_3) = \text{cost}(ES^*(\text{parse}_1, \text{gold}_1))$, we get 0 distance, and the parser does not get penalized on recovering a correct structure lacking in gold3.

In order to turn distance measures into a measure of parse accuracy, we have to normalize it relative to the maximal number of operations that is conceivable. In the worst case scenario, we would remove all the internal nodes in the parse and add all the internal nodes into the new gold, so our normalization factor $\iota$ is defined as $\iota(\text{parse}_1, \text{gold}_3) = |\text{parse}_1| + |\text{gold}_3| - 2(|\text{span}(|\text{gold}_3|) + 1)$. We can now score parse1 relative to the new gold as:

$$\text{score}(\text{parse}_1, \text{gold}_3) = 1 - \frac{\delta_{\text{new}}(\text{parse}_1, \text{gold}_3)}{\iota(\text{parse}_1, \text{gold}_3)}$$

To normalize the scores over the parses in an entire test set of size $n$ we can take a mathematical average of these scores for each sentence in the test data:

$$\frac{\sum_{i=1}^{[\text{gold}_3]} \text{score}(\text{parse}_1, \text{gold}_3)}{[\text{gold}_3]}$$

We can alternatively perform macro averaging of all edit distance costs, normalized by the maximally possible edits on the data. This is the metric we use.

$$1 - \frac{\sum_{i=1}^{[\text{gold}_3]} \delta_{\text{new}}(\text{parse}_1, \text{gold}_3)}{\sum_{i=1}^{[\text{gold}_3]} \iota(\text{parse}_1, \text{gold}_3)}$$

4 Experiments

We demonstrate the application of our evaluation procedure to comparing dependency parsing results on the Penn Treebank (PTB) using different conversions of the PTB into dependency structures. We obtain the different conversions using the LTH software, a general purpose tool for constituency-to-dependency conversion (Johansson and Nugues, 2007). We used five different conversion settings:

- **Default** the LTH conversion default settings
- **OldLTH** the conversion used in Johansson and Nugues (2007)
- **CoNLL** the conversion used in the CoNLL shared task (Nivre et al., 2007)
- **Lexical** Same as CoNLL, but selecting only lexical heads when a choice exists
- **Functional** Same as CoNLL, but selecting only functional heads when a choice exists
The values for the different settings are elaborated in the supplementary material. Briefly, the Default, OldLTH and CoNLL differ in particular in their coordination structure, and the Functional vs. Lexical conversions differ in their selection of a head.

We used MaltParser as optimized for English in (Nivre et al., 2010) and trained on sections 2-21 of the PTB to parse section 23. We converted all parse and gold files into functional trees, and, for each set of parsing experiments that are evaluated with respect to one another, we calculated the new gold standard using generalization. We evaluated the experiments using labeled and unlabeled attachment scores relative to the native gold standard, and used TED-based macro averages for intra- and cross-experimental evaluation. Our TED implementation builds on the algorithm of (Zhang and Shasha, 1989). We report LAS and macro average of simple TED between each parse tree and a single gold standard using generalization. We evaluated the experiments using labeled and unlabeled attachment scores relative to the native gold standard, and used TED-based macro averages for intra- and cross-experimental evaluation. Our TED implementation builds on the algorithm of (Zhang and Shasha, 1989). We report LAS and macro average of simple TED between each parse tree and a single gold standard. We report the new TED metrics for sets of experiments compared to one another.4 We include a sample of all files and a printout of the TED output in our supplementary material.

The left hand side of table 2 presents our empirical results for our first set of experiments, in which we compare parsing results of the Default, OldLTH and CoNLL conversions. corresponding to annotation schemes used in the past to parse the PTB.

The first three rows we compare results of parsed dependencies trained on one format against gold dependency trees from its own or a different format. For all of the proposed metrics, evaluating against a gold standard that does not share its native annotation causes a dramatic decrease in performance. This suggests that learning annotation peculiarities plays a great role in obtaining good parsing result on the simple traditional pipeline. However, does this decrease reflect a true gap in parsing performance? Clearly, we cannot conclude that, because we are in effect comparing apples to oranges.

In the next three rows we report the results for comparing all pair-wise sets of experiments using our proposed procedure. Here we see that the TED-based scores on the different trees in the data reflect very similar results for the experiments comparing CoNLL to the default LTH settings. There is still an observable gap in parser performance between Old LTH vs. ConLL, and Old LTH vs. default. When comparing the results of individual experiments, it looked as though Default is better than OldLTH, which is better than CoNLL. But looking at the intersected scores, it is actually the other way around (although differences are small). This suggests that apparent improvements, when only looking at native scores, can actually be misleading.

In the last raw we compare all parsing results against a single gold representation that we obtained by generalizing the three different conversions for-

