Towards Framework-Independent Evaluation of Deep Linguistic Parsers

Yusuke Miyao[1] Kenji Sagae[1] Jun'ichi Tsujii[1,2,3]
[1]Department of Computer Science
University of Tokyo
[2]School of Computer Science
University of Manchester
[3]National Center for Text Mining

Proceedings of the GEAF 2007 Workshop

Tracy Holloway King and Emily M. Bender (Editors)

CSLI Studies in Computational Linguistics ONLINE

Ann Copestake (Series Editor)

2007

CSLI Publications

http://csli-publications.stanford.edu/

Abstract

This paper describes practical issues in the framework-independent evaluation of deep and shallow parsers. We focus on the use of two dependency-based syntactic representation formats in parser evaluation, namely, Carroll et al. (1998)'s *Grammatical Relations* and de Marneffe et al. (2006)'s *Stanford Dependency* scheme. Our approach is to convert the output of parsers into these two formats, and measure the accuracy of the resulting converted output. Through the evaluation of an HPSG parser and Penn Treebank phrase structure parsers, we found that mapping between different representation schemes is a non-trivial task that results in lossy conversions that may obscure important differences between different parsing approaches. We discuss sources of disagreements in the representation of syntactic structures in the two dependency-based formats, indicating possible directions for improved framework-independent parser evaluation.

1 Introduction

Despite the rapid progress made in recent years on deep linguistic parsing (Cahill et al., 2002; Hockenmaier, 2003; Kaplan et al., 2004; Burke et al., 2004b; Clark and Curran, 2004; Malouf and van Noord, 2004; Oepen et al., 2004; Toutanova et al., 2004; Miyao and Tsujii, 2005), shallow phrase-structure parsers (Collins, 1997; Charniak and Johnson, 2005) are still often chosen over linguistically richer approaches in natural language processing (NLP) research and applications where syntactic analysis is needed. This is due in part to the perception that deep parsing is not robust and efficient enough for handling practical tasks, and that its accuracy is below that of shallow parsing approaches. In addition, the advantages of deep syntactic analysis over shallow phrase-structures, although clear to those in the deep parsing community, has not been demonstrated convincingly to the general NLP community. While shallow parsers may in fact be better suited for some NLP tasks, an informed decision on that regard requires a fair comparison between different kinds of parsers, especially when they deal with different ways of representing syntactic information. However, comparison of different parsing approaches is challenging even among deep parsers, since accuracy measurements used in different systems are largely incompatible, making it difficult to determine the advantages of specific deep parsing approaches. Meanwhile, the most widely used evaluation metric in current parsing research, precision and recall of labeled brackets, follows a view of syntax that is simplistic and at the same time quite specific to one particular type of syntactic representation. While the use of bracketing precision and recall in simplified trees from the Penn Treebank (Marcus et al., 1994) fueled much of the development of current wide-coverage data-driven parsing by providing a way to evaluate parsers on a common test set, it is now too

[†]We thank John Carroll for providing the gold-standard GR data, and for numerous insightful discussions and comments. This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan) and Grant-in-Aid for Young Scientists (MEXT, Japan).

limited to deal with recent developments that go beyond what is represented in that test set[1]. Hence, framework-independent parser evaluation is necessary not only for informed development of NLP applications, where different types of parsers may be more or less suited for certain NLP tasks (Clegg and Shepherd, 2007), but also for progress in parsing research itself, where it would allow for a more direct comparison between different parsing approaches (Clark and Curran, 2007).

This paper discusses several challenges and practical issues in framework-independent evaluation of syntactic parsers. Specifically, we focus on two existing proposals for representing syntactic relationships between words, and examine practical issues through the evaluation of parsing accuracy of a deep parser based on Head-Driven Phrase Structure Grammar (HPSG), *Enju* (Miyao and Tsujii, 2005). The first representation scheme we consider is *Grammatical Relations* (GR) (Carroll et al., 1998; Carroll and Briscoe, 2004; Briscoe, 2006), which aims to provide a better parser evaluation framework than PARSEVAL measures (Black et al., 1991) of constituent bracketing precision and recall. This scheme has been adopted for the evaluation of RASP (Briscoe and Carroll, 2006; Briscoe et al., 2006), shallow parsers derived from Penn Treebank (PTB) (Preiss, 2003), and recently, a deep parser based on Combinatory Categorial Grammar (CCG) (Clark and Curran, 2007). The other is the *Stanford Dependency* scheme (SD) (de Marneffe et al., 2006), which was proposed for providing NLP applications with more useful syntactic representations than phrase structures. Although gold standard data is not available, a program attached to the Stanford parser (Klein and Manning, 2003) automatically converts PTB-style phrase structures into this format. This conversion is only approximate, making SD-based evaluation problematic. In addition, the lack of detailed documentation on the specific syntactic representation choices highlights that this format was not originally intended for parser evaluation. However, because of its recent use in the evaluation of shallow PTB-style parsers in the biomedical domain (Clegg and Shepherd, 2007; Pyysalo et al., 2007a), and the availability of a conversion tool that uses shallow PTB-style trees[2] as input, we investigate the use of SD as a scheme for framework-independent parser evaluation.

Our basic strategy for the evaluation of Enju is to establish a program for converting Enju's output into these two formats, and measure accuracy of converted output. We also develop a conversion program from SD to GR, which allows for GR-based evaluation of PTB-style parsers (Collins, 1997; Charniak, 2000), since a conversion tool from shallow PTB-style output to SD is available. We can therefore compare the performances of Enju and shallow PTB parsers directly, in addition to previously reported results for RASP (Briscoe and Carroll, 2006; Briscoe et al., 2006) and the C&C CCG parser (Clark and Curran, 2007).

One might expect that format conversion is straightforward among GR, SD,

[1]By "test set" we refer to the set of shallow brackets extracted from the Penn Treebank data. While the original treebank data includes richer syntactic analyses, information such as long-distance dependencies, ellipsis, and functional tags are removed in the extraction of shallow brackets.

