Cross-Framework Evaluation for Statistical Parsing

Reut Tsarfaty   Joakim Nivre   Evelina Andersson
Uppsala University, Box 635, 75126 Uppsala, Sweden
tsarfaty@stp.lingfil.uu.se,{joakim.nivre,evelina.andersson}@lingfil.uu.se

Abstract

A serious bottleneck of comparative parser evaluation is the fact that different parsers subscribe to different formal frameworks and theoretical assumptions. Converting outputs from one framework to another is less than optimal as it easily introduces noise into the process. Here we present a principled protocol for evaluating parsing results across frameworks based on function trees, tree generalization and edit distance metrics. This extends a previously proposed framework for cross-theory evaluation and allows us to compare a wider class of parsers. We demonstrate the usefulness and language independence of our procedure by evaluating constituency and dependency parsers on English and Swedish.

1 Introduction

The goal of statistical parsers is to recover a formal representation of the grammatical relations that constitute the argument structure of natural language sentences. The argument structure encompasses grammatical relationships between elements such as *subject, predicate, object*, etc., which are useful for further (e.g., semantic) processing. The parses yielded by different parsing frameworks typically obey different formal and theoretical assumptions concerning how to represent the grammatical relationships in the data (Rambow, 2010). For example, grammatical relations may be encoded on top of dependency arcs in a dependency tree (Mel'čuk, 1988), they may decorate nodes in a phrase-structure tree (Marcus et al., 1993; Maamouri et al., 2004; Sima'an et al., 2001), or they may be read off of positions in a phrase-structure tree using hard-coded conversion procedures (de Marneffe et al., 2006). This diversity poses a challenge to cross-experimental parser evaluation, namely: How can we evaluate the performance of these different parsers relative to one another?

Current evaluation practices assume a set of correctly annotated test data (or gold standard) for evaluation. Typically, every parser is evaluated with respect to its own formal representation type and the underlying theory which it was trained to recover. Therefore, numerical scores of parses across experiments are incomparable. When comparing parses that belong to different formal frameworks, the notion of a single gold standard becomes problematic, and there are two different questions we have to answer. First, what is an appropriate gold standard for cross-parser evaluation? And secondly, how can we alleviate the differences between formal representation types and theoretical assumptions in order to make our comparison sound – that is, to make sure that we are not comparing apples and oranges?

A popular way to address this has been to pick one of the frameworks and convert all parser outputs to its formal type. When comparing constituency-based and dependency-based parsers, for instance, the output of constituency parsers has often been converted to dependency structures prior to evaluation (Cer et al., 2010; Nivre et al., 2010). This solution has various drawbacks. First, it demands a conversion script that maps one representation type to another when some theoretical assumptions in one framework may be incompatible with the other one. In the constituency-to-dependency case, some constituency-based structures (e.g., coordination

Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, pages 44–54,
Avignon, France, April 23 - 27 2012. ©2012 Association for Computational Linguistics

and ellipsis) do not comply with the single head assumption of dependency treebanks. Secondly, these scripts may be labor intensive to create, and are available mostly for English. So the evaluation protocol becomes language-dependent.

In Tsarfaty et al. (2011) we proposed a general protocol for handling annotation discrepancies when comparing parses across different dependency theories. The protocol consists of three phases: converting all structures into function trees, for each sentence, generalizing the different gold standard function trees to get their common denominator, and employing an evaluation measure based on tree edit distance (TED) which discards edit operations that recover theory-specific structures. Although the protocol is potentially applicable to a wide class of syntactic representation types, formal restrictions in the procedures effectively limit its applicability only to representations that are isomorphic to dependency trees.

The present paper breaks new ground in the ability to soundly compare the accuracy of different parsers relative to one another given that they employ different formal representation types and obey different theoretical assumptions. Our solution generally confines with the protocol proposed in Tsarfaty et al. (2011) but is re-formalized to allow for arbitrary linearly ordered labeled trees, thus encompassing constituency-based as well as dependency-based representations. The framework in Tsarfaty et al. (2011) assumes structures that are isomorphic to dependency trees, bypassing the problem of arbitrary branching. Here we lift this restriction, and define a protocol which is based on generalization and TED measures to soundly compare the output of different parsers.

We demonstrate the utility of this protocol by comparing the performance of different parsers for English and Swedish. For English, our parser evaluation across representation types allows us to analyze and precisely quantify previously encountered performance tendencies. For Swedish we show the first ever evaluation between dependency-based and constituency-based parsing models, all trained on the Swedish treebank data. All in all we show that our extended protocol, which can handle linearly-ordered labeled trees with arbitrary branching, can soundly compare parsing results across frameworks in a representation-independent and language-independent fashion.

2 Preliminaries: Relational Schemes for Cross-Framework Parse Evaluation

Traditionally, different statistical parsers have been evaluated using specially designated evaluation measures that are designed to fit their representation types. Dependency trees are evaluated using attachment scores (Buchholz and Marsi, 2006), phrase-structure trees are evaluated using ParsEval (Black et al., 1991), LFG-based parsers postulate an evaluation procedure based on f-structures (Cahill et al., 2008), and so on. From a downstream application point of view, there is no significance as to which formalism was used for generating the representation and which learning methods have been utilized. The bottom line is simply which parsing framework most accurately recovers a useful representation that helps to unravel the human-perceived interpretation.