<table>
<thead>
<tr>
<th>Parse</th>
<th>Gold</th>
<th>default</th>
<th>old LTH</th>
<th>CoNLL/07</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>UAS</td>
<td>0.9142</td>
<td>0.6077</td>
<td>0.7772</td>
</tr>
<tr>
<td></td>
<td>LAS</td>
<td>0.8820</td>
<td>0.4801</td>
<td>0.6454</td>
</tr>
<tr>
<td></td>
<td>U-TED</td>
<td>0.9498</td>
<td>0.8686</td>
<td>0.9163</td>
</tr>
<tr>
<td></td>
<td>L-TED</td>
<td>0.9266</td>
<td>0.7672</td>
<td>0.8467</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parse</th>
<th>Gold</th>
<th>default</th>
<th>old LTH</th>
<th>CoNLL/07</th>
</tr>
</thead>
<tbody>
<tr>
<td>old LTH</td>
<td>UAS</td>
<td>0.8653</td>
<td>0.8955</td>
<td>0.6508</td>
</tr>
<tr>
<td></td>
<td>LAS</td>
<td>0.4816</td>
<td>0.8644</td>
<td>0.5771</td>
</tr>
<tr>
<td></td>
<td>U-TED</td>
<td>0.8690</td>
<td>0.9496</td>
<td>0.8889</td>
</tr>
<tr>
<td></td>
<td>L-TED</td>
<td>0.7674</td>
<td>0.9259</td>
<td>0.8019</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parse</th>
<th>Gold</th>
<th>default</th>
<th>old LTH</th>
<th>CoNLL/07</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoNLL/07</td>
<td>UAS</td>
<td>0.7754</td>
<td>0.6474</td>
<td>0.8917</td>
</tr>
<tr>
<td></td>
<td>LAS</td>
<td>0.6479</td>
<td>0.5722</td>
<td>0.8736</td>
</tr>
<tr>
<td></td>
<td>U-TED</td>
<td>0.9180</td>
<td>0.8893</td>
<td>0.9450</td>
</tr>
<tr>
<td></td>
<td>L-TED</td>
<td>0.8498</td>
<td>0.8014</td>
<td>0.9223</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parse</th>
<th>Gold</th>
<th>default-oldLTH generalization</th>
<th>functional/generalization</th>
<th>Lexical/generalization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>U-TED</td>
<td>0.9422</td>
<td>0.9484</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>L-TED</td>
<td>0.9198</td>
<td>0.9224</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parse</th>
<th>Gold</th>
<th>default-OldLTH generalization</th>
<th>CoNLL-Lexical/generalization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>U-TED</td>
<td>0.80450</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>L-TED</td>
<td>0.9280</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parse</th>
<th>Gold</th>
<th>default-oldLTH-CoNLL generalization</th>
<th>Functional-Lexical/generalization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>U-TED</td>
<td>0.9411</td>
<td>0.9439</td>
</tr>
<tr>
<td></td>
<td>L-TED</td>
<td>0.9236</td>
<td>0.9240</td>
</tr>
</tbody>
</table>

Table 2: Cross-experimental dependency parsing evaluation for multiple conversion alternatives. The conversion settings are detailed in the supplementary material. We report standard LAS scores and TED macro-average metrics.

4Unlabeled TED scores are calculated like labeled scores except that the cost of the “relabling” operation is defined as 0.
mats. Here we see that all different results are on a par with one another, with very minor gaps in performance, if any. This last observation suggests that annotational variations and idiosyncrasies that may have crucial effects when parses are compared against their own native annotation standard, in effect may have a lot less influence on parsing results once we bring different data into the same formal and theoretical common ground for comparison.

The right hand side of our results table reports results for a parallel set of experiments, now considering the effects of different linguistic conversion choices on parsing accuracy — the Functional data consistently selects functional elements as head, and the Lexical data selects lexical heads consistently. Here, again, the comparison between parses and non-native gold standard trees shows significant differences, while applying our cross-experimental procedure shows very little to no gaps in performance performance across the same experiments.

It is striking that even for such fundamentally different annotation choices, from a linguistic point of view, empirically there is almost no difference in parsing results. This suggests that even when linguistically profound alternatives express the same predicate-argument content, tuning the annotation of dependencies is not necessarily crucial for obtaining good results on the set of grammatical relations.

5 Discussion and Future Work

This paper addressed the problem of cross-experiment evaluation. As it turns out, this problem arises in CL/NLP in different shapes and forms; when comparing the same parser on different annotation schemes, when comparing parsers from different formalisms, and when comparing parser performance across different languages.

We consider our contribution successful if after reading it the reader develops a healthy suspicion to blunt comparison of numbers across experiments, or better yet, across different papers. Cross experimental comparison should be a careful and well thought through endeavor, in which we retain as much information as we can from our data and parsed structures, avoid lossy conversions, and focus on an object of evaluation which is the pertinent linguistic content for the task, and shared by all variants.

Our proposal introduces one way to do so in a streamlined, formally worked out and empirically robust way. While individual components may be further improved, the proposed setup and implementation can be straightforwardly applied to cross-parser and cross-framework comparison. (A conversion from constituency-based trees or LFG-based trees into functional trees is straightforward to define: simply replace the node labels with the grammatical function of their dominating arc – and the rest of the pipeline follows).

A pre-condition for such cross-framework evaluation is that all representations realize the same set of grammatical relations, either by, e.g., annotating arcs in dependency trees or by decorating nodes in constituency trees. For some treebanks this is already the case, (Nivre and Megyesi, 2007; Skut et al., 1997; Hinrichs et al., 2004) while for others this is still lacking. However, research on extracting typed dependencies (de Marneffe et al., 2006) and proposals for relational cross-parser evaluation (Briscoe et al., 2002) suggest that evaluation through a single set of relations as the common denominator is a linguistically sound and practically useful way to go. If we can develop resources that retain the same set of functions for multiple representation schemes, this would be a fruitful path to explore.

In the future we plan to use this pipeline for comparison of constituency and dependency parsers, for English and other languages. We further intend to inquire whether we can find a set of abstract grammatical relations for evaluating structures in multiple languages. Such inquiries may pave the way for cross-language parser evaluation.

6 Conclusion

We presented an end-to-end procedure for faithfully comparing dependency parsing results across experiments. We described the three fundamental components in the pipeline: (i) neutralizing representation peculiarities, (ii) formal operations for harmonizing information from different sources, and (iii) distance-based metric that consider multiple sources. When applied to parsing results of different dependency schemes, dramatic gaps observed in comparing results obtained in isolation decrease or aldißolve when using our proposed pipeline.
References