[2]We use "shallow PTB-style trees" to refer to the Penn Treebank trees with empty-nodes and function-tags removed.

and Enju's output, because they all represent labeled dependencies between words and are similar in concept. In fact, however, our experiments revealed that format conversion is not trivial. We had to implement complex mapping rules for Enju-to-GR/SD and SD-to-GR conversion, and there remain a lot of disagreements for which resolution is unlikely and which may obscure not just differences in performance among individual parsers, but also differences in the strengths of general parsing approaches.

The idea of parser evaluation across frameworks is not new, and its difficulty has been reported repeatedly in the literature (Carroll et al., 1998; Kaplan et al., 2004; Burke et al., 2004a; Clark and Curran, 2007). The results in this paper add to this discussion by focusing on actual challenges in format conversion, providing in-depth analyses of sources of format disagreements. It is our hope that such work will provide the direction for the development of a better scheme for framework-independent evaluation of deep and shallow parsers.

Section 2 presents an overview of the two schemes for parser evaluation. Section 3 describes methods for conversion from Enju's output to GR/SD, and from SD to GR. Section 4 shows experimental results on the accuracy evaluation of Enju, PTB parsers, RASP, and a CCG parser. Section 5 discusses sources of difficulties in format conversion.

2 Parser Evaluation Schemes

In the context of wide-coverage deep parsing, the de facto standard metric for parsing accuracy is precision/recall of labeled dependency relations such as predicate argument dependencies (Kaplan et al., 2004; Clark and Curran, 2004; Miyao and Tsujii, 2005). However, dependency relations used to evaluate different parsers are based on each parser's formalism and resources. For example, the PARC 700 Dependency Bank (King et al., 2003) was used for the evaluation of LFG parsers (Kaplan et al., 2004; Burke et al., 2004a), a CCG treebank (CCGBank) (Hockenmaier and Steedman, 2002) was used for the evaluation of CCG parsing models (Hockenmaier, 2003; Clark and Curran, 2004), and HPSG treebanks, which were created manually (Oepen et al., 2004) or derived from PTB data (Miyao et al., 2005), were used for the evaluation of HPSG parsers (Toutanova et al., 2004; Miyao and Tsujii, 2005; Ninomiya et al., 2007; Sagae et al., 2007). Direct relationships among different dependency schemes are unclear, and we have no way for fair comparison of these parsers.

One issue that must be considered in parser evaluation is that an evaluation scheme must represent information needed by applications and cover real-world texts, because the goal of parser development is usability in NLP applications. Another important issue is that the evaluation framework should account for syntactic structures that are not tied specifically to any single formalism. For example, an evaluation scheme should be sensitive to grammatical phenomena such as control/raising and long-distance dependencies, even though such structures are not

```
(ncsubj market They _)
(iobj market on)
(dobj market cable-TV)
(dobj on opportunities)
(det opportunities the)
(ncmod _ opportunities grazing)
(cmod _ opportunities seeks)
(ncsubj seeks CNN _)
(ncsubj discourage CNN _)
(dobj discourage opportunities)
(xcomp to seeks discourage)
(ncmod _ opportunities very)
```

Figure 1: GR annotation for *They market cable-TV on the very grazing opportunities CNN seeks to discourage.*

accounted for in shallow PTB parsers. At the same time, the inclusion of such syntactic phenomena must not make it unnecessarily difficult to evaluate parsers that output shallow brackets.

Considering these issues, we focus on two dependency-based schemes, *Grammatical Relations* (GR) (Carroll et al., 1998; Carroll and Briscoe, 2004; Briscoe, 2006) and the *Stanford Dependency* (SD) scheme (de Marneffe et al., 2006), which were proposed outside the deep parsing community, while aiming to represent not only surface syntactic structures but also deep structures such as long distance dependencies. In what follows, we describe these schemes and compare them briefly.

2.1 Grammatical Relations (GR)

The *Grammatical Relation* scheme (GR) was proposed aiming at a framework-independent metric for parsing accuracy (Carroll et al., 1998). A set of 700 sentences extracted from Section 23 of the Penn Treebank (the same set as the PARC 700 Dependency Bank) was manually annotated and made publicly available as gold standard data (Briscoe and Carroll, 2006), in addition to an older set of 500 sentences from the SUSANNE corpus[3]. While this evaluation scheme is not as widely used as PARSEVAL, it has recently gained some traction as a more framework-independent alternative, and has been used in the evaluation of parsers including RASP (Carroll and Briscoe, 2004; Briscoe and Carroll, 2006; Briscoe et al., 2006) and the C&C CCG parser (Clark and Curran, 2007). Preiss (2003) reported GR-based evaluation of PTB parsers including the Collins parser (Collins, 1997) and the Charniak parser (Charniak, 2000), although the SUSANNE-based gold data was used, and the results are not directly comparable to the results in this paper, where we use the data based on the PARC 700 selection of Penn Treebank sentences.

[3]http://www.informatics.sussex.ac.uk/research/groups/nlp/carroll/greval.html.

Figure 1 shows an example of GR annotation. GR represents labeled syntactic dependencies between words. For example, `ncsubj` means a non-clausal subject (e.g. `(ncsubj market They _)`), `dobj` indicates a direct object (e.g. `(dobj market cable-TV)`), and `ncmod` expresses a non-clausal modifier (e.g. `(ncmod _ opportunities grazing)`). Most relations are binary, while a few relation types have additional slots that represent subtypes of the relations. For example, `(xcomp to seeks discourage)` means that *discourage* is a to-infinitival complement of *seeks*. Refer to Briscoe (2006) for the definition of these relation types.

GR annotations are almost purely *syntactic* and therefore lack the means to evaluate the true potential of deep linguistic parsers that compute relationships based on semantics. However, it should be noted that GR represents non-local dependencies such as control/raising and movement. In this example, `(ncsubj discourage CNN _)` indicates a control relation, `(dobj discourage opportunities)` expresses a moved object of *discourage*, and `(cmod _ opportunities seeks)` means a relation between a relative clause and its antecedent. Since these relations are not explicitly represented by PTB parsers, this scheme may serve as a starting point in the identification of the added benefits of deep parsing and the discussion of problems in framework-independent evaluation. On the other hand, identifying most of the relationships in the GR scheme in the output of shallow phrase structure parsers requires matching of tree patterns, which makes it challenging to evaluate those parsers.