Relational schemes, that is, schemes that encode the set of grammatical relations that constitute the predicate-argument structures of sentences, provide an interface to semantic interpretation. They are more intuitively understood than, say, phrase-structure trees, and thus they are also more useful for practical applications. For these reasons, relational schemes have been repeatedly singled out as an appropriate level of representation for the evaluation of statistical parsers (Lin, 1995; Carroll et al., 1998; Cer et al., 2010).

The annotated data which statistical parsers are trained on encode these grammatical relationships in different ways. Dependency treebanks provide a ready-made representation of grammatical relations on top of arcs connecting the words in the sentence (Kübler et al., 2009). The Penn Treebank and phrase-structure annotated resources encode partial information about grammatical relations as dash-features decorating phrase structure nodes (Marcus et al., 1993). Treebanks like Tiger for German (Brants et al., 2002) and Talbanken for Swedish (Nivre and Megyesi, 2007) explicitly map phrase structures onto grammatical relations using dedicated edge labels. The Relational-Realizational structures of Tsarfaty and Sima'an (2008) encode relational networks (sets of relations) projected and realized by syntactic categories on top of ordinary phrase-structure nodes.

Function trees, as defined in Tsarfaty et al. (2011), are linearly ordered labeled trees in which every node is labeled with the grammatical func-

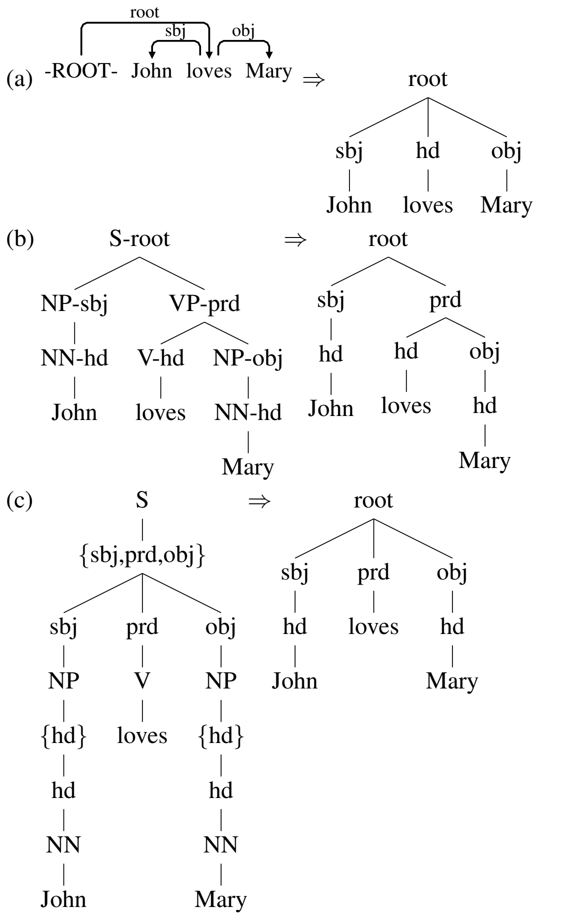

Figure 1: Deterministic conversion into function trees. The algorithm for extracting a function tree from a dependency tree as in (a) is provided in Tsarfaty et al. (2011). For a phrase-structure tree as in (b) we can replace each node label with its function (dash-feature). In a relational-realizational structure like (c) we can remove the projection nodes (sets) and realization nodes (phrase labels), which leaves the function nodes intact.

tion of the dominated span. Function trees benefit from the same advantages as other relational schemes, namely that they are intuitive to understand, they provide the interface for semantic interpretation, and thus may be useful for downstream applications. Yet they do not suffer from formal restrictions inherent in dependency structures, for instance, the single head assumption.

For many formal representation types there exists a fully deterministic, heuristics-free, procedure mapping them to function trees. In Figure 1 we illustrate some such procedures for a simple transitive sentence. Now, while all the structures at the right hand side of Figure 1 are of the same formal type (function trees), they have different tree structures due to different theoretical assumptions underlying the original formal frameworks.

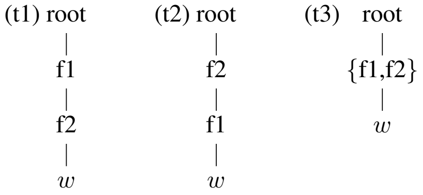

Figure 2: Unary chains in function trees

Once we have converted framework-specific representations into function trees, the problem of cross-framework evaluation can potentially be reduced to a cross-theory evaluation following Tsarfaty et al. (2011). The main idea is that once all structures have been converted into function trees, one can perform a formal operation called generalization in order to harmonize the differences between theories, and measure accurately the distance of parse hypotheses from the generalized gold. The generalization operation defined in Tsarfaty et al. (2011), however, cannot handle trees that may contain unary chains, and therefore cannot be used for arbitrary function trees.

Consider for instance (t1) and (t2) in Figure 2. According to the definition of subsumption in Tsarfaty et al. (2011), (t1) is subsumed by (t2) and vice versa, so the two trees should be identical – but they are not. The interpretation we wish to give to a function tree such as (t1) is that the word w has both the grammatical function f1 and the grammatical function f2. This can be graphically represented as a set of labels dominating w, as in (t3). We call structures such as (t3) *multi-function trees*. In the next section we formally define multi-function trees, and then use them to develop our protocol for cross-framework and cross-theory evaluation.

3 The Proposal: Cross-Framework Evaluation with Multi-Function Trees

Our proposal is a three-phase evaluation protocol in the spirit of Tsarfaty et al. (2011). First, we obtain a *formal* common ground for all frameworks in terms of multi-function trees. Then we obtain a *theoretical* common ground by means of tree-generalization on gold trees. Finally, we calculate TED-based scores that discard the cost of annotation-specific edits. In this section, we define multi-function trees and update the tree-generalization and TED-based metrics to handle multi-function trees that reflect different theories.

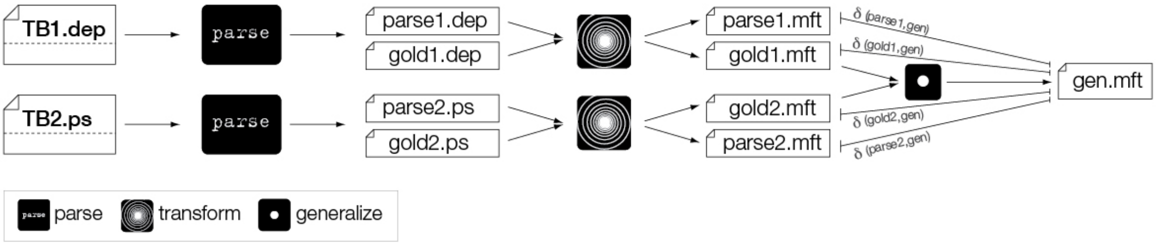

Figure 3: **The Evaluation Protocol.** Different formal frameworks yield different parse and gold formal types. All types are transformed into multi-function trees. All gold trees enter generalization to yield a new gold for each sentence. The different δ arcs represent the different edit scripts used for calculating the TED-based scores.

3.1 Defining Multi-Function Trees

An ordinary function tree is a linearly ordered tree $T = (V, A)$ with yield $w_1, ..., w_n$, where internal nodes are labeled with grammatical function labels drawn from some set $\mathcal{L}$. We use $span(v)$ and $label(v)$ to denote the yield and label, respectively, of an internal node v. A multi-function tree is a linearly ordered tree $T = (V, A)$ with yield $w_1, ..., w_n$, where internal nodes are labeled with *sets of* grammatical function labels drawn from $\mathcal{L}$ and where $v \neq v'$ implies $span(v) \neq span(v')$ for all internal nodes v, v'. We use $labels(v)$ to denote the label set of an internal node v.

We interpret multi-function trees as encoding sets of functional constraints over spans in function trees. Each node v in a multi-function tree represents a constraint of the form: for each $l \in labels(v)$, there should be a node v' in the function tree such that $span(v) = span(v')$ and $label(v') = l$. Whenever we have a conversion for function trees, we can efficiently collapse them into multi-function trees with no unary productions, and with label sets labeling their nodes. Thus, trees (t1) and (t2) in Figure 2 would both be mapped to tree (t3), which encodes the functional constraints encoded in either of them.

For dependency trees, we assume the conversion to function trees defined in Tsarfaty et al. (2011), where head daughters always get the label 'hd'. For PTB style phrase-structure trees, we replace the phrase-structure labels with functional dash-features. In relational-realization structures we remove projection and realization nodes. Deterministic conversions exist also for Tiger style treebanks and frameworks such as LFG, but we do not discuss them here.[1]

3.2 Generalizing Multi-Function Trees

Once we obtain multi-function trees for all the different gold standard representations in the system, we feed them to a generalization operation as shown in Figure 3. The goal of this operation is to provide a consensus gold standard that captures the linguistic structure that the different gold theories agree on. The generalization structures are later used as the basis for the TED-based evaluation. Generalization is defined by means of subsumption. A multi-function tree subsumes another one if and only if all the constraints defined by the first tree are also defined by the second tree. So, instead of demanding equality of labels as in Tsarfaty et al. (2011), we demand set inclusion:

> **T-Subsumption**, denoted $\sqsubseteq_t$, is a relation between multi-function trees that indicates that a tree π_1 is consistent with and more general than tree π_2. Formally: $\pi_1 \sqsubseteq_t \pi_2$ iff for every node $n \in \pi_1$ there exists a node $m \in \pi_2$ such that $span(n) = span(m)$ and $labels(n) \subseteq labels(m)$.
>
> **T-Unification**, denoted $\sqcup_t$, is an operation that returns the most general tree structure that contains the information from both input trees, and fails if such a tree does not exist. Formally: $\pi_1 \sqcup_t \pi_2 = \pi_3$ iff $\pi_1 \sqsubseteq_t \pi_3$ and $\pi_2 \sqsubseteq_t \pi_3$, and for all π_4 such that $\pi_1 \sqsubseteq_t \pi_4$ and $\pi_2 \sqsubseteq_t \pi_4$ it holds that $\pi_3 \sqsubseteq_t \pi_4$.
>
> **T-Generalization**, denoted $\sqcap_t$, is an operation that returns the most specific tree that is more general than both trees. Formally, $\pi_1 \sqcap_t \pi_2 = \pi_3$ iff $\pi_3 \sqsubseteq_t \pi_1$ and $\pi_3 \sqsubseteq_t \pi_2$, and for every π_4 such that $\pi_4 \sqsubseteq_t \pi_1$ and $\pi_4 \sqsubseteq_t \pi_2$ it holds that $\pi_4 \sqsubseteq_t \pi_3$.