Relation types in the GR scheme are arranged in a hierarchy. Upper types represent more generalized and coarse-grained relations. This hierarchy is used for partial matching of relation types, which is intended for reducing disagreements involving relation types. For example, when a parser outputs `(ncmod _ market on)`, where the gold standard relation is `(iobj market on)`, this output is regarded as incorrect at the leaf level, but judged as correct at upper levels, `arg_mod` and `dependent`. This matching in the hierarchy is considered in scoring as described below.

Standard metrics for the GR scheme are *microaveraged* and *macroaveraged* scores. Microaveraged scores are similar to standard precision/recall/f-score, but take accuracy of non-leaf relation types into consideration. For example, `ncmod` is a subtype of `mod`, `arg_mod`, and `dependent`. A single `ncmod` dependency is regarded as expressing these four relations, and correctness of each of these relations is counted. Hence, as indicated above, disagreements of relation types are discounted, because higher level types are easier to identify. In general, microaveraged scores are higher than the accuracy of leaf relation types. A macroaveraged score is an average of the accuracy for each relation type, and frequencies of relation types are ignored. Hence, infrequent relation types affect macroaveraged scores. A program for computing microaveraged/macroaveraged scores is publicly available[4]. We also report the overall accuracy of leaf relation types only, which is the same metric used in our evaluation using SD.

[4]http://www.informatics.sussex.ac.uk/research/groups/nlp/carroll/greval.html

```
nsubj(market-2, They-1)
dobj(market-2, cable-3)
det(opportunities-8, the-5)
amod(opportunities-8, very-6)
nn(opportunities-8, grazing-7)
prep_on(market-2, opportunities-8)
nsubj(seeks-10, CNN-9)
rcmod(opportunities-8, seeks-10)
aux(discourage-12, to-11)
xcomp(seeks-10, discourage-12)
```

Figure 2: SD annotation for *They market cable-TV on the very grazing opportunities CNN seeks to discourage.*

2.2 Stanford Dependency (SD) scheme

The *Stanford Dependency* (SD) scheme was originally proposed for providing dependency relations that are more useful for applications than phrase structures (de Marneffe et al., 2006). This scheme was designed based on Carroll et al. (1998)'s grammatical relations and King et al. (2003)'s dependency bank, and modified to represent more fine-grained and semantically valuable relations such as apposition and temporal modification, while at the same time leaving out certain relations that are particularly problematic for the parsers it was intended to work with. Although no hand-annotated data is available, a program to convert PTB-style phrase structures into SD relations is available as part of the Stanford Parser[5]. That is, in principle, any PTB-style treebank can be converted into SD gold standard data. In practice, however, the conversion from phrase structure trees to SD is only approximate, and converting gold standard phrase structure trees results in only partially correct SD annotations. Unfortunately, the accuracy of these annotations is unknown. This scheme was recently used for the evaluation of shallow PTB-style parsers in the biomedical domain (Clegg and Shepherd, 2007; Pyysalo et al., 2007a), using GENIA (Kim et al., 2003) and BioInfer (Pyysalo et al., 2007b) as sources of gold standard PTB-style data.

Figure 2 shows an example of SD annotation for the same sentence as Figure 1. SD looks very similar to GR, and represents many equivalent relations such as `nsubj(market-2, They-1)`, `dobj(market-2, cable-3)`, and `nn(opportunities-8, grazing-7)`. The example also includes long-distance dependencies, but they are not as explicit as in GR, and often not as reliable. Here, `xcomp(seeks-10, discourage-12)` indicates a control relation between *seeks* and *discourage*, and `rcmod(opportunities-8, seeks-10)` expresses a relation between a relative clause and its antecedent. However, relations indicating that *CNN* is the subject of *discourage* and *opportunities* is the direct object of *discourage* are not represented. Refer to de Marneffe et al. (2006) for details of these

[5]http://nlp.stanford.edu/software/lex-parser.shtml

```
(ncsubj ordered Regulators _)
(ncsubj stop CenTrust _)
(ncsubj buying CenTrust _)
(ncmod _ ordered also)
(xcomp to ordered stop)
(xcomp _ stop buying)
(dobj buying stock)
(det stock the)
(passive preferred)
(ncsubj preferred stock obj)
(ncmod _ stock preferred)
(ncmod prt buying back)
(dobj ordered CenTrust)
```

```
nsubj(ordered-3, Regulators-1)
advmod(ordered-3, also-2)
dobj(ordered-3, CenTrust-4)
aux(stop-6, to-5)
xcomp(ordered-3, stop-6)
partmod(stop-6, buying-7)
prt(buying-7, back-8)
det(stock-11, the-9)
amod(stock-11, preferred-10)
dobj(buying-7, stock-11)
```

Figure 3: GR (left) and SD (right) for *Regulators also ordered CenTrust to stop buying back the preferred stock.*

relation types, and note that while some of these relations are described as part of the SD scheme, they are not implemented in the provided conversion software.

Although SD relation types are also organized in a hierarchy, it is intended for convenience in use by applications, and they are not aimed at parser evaluation purposes. Hence, we use standard precision, recall, and f-score as metrics for SD-based evaluation.

2.3 Comparison of the two schemes

As noted above, and illustrated in Figure 3, the GR and SD schemes are very similar in concept, and they represent equivalent dependencies in many cases. In this example, they share subject/object relations such as *(ordered, Regulators)*, clausal complements such as *(ordered, stop)*, and modifiers such as *(ordered, also)*. However, disagreements can also be found; for example, *preferred* is recognized as a past-participle modifier in GR (which is indicated as (`ncsubj preferred stock obj`)), while it is an adjectival modifier in SD (which is represented as `amod(stock-11, preferred-10)`). This comes from a difference in the part-of-speed (POS) of *preferred*. Representations of long-distance dependencies are also different, as previously mentioned. In this example, the subject of *stop* and *buying* is expressed in GR, while not in SD. Finally, we once again note that SD data converted from gold standard PTB data includes errors: in figure 3 *buying* is incorrectly recognized as a participial modifier to *stop*.