The generalization tree contains all nodes that exist in both trees, and for each node it is labeled by

[1] All the conversions we use are deterministic and are defined in graph-theoretic and language-independent terms. We make them available at `http://stp.lingfil.uu.se/~tsarfaty/unipar/index.html`.

the intersection of the label sets dominating the same span in both trees. The unification tree contains nodes that exist in one tree or another, and for each span it is labeled by the union of all label sets for this span in either tree. If we generalize two trees and one tree has no specification for labels over a span, it does not share anything with the label set dominating the same span in the other tree, and the label set dominating this span in the generalized tree is empty. If the trees do not agree on any label for a particular span, the respective node is similarly labeled with an empty set. When we wish to unify theories, then an empty set over a span is unified with any other set dominating the same span in the other tree, without altering it.

Digression: Using Unification to Merge Information From Different Treebanks In Tsarfaty et al. (2011), only the generalization operation was used, providing the common denominator of all the gold structures and serving as a common ground for evaluation. The unification operation is useful for other NLP tasks, for instance, combining information from two different annotation schemes or enriching one annotation scheme with information from a different one. In particular, we can take advantage of the new framework to enrich the node structure reflected in one theory with grammatical functions reflected in an annotation scheme that follows a different theory. To do so, we define the Tree-Labeling-Unification operation on multi-function trees.

> **TL-Unification**, denoted $\sqcup_{tl}$, is an operation that returns a tree that retains the structure of the first tree and adds labels that exist over its spans in the second tree. Formally: $\pi_1 \sqcup_{tl} \pi_2 = \pi_3$ iff for every node $n \in \pi_1$ there exists a node $m \in \pi_3$ such that $span(m) = span(n)$ and $labels(m) = labels(n) \cup labels(\pi_2, span(n))$.

Where $labels(\pi_2, span(n))$ is the set of labels of the node with yield $span(n)$ in π_2 if such a node exists and $\emptyset$ otherwise. We further discuss the TL-Unification and its use for data preparation in §4.

3.3 TED Measures for Multi-Function Trees

The result of the generalization operation provides us with multi-function trees for each of the sentences in the test set representing sets of constraints on which the different gold theories agree. We would now like to use distance-based metrics in order to measure the gap between the gold and predicted theories. The idea behind distance-based evaluation in Tsarfaty et al. (2011) is that recording the edit operations between the native gold and the generalized gold allows one to discard their cost when computing the cost of a parse hypothesis turned into the generalized gold. This makes sure that different parsers do not get penalized, or favored, due to annotation specific decisions that are not shared by other frameworks.

The problem is now that TED is undefined with respect to multi-function trees because it cannot handle complex labels. To overcome this, we convert multi-function trees into *sorted function trees*, which are simply function trees in which any label set is represented as a unary chain of single-labeled nodes, and the nodes are sorted according to the canonical order of their labels.[2] In case of an empty set, a 0-length chain is created, that is, no node is created over this span. Sorted function trees prevent reordering nodes in a chain in one tree to fit the order in another tree, since it would violate the idea that the set of constraints over a span in a multi-function tree is unordered.

The edit operations we assume are **add-node**(l, i, j) and **delete-node**(l, i, j) where $l \in \mathcal{L}$ is a grammatical function label and $i < j$ define the span of a node in the tree. Insertion into a unary chain must confine with the canonical order of the labels. Every operation is assigned a cost. An edit script is a sequence of edit operations that turns a function tree π_1 into π_2, that is:

$$ES(\pi_1, \pi_2) = \langle e_1, \ldots, e_k \rangle$$

Since all operations are anchored in spans, the sequence can be determined to have a unique order of traversing the tree (say, DFS). Different edit scripts then only differ in their set of operations on spans. The edit distance problem is finding the minimal cost script, that is, one needs to solve:

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

In the current setting, when using only add and delete operations on spans, there is only one edit script that corresponds to the minimal edit cost. So, finding the minimal edit script entails finding a single set of operations turning π_1 into π_2.

[2] The ordering can be alphabetic, thematic, etc.

We can now define δ for the ith framework, as the error of $parse_i$ relative to its native gold standard $gold_i$ and to the generalized gold *gen*. This is the edit cost minus the cost of the script turning $parse_i$ into *gen* intersected with the script turning $gold_i$ into *gen*. The underlying intuition is that if an operation that was used to turn $parse_i$ into *gen* is used to discard theory-specific information from $gold_i$, its cost should not be counted as error.

$$\delta(parse_i, gold_i, gen) = cost(ES^*(parse_i, gen))$$
$$-cost(ES^*(parse_i, gen) \cap ES^*(\text{gold}_i, \text{gen}))$$

In order to turn distance measures into parse-scores we now normalize the error relative to the size of the trees and subtract it from a unity. So the **Sentence Score** for parsing with framework i is:

$$score(parse_i, gold_i, gen) =$$
$$1 - \frac{\delta(parse_i, gold_i, gen)}{|parse_i| + |gen|}$$

Finally, **Test-Set Average** is defined by macro-avaraging over all sentences in the test-set:

$$1 - \frac{\sum_{j=1}^{|\text{testset}|} \delta(parse_{ij}, gold_{ij}, gen_j)}{\sum_{j=1}^{|\text{testset}|} |parse_{ij}| + |gen_j|}$$

This last formula represents the TEDEVAL metric that we use in our experiments.

A Note on System Complexity Conversion of a dependency or a constituency tree into a function tree is linear in the size of the tree. Our implementation of the generalization and unification operation is an exact, greedy, chart-based algorithm that runs in polynomial time ($O(n^2)$ in n the number of terminals). The TED software that we utilize builds on the TED efficient algorithm of Zhang and Shasha (1989) which runs in $O(|T_1||T_2| \min(d_1, n_1) \min(d_2, n_2))$ time where d_i is the tree degree (depth) and n_i is the number of terminals in the respective tree (Bille, 2005).

4 Experiments

We validate our cross-framework evaluation procedure on two languages, English and Swedish. For English, we compare the performance of two dependency parsers, MaltParser (Nivre et al., 2006) and MSTParser (McDonald et al., 2005), and two constituency-based parsers, the Berkeley parser (Petrov et al., 2006) and the Brown parser (Charniak and Johnson, 2005). All experiments use Penn Treebank (PTB) data. For Swedish, we compare MaltParser and MSTParser with two variants of the Berkeley parser, one trained on phrase structure trees, and one trained on a variant of the Relational-Realizational representation of Tsarfaty and Sima'an (2008). All experiments use the Talbanken Swedish Treebank (STB) data.

4.1 English Cross-Framework Evaluation

We use sections 02–21 of the WSJ Penn Treebank for training and section 00 for evaluation and analysis. We use two different native gold standards subscribing to different theories of encoding grammatical relations in tree structures:

◦ THE DEPENDENCY-BASED THEORY is the theory encoded in the basic Stanford Dependencies (SD) scheme. We obtain the set of basic stanford dependency trees using the software of de Marneffe et al. (2006) and train the dependency parsers directly on it.

◦ THE CONSTITUENCY-BASED THEORY is the theory reflected in the phrase-structure representation of the PTB (Marcus et al., 1993) enriched with function labels compatible with the Stanford Dependencies (SD) scheme. We obtain trees that reflect this theory by TL-Unification of the PTB multi-function trees with the SD multi-function trees (PTB$\sqcup_{tl}$SD) as illustrated in Figure 4.

The theory encoded in the multi-function trees corresponding to SD is different from the one obtained by our TL-Unification, as may be seen from the difference between the flat SD multi-function tree and the result of the PTB$\sqcup_{tl}$SD in Figure 4. Another difference concerns coordination structures, encoded as binary branching trees in SD and as flat productions in the PTB$\sqcup_{tl}$SD. Such differences are not only observable but also quantifiable, and using our redefined TED metric the cross-theory overlap is 0.8571.

The two dependency parsers were trained using the same settings as in Tsarfaty et al. (2011), using SVMTool (Giménez and Màrquez, 2004) to predict part-of-speech tags at parsing time. The two constituency parsers were used with default settings and were allowed to predict their own part-of-speech tags. We report three different evaluation metrics for the different experiments:

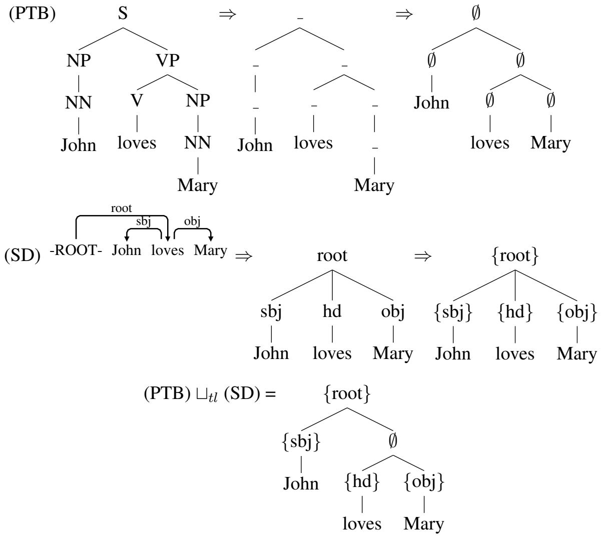

Figure 4: Conversion of PTB and SD tree to multi-function trees, followed by TL-Unification of the trees. Note that some PTB nodes remain without an SD label.

- ○ LAS/UAS (Buchholz and Marsi, 2006)
- ○ PARSEVAL (Black et al., 1991)
- ○ TEDEVAL as defined in Section 3

We use LAS/UAS for dependency parsers that were trained on the same dependency theory. We use ParseEval to evaluate phrase-structure parsers that were trained on PTB trees in which dash-features and empty traces are removed. We use our implementation of TEDEVAL to evaluate parsing results across all frameworks under two different scenarios:[3] TEDEVAL SINGLE evaluates against the native gold multi-function trees. TEDEVAL MULTIPLE evaluates against the generalized (cross-theory) multi-function trees. Unlabeled TEDEVAL scores are obtained by simply removing all labels from the multi-function nodes, and using unlabeled edit operations. We calculate pairwise statistical significance using a shuffling test with 10K iterations (Cohen, 1995).