An advantage of GR is the availability of hand-annotated data, although the data size is relatively small. Another advantage is that partial matching of relation types may reduce the labor of format conversion. An advantage of SD is that we can use any data annotated with the more common PTB annotation policy. In addition, evaluation of PTB parsers (Collins, 1997; Charniak and Johnson, 2005) is convenient because software for format conversion is already available. However, conversion errors make the evaluation only approximate, and the lack of a detailed definition of relation types is an obstacle to further development of conversion rules. This leads to a greater problem in framework-independent evaluation, since many of the same conversion errors are present in the SD data converted from gold standard PTB data and the converted output of shallow PTB-style parsers. For example, the SD data used in experiments includes many relations assigned “dep”, which is the most underspecified relation type. This relation is chosen as output when the conversion program could not determine a relation type properly. In fact, more than 5% of relations are assigned “dep”, meaning that the actual upper bound for PTB-to-SD conversion is below 95%. Because these errors are undocumented, in practice they result in inflated accuracy figures for shallow PTB parsers when compared to the accuracy of parsers that use other formats that must be converted with different software (and may contain a number of different conversion errors). In other words, the accuracy of parsers that use PTB-like output is overestimated, and the accuracy of parsers that use other output formats is likely to be underestimated.

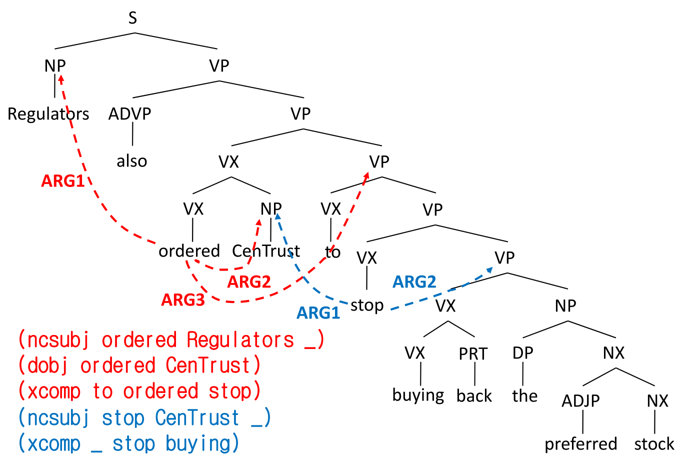
3 Format Conversion

Our strategy for format conversion is based on post-processing. That is, we convert the output of parsers without changing the original parsers. During the development of conversion programs, we validate our progress using a development set of gold standard data in the format used by each parser, i.e., we run the conversion on parts of the HPSG treebank (Miyao et al., 2005) and the Penn Treebank. Automatic parsing results are used in the final evaluation. This is because accuracy obtained by converting gold standard data indicates the quality of conversion, and we can separate issues of format conversion from actual parsing errors. Accuracy figures from converted gold standard are also meaningful as upper-bounds of scores obtained with these evaluation schemes.

3.1 From Enju’s XML format to GR/SD

We implemented conversion rules for the Enju XML format (Miyao, 2007). This format represents constituent structures and predicate argument structures in an XML format. To start with, we mapped predicate argument relations into GR/SD. Figure 4 shows an example of the XML format of Enju, and its mapping to GR. Arguments of *ordered* and *stop* can be mapped into GR in a fairly straightforward manner. Relation types are determined depending on argument labels (e.g. `ARG1` and `ARG2`), categories and POS tags of predicate words (e.g. `VBD`), and syntactic categories of argument constituents (e.g. `NP` and `VP`).

However, this simple method produced poor accuracy, mainly because of non-trivial disagreements between the formats. Hence, we had to implement heuristic conversion rules to fix these disagreements.



```
(ncsubj ordered Regulators _)
(dobj ordered CenTrust)
(xcomp to ordered stop)
(ncsubj stop CenTrust _)
(xcomp _ stop buying)
```

Figure 4: Enju XML format and mapping to GR

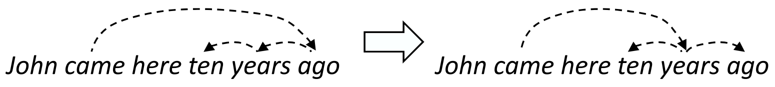


Figure 5: Conversion of lexical heads

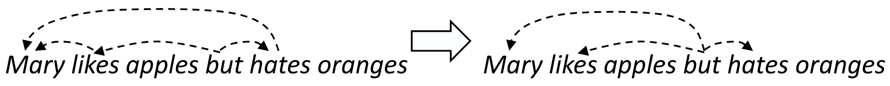

Mary likes apples but hates oranges
Mary likes apples but hates oranges

Figure 6: Conversion of coordination

It is often the case that certain types of relations are expressed in one format, but not in the other. For example, GR has “text adjunct” as a distinct relation, while Enju (and SD) does not distinguish such a relation type from others. Text adjunct is a text region delimited by some punctuation (Briscoe, 2006). Our conversion program outputs parentheses and appositive relative clauses as text adjuncts, but does not identify other text adjuncts. Another example is that GR does not represent internal structures of named entities, while Enju does. Hence, we detect text regions of named entities using simple patterns on POS tags, and remove dependencies inside named entity regions.

A major source of format disagreements was differences of lexical heads. Figure 5 shows an example of a temporal modifier. In Enju’s output (left), the lexical head of *ten years ago* is *ago*, while in the GR scheme (right), it is *years*. Hence, our conversion rule changes lexical heads of such temporal modifiers. Similar conversion is applied to such constructions as number expressions.

Coordination was another major source of disagreements. Figure 6 shows an example of VP coordination. Enju outputs subject relations of conjunct VPs separately (left), while in GR the head of coordinated phrase is a coordinator and it has a subject relation (right). We therefore reduce two subject relations in Enju's output into one.

We also found systematic disagreements in specific constructions including relative clauses, quotations, copulas, and small clauses. For example, GR and SS represent a syntactic relation between the head verb of a relative clause and its antecedent. However, Enju does not output such relations explicitly, and instead, expresses a relation between a relativizer and its antecedent. We therefore developed conversion rules specialized to these constructions.

3.2 From SD to GR

Although the typed dependencies in the Stanford Dependency scheme are superficially similar to those in the Grammatical Relations scheme, conversion from SD to GR is problematic for many of the same reasons cited above in our discussion on conversion from Enju's XML format. However, a more serious problem with the use of SD (and SD to GR conversion) for parser evaluation is the lack of a gold-standard SD corpus. In our experiments, SD annotations are obtained from shallow PTB-style phrase-structure trees (which correspond to PTB trees with empty-node and function-tag information removed), using a conversion program included in the Stanford Parser. As can be expected, given our present discussion about parser format conversion, the PTB to SD conversion program is far from perfect. In addition to noticeable errors in the output of the conversion program, more than 5% of the dependencies are left completely underspecified, labeled only with the general `dep` type (which does not correspond to a specific grammatical relation, indicating only a head-dependent relation between two words). However, the accuracy of this conversion is not known, and cannot be easily computed without a gold-standard corpus for SD.

Conversion from SD to GR followed a similar pattern than the one described above for conversion from Enju XML to GR. First, a simple mapping between corresponding relations was attempted. As was the case with Enju, the resulting conversion was poor for all but the simplest relations (`det` and `aux`). A telling example is the conversion of SD's `nsubj` to GR's `ncsubj`. Although the two relations appear very similar, a number of undocumented differences make the conversion less straightforward than a simple mapping. For example, sentences involving the copula are treated differently by the two schemes, with GR attaching the subject as a dependent of the verb, and SD attaching the verb as a dependent of the predicate nominal. While some additional information required by GR's `ncsubj` (such as whether the subject position is inverted, or the initial relation of the subject) can be determined reliably by looking at aspects of the SD structure that go beyond the `nsubj` dependency, some information (such as subjects in control structures) simply cannot be determined from SD structures. Although de Marneffe et al.'s description of the SD scheme seems to indicate that enough information for such a conversion should be available, the actual implementation of the SD scheme in the

Stanford Parser lacks information relating to, for example, control structures and long-distance dependencies. This is understandable, given the difficulty in producing such information from shallow PTB-style trees. However, undocumented differences between the description and implementation of SD make the conversion even more difficult. A reasonably accurate conversion to GR's `ncsubj` was finally obtained with the use of development data, by inspecting specific examples annotated in each format. However, the efficacy of this approach varied in other types of relations, especially since the amount of development data available was limited.

One of the most problematic aspects of SD to GR conversion is the distinction of complements from adjunct, especially in prepositional phrases. SD does not assign a grammatical function to PPs, making it difficult to determine the correct relation in the GR scheme. As a result, the identification of indirect objects (`iobj`) has low accuracy compared to other relations. Of course, this also affects the accuracy of non-clausal modifiers. Another source of conversion errors is coordination, which is annotated in such a way in SD that its scope cannot be determined. As with Enju XML, recognizing the text adjunct (`ta`) relation is challenging, since it is not represented in the SD scheme. Although issues relating to headedness were less problematic in SD to GR conversion than in Enju XML to SD/GR, there were still differences, probably related to the use of an automatic conversion from the original PTB data to SD, compared to the manual annotation of the GR gold-standard data.

Finally, we reiterate that the conversion errors in our conversion from SD to GR are added on top of the PTB to SD conversion errors made by the Stanford parser implementation when a complete PTB to GR conversion is performed. For this reason, the GR results we obtain from parsers that produce PTB-style output do not do these parsers justice. While it is possible that a one-step conversion, from shallow PTB-style trees directly to GR, could produce more accurate results, an attempt by Preiss (2003) shows that this is not guaranteed to be much more successful, or at least is far from trivial. While our PTB to GR conversion does not provide completely fair grounds for comparison between shallow PTB-style parsers and Enju, a deep parser, it does serve to highlight some of the challenges in attempting such a comparison.

4 Experiments

Table 1 shows the sizes of the data sets used in experiments. For the development of conversion rules, we used 140 sentences extracted from the GR-annotated version of the PARC 700 Dependency Bank and the same set of sentences annotated automatically with SD (by running the Stanford Parser's automatic conversion on the corresponding Penn Treebank gold standard trees). For the final test, we used 560 sentences of the GR data and the same set of SD-annotated sentences. The GR data for the final test is the same set as previous works on GR-based evaluation

Table 1: Statistics of test data

scheme	# sent.	# rels.	# avg. rels/sent.	# rel. types
GR	560	10386	18.55	18
SD	560	9343	16.68	40

(Briscoe and Carroll, 2006; Briscoe et al., 2006; Clark and Curran, 2007).

The parsers we evaluate are Enju 2.2[6], Charniak and Johnson (2005)'s reranking parser (C&J parser), Charniak (2000)'s parser, and the Stanford parser (Klein and Manning, 2003). We also show previously reported microaveraged and macroaveraged scores for the GR evaluation of RASP (Briscoe and Carroll, 2006; Briscoe et al., 2006) and the C&C CCG parser (C&C parser) (Clark and Curran, 2007). Enju 2.2 includes a feature forest model (Miyao and Tsujii, 2005) and an extremely lexicalized model (Ninomiya et al., 2007), while excluding more advanced technologies such as deterministic parsing (Matsuzaki et al., 2007) and combination with shallow dependency parsing (Sagae et al., 2007).

Tables 2 and 3 show the accuracy of Enju and the PTB-style parsers obtained after the format conversion. In these tables, "auto" denotes figures obtained from automatic parsing results, while "gold" indicates accuracy figures obtained by converting gold standard data (establishing upper-bounds for the corresponding "auto" figures). In the case of Enju, "gold" figures are obtained by converting the HPSG treebank, and indicate the upper bound in accuracy in these evaluation schemes. Because the HPSG treebank lacks several sentences due to failures in the PTB-to-HPSG conversion (Miyao et al., 2005) that created the HPSG treebank, we excluded missing sentences from the evaluation of "gold". The evaluation of "auto" includes all sentences in the test data. For the evaluation of PTB parsers on GR, we applied our SD-to-GR conversion program to the output of the existing PTB-to-SD conversion software. In this case, "gold" indicates the accuracy obtained in the two-step conversion from PTB to GR. The "gold" accuracy for SD is 100%, because in SD evaluations we take the output of the Stanford Parser's PTB-to-SD conversion to be correct. Although we know the conversion is in fact not 100% correct, in our SD-based evaluation we do take the conversion of gold standard PTB trees to be our gold standard SD corpus, since a manually curated gold standard corpus is not available.