Tables 1 and 2 present the results of our cross-framework evaluation for English Parsing. In the left column of Table 1 we report ParsEval scores for constituency-based parsers. As expected, F-Scores for the Brown parser are higher than the F-Scores of the Berkeley parser. F-Scores are however not applicable across frameworks. In the rightmost column of Table 1 we report the LAS/UAS results for all parsers. If a parser yields a constituency tree, it is converted to and evaluated on SD. Here we see that MST outperforms Malt, though the differences for labeled dependencies are insignificant. We also observe here a familiar pattern from Cer et al. (2010) and others, where the constituency parsers significantly outperform the dependency parsers after conversion of their output into dependencies.

The conversion to SD allows one to compare results across formal frameworks, but not without a cost. The conversion introduces a set of annotation specific decisions which may introduce a bias into the evaluation. In the middle column of Table 1 we report the TEDEVAL metrics measured against the generalized gold standard for all parsing frameworks. We can now confirm that the constituency-based parsers significantly outperform the dependency parsers, and that this is not due to specific theoretical decisions which are seen to affect LAS/UAS metrics (Schwartz et al., 2011). For the dependency parsers we now see that Malt outperforms MST on labeled dependencies slightly, but the difference is insignificant.

The fact that the discrepancy in theoretical assumptions between different frameworks indeed affects the conversion-based evaluation procedure is reflected in the results we report in Table 2. Here the leftmost and rightmost columns report TEDEVAL scores against the own native gold (SINGLE) and the middle column against the generalized gold (MULTIPLE). Had the theories for SD and PTB$\sqcup_{tl}$SD been identical, TEDEVAL SINGLE and TEDEVAL MULTIPLE would have been equal in each line. Because of theoretical discrepancies, we see small gaps in parser performance between these cases. Our protocol ensures that such discrepancies do not bias the results.

4.2 Cross-Framework Swedish Parsing

We use the standard training and test sets of the Swedish Treebank (Nivre and Megyesi, 2007) with two gold standards presupposing different theories:

- THE DEPENDENCY-BASED THEORY is the dependency version of the Swedish Treebank. All trees are projectivized (STB-Dep).
- THE CONSTITUENCY-BASED THEORY is the standard Swedish Treebank with grammatical function labels on the edges of constituency structures (STB).

[3] Our TedEval software can be downloaded at `http://stp.lingfil.uu.se/~tsarfaty/unipar/download.html`.

Formalism	**PS Trees**	**MF Trees**	**Dep Trees**
Theory	**PTB $\sqcup_{\mathrm{lt}}$ SD**	**(PTB $\sqcup_{\mathrm{lt}}$ SD) $\sqcap_{\mathrm{t}}$ SD**	**SD**
Metrics	ParsEval	TedEval	AttScores
Malt	N/A	U: 0.9525 L: *0.9088*	U: 0.8962 L: 0.8772
MST	N/A	U: *0.9549* L: 0.9049	U: *0.9059* L: *0.8795*
Berkeley	F-Scores 0.9096	U: 0.9677 L: 0.9227	U: 0.9254 L: 0.9031
Brown	F-Scores **0.9129**	U: **0.9702** L: **0.9264**	U: **0.9289** L: **0.9057**

Table 1: English cross-framework evaluation: Three measures as applicable to the different schemes. Boldface scores are highest in their column. Italic scores are the highest for dependency parsers in their column.

Formalism	**PS Trees**	**MF Trees**	**Dep Trees**
Theory	**PTB $\sqcup_{\mathrm{lt}}$ SD**	**(PTB $\sqcup_{\mathrm{lt}}$ SD) $\sqcap_{\mathrm{t}}$ SD**	**SD**
Metrics	TedEval Single	TedEval Multiple	TedEval Single
Malt	N/A	U: 0.9525 L: *0.9088*	U: 0.9524 L: *0.9186*
MST	N/A	U: *0.9549* L: 0.9049	U: *0.9548* L: 0.9149
Berkeley	U: 0.9645 L: 0.9271	U: 0.9677 L: 0.9227	U: 0.9649 L: 0.9324
Brown	U: **0.9667** L: **0.9301**	U: **9702** L: **9264**	U: **0.9679** L: **0.9362**

Table 2: English cross-framework evaluation: TedEval scores against gold and generalized gold. Boldface scores are highest in their column. Italic scores are highest for dependency parsers in their column.

Formalism	**PS Trees**	**MF Trees**	**Dep Trees**
Theory	**STB**	**STB $\sqcap_{\mathrm{t}}$ Dep**	**Dep**
Metrics	ParsEval	TedEval	AttScore
Malt	N/A	U: 0.9266 L: **0.8225**	U: 0.8298 L: 0.7782
MST	N/A	U: 0.9275 L: 0.8121	U: **0.8438** L: **0.7824**
Bkly/STB-RR	F-Score **0.7914**	U: **0.9281** L: 0.7861	N/A
Bkly/STB-PS	F-Score 0.7855	N/A	N/A

Table 3: Swedish cross-framework evaluation: Three measures as applicable to the different schemes. Boldface scores are the highest in their column.