First, we note that these results show that the accuracy levels obtained by converting gold standard data are fairly low when format conversion is needed. This means that format conversion is far from perfect. For both GR and SD evaluation of Enju, "gold" accuracy figures are slightly higher than 80%, indicating that nearly 20% of dependencies cannot be converted properly. This is discouraging because reported accuracy levels of shallow and deep parsing are around 90%. However,

[6]Available at `http://www-tsujii.is.s.u-tokyo.ac.jp/enju/`

Table 2: Accuracy for GR

	gold			auto		
	precision	recall	f-score	precision	recall	f-score
Enju	84.27	83.67	83.97	80.60	78.74	79.66
C&J parser	78.60	68.51	73.21	75.86	62.92	68.79
Charniak parser	78.60	68.51	73.21	75.18	62.97	68.53
Stanford parser	78.60	68.51	73.21	70.88	60.24	65.13

this is an indication that previously reported accuracy figures might be inflated. The accuracy of "gold" PTB conversion to GR is even worse, since in this case we do suffer from the errors in PTB-to-SD conversion, and the errors in the subsequent SD-to-GR conversion. As we have described, these schemes are superficially similar, but this result reveals the difficulty of format conversion even between SD and GR. A possible reason is that a significant portion of GR dependencies could not be produced accurately from shallow phrase structures, which resulted in lower recall.

Results for "auto" reveal that Enju outperforms PTB parsers significantly in our GR evaluation, which is, as previously noted, unfair to the PTB parsers that suffer a double penalty in conversion. In our SD evaluation, which in turn heavily favors the PTB parsers and penalizes Enju, as previously discussed, the PTB parsers do have higher accuracy than Enju. It is then obvious that these contradictory results are heavily affected by the quality of format conversion, and this highlights how challenging (and even misleading) cross-framework evaluations can be. If we focus on an argument on the neutral nature of the GR scheme, we might be able to say that Enju is better in recognizing deeper dependency relations. However, this result relies on SD-to-GR conversion after PTB-to-SD conversion for PTB parsers, and it is likely that the figures for PTB parsers may be improved by directly converting their PTB-style to GR. However, it should be noted that our results for PTB parsers are better than the results reported by Preiss (2003) that implemented direct conversion from PTB to GR, although actual figures are not comparable because the test data is different, and the test set used by Preiss was in a different domain.

Tables 4 and 5 show microaveraged and macroaveraged scores for GR, respectively. We also show previously reported results for RASP (Briscoe and Carroll, 2006; Briscoe et al., 2006) and the C&C parser (Clark and Curran, 2007), which used the same evaluation scheme. Table 6 shows the accuracy of Enju for each relation type. As described in Section 2, microaveraged scores are higher than the accuracy in Table 2, which means that disagreements of relation types are reduced to some extent. However, nearly 13% of relations still cannot be produced. Similar results were also reported for the CCG parser, and this suggests that format disagreements may not be a simple matter of relation type mismatch.

Although the problem of format conversion remains, and we do not claim to

Table 3: Accuracy for SS

	gold			auto		
	precision	recall	f-score	precision	recall	f-score
Enju	83.43	81.44	82.42	77.38	74.54	75.93
C&J parser	100.00	100.00	100.00	88.36	88.45	88.40
Charniak parser	100.00	100.00	100.00	87.05	87.10	87.07
Stanford parser	100.00	100.00	100.00	85.36	83.16	84.25

Table 4: Microaveraged scores for GR

	gold			auto		
	precision	recall	f-score	precision	recall	f-score
Enju	87.49	86.79	87.14	83.57	81.73	82.64
C&J parser	80.84	69.16	74.54	79.08	67.46	72.81
Charniak parser	80.84	69.16	74.54	78.41	67.68	72.65
Stanford parser	80.84	69.16	74.54	74.76	64.83	69.44
RASP	—	—	—	77.66	74.98	76.29
C&C parser	86.86	82.75	84.76	82.44	81.28	81.86

Table 5: Macroaveraged scores for GR

	gold			auto		
	precision	recall	f-score	precision	recall	f-score
Enju	81.19	75.70	78.35	77.87	71.10	74.33
C&J parser	62.64	49.30	55.17	60.20	47.97	53.39
Charniak parser	62.64	49.30	55.17	59.39	48.08	53.14
Stanford parser	62.64	49.30	55.17	57.93	46.81	51.78
RASP	—	—	—	62.12	63.77	62.94
C&C parser	71.73	65.85	68.67	65.61	63.28	64.43

have achieved the best possible results with the PTB parsers, Tables 4 and 5 show that the accuracy of Enju is significantly higher than those of other parsers evaluated using the same test set, including PTB parsers, RASP, and the C&C parser. In particular, Enju achieved impressively higher macroaveraged scores, indicating that Enju is able to recognize infrequent relation types accurately.

Table 6: Relation type-wise accuracy

	gold			auto		
	precision	recall	f-score	precision	recall	f-score
ncmod	77.01	85.17	80.88	72.82	79.42	75.98
xmod	60.48	61.21	60.84	51.83	55.62	53.66
cmod	75.44	55.48	63.94	66.67	52.38	58.67
pmod	0.00	0.00	0.00	0.00	0.00	0.00
det	96.35	97.85	97.09	94.24	94.49	94.37
ncsubj	88.39	86.46	87.41	83.23	81.02	82.11
xsubj	100.00	57.14	72.73	75.00	42.86	54.55
csubj	75.00	100.00	85.71	100.00	100.00	100.00
dobj	91.80	93.53	92.66	88.35	89.36	88.85
obj2	56.52	68.42	61.90	61.90	65.00	63.41
iobj	82.24	60.08	69.43	82.25	58.33	68.26
xcomp	82.48	78.22	80.29	80.90	71.13	75.70
ccomp	87.39	79.39	83.20	81.92	73.45	77.45
pcomp	100.00	63.64	77.78	94.12	66.67	78.05
aux	95.88	95.36	95.62	94.66	92.77	93.70
conj	89.93	84.74	87.26	81.17	73.31	77.04
ta	61.54	25.91	36.47	59.46	22.68	32.84

5 Analysis of Format Disagreements

In what follows, we discuss sources of disagreements we found through our experiments. Figure 7 shows classification of dependency mismatches between the converted HPSG treebank and GR gold standard. That is, these come from format disagreements, and do not include parsing errors.

Text adjunct As described in Section 3, GR has a relation type called *text adjunct*, which is not explicitly identified by Enju. Although our conversion program tries to produce such relations, Table 7 shows that a significant number of text adjuncts were not recognized correctly.

Argument/modifier distinction It is widely recognized that a clear distinction between arguments and modifiers is difficult even for humans. In fact, there are no formal criteria for argument/modifier distinction in GR/SD annotation, and they are different even between GR and SD. Our conversion program approximately reproduces their intended distinctions, but a significant portion of them remain mismatched. One reason is that Enju outputs most prepositional phrases as modifiers. However, we found many other cases such as a distinction between adverbial clauses and clausal complements.

Table 7: Classification of dependency disagreements

Remaining disagreements	107
text adjunct	35
argument/modifier distinction	34
lexical head	25
POS	7
attachment	6
Conversion errors	36
named entity	15
number expression	6
coordination	6
others	9
Limitation of the HPSG treebank	14
noun phrase structure	10
others	4
Errors in the gold standard	13

Lexical head Although headedness is considered an agreed upon notion of syntactic structures, it is not obvious for a certain portion of syntactic constructions. For example, the head of "30.5 million" is "30.5" in GR and SD, while it is "million" in Enju. This example is rather simple and could be converted easily, but the important implication here is that there is a potential disagreement in headedness in many different types of constructions such as this one. This is critical because dependency-based evaluation heavily relies on the identification of lexical heads, and disagreements on heads may unfairly decrease apparent accuracy.

A similar problem is the necessary portion of arbitrary annotation policies that is inherent in this type of exercise. A typical example is the dependency structure of "*more than 2%*". In GR, "*more*" is the head of this phrase, and "*than 2%*" is a modifier of "*more*". However, in SD and Enju, "*more than 2*" constitutes a quantifier phrase, and "*more*" modifies "*%*". Either of these is acceptable, and we should regard this as a difference of annotation policies.

Part of Speech (POS) While the POS tags of some words are legitimately ambiguous, their differences significantly hurt accuracy because different relation types are assigned to words with different POS tags. For example, in Figure 3, GR recognizes "*preferred*" as a past participle, while SD (and Enju) treats it as an adjective, which results in assigning different relation types.

Conversion errors Our conversion rules for named entities, number expressions, coordination, and others did not work properly in some cases, and conversion errors remained. We expect that these errors can be reduced with further improvement of our conversion rules, although this complicates the process of format conversion.

Limitations of the HPSG treebank The HPSG treebank does not represent some syntactic structures correctly. An example is internal structures of noun phrases. In the HPSG treebank, most noun phrases are annotated as right-branching trees, which are not necessarily correct. This is because the HPSG treebank was translated from the Penn Treebank, in which the internal structure of noun phrases is not annotated.

Errors in the gold standard We found a few cases that we simply disagreed or did not understand the intention of the GR gold standard annotation. Clearly, this type of problem is much more serious in our SD evaluation, because the SD evaluation sets are automatically converted from PTB, and they contain unjustifiable relations caused by imperfect conversion.

6 Summary and Future Directions

In this paper, we described an attempt to perform framework-independent parser evaluation. We focused on two dependency-based schemes for parser evaluation, namely, GR and SD, and evaluated the accuracy of an HPSG parser and shallow PTB-style parsers by converting their output into the dependency formats in these two schemes. In a series of experiments, we found that non-trivial conversion of parser output format was required. Experimental results showed that nearly 20% of dependency relations are problematic even when we converted a gold-standard HPSG treebank, demonstrating the difficulty of format conversion. In practice, it is difficult to have reliable conversion between different dependency representations, even between GR and SD, which are superficially similar. While we identified several of the major problems in our format conversion programs, their solution is unclear and would likely require a more complex conversion process. These remaining problems may obscure the results of parser evaluation. In fact, the results we obtain using the two evaluation schemes do not agree, confirming previous findings that framework-independent evaluation remains a challenge. Our experience suggests that GR evaluation is a step in the right direction, but a more accurate conversion procedure from PTB-style output to GR format is necessary.

From these observations, we conclude that a possible direction for improved parser evaluation includes machinery for dealing with multiple valid heads and dependency types in gold standard data. Following the discussion in Section 5, it is important to reduce the disagreements in relation types and lexical heads. While GR provides a partial solution to the former through its hierarchy of relations, a significant portion of remaining problems are relation type mismatches caused by the argument/modifier distinction, text adjuncts, and ambiguity of POS tags. For the latter, a possible solution would be to annotate multiple candidate dependencies, any of which may be matched to parser output. It may also be desirable to determine the relative importance of relations in the evaluation. For example, in current schemes, the attachment of prepositional phrases is weighted identically to the attachment of determiners. While this is addressed in GR evaluation by having separate figures for precision and recall of each relation, complex results that include several dimensions can be difficult to interpret. Another example involves dependencies concerning idiomatic expressions, which may need to be excluded from the evaluation, since their structures vary significantly in different frameworks.

While this paper focused on parser evaluation at an intermediate representation, another direction is evaluation in end-to-end applications, such as information extraction and machine translation. In application-oriented, or task-based evaluation, some differences between parsers might be obscured because many other components contribute to overall system performance. However, this type of evaluation is indispensable for further understanding of how the characteristics of specific parsers make them more suitable in certain situations, and even to validate the results of more straightforward synthetic evaluations using gold-standard parsed data.