Formalism	**PS Trees**	**MF Trees**	**Dep Trees**
Theory	**STB**	**STB $\sqcap_{\mathrm{t}}$ Dep**	**Dep**
Metrics	TedEval Single	TedEval Multiple	TedEval Single
Malt	N/A	U: 0.9266 L: **0.8225**	U: 0.9264 L: **0.8372**
MST	N/A	U: 0.9275 L: 0.8121	U: **0.9272** L: 0.8275
Bkly-STB-RR	U: **0.9239** L: **0.7946**	U: **0.9281** L: 0.7861	N/A

Table 4: Swedish cross-framework evaluation: TedEval scores against the native gold and the generalized gold. Boldface scores are the highest in their column.

Because there are no parsers that can output the complete STB representation including edge labels, we experiment with two variants of this theory, one which is obtained by simply removing the edge labels and keeping only the phrase-structure labels (STB-PS) and one which is loosely based on the Relational-Realizational scheme of Tsarfaty and Sima'an (2008) but excludes the projection set nodes (STB-RR). RR trees only add function nodes to PS trees, and it holds that STB-PS$\sqcap_t$STB-RR=STB-PS. The overlap between the theories expressed in multifunction trees originating from STB-Dep and STB-RR is 0.7559. Our evaluation protocol takes into account such discrepancies while avoiding biases that may be caused due to these differences.

We evaluate MaltParser, MSTParser and two versions of the Berkeley parser, one trained on STB-PS and one trained on STB-RR. We use predicted part-of-speech tags for the dependency parsers, using the HunPoS tagger (Megyesi, 2009), but let the Berkeley parser predict its own tags. We use the same evaluation metrics and procedures as before. Prior to evaluating RR trees using ParsEval we strip off the added function nodes. Prior to evaluating them using TedEval we strip off the phrase-structure nodes.

Tables 3 and 4 summarize the parsing results for the different Swedish parsers. In the leftmost column of table 3 we present the constituency-based evaluation measures. Interestingly, the Berkeley RR instantiation performs better than when training the Berkeley parser on PS trees. These constituency-based scores however have a limited applicability, and we cannot use them to compare constituency and dependency parsers. In the rightmost column of Table 3 we report the LAS/UAS results for the two dependency parsers. Here we see higher performance demonstrated by MST on both labeled and unlabeled dependencies, but the differences on labeled dependencies are insignificant. Since there is no automatic procedure for converting bare-bone phrase-structure Swedish trees to dependency trees, we cannot use

LAS/UAS to compare across frameworks, and we use TEDEVAL for cross-framework evaluation.

Training the Berkeley parser on RR trees which encode a mapping of PS nodes to grammatical functions allows us to compare parse results for trees belonging to the STB theory with trees obeying the STB-Dep theory. For unlabeled TEDEVAL scores, the dependency parsers perform at the same level as the constituency parser, though the difference is insignificant. For labeled TEDEVAL the dependency parsers significantly outperform the constituency parser. When considering only the dependency parsers, there is a small advantage for Malt on labeled dependencies, and an advantage for MST on unlabeled dependencies, but the latter is insignificant. This effect is replicated in Table 4 where we evaluate dependency parsers using TEDEVAL against their own gold theories. Table 4 further confirms that there is a gap between the STB and the STB-Dep theories, reflected in the scores against the native and generalized gold.

5 Discussion

We presented a formal protocol for evaluating parsers across frameworks and used it to soundly compare parsing results for English and Swedish. Our approach follows the three-phase protocol of Tsarfaty et al. (2011), namely: (i) obtaining a formal common ground for the different representation types, (ii) computing the theoretical common ground for each test sentence, and (iii) counting only what counts, that is, measuring the distance between the common ground and the parse tree while discarding annotation-specific edits.

A pre-condition for applying our protocol is the availability of a relational interpretation of trees in the different frameworks. For dependency frameworks this is straightforward, as these relations are encoded on top of dependency arcs. For constituency trees with an inherent mapping of nodes onto grammatical relations (Merlo and Musillo, 2005; Gabbard et al., 2006; Tsarfaty and Sima'an, 2008), a procedure for reading relational schemes off of the trees is trivial to implement.

For parsers that are trained on and parse into bare-bones phrase-structure trees this is not so. Reading off the relational structure may be more costly and require interjection of additional theoretical assumptions via manually written scripts. Scripts that read off grammatical relations based on tree positions work well for configurational languages such as English (de Marneffe et al., 2006) but since grammatical relations are reflected differently in different languages (Postal and Perlmutter, 1977; Bresnan, 2000), a procedure to read off these relations in a language-independent fashion from phrase-structure trees does not, and should not, exist (Rambow, 2010).

The crucial point is that even when using external scripts for recovering a relational scheme for phrase-structure trees, our protocol has a clear advantage over simply scoring converted trees. Manually created conversion scripts alter the theoretical assumptions inherent in the trees and thus may bias the results. Our generalization operation and three-way TED make sure that theory-specific idiosyncrasies injected through such scripts do not lead to over-penalizing or over-crediting theory-specific structural variations.