References

Black, E., Abney, S., Flickinger, D., Gdaniec, C., Grishman, R., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B. and Strzalkowski, T. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proc. DARPA Speech and Natural Language Workshop*, pages 306–311.

Briscoe, T. 2006. An introduction to tag sequence grammars and the RASP system parser. Computer Laboratory Technical Report 662, University of Cambridge.

Briscoe, T. and Carroll, J. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the PARC DepBank. In *Proc. COLING/ACL 2006 Poster Session*.

Briscoe, T., Carroll, J. and Watson, R. 2006. The second release of the RASP system. In *Proc. COLING/ACL-06 Demo Session.*

Burke, M., Cahill, A., O'Donovan, R., van Genabith, J. and Way, A. 2004a. Evaluation of an automatic annotation algorithm against the PARC 700 Dependency Bank. In *Proc. 9th International Conference on LFG*, pages 101–121.

Burke, M., Cahill, A., O'Donovan, R., van Genabith, J. and Way, A. 2004b. Treebank-based acquisition of wide-coverage, probabilistic LFG resources: project overview, results and evaluation. In *Proc. "Beyond Shallow Analyses" workshop at IJCNLP-04.*

Cahill, A., McCarthy, M., van Genabith, J. and Way, A. 2002. Parsing text with a PCFG derived from Penn-II with an automatic F-structure annotation procedure. In *Proc. 7th International Conference on LFG*, pages 76–95.

Carroll, J. and Briscoe, T. 2004. High precision extraction of grammatical relations. In H. Bunt, J. Carroll and G. Satta (eds.), *New Developments in Parsing Technology*, Kluwer Academic.

Carroll, J., Briscoe, T. and Sanfilippo, A. 1998. Parser evaluation: a survey and a new proposal. In *Proc. LREC 1998*, pages 447–454.

Charniak, E. 2000. A maximum-entropy-inspired parser. In *Proc. NAACL 2000*, pages 132–139.

Charniak, E. and Johnson, M. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proc. ACL 2005*.

Clark, S. and Curran, J. 2007. Formalism-independent parser evaluation with CCG and DepBank. In *Proc. ACL 2007*.

Clark, S. and Curran, J. R. 2004. Parsing the WSJ using CCG and log-linear models. In *Proc. 42th ACL*.

Clegg, A. B. and Shepherd, A. 2007. Benchmarking natural-language parsers for biological applications using dependency graphs. *BMC Bioinformatics* 8(1), 24.

Collins, M. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. 35th ACL*.

de Marneffe, M.-C., MacCartney, B. and Manning, C. D. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. LREC 2006*.

Hockenmaier, J. 2003. Parsing with generative models of predicate-argument structure. In *Proc. 41st ACL*, pages 359–366.

Hockenmaier, J. and Steedman, M. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proc LREC-2002*, Las Palmas, Spain.

Kaplan, R. M., Riezler, S., King, T. H., Maxwell, J. T. and Vasserman, A. 2004. Speed and accuracy in shallow and deep stochastic parsing. In *Proc. HLT/NAACL'04*.

Kim, J. D., Ohta, T., Tateisi, Y. and Tsujii, J. 2003. GENIA corpus — a semantically annotated corpus for bio-textmining. *Bioinformatics* 19, i180–182.

King, T. H., Crouch, R., Riezler, S., Dalrymple, M. and Kaplan, R. M. 2003. The PARC 700 Dependency Bank. In *Proc. LINC'03*.

Klein, D. and Manning, C. D. 2003. Accurate unlexicalized parsing. In *Proc. ACL 2003*.

Malouf, R. and van Noord, G. 2004. Wide coverage parsing with stochastic attribute value grammars. In *Proc. IJCNLP-04 Workshop "Beyond Shallow Analyses"*.

Marcus, M., Santorini, B. and Marcinkiewicz, M. A. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2), 313–330.

Matsuzaki, T., Miyao, Y. and Tsujii, J. 2007. Efficient HPSG parsing with supertagging and CFG-filtering. In *In Proc. IJCAI 2007*.

Miyao, Y. 2007. Enju 2.2 Output Specifications. Technical Report TR-NLP-UT-2007-1, Tsujii Laboratory, University of Tokyo.

Miyao, Y., Ninomiya, T. and Tsujii, J. 2005. Corpus-oriented grammar development for acquiring a Head-driven Phrase Structure Grammar from the Penn Treebank. In Keh-Yih Su, Jun'ichi Tsujii, Jong-Hyeok Lee and Oi Yee Kwong (eds.), *Natural Language Processing - IJCNLP 2004*, volume 3248 of *LNAI*, pages 684–693, Springer-Verlag.

Miyao, Y. and Tsujii, J. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proc. ACL 2005*, pages 83–90.

Ninomiya, T., Matsuzaki, T., Miyao, Y. and Tsujii, J. 2007. A log-linear model with an n-gram reference distribution for accurate HPSG parsing. In *Proc. IWPT 2007*.

Oepen, S., Flickinger, D. and Bond, F. 2004. Towards holistic grammar engineering and testing — grafting treebank maintenance into the grammar revision cycle. In *Proc. IJCNLP-04 Workshop "Beyond Shallow Analyses"*.

Preiss, J. 2003. Using grammatical relations to compare parsers. In *Proc. EACL 2003*, pages 291–298.

Pyysalo, S., Ginter, F., Haverinen, K., Laippala, V., Heimonen, J. and Salakoski, T. 2007a. On the unification of syntactic annotations under the Stanford dependency scheme: A case study on BioInfer and GENIA. In *Proc. BioNLP 2007*, pages 25–32.

Pyysalo, S., Ginter, F., Heimonen, J., Bjorne, J., Boberg, J., Jarvinen, J. and Salakoski, T. 2007b. BioInfer: a corpus for information extraction in the biomedical domain. *BMC Bioinformatics* 8(50).

Sagae, K., Miyao, Y. and Tsujii, J. 2007. HPSG parsing with shallow dependency constraints. In *Proc. ACL 2007*.

Toutanova, K., Markova, P. and Manning, C. 2004. The leaf projection path view of parse trees: exploring string kernels for HPSG parse selection. In *EMNLP 2004*.