Certain linguistic structures cannot yet be evaluated with our protocol because of the strict assumption that the labeled spans in a parse form a tree. In the future we plan to extend the protocol for evaluating structures that go beyond linearly-ordered trees in order to allow for non-projective trees and directed acyclic graphs. In addition, we plan to lift the restriction that the parse yield is known in advance, in order to allow for evaluation of joint parse-segmentation hypotheses.

6 Conclusion

We developed a protocol for comparing parsing results across different theories and representation types which is framework-independent in the sense that it can accommodate any formal syntactic framework that encodes grammatical relations, and it is language-independent in the sense that there is no language specific knowledge encoded in the procedure. As such, this protocol is adequate for parser evaluation in cross-framework and cross-language tasks and parsing competitions, and using it across the board is expected to open new horizons in our understanding of the strengths and weaknesses of different parsers in the face of different theories and different data.

Acknowledgments We thank David McClosky, Marco Khulmann, Yoav Goldberg and three anonymous reviewers for useful comments. We further thank Jennifer Foster for the Brown parses and parameter files. This research is partly funded by the Swedish National Science Foundation.

References

Philip Bille. 2005. A survey on tree edit distance and related. problems. *Theoretical Computer Science*, 337:217–239.

Ezra Black, Steven P. Abney, D. Flickenger, Claudia Gdaniec, Ralph Grishman, P. Harrison, Donald Hindle, Robert Ingria, Frederick Jelinek, Judith L. Klavans, Mark Liberman, Mitchell P. Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the DARPA Workshop on Speech and Natural Language*, pages 306–311.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The Tiger treebank. In *Proceedings of TLT*.

Joan Bresnan. 2000. *Lexical-Functional Syntax*. Blackwell.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL-X*, pages 149–164.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Stefan Riezler, Josef van Genabith, and Andy Way. 2008. Wide-coverage deep statistical parsing using automatic dependency structure annotation. *Computational Linguistics*, 34(1):81–124.

John Carroll, Edward Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: A survey and a new proposal. In *Proceedings of LREC*, pages 447–454.

Daniel Cer, Marie-Catherine de Marneffe, Daniel Jurafsky, and Christopher D. Manning. 2010. Parsing to Stanford Dependencies: Trade-offs between speed and accuracy. In *Proceedings of LREC*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL*.

Paul Cohen. 1995. *Empirical Methods for Artificial Intelligence*. The MIT Press.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, pages 449–454.

Ryan Gabbard, Mitchell Marcus, and Seth Kulick. 2006. Fully parsing the Penn treebank. In *Proceeding of HLT-NAACL*, pages 184–191.

Jesús Giménez and Lluís Màrquez. 2004. SVMTool: A general POS tagger generator based on support vector machines. In *Proceedings of LREC*.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Number 2 in Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Dekang Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of IJCAI-95*, pages 1420–1425.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic treebank: Building a large-scale annotated Arabic corpus. In *Proceedings of NEMLAR International Conference on Arabic Language Resources and Tools*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530, Morristown, NJ, USA. Association for Computational Linguistics.

Beata Megyesi. 2009. The open source tagger HunPoS for Swedish. In *Proceedings of the 17th Nordic Conference of Computational Linguistics (NODALIDA)*, pages 239–241.

Igor Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

Paola Merlo and Gabriele Musillo. 2005. Accurate function parsing. In *Proceedings of EMNLP*, pages 620–627.

Joakim Nivre and Beata Megyesi. 2007. Bootstrapping a Swedish Treebank using cross-corpus harmonization and annotation projection. In *Proceedings of TLT*.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, pages 2216–2219.

Joakim Nivre, Laura Rimell, Ryan McDonald, and Carlos Gómez-Rodríguez. 2010. Evaluation of dependency parsers on unbounded dependencies. In *Proceedings of COLING*, pages 813–821.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of ACL*.

Paul M. Postal and David M. Perlmutter. 1977. Toward a universal characterization of passivization. In *Proceedings of the 3rd Annual Meeting of the Berkeley Linguistics Society*, pages 394–417.

Owen Rambow. 2010. The simple truth about dependency and phrase structure representations: An opinion piece. In *Proceedings of HLT-ACL*, pages 337–340.

Roy Schwartz, Omri Abend, Roi Reichart, and Ari Rappoport. 2011. Neutralizing linguistically problematic annotations in unsupervised dependency parsing evaluation. In *Proceedings of ACL*, pages 663–672.

Khalil Sima'an, Alon Itai, Yoad Winter, Alon Altman, and Noa Nativ. 2001. Building a Tree-Bank for Modern Hebrew Text. In *Traitement Automatique des Langues*.

Reut Tsarfaty and Khalil Sima'an. 2008. Relational-Realizational parsing. In *Proceedings of CoLing*.

Reut Tsarfaty, Joakim Nivre, and Evelina Andersson. 2011. Evaluating dependency parsing: Robust and heuristics-free cross-framework evaluation. In *Proceedings of EMNLP*.

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. In *SIAM Journal of Computing*, volume 18, pages 1245–1262